HIDDEN CONDITIONAL RANDOM FIELDS FOR ACTION RECOGNITION

Department of Information and Computing Sciences
Utrecht University
Utrecht, the Netherlands

Master Thesis of

Lifang Chen

Student Number: 3554635

University Supervisor:
Dr. Robby T. Tan

Company Supervisor:
Dr. Nico van der Aa

June 2013

Acknowledgement

This master thesis project is a cooperation between Utrecht University and Noldus InnovationWorks. Noldus InnovationWorks is the research and innovation laboratory of Noldus Information Technology, where novel technologies, concepts and product prototypes for behavioral research on humans and animals are researched, developed, field-tested and commercialized. The project is carried out at Noldus' headquarter located in Wageningen, The Netherlands.

I would like to thank Dr. Nico van der Aa for his insightful and patient guidance. I would also like to thank Dr. Robby Tan for his supervision and suggestions throughout the whole thesis project. Elsbeth van Dam has shared her valuable work and experience on animal action classification. Many other people have been interested in this project and enthusiastically joined the discussions.

Abstract

In this thesis we apply Hidden Conditional Random Fields (HCRF) to action recognition. HCRF is a classification method modelling the structure among local observations. In our system, an image is modelled as a set of hidden part labels conditioned on their local features. For each action class, the probability of an assignment of part labels to local patch features is modelled by a Conditional Random Field (CRF). These class conditional CRFs are combined into an unified framework of HCRF, which treats the assignment of part labels as hidden variables. This model also combines the local patch features with the global feature of an image under the framework of HCRF. The model parameter is trained with a maximum likelihood criteria. We have also evaluated a baseline model of HCRF, called the root model. It only uses the global feature and it does not include the hidden part labels. The root model is trained with the maximum likelihood criteria as well.

An extension of HCRF, Max-Margin Hidden Conditional Random Field (MMHCRF), has also been applied to action recognition. MMHCRF extends HCRF by training with a maximum margin criteria. That is, it sets the model parameter in the way that the margin between the score of the correct action label and the scores of the other labels is maximized. We have also evaluated a baseline model of MMHCRF. Similar to the root model, this baseline model only uses the global feature, but it trains the model parameter with the max-margin criteria.

Based on HCRF and the root model, we have proposed a Part Labels method. This method learns the hidden part labels of each image using the model parameter trained by HCRF. It uses these part labels as a new set of local features and combines them with the global feature. It trains these features in the same way as the root model.

We have implemented and evaluated these five models on the Weizmann dataset, a human action dataset, and an animal behaviour dataset, called Noldus ABR dataset. Our experiments show that only modelling the spatial structures in 2D space is not sufficient for action recognition. It has been demonstrated that the classification results of the simpler models such as the root model and the multi-class SVM are comparable to the more complex model such as HCRF. We have also found that the performance of MMHCRF is heavily dependent on its model parameter initialization and other parameter settings. It is not a robust method compared to HCRF. The Part Labels method is also less robust than HCRF, but it can be an option to improve the performance as it explicitly used the information of the learned part labels. One of the goals of this project is to investigate alternatives for the automatic rodent behaviour recognition module developed at Noldus IT. We have improved its action classification performance by 15%, promising a more robust action recognition tool for rodent behaviour.

Contents

Chapter 1

Introduction

1.1 Background

Human visual perception is capable of recognizing complex human and animal actions from videos. Vision-based *action recognition*, which is the process of labelling video sequences with action labels [1], allows computers to have the recognition ability similar to human eyes. Action recognition is an important research field in computer vision. It can be applied to both human actions and animal behaviours. The applications of human action recognition include human-computer interaction, user interface design, robot learning, and automatic surveillance. Automatic annotation of animal behaviours, such as rodent behaviours, is crucial for neuroscience and pharmacology research [2].

Action recognition is a challenging task because of the degree of intra-class and inter-class variations. As an example, consider the action walking. The intra-class variations apply to the spread in walking speed and step size performed by different people. On the other hand, actions like jogging are very similar to walking, which is an example of inter-class variations. What makes it even more difficult is the variations in environment and record settings, like viewpoint change, dynamic background and occlusions. Figure 1.1 shows the examples of walking and jogging. A good action recognition method should be able to generalize over variations within one action, distinguish between different actions, and be invariant to environment and record settings.

Research in action recognition share a common framework, where image features are extracted from videos first and next these features are classified into different actions. These are supervised learning methods.

Feature extraction methods have been intensively explored to better describe videos and their contained actions. There are two major categories: global representations and local representations. Global representations describe images or videos as a whole. Global representations are powerful because they have the most information. Local representations describe images or videos as a collection of patches. A common approach is to detect interest points first, and subsequently calculate patches around these points. Local representations provide concise descriptors for videos, but they mostly ignore the overall structure information.

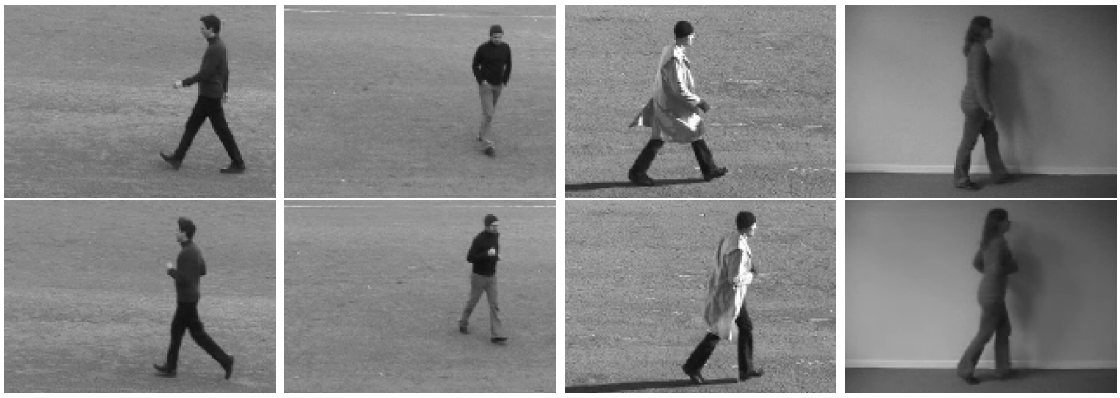


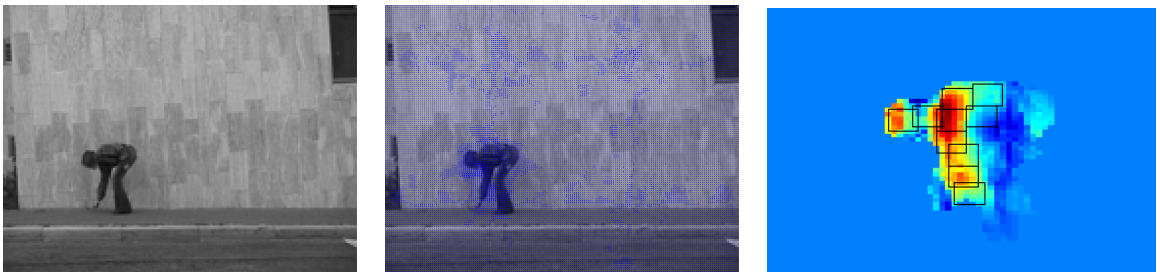
Figure 1.1: Inter-variations and intra-variations. Images in the first row show four individuals walking. There are variations on background, clothes, camera view points and step sizes. Images in the second row show the same individuals jogging. These examples are taken from the KTH human action dataset [3].

Optical flow [4] is the oriented difference between the corresponding pixels on subsequent frames. It is a widely used motion feature which naturally captures the appearance invariant motion information. The pixel-wise characteristic of optical flow determines that it can easily be used for both global and local representations. Figure 1.2b is an example of the optical flow. The blue arrows point to the directions where pixels move toward to. Figure 1.2b is a global representation. Figure 1.2c shows the local patches found out in this frame. These patches locate on the area with the most optical flow energy. Optical flow within these patches forms a local representation.

When features are extracted from a frame or a video, action recognition can be considered as a classification problem. The frequently used classification methods are k-Nearest Neighbor (kNN) [5], Support Vector Machines (SVM) [6] and boosting [7]. These approaches assume features in a feature vector are independent of each other, which is often not the case. Moreover, they ignore the spatial and temporal structures of an action, which is important for action recognition. As an example for the spatial structure, the patches in an image may correspond to different body parts of a person, whose spatial arrangement is important for the classification task. The temporal structure is of apparent importance for action recognition: consecutive frames may correspond to different phases of an action. Graphical models such as hidden Markov model [8], conditional random fields [9] and hidden conditional random fields [10] are introduced into action recognition to model the sophisticated spatial and temporal structures.

1.2 Motivation

In the field of action recognition, the design and calculation of features have been extensively explored. There is a large pool of features available and they can meet the requirements of different applications, either simple or complex, time-consuming or in real-time, universally applicable or domain specific. On the contrary, the choice of action classification methods is limited. The most commonly used classification methods are still the universally applicable



(a) Original image (b) Global optical flow (c) Local patches

Figure 1.2: The global and local representation of optical flow. Figure 1.2a shows the original image of action "bend". Figure 1.2b shows the global optical flow features computed from this image and its following frame. Figure 1.2c shows the local patches found on this frame.

machine learning methods, such as kNN and SVM. These methods do not explicitly take into account the spatial or temporal structures which are important for action recognition. Thus we consider action classification as a research topic with great potential and worth more research attention. It is necessary to investigate more advanced classification methods that are more suited for handling the level of complexity in action recognition.

Recently, a few methods have been proposed to model the spatial or temporal interdependencies of actions. Hidden Conditional Random Fields (HCRF) [10] stands out among these methods because it can model the spatial and temporal structures without assuming conditional independence among the underline features. Therefore, it is suited to including rich, overlapping features.

One of the research projects carried out by Noldus Information Technology, Automated Behaviour Recognition system (ABR) [2], is to automatically annotate rat behaviours such as "eat" and "groom". This project provides a rat behaviour dataset which is manually annotated frame by frame. It has chosen a set of complex and highly dependent features. The details of these features will be explained in Chapter 6. These features are classified using a quadratic classifier based on normal densities [11]. Yet like most other conventional action classifiers, this classifier ignores the dependencies among features. In order to solve this problem, we consider HCRF.

HCRF models the spatial and temporal structures by introducing structured hidden variables. In action classification tasks using local patch features, HCRF does not use the local features directly for classification, but first assign a part label to each patch, and then using these part labels for action classification. These part labels are not provided by the training data, so they are hidden variables. There is evidence [12, 10] showing that incorporating hidden variables can improve the performance. HCRF is not the only method to introduce hidden variables. The commonly used bag-of-words approach, for instance, assigns a "word" to each patch. The assignment of words is not available in the training data, but learned during the training process, so words are also hidden variables. The concept of words for bag-of-words is similar as part labels for HCRF.

But there are certain structures, or dependencies, among these hidden variables: one of the most notable dependencies is the spatial dependency among patches. For example, two

patches on the upper arm tend to have the same part label. Bag-of-words ignores this spatial dependency, and assumes all patches in an image are independent of each other. HCRF, however, relaxes this independency assumption by modelling interactions between nearby patches.

There are other models, such as Hidden Markov Model (HMM) [8] and Conditional Random Fields (CRF), being used to model the spatial and temporal structures. HMM can also assign hidden part labels to the patches, as HCRF does. The limitation of HMM is that it assumes the observed data (i.e., local patch features) are independent with each other given their hidden part labels. This is often too restrictive for action recognition. HCRF relaxes this assumption by allowing local features to be overlapping and dependent on each other.

CRF is a structural prediction method to model the structures underline the local patch features. It does not require the conditional independence assumption as HMM does. However, the output of CRF is not a single class label, which is required by action classification, but a set of part labels assigned to the local patches. In order to train CRFs for action recognition, the correct assignment of part labels to the local patches must be provided in the training data. But this information is often not available in public datasets, and manual annotation is a time-consuming task. HCRF is based on CRF, but it directly outputs the action label and does not require the assignment of part labels in the training data.

HCRF was introduced into action recognition by Wang and Mori [13]. This work models the spatial structure underline the local patches. In addition, it combines the benefit of the global features with that of the local patch features, under the framework of HCRF. The goal of this thesis is to evaluate the application of HCRF on action recognition, and to apply this framework to the Noldus ABR system.

1.3 Overview

In this section we provide an overview of the models investigated in this thesis. Our work is based on the methods proposed by Wang and Mori [13, 14]. Both [13] and [14] model an image as a constellation of body parts conditioned on the features of local patches. They also combine the local patch features with the global features. But they use different training criteria to learn the model parameter. We have proposed a new Part Labels method based on HCRF. This method extracts the body parts information from the trained HCRF and uses them together with the global features for classification.

The method proposed by [13] is built upon the optical flow features in [15]. These features are calculated both on the whole image and on the detected interest points. Figure 1.2 is a visualization of these features. This method models an image as a constellations of parts conditioned on these features. For each action class, a given assignment of part labels to local patches is modelled as a CRF [9]. It combines these CRFs into a unified framework HCRF by incorporating the part labels as hidden variables. The parameters of HCRF are trained by maximizing the conditional log-likelihood. The merits of this method are that it models the spatial structures among local patches and that it combines the strength of both local and global representations.

In order to evaluate HCRF, we have also implemented a baseline model of HCRF. This

method, called the root model, does not model the spatial structure among local patches. It only uses the global feature of an image and trains them by maximizing their conditional log-likelihood, which is the same as HCRF.

Wang and Mori [14] have also proposed a Max-Margin Hidden Conditional Random Fields (MMHCRF), which is similar with HCRF but trains the model parameter by maximizing the margin. The goal of the max-margin learning algorithm is to train the model parameters in a way that the score of the correct action label is higher than the scores of other labels by a large margin, whereas the goal of the maximum log-likelihood approach is to maximize the probability of the correct action label. Given the trained model, both HCRF and MMHCRF classify an image as the action label with the highest score.

Similar with the root model and HCRF, we also evaluate a baseline model of MMHCRF. This model also only uses the global feature for training and classification. But it trains the features with a max-margin criteria, which is the same as MMHCRF. Without the part labels, this model is actually a standard multi-class SVM [16].

Our proposed Part Labels method utilizes the model parameter trained by HCRF to find the best assignment of part labels to local patches. These part labels are considered as refined descriptors of the local patches. We concatenate them with global features to form a new feature vector. These new feature vectors are used for a new phase of training using a gradient-based method. For each new image, we use the HCRF model parameter to find its part labels, concatenate them with its original features and classify them using the new model parameters.

1.4 Contributions

This thesis project aims to build a system capable of automatic action recognition using HCRF. We have implemented and evaluated the application of HCRF on action recognition, based on the method proposed by Wang and Mori [13]. We have also investigated the max-margin version of HCRF [14]. We have made a comparison of these two methods. Based on HCRF we proposed a method to extract part labels as local features for new classification. The contributions of this thesis are:

- The development, evaluation and comparison of the root model, HCRF, multi-class SVM, MMHCRF and the Part Labels method.
- Proposed the Part Labels method for action recognition. It extracts part labels learned from HCRF, concatenates them with original features, and uses this new feature vector for action recognition.
- Apply the HCRF framework on the Noldus ABR dataset [2], and substantially improved the classification rate on this dataset.
- Discovered the strength of the root model, an efficient classification method which only uses the global features.
- Improved the method for motion features computation. We move the optical flow calculation step to the front of the video stabilization step. This adjustment has improves

the performance of the root model.

- Provided two methods to deal with the root potential in the HCRF objective function.

1.5 Layout

This thesis is organized as follows: Chapter 2 discusses the related work on action recognition. Chapter 3 describes the theory of HCRF. Chapter 4 describes the details of MMHCRF and makes a comparison between HCRF and MMHCRF. Chapter 5 explains the method to combine global features and part labels. Chapter 6 describes the evaluation of the action classification methods. Chapter 7 gives a conclusion and lists possible future works.

Chapter 2

Related work

Action recognition is a hot topic in the field of computer vision. The amount of literature on action recognition has grown rapidly over the past few years. The task of action recognition is generally split into two parts: feature extraction and classification. In this chapter we summarize methods that are related to our topic on both parts.

2.1 Feature Extraction

Feature describes the characteristic of an image or a video in a way that it is sufficiently rich to allow robust action classification and is invariant to individual appearance, background, viewpoint and action execution. Researchers have explored an extensively number of methods to extract features from videos. Temporal information is an important element in action performance. Optical flow [15] is a frequently used motion feature extracted from every two consecutive frames. Optical flow captures the motion information between frames by calculating the pixel-wise oriented difference.

Features can also be divided into two categories by scope: global features and local features. Global representations describe the region of interest as a whole. Efos et al. have [4] designed a motion feature that splits the optical flow into horizontal and vertical, positive and negative channels and blur them to only capture the important location information. This motion feature works well on low-resolution images and it is robust against errors in optical flow calculation.

Local features describe an image or a video as a collection of patches. It needs to detect the interest points where changes of movement occur. Felzenszwalb et al. [12] proposed a simple heuristic to find salient patches having the highest positive optical flow energy. In this way the chosen patches are on the most intensively moving body parts, like arms in the action "wave hands". Local patch features around these interest points can be calculated using descriptors such as optical flow [15].

Wang and Mori [13] use the motion feature proposed by [4] and combine its global representation with its local representations where the patches are identified using a heuristic similar to [12]. They also incorporate location information into a local representation. Van Dam et

al. [2] introduced a set of features especially designed for rodent behaviour recognition. This feature includes both motion features and tracking features. Motion features are extracted from optical flow statistics on different rat body parts. Tracking features describes the animal location and shape, which are obtained from animal tracking software. This set of features contain both global features on the whole animal and local features on different body parts.

2.2 Action Classification

After features are extracted, assigning action labels to videos becomes a classification problem. The commonly used classification methods are kNN [5], SVM [6] and boosting [7]. These methods can also be combined together. For example, the bag-of-words [17] framework uses clustering to find the word vocabulary, maps the descriptors to these words to obtain the word frequency histogram, and classifies these histograms using SVM or boosting. In this way, the bag-of-words approach introduces the hidden variables "words" to grasp the characteristic of each local patch.

Next to these conventional classifiers, more complex graphical models are introduced into action classification. These models are either generative or discriminative. Generative approaches model a joint probability distribution over both local features and their part labels. It requires a prior model over features. When features are dependent on each other, it is difficult to model this prior. To model it tractably, generative approaches often assume the features are conditionally independent on their labels. A typical generative approach is HMM. HMM can use hidden states corresponding to different phases in an action [8]. The disadvantage of generative models is that the independency assumption is often too restrictive for action recognition, as the features in action recognition often dependent on each other.

Discriminative approaches overcome this problem by directly modelling a conditional distribution over action classes given features. It does not need to model the prior on features, thus the independence assumption is relaxed. CRF [18] is a discriminative approach that can use dependent and overlapping features. But CRF requires fully labelled data where each observation node has a intermediate level label. For example, if patches in a frame are considered as the observation nodes, each of these patches should be assigned a part label, like "hands up" or "put down leg". Unfortunately most available datasets do not provide this intermediate labelling. Quattoni et al. [10] proposes the HCRF model which extends CRF to incorporate these intermediate part labels as hidden variables. The assignments of these hidden variables are learned during training, not required in the dataset.

HCRF was originally proposed for object recognition [10]. Later it has also been applied to gesture recognition [19], where the model captures temporal dependencies across frames. Wang and Mori [13] used HCRF for action recognition by modelling the spatial dependencies of patches within a frame. They use HCRF to model a human action as a constellation of parts conditioned on image features. A major contribution of this work is that they have improved the classification performance by combining the flexibility of local representation and the large-scale global representation under the unified framework of HCRF.

Max-margin methods have been successful in machine learning [20, 12, 16]. Max-margin approache sets separating hyperplanes in the way that the margin between the correct label

and all other ones is maximized, so that the score of the correct label is much higher than the incorrect ones.

Felzenszwalb et al. [12] have proposed the Latent Support Vector Machine (LSVM). LSVM learns a discriminative model with structured hidden (or latent) variables similar to HCRF. But it trains the model parameter with a max-margin approach, instead of the maximum likelihood approach adopted by HCRF. LSVM is a binary classifier which does not directly handle multi-class classification. Crammer and Singer [16] introduced the standard multi-class SVM which extends the binary SVM to directly support multi-class classification. In the way similar with [16], Wang and Mori [14, 21] proposed MMHCRF to extend LSVM to directly handle multi-class classification.

Our work is based on the HCRF model [13] and the MMHCRF model [14]. Both methods model the spatial structure of an image by introducing structured hidden variables. But HCRF learns the model parameter with an maximum likelihood approach, while MMHCRF adopts an max-margin approach. We have also proposed our own method based on the HCRF model.

Chapter 3

Hidden Conditional Random Fields

3.1 CRF and HCRF

CRF [9] is a popular graphical modelling method to predict a set of variables that depend on each other as well as other observed variables. CRF is an undirected graphical model which defines a conditional distribution over labelling given an observation. Given an observed feature vector $\mathbf{x}$, CRF is used to predict a vector $\mathbf{h} = \{h_1, h_2, \ldots, h_m\}$ of random variables. In our case, given the motion feature of an image including a rat, we want to predict the movement pattern of rat body parts: head, middle, rear. The complex dependencies between the output variables are represented by a graphical model. For example, the movements of head and middle body are related to each other, so as the movements of middle body and rear. CRF is a discriminative approach to directly model the conditional likelihood $P(\mathbf{h}|\mathbf{x})$. The parameters of CRF are estimated by maximizing the joint conditional likelihood on all training samples. Its advantage is that it does not assume conditional independence among features $\mathbf{x}$.

In most cases of action recognition, training data does not provide the assignment of part labels. It only has one action label y for each sample. HCRF incorporate hidden variables to solve this problem. It models an image as a constellation of parts conditioned on local patch features. For a given action class, the probability of a certain assignment to the patches $P(\mathbf{h}|\mathbf{x}, y)$ is modelled using CRF. HCRF combines these CRFs conditioned on each class into one framework. In this framework, the assignment of parts are hidden variables. In other words, the assignment of parts $\mathbf{h}$ are learned during training. The training data does not need to provide them. The parameters of HCRF are also estimated in a maximum likelihood approach. In the rest of this chapter, we will describe the HCRF model and its parameter estimation process.

3.2 Hidden Part Model

In this section we describe how to model a frame I in a video sequence. Let $\mathbf{x}$ be the feature extracted from I, and y be its action label. Denote $\mathcal{Y}$ as the set of possible action classes.

For example, $\mathcal{Y} = \{walk, drink\}$. Assume I contains a set of patches $\{I_1, I_2, \ldots, I_m\}$, and its corresponding features can be written as $\mathbf{x} = \{x_0, x_1, \ldots, x_m\}$. x_0 is the global feature vector which is extracted from the whole frame, and $x_i\,(i = 1 \ldots m)$ is the local feature vector extracted from patch I_i. For example, in our experiment x_0 contains all features extracted from the whole rat body, and each patch x_i contains features extracted from a rat body part, such as head, middle body or rear. Our training set consists of labelled frames $(\mathbf{x}_t, y_t)$ for $t = 1 \ldots T$.

Assume we can assign each patch I_i with a hidden part label h_i. Thus each frame I has a vector of hidden part labels $\mathbf{h} = \{h_1, h_2, \ldots, h_m\}$. Denote $\mathcal{H}$ as the finite set of possible part labels such that each $h_i \in \mathcal{H}$. The meaning behind a hidden part label is the motion pattern of a body part. For example, the part labels for the head can be patterns of move forward or move backward. The values of $\mathbf{h}$ are not observed in training examples. Instead, they will be learned during training, thus they will become the hidden variables of the model.

Assume there are dependencies between some pairs of (h_j, h_k). For example, in the case of walking, head and rear might have the dependence that they both tend to move forward. Assume there is an undirected graph structure $G = (V, E)$ for each frame, in which $h_i\,(i = 1, 2, \ldots m)$ are the set of vertices V, and the constraint between h_j and h_k is edge $(j, k) \in E$. Note that the graph structure can be different from image to image. Figure 3.1 shows the model structure.

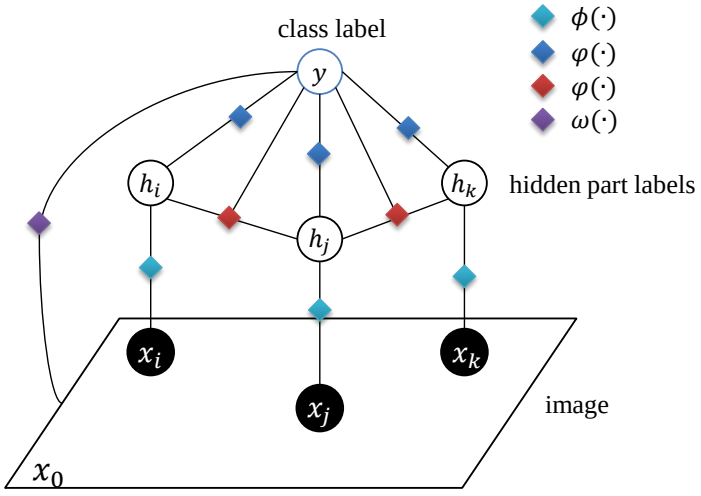


Figure 3.1: Illustration of the model. Each circle corresponds to a variable, and each square corresponds to a factor in the model.

The graph G encodes the connectivity between the hidden part labels. Intuitively, G determines the ability of our model to capture conditional dependencies between the hidden part labels. Theoretically, such conditional dependence can be encoded using any graph structure, but in this thesis we assume G is a tree for two reasons. First of all, Quattoni et al. [22] has proved that a minimal spanning tree model has equivalent performance with models using more densely connected graphs, therefore a tree structure encodes enough dependency constraints. Secondly, a more densely connected graph leads to an increase in the computational complexity of performing inference in such models. The inference problem of a tree-structure model can be exactly solved by belief propagation algorithm who will be explained in Section

3.5. We will explain how to find the graph structure in Section 3.6.

Given the definitions of feature $\mathbf{x}$, part labels $\mathbf{h}$, and class label y, we can define a potential function $\theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y)$ which is parametrized by the model parameter θ:

$$\theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y) = \sum_{j \in V} \alpha^\intercal \cdot \phi\left(x_j, h_j\right) + \sum_{j \in V} \beta^\intercal \cdot \varphi\left(y, h_j\right) + \sum_{(j,k) \in E} \gamma^\intercal \cdot \psi\left(y, h_j, h_k\right) + \eta^\intercal \cdot \omega\left(y, x_0\right), \quad (3.1)$$

where α, β, γ and η are the components of θ, in other words $\theta = \{\alpha, \beta, \gamma, \eta\}$. Φ is linear with respect to θ. $\phi(\cdot)$, $\varphi(\cdot)$, $\psi(\cdot)$ and $\omega(\cdot)$ are functions defining the features of the model. We will explain their definitions in detail in the rest of this section.

Unary potential $\alpha^\intercal \cdot \phi\left(x_j, h_j\right)$. This potential function models how likely patch x_j is assigned with part label h_j: it measures the compatibility between x_j and h_j. It is parametrized as

$$\alpha^\intercal \cdot \phi\left(x_j, h_j\right) = \sum_{c \in \mathcal{H}} \alpha_c^\intercal \cdot \mathbf{1}_{\{h_j = c\}} \cdot x_j, \quad (3.2)$$

where α_c measures the compatibility between feature vector x_j and part label $h_j = c$. α is the concatenation of α_c for all $c \in \mathcal{H}$. x_j is the feature vector of patch I_j. It incorporates features describing both patch appearance and spatial location. For example, the appearance features can be the optical flow calculated out of this patch; the location features can be the relative location of this patch on the image. α_c and x_j are two vectors with the same length. An illustration of this potential function is showed in Figure 3.2. We can see that the length of α is $|\mathcal{H}||x_j|$, where $|x_j|$ is the length of feature vector on patch j. We assume all feature vectors of patches to have the same length, in other words, $|x_1| = |x_2| = \cdots = |x_m|$.

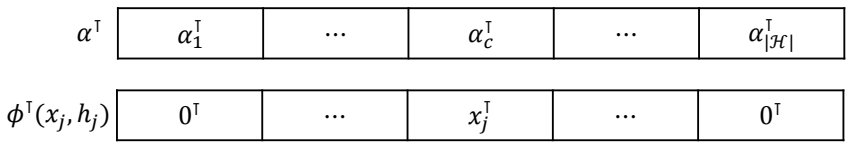


Figure 3.2: Illustration of unary potential function.

Unary potential $\beta^\intercal \cdot \varphi\left(y, h_j\right)$. This potential function measures how likely an image with class label y contains a patch with part label h_j. It models the compatibility between class label y and part label h_j. It is parametrized as

$$\beta^\intercal \cdot \varphi\left(y, h_j\right) = \sum_{a \in \mathcal{Y}} \sum_{b \in \mathcal{H}} \beta_{a,b} \cdot \mathbf{1}_{\{y = a\}} \cdot \mathbf{1}_{\{h_j = b\}}, \quad (3.3)$$

where $\beta_{a,b}$ measures the compatibility between class label $y = a$ and part label $h_j = b$. An illustration of this potential function is showed in Figure 3.3. The length of β is $|\mathcal{Y}||\mathcal{H}|$.

Pairwise potential $\gamma^\intercal \cdot \psi\left(y, h_j, h_k\right)$. This potential function measures how likely an image with class label y contains a pair of part labels h_j and h_k, where $(j, k) \in E$ is an edge in

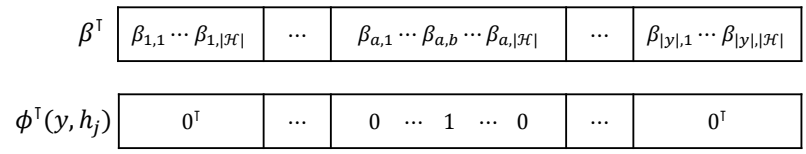


Figure 3.3: Illustration of unary potential function.

graph G. This potential function respects the structure of graph G. It is parameterized as

$$\gamma^\intercal \cdot \psi\left(y, h_j, h_k\right) = \sum_{a \in \mathcal{Y}} \sum_{b \in \mathcal{H}} \sum_{c \in \mathcal{H}} \gamma_{a,b,c} \cdot \mathbf{1}_{\{y=a\}} \cdot \mathbf{1}_{\{h_j=b\}} \cdot \mathbf{1}_{\{h_k=c\}}, \tag{3.4}$$

where $\gamma_{a,b,c}$ measures the compatibility of class label $y = a$, $h_j = b$ and $h_k = c$ for edge $(j, k) \in E$. This potential function is illustrated in Figure 3.4. The length of γ is $|\mathcal{Y}||\mathcal{H}|^2$.

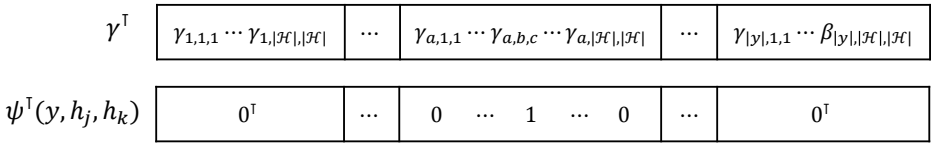


Figure 3.4: Illustration of pairwise potential function.

Root potential $\eta^\intercal \cdot \omega\left(y, x_0\right)$. This potential function measures the compatibility of class label y and the global feature of the whole image. It is parametrized as

$$\eta^\intercal \cdot \omega\left(y, x_0\right) = \sum_{a \in \mathcal{Y}} \eta_a^\intercal \cdot \mathbf{1}_{\{y=a\}} \cdot x_0, \tag{3.5}$$

where x_0 is the global feature vector. η_a measures the compatibility between the global feature and class label $y = a$. η is the concatenation of η_a for all $a \in \mathcal{Y}$. This potential function is illustrated in Figure 3.5. The length of η is $|\mathcal{Y}||x_0|$, where $|x_0|$ is the length of global feature vector.

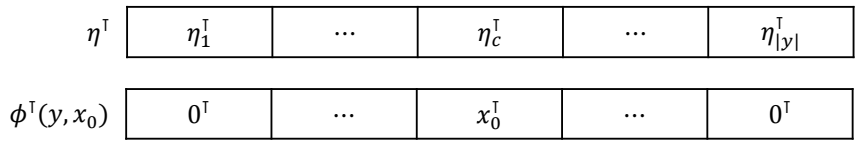


Figure 3.5: Illustration of root potential function.

This is the definition of the hidden part model, the potential function and its parametrization. This potential function is similar to the one originally used in object recognition [10]. But there are two differences. First, the potential function in [10] only has the unary potential and pairwise potential, or the first three components of Eq.(3.1). But this definition also has a root potential $\eta^\intercal \cdot \omega\left(y, x_0\right)$ to describe the relationship between the global feature and action label directly. This potential function combines the local patch features with the global features. Second, the unary potential $\alpha^\intercal \cdot \phi\left(x_j, h_j\right)$ uses both appearance feature and location information of the patches in the feature vector x_j, while [10] only uses the appearance feature.

3.3 Conditional Probabilistic Model

Given the definition of potential function $\theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y)$, we could define a conditional probabilistic model:

$$P(y, \mathbf{h}|\mathbf{x}, \theta) = \frac{\exp\left(\theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y)\right)}{\sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} \exp\left(\theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y')\right)}, \tag{3.6}$$

where y is the class label, $\mathbf{h}$ is the set of part labels for each patch, $\mathbf{x}$ is the feature vector, and θ is the weight parameters of the model. Its denominator is a normalization term which sums over all possible class label $y' \in \mathcal{Y}$ and all possible combinations of $\mathbf{h}$. It follows that when the feature of an image $\mathbf{x}$ and model parameter θ are known, the probability of this image has class label y is the summation of conditional probabilities $P(y, \mathbf{h}|\mathbf{x}, \theta)$ over all possible assignments of part labels $\mathbf{h}$:

$$P(y|\mathbf{x}, \theta) = \sum_{\mathbf{h}} P\left(y, \mathbf{h}|\mathbf{x}, \theta\right) = \frac{\sum_{\mathbf{h}} \exp\left(\theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y)\right)}{\sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} \exp\left(\theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y')\right)}. \tag{3.7}$$

From Eq.(3.6) and Eq.(3.7), we can use Bayes' rule to derive the joint probability of assigning a set of part labels $\mathbf{h}$ to patches of an image when its features $\mathbf{x}$, class label y and weight parameters θ are known:

$$P\left(\mathbf{h} \mid y, \mathbf{x}, \theta\right) = \frac{P(y, \mathbf{h}|\mathbf{x}, \theta)}{P(y|\mathbf{x}, \theta)} = \frac{\exp\left(\theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y)\right)}{\sum_{\mathbf{h}} \exp\left(\theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y)\right)}. \tag{3.8}$$

Following previous work on CRF [18, 9], we want to maximize the joint conditional probability $P(y|\mathbf{x}, \theta)$ for all training examples. The objective function used for training parameters θ is defined as:

$$L\left(\theta\right) = \sum_{t} \log P\left(y_t|\mathbf{x}_t, \theta\right) - \frac{1}{2\sigma^2}||\theta||^2. \tag{3.9}$$

The first term in Eq.(3.9) is the conditional log-likelihood on the training images. The second term is a penalized term to prevent the L_2 norm of the model parameter $||\theta||$ becoming too big. It is the log of a Gaussian prior with variance σ^2. That is, we assume the model parameter follows a normal distribution $P\left(\theta\right) \sim N(0, \sigma^2)$ to constrain $||\theta||$ [23]. The optimal θ is learned by maximizing the objective function in Eq.(3.9), thus

$$\theta^* = \arg\max_{\theta} L\left(\theta\right). \tag{3.10}$$

The optimal θ^* which maximize L can not be computed analytically; instead we need to employ iterative methods to estimate it. We will explain how to search for the optimal θ^* in the next section.

3.4 Parameter Estimation

In this section we estimate the optimal weight parameter $\theta^* = \arg\max_{\theta} L\left(\theta\right)$ from a set of training samples. We use iterative gradient-based optimization methods such as limited-memory BFGS [24] and stochastic gradient ascent [25] to search for the optimal θ. These methods require repeated evaluation of objective function L and its derivatives with respect

to each model parameter in θ. But similarly as with other hidden state models like HMMs, adding hidden states $\mathbf{h}$ makes the objective function $L(\theta)$ not convex [10]. Therefore this method does not guarantee reaching the global optimal point. But we can still find θ that is locally optimal.

We will describe how to efficiently calculate the gradient of $L(\theta)$. Denote the log-likelihood of the t-th training example as

$$\begin{aligned} L_t(\theta) &= \log P(y_t \mid \mathbf{x}_t, \theta) \\ &= \log \frac{\sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y_t)\right)}{\sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y')\right)} \\ &= \log \sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y_t)\right) - \log \sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y_t)\right). \end{aligned} \tag{3.11}$$

So $L(\theta)$ and its derivative with respect to θ can be written as:

$$L(\theta) = \sum_t L_t(\theta) - \frac{1}{2\sigma^2}||\theta||^2, \tag{3.12}$$

$$\frac{\partial L(\theta)}{\partial \theta} = \sum_t \frac{\partial L_t(\theta)}{\partial \theta} - \frac{\theta}{\sigma^2}. \tag{3.13}$$

In the following we will calculate $\dfrac{\partial L_t(\theta)}{\partial \theta}$, the gradient where the t-th training example contributes to. First let us consider the gradient of $L_t(\theta)$ with respect to α, the weight parameter in the unary potential in Eq.(3.2). Note that α is a vector, so its gradient $\partial L_t(\theta)/\partial \alpha$ is a vector of the same length as α and feature vector $\phi(x_j, h_j)$, that is $|\mathcal{H}||x_j|$.

$$\begin{aligned} \frac{\partial L_t(\theta)}{\partial \alpha} &= \frac{\sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y_t)\right) \cdot \dfrac{\partial \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y_t)}{\partial \alpha}}{\sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y_t)\right)} \\ &\quad - \frac{\sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y')\right) \cdot \dfrac{\partial \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y')}{\partial \alpha}}{\sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y')\right)} \\ &= \sum_{\mathbf{h}} P(\mathbf{h} \mid y_t, \mathbf{x}_t, \theta) \frac{\partial \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y_t)}{\partial \alpha} - \sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} P\left(y', \mathbf{h} \mid \mathbf{x}_t, \theta\right) \frac{\partial \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y')}{\partial \alpha} \\ &= \sum_{\mathbf{h}} P(\mathbf{h} \mid y_t, \mathbf{x}_t, \theta) \sum_{j \in V} \phi(x_{t,j}, h_j) - \sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} P\left(y', \mathbf{h} \mid \mathbf{x}_t, \theta\right) \sum_{j \in V} \phi(x_{t,j}, h_j), \end{aligned} \tag{3.14}$$

where $P(\mathbf{h} \mid y, \mathbf{x}, \theta)$ and $P(y, \mathbf{h} \mid \mathbf{x}, \theta)$ are defined in Eq.(3.8) and Eq.(3.6) respectively.

Yet calculating the derivatives following Eq.(3.14) with brute force is intractable, because there are exponentially many possible assignments of $\mathbf{h}$. If the image has m patches, there are $|\mathcal{H}|^m$ possible $\mathbf{h}$. Summing over this number of terms is prohibitively expensive.

Fortunately, there exists a belief propagation algorithm [26] to calculate marginal probabilities efficiently. So we would like to write Eq.(3.14) in terms of marginal probabilities and their

normalization term [26], all of which can be easily calculated using the BP algorithm. We will describe how to compute these marginal probabilities using the BP algorithm in the next section.

Let us define the marginal probabilities and their normalization term:

$$\forall y \in \mathcal{Y}, \qquad Z\left(y \mid \mathbf{x}, \theta\right) = \sum_{\mathbf{h}} \exp\left(\theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y)\right), \tag{3.15}$$

$$\forall y \in \mathcal{Y}, \forall j \in V, \forall a \in \mathcal{H}, \qquad P\left(h_j = a \mid y, \mathbf{x}, \theta\right) = \sum_{\mathbf{h}: h_j = a} P\left(\mathbf{h} \mid y, \mathbf{x}, \theta\right), \tag{3.16}$$

$$\forall y \in \mathcal{Y}, \forall (j,k) \in E, \forall a \in \mathcal{H}, \forall b \in \mathcal{H}, \quad P\left(h_j = a, h_k = b \mid y, \mathbf{x}, \theta\right) = \sum_{\mathbf{h}: h_j = a, h_k = b} P\left(\mathbf{h} \mid y, \mathbf{x}, \theta\right), \tag{3.17}$$

Eq.(3.15) defines a normalization term $Z(y|\mathbf{x}, \theta)$ that sums over all possible $\mathbf{h}$. Eq.(3.16) defines a marginal probability over an individual variable h_j. Eq.(3.17) defines a marginal probability over pairs of variables h_j and h_k, which correspond to edges in graph G.

Using Eq.(3.16), we can rewrite gradients in Eq.(3.14) as:

$$\frac{\partial L_t\left(\theta\right)}{\partial \alpha} = \sum_{j \in V} \sum_{a \in \mathcal{H}} P\left(h_j = a \mid y_t, \mathbf{x}_t, \theta\right) \phi\left(x_{t,j}, h_j\right) - \sum_{y' \in \mathcal{Y}} \sum_{j \in V} \sum_{a \in \mathcal{H}} P\left(h_j = a, y' \mid \mathbf{x}_t, \theta\right) \phi\left(x_{t,j}, h_j\right). \tag{3.18}$$

Similarly, we can obtain the gradients of $L_t\left(\theta\right)$ with respect to β,γ and η:

$$\frac{\partial L_t\left(\theta\right)}{\partial \beta} = \sum_{j \in V} \sum_{a \in \mathcal{H}} P\left(h_j = a \mid y_t, \mathbf{x}_t, \theta\right) \varphi\left(y_t, h_j\right) - \sum_{y' \in \mathcal{Y}} \sum_{j \in V} \sum_{a \in \mathcal{H}} P\left(h_j = a, y' \mid \mathbf{x}_t, \theta\right) \varphi\left(y', h_j\right), \tag{3.19}$$

$$\begin{aligned} \frac{\partial L_t\left(\theta\right)}{\partial \gamma} = & \sum_{(j,k) \in E} \sum_{a \in \mathcal{H}} \sum_{b \in \mathcal{H}} P\left(h_j = a, h_k = b \mid y_t, \mathbf{x}_t, \theta\right) \psi\left(y_t, h_j, h_k\right) \\ & - \sum_{y' \in \mathcal{Y}} \sum_{(j,k) \in E} \sum_{a \in \mathcal{H}} \sum_{b \in \mathcal{H}} P\left(h_j = a, h_k = b, y' \mid \mathbf{x}_t, \theta\right) \psi\left(y', h_j, h_k\right), \end{aligned} \tag{3.20}$$

$$\frac{\partial L_t\left(\theta\right)}{\partial \eta} = \omega\left(y_t, x_{t,0}\right) - \sum_{y' \in \mathcal{Y}} P\left(y' \mid \mathbf{x}_t, \theta\right) \omega\left(y', x_{t,0}\right). \tag{3.21}$$

Note that on the right side of Eq.(3.18), Eq.(3.19) and Eq.(3.20), $P\left(h_j = a, y' \mid \mathbf{x}_t, \theta\right)$ and $P\left(h_j = a, h_k = b, y' \mid \mathbf{x}_t, \theta\right)$ are not marginal probabilities as defined above. But they can be written in terms of marginal probabilities by the product rule :

$$P\left(h_j = a, y' \mid \mathbf{x}_t, \theta\right) = P\left(h_j = a \mid y', \mathbf{x}_t, \theta\right) P\left(y' \mid \mathbf{x}_t, \theta\right), \tag{3.22}$$

$$P\left(h_j = a, h_k = b, y' \mid \mathbf{x}_t, \theta\right) = P\left(h_j = a, h_k = b \mid y', \mathbf{x}_t, \theta\right) P\left(y' \mid \mathbf{x}_t, \theta\right). \tag{3.23}$$

And $P(y|\mathbf{x},\theta)$ is easily calculated using the normalization term $Z(y|\mathbf{x},\theta)$ defined in Eq.(3.15):

$$P(y|\mathbf{x},\theta) = \frac{\sum_{\mathbf{h}} \exp\left(\theta^{\intercal} \cdot \Phi(\mathbf{x},\mathbf{h},y)\right)}{\sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} \exp\left(\theta^{\intercal} \cdot \Phi(\mathbf{x},\mathbf{h},y')\right)} = \frac{Z(y|\mathbf{x},\theta)}{\sum_{y' \in \mathcal{Y}} Z(y'|\mathbf{x},\theta)}. \tag{3.24}$$

Thus the gradients (3.18), (3.19), (3.20) and (3.21) can be expressed in terms of the three components defined in Eq.(3.15), (3.16) and (3.17): normalization term $Z\left(y|\mathbf{x},\theta\right)$, marginal probability over one part label $P\left(h_j = a \mid y,\mathbf{x},\theta\right)$ and marginal probability over two part labels $P\left(h_j = a, h_k = b \mid y,\mathbf{x},\theta\right)$. All of these components can be calculated using belief propagation in a time that grows only linearly with the number of part labels. Appendix A contains the pseudo code of this parameter estimation process.

3.5 Inference with Belief Propagation

In this section we describe how to use belief propagation [26] to calculate marginal probabilities and the normalization term. Belief Propagation (BP) is an algorithm to efficiently solve *inference* and *decoding* problems in graphical modelling. In our context, the inference problem is to infer the probability that a part label is assigned to a patch given features and action label. This probability is called marginal probability. The decoding problem is to find the most likely assignment of part labels to patches. In this section we will focus on the inference problem, because the decoding problem simply substitutes the summation operation in inference with maximization. We refer the patches as "nodes" and the set of possible part labels as "states" of the nodes.

Before stating how BP algorithm solves the inference problem, it is necessary to explain some basic ideas of graphical modelling. As an undirected graph model, HCRF is a *Markov network*, in which a node is conditionally independent of all other nodes given its neighbours. Intuitively, this means that the neighbours of a node contain all of the information necessary to predict its state. We can factorize the joint probability $P\left(\mathbf{h} \mid y,\mathbf{x},\theta\right)$ according to cliques of the graph $G = (V, E)$. If G is a tree, the factors contain either one node or two nodes. This factorization allows us to represent $P\left(\mathbf{h} \mid y,\mathbf{x},\theta\right)$ more efficiently, because each clique may be much smaller than the full set V.

To compute the marginal probability of a node h_j, the BP algorithm assumes that each of the neighbouring factors of h_j makes a multiplicative contribution to the marginal probability of h_j, called a *message*. Each of these messages can be computed separately when the graph is a tree. This means that we can split up the summation required for the marginal probability calculation into a product of independent sub-problems. The calculation of messages can also be divided into independent sub-problems recursively. In this way, the BP algorithm avoids redundant summation computation. If G is a tree, belief propagation computes the marginal probabilities exactly. If G is not a tree, the message might be passed in loops, thus is not guaranteed to converge.

Mathematically, marginal probabilities are defined in terms of sums over all the possible states of all other nodes in the system. In belief propagation, they are often refereed as "beliefs". In our case, $P\left(h_j = a \mid y,\mathbf{x},\theta\right)$ is the belief at node j, and $P\left(h_j = a, h_k = b \mid y,\mathbf{x},\theta\right)$ is the two-node belief at edge (j,k).

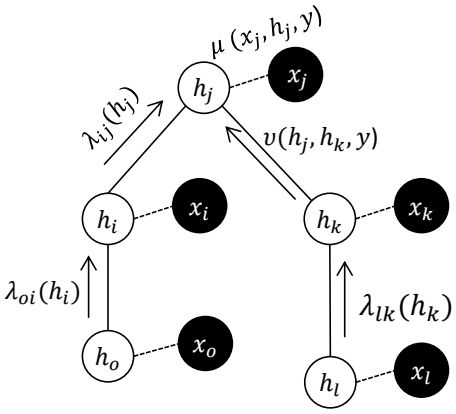


Figure 3.6: Illustration of the transformed graphic model. Empty circles represent the part labels, and the filled-in circles represent the features at each patch.

From the definition of Eq.(3.16) and (3.17), the beliefs at node j and edge (j,k) are:

$$P\left(h_j = a \mid y, \mathbf{x}, \theta\right) = \frac{\sum_{\mathbf{h}:h_j=a} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y)\right)}{\sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y')\right)}, \tag{3.25}$$

$$P\left(h_j = a, h_k = b \mid y, \mathbf{x}, \theta\right) = \frac{\sum_{\mathbf{h}:h_j=a,h_k=b} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y)\right)}{\sum_{y' \in \mathcal{Y}} \sum_{\mathbf{h}} \exp\left(\theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y')\right)}. \tag{3.26}$$

In these two equations, direct summation over part labels is computationally expensive. We could split the summations into a product of independent sub-problems. The BP algorithm considers each neighbour of the goal node (h_j in the case of $P\left(h_j = a \mid y, \mathbf{x}, \theta\right)$) contributes to the marginal probability of the goal node. Each contribution is called a *message*. We can compute these messages separately when the graph is a tree.

This assumption is based on the property of Markov network: the state of a node depends only on the states of its neighbours. To fit HCRF into the form of a Markov network, we need to rewrite the potential function in Eq. (3.1) into the form with only a unary potential (or node potential) and a pairwise potential (or edge potential):

$$\theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y) = \sum_{j \in V} \sum_{p} \theta_p^1 \cdot f_p^1\left(y, x_j, h_j\right) + \sum_{(j,k) \in E} \sum_{q} \theta_q^2 \cdot f_q^2\left(y, h_j, h_k\right), \tag{3.27}$$

where f_p^1 is the feature function depending on a single part label, and f_q^2 is the feature function depending on a pair of part labels. θ_p^1 and θ_q^2 are the model parameters corresponding to the feature functions. The summation over p and q are the number of feature functions.

The intuition behind this is to transform the graphical model in Figure 3.1 to the model in Figure 3.6. In this model, part labels $\{h_1, \ldots, h_m\}$ form a graph $G = (V, E)$. The part labels $h_j\,(1, \ldots, m)$ are the nodes V, and the constrains between two nodes h_j and h_k are edges E. Each node h_j corresponds to its patch features x_j. When this graph structure is a tree, there exists an exact inference.

The difference between two models is that this model does not consider the class label y and the root potential $\eta^\intercal \cdot \omega\left(y, x_0\right)$ explicitly. It only takes into account the connection between

nodes, and the connection between a node and its feature. In Eq.(3.25) and Eq.(3.26), class label y is fixed, so the top layer of model 3.1 could be safely removed. There are two ways to deal with the root potential. We will discuss them in the following two subsections.

3.5.1 Fit Root Potential

The first way to transform the model is to treat root potentials as part of the connection between the node and its feature. We can rewrite Eq.(3.1) in this form:

$$\theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y) = \sum_{j \in V} \left[\alpha^{\intercal} \cdot \phi\left(x_j, h_j\right) + \beta^{\intercal} \cdot \varphi\left(y, h_j\right) + \frac{1}{m} \eta^{\intercal} \cdot \omega\left(y, x_0\right) \right] + \sum_{(j,k) \in E} \gamma^{\intercal} \cdot \psi\left(y, h_j, h_k\right), \tag{3.28}$$

where m is the number of nodes. Because y is considered as a constant and the root potential $\eta^{\intercal} \cdot \omega\left(y, \mathbf{x}\right)$ is not relevant to the state of nodes, the root potential can also be considered as a constant. So $\eta^{\intercal} \cdot \omega\left(y, \mathbf{x}\right) = \sum_{j \in V} \frac{1}{m} \eta^{\intercal} \cdot \omega\left(y, x_0\right)$. The intuition of this approach is that the influence of global image features is evenly distributed among all nodes.

Based on this model, we could factorize the joint probability $P\left(\mathbf{h} \mid y, \mathbf{x}, \theta\right)$ by defining two joint compatibility functions as the factors:

$$\forall j \in V, \qquad \mu\left(x_j, h_j, y\right) = \exp\left(\alpha^{\intercal} \cdot \phi\left(x_j, h_j\right) + \beta^{\intercal} \cdot \varphi\left(y, h_j\right) + \frac{1}{m} \eta^{\intercal} \cdot \omega\left(y, x_0\right) \right), \tag{3.29}$$

$$\forall \left(j, k\right) \in E, \quad \upsilon\left(h_j, h_k, y\right) = \exp\left(\gamma^{\intercal} \cdot \psi\left(y, h_j, h_k\right)\right), \tag{3.30}$$

where $\mu\left(x_j, h_j, y\right)$ is the "evidence" for h_j, and $\upsilon\left(h_j, h_k, y\right)$ is the compatibility of h_j and h_k. Using these two terms we can rewrite Eq.(3.8) as

$$P\left(\mathbf{h} \mid y, \mathbf{x}, \theta\right) = \frac{1}{Z\left(y \mid \mathbf{x}, \theta\right)} \prod_{j \in V} \mu\left(x_j, h_j, y\right) \prod_{(j,k) \in E} \upsilon\left(h_j, h_k, y\right), \tag{3.31}$$

where $\frac{1}{Z\left(y \mid \mathbf{x}, \theta\right)}$ is the normalization term defined in Eq.(3.15). From the definition of marginal probabilities in Eq.(3.25), the belief can be written as:

$$P\left(h_j = a \mid y, \mathbf{x}, \theta\right) = \frac{1}{Z\left(y \mid \mathbf{x}, \theta\right)} \sum_{\mathbf{h}/h_j} \prod_{i \in V} \mu\left(x_i, h_i, y\right) \prod_{(i,k) \in E} \upsilon\left(h_i, h_k, y\right), \tag{3.32}$$

where $\mathbf{h}/h_j$ means all part labels $\mathbf{h}$ except h_j.

BP algorithm defines a message $\lambda_{ij}\left(h_j\right)$, which can be understood as a message from node i to node j about how likely node j is in the state h_j (see Figure 3.6). Messages are calculated in a bottom-up approach. It starts from the leaves, and flows into the goal node $\left(h_j\right)$. A message $\lambda_{ij}\left(h_j\right)$ takes all upstream messages to node i, local evidence node i and the compatibility of node i and j (see Figure 3.6). The message update rule is:

$$\lambda_{ij}\left(h_j\right) \leftarrow \sum_{h_i \in \mathcal{H}} \mu\left(x_i, h_i, y\right) \upsilon\left(h_i, h_j, y\right) \prod_{o \in N(i) \setminus j} \lambda_{oi}\left(h_i\right). \tag{3.33}$$

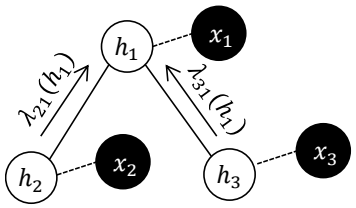


Figure 3.7: An example of marginal probability calculation.

We can see that message $\lambda_{ij}(h_j)$ takes all messages going to node i except for the one from node j. The summation on the right hand sums over all possible states of h_i.

The belief at node j is proportional to the product of the local evidence at that node ($\mu(x_j, h_j, y)$), and all messages coming into node j:

$$P(h_j = a \mid y, \mathbf{x}, \theta) = \frac{1}{Z(y \mid \mathbf{x}, \theta)} \mu(x_j, h_j, y) \prod_{i \in N(j)} \lambda_{ij}(h_j), \tag{3.34}$$

where $N(j)$ donates the neighbouring nodes of j. We do not provide the proof of the message update rule and belief calculation here, but we will give a small example to show its correctness. Consider the small network in Figure 3.7. According to the belief propagation rule, the belief at node 1 is

$$P(h_1 \mid y, \mathbf{x}, \theta) = \frac{1}{Z(y \mid \mathbf{x}, \theta)} \mu(x_1, h_1, y) \lambda_{21}(h_1) \lambda_{31}(h_1). \tag{3.35}$$

Using the message update rule for $\lambda_{21}(h_1)$ and $\lambda_{31}(h_1)$, we can get

$$P(h_1 \mid y, \mathbf{x}, \theta) = \frac{1}{Z(y \mid \mathbf{x}, \theta)} \mu(x_1, h_1, y) \sum_{h_2 \in \mathcal{H}} \mu(x_2, h_2, y) \upsilon(h_2, h_1, y) \sum_{h_3 \in \mathcal{H}} \mu(x_3, h_3, y) \upsilon(h_3, h_1, y). \tag{3.36}$$

This is equal to the belief written in the form of Eq.(3.32):

$$P(h_1 \mid y, \mathbf{x}, \theta) = \frac{1}{Z(y \mid \mathbf{x}, \theta)} \sum_{h_2 \in \mathcal{H}} \sum_{h_3 \in \mathcal{H}} \mu(x_1, h_1, y) \mu(x_2, h_2, y) \mu(x_3, h_3, y) \upsilon(h_2, h_1, y) \upsilon(h_3, h_1, y). \tag{3.37}$$

Analogously, the pairwise marginal probability is proportional to the product of local evidence on nodes ($\mu(x_j, h_j, y)$ and $\mu(x_k, h_k, y)$), the compatibility of two nodes ($\upsilon(h_j, h_k, y)$) and all messages coming to node j and k;

$$P(h_j = a, h_k = b \mid y, \mathbf{x}_t, \theta) = \frac{1}{Z(y \mid \mathbf{x}, \theta)} \mu(x_j, h_j, y) \mu(x_k, h_k, y) \upsilon(h_j, h_k, y) \prod_{i \in N(j) \backslash k} \lambda_{ij}(h_j) \prod_{l \in N(k) \backslash j} \lambda_{lk}(h_k). \tag{3.38}$$

Finally, we can compute the normalization term:

$$Z(y \mid \mathbf{x}, \theta) = \sum_{h_j \in \mathcal{H}} \sum_{\mathbf{h} \backslash h_j} \exp(\theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y)) = \sum_{h_j \in \mathcal{H}} \mu(x_j, h_j, y) \prod_{i \in N(j)} \lambda_{ij}(h_j). \tag{3.39}$$

3.5.2 Normal HCRF

Mathematically, Eq.(3.28) is equal to Eq.(3.1), but it is not a graceful way to interpret intuitively. In this section we will explain a more natural way to transform the model in Figure 3.1 to the model in Figure 3.6. The idea behind this method is to only use the part of the potential function related to the part labels $\mathbf{h}$.

Let us define a potential function without root potential:

$$\begin{aligned}\Delta\left(y,\mathbf{h},\mathbf{x};\theta\right) &= \theta^{\intercal}\cdot\Phi(\mathbf{x},\mathbf{h},y)-\eta^{\intercal}\cdot\omega\left(y,\mathbf{x}\right)\\ &= \sum_{j\in V}\alpha^{\intercal}\cdot\phi\left(x_j,h_j\right)+\sum_{j\in V}\beta^{\intercal}\cdot\varphi\left(y,h_j\right)+\sum_{(j,k)\in E}\gamma^{\intercal}\cdot\psi\left(y,h_j,h_k\right).\end{aligned} \tag{3.40}$$

This definition has the same form as a normal HCRF defined in [10]. The discarded part $\eta^{\intercal}\cdot\omega\left(y,\mathbf{x}\right)$ is not related to part labels $\mathbf{h}$, i.e. it does not change when $\mathbf{h}$ changes. It follows that

$$P\left(\mathbf{h}\mid y,\mathbf{x},\theta\right)=\frac{\exp\left(\Delta\left(y,\mathbf{h},\mathbf{x};\theta\right)+\eta^{\intercal}\cdot\omega\left(y,\mathbf{x}\right)\right)}{\sum_{\mathbf{h}}\exp\left(\Delta\left(y,\mathbf{h},\mathbf{x};\theta\right)+\eta^{\intercal}\cdot\omega\left(y,\mathbf{x}\right)\right)}=\frac{\exp\left(\Delta\left(y,\mathbf{h},\mathbf{x};\theta\right)\right)}{\sum_{\mathbf{h}}\exp\left(\Delta\left(y,\mathbf{h},\mathbf{x};\theta\right)\right)}. \tag{3.41}$$

Similar as the previous section, we can rewrite Eq.(3.40) into the form of Eq.(3.27):

$$\Delta\left(y,\mathbf{h},\mathbf{x};\theta\right)=\sum_{j\in V}\left[\alpha^{\intercal}\cdot\phi\left(x_j,h_j\right)+\beta^{\intercal}\cdot\varphi\left(y,h_j\right)\right]+\sum_{(j,k)\in E}\gamma^{\intercal}\cdot\psi\left(y,h_j,h_k\right). \tag{3.42}$$

The meaning of this is easier to understand than Eq.(3.28). The first part of this function is the connection between part label and its features. It represents the dash line between empty circles and filled-in circles in Figure 3.6. The second part is the connection between two part labels. It represents the solid line between empty circles in Figure 3.6. Based on this form we could define the joint compatibility functions:

$$\forall j\in V, \qquad \mu^{'}\left(x_j,h_j,y\right)=\exp\left(\alpha^{\intercal}\cdot\phi\left(x_j,h_j\right)+\beta^{\intercal}\cdot\varphi\left(y,h_j\right)\right), \tag{3.43}$$

$$\forall\left(j,k\right)\in E, \qquad \upsilon^{'}\left(h_j,h_k,y\right)=\exp\left(\gamma^{\intercal}\cdot\psi\left(y,h_j,h_k\right)\right). \tag{3.44}$$

Denote the normalization term as:

$$Z^{'}\left(y\mid\mathbf{x},\theta\right)=\sum_{\mathbf{h}}\exp\left(\Delta\left(y,\mathbf{h},\mathbf{x};\theta\right)\right). \tag{3.45}$$

Using these two compatibility functions and the normalization term defined above, we could rewrite conditional probability $P\left(\mathbf{h}\mid y,\mathbf{x},\theta\right)$ as:

$$P\left(\mathbf{h}\mid y,\mathbf{x},\theta\right)=\frac{1}{Z^{'}\left(y\mid\mathbf{x},\theta\right)}\prod_{j\in V}\mu^{'}\left(x_j,h_j,y\right)\prod_{(j,k)\in E}\upsilon^{'}\left(h_j,h_k,y\right). \tag{3.46}$$

According to the same deduction process as in the previous section, the message update rule is:

$$\lambda_{ij}^{'}\left(h_j\right)\leftarrow\sum_{h_i\in\mathcal{H}}\mu^{'}\left(x_i,h_i,y\right)\upsilon^{'}\left(h_i,h_j,y\right)\prod_{o\in N(i)\backslash j}\lambda_{oi}^{'}\left(h_i\right). \tag{3.47}$$

The marginal probabilities are:

$$P\left(h_j = a \mid y, \mathbf{x}_t, \theta\right) = \frac{1}{Z'\left(y \mid \mathbf{x}, \theta\right)} \mu'\left(x_j, h_j, y\right) \prod_{i \in N(j)} \lambda'_{ij}\left(h_j\right), \tag{3.48}$$

$$P\left(h_j = a, h_k = b \mid y, \mathbf{x}_t, \theta\right) =$$
$$\frac{1}{Z'\left(y \mid \mathbf{x}, \theta\right)} \mu'\left(x_j, h_j, y\right) \mu'\left(x_k, h_k, y\right) \upsilon'\left(h_j, h_k, y\right) \prod_{i \in N(j) \setminus k} \lambda'_{ij}\left(h_j\right) \prod_{l \in N(k) \setminus j} \lambda'_{lk}\left(h_k\right). \tag{3.49}$$

The normalization term is computed in this way:

$$Z'\left(y \mid \mathbf{x}, \theta\right) = \sum_{h_j \in \mathcal{H}} \sum_{\mathbf{h} \setminus h_j} \exp\left(\Delta\left(y, \mathbf{h}, \mathbf{x}; \theta\right)\right) = \sum_{h_j \in \mathcal{H}} \mu'\left(x_j, h_j, y\right) \prod_{i \in N(j)} \lambda'_{ij}\left(h_j\right). \tag{3.50}$$

From the definition of $Z'\left(y \mid \mathbf{x}, \theta\right)$, we can easily get its relationship with the goal normalization term $Z\left(y \mid \mathbf{x}, \theta\right)$:

$$Z\left(y \mid \mathbf{x}, \theta\right) = Z'\left(y \mid \mathbf{x}, \theta\right) \cdot \exp\left(\eta^{\intercal} \cdot \omega\left(y, \mathbf{x}\right)\right) \tag{3.51}$$

The benefit of this method is that it calculates the marginal probabilities in exactly the same way as a normal HCRF model does. The only modification is that the normalization term $Z\left(y \mid \mathbf{x}, \theta\right)$ needs to be multiplied by the normalization term $Z'\left(y \mid \mathbf{x}, \theta\right)$, which is obtained from the normal HCRF with the exponential of the root potential in the end.

These two methods are mathematically identical. They are just two different ways to interpret and implement the HCRF model. These two methods have the same output when implemented.

3.6 Root Model

In this section we will describe the root model, a baseline model of HCRF. This model only uses the global feature and does not include the hidden part labels. It is trained with the maximum likelihood criteria and an iterative gradient ascent method similar to HCRF.

The root model is important for HCRF for three reasons. First, it can be used to initialize the root filter η of HCRF. The objective function of HCRF is not convex, thus the initialization of model parameters becomes important. A good starting point of the model parameters could lead to a good local optimum. The trained model parameter of the root model, called root filter, can be a good estimation of the root filter η in the HCRF model. Second, the root filter can be used to find the local patches during training and testing stage. The root model gives higher weights to the pixels important for an action and lower weights to the other pixels. We can find the salient patches of an image by applying the root filter on the image. Thirdly, the root model is a simplified version of HCRF. It can be used as to evaluate the contribution of the root potential to the HCRF model.

The root filter is trained without using hidden part labels. That is, we learn the root filter η by only considering the global feature vector x_0. Figure 3.8 is an illustration of the root model. This simplified model reduces the number of parameters needed to be considered initially.

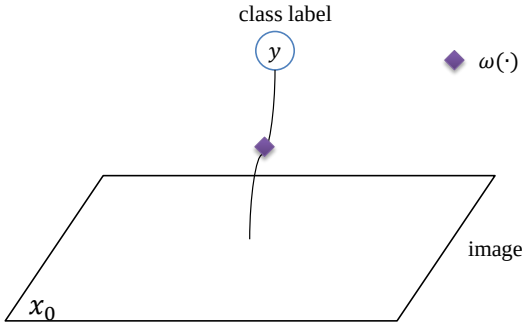


Figure 3.8: Illustration of the root filter model.

We only use the last part of the potential function in Eq.(3.1) for modelling. The probability of a class label y conditional on the global feature x_0 and root filter parameter η is:

$$P_{root}(y|x_0;\eta) = \frac{\exp\left(\eta^{\intercal} \cdot \omega\left(y, x_0\right)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(\eta^{\intercal} \cdot \omega\left(y', x_0\right)\right)}. \tag{3.52}$$

Given a set of training samples $(x_{t,0}, y_t)$, the objective function for the root filter can be formulated as the summation of log-likelihood for each sample:

$$L_{root}\left(\eta\right) = \sum_t L^t_{root}\left(\eta\right) - \frac{1}{2\sigma^2_{root}}||\eta||^2 = \sum_t \log P_{root}\left(y_t|x_{t,0};\eta\right) - \frac{1}{2\sigma^2_{root}}||\eta||^2. \tag{3.53}$$

The second term is the L2 regularization term to prevent the norm of η becoming too big. That is, we assume η follows a normal distribution $P\left(\eta\right) \sim N(0, \sigma^2_{root})$. Note that this objective function is concave, so we can find the global optimal solution.

The optimization problem is to find the optimal η^* that gives the maximum of the objective function:

$$\eta^* = \arg\max_{\eta} L_{root}\left(\eta\right). \tag{3.54}$$

We could use gradient ascend or limited-memory BFGS to search for the optimal η^*. Similar to Eq.(3.21), the gradient of the log-likelihood with respect to the t-th training image can be calculated as:

$$\frac{\partial L^t_{root}\left(\eta\right)}{\partial \eta} = \omega\left(y_t, x_{t,0}\right) - \sum_{y' \in \mathcal{Y}} P_{root}\left(y' \mid x_{t,0};\eta\right) \omega\left(y', x_{t,0}\right). \tag{3.55}$$

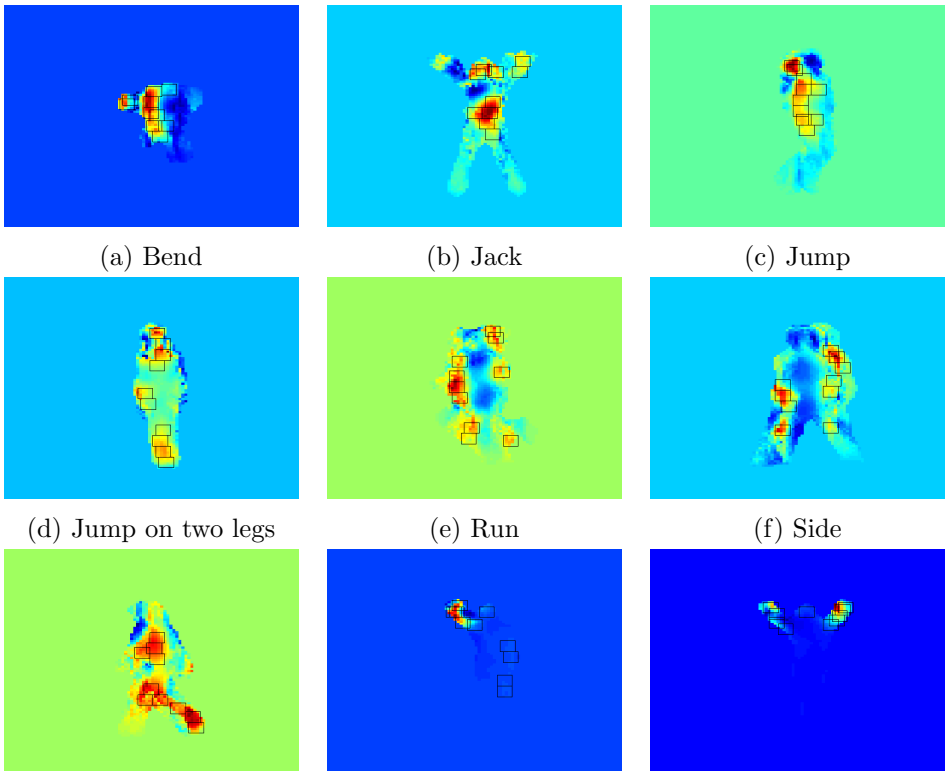
During the training process, we do not need to solve the inference problem, because the root model does not have the hidden part labels. For this reason, the training process of the root model is much more efficient than HCRF.

The root filter η^* is used as a starting point for η in the potential function Eq.(3.1). The other parameters α, β and γ are initialized randomly. Other than initialization, the root filter can also be used to find salient patches. We will describe it in the next subsection.

3.6.1 Patch Initialization

In this subsection we use the learned root filter to select salient patches. This method is similar to a heuristic proposed by Felzenszwalb et al. [12]. It only applies to pixel-wide global motion features such as optical flow.

For each image with global feature x_0 and class label y, we apply the root filter η_y on x_0: $\eta_y^\intercal \cdot x_0$. Then select a rectangle region of certain size in the image that has the most positive energy. We set the weights in this region to zero and repeat the selection process until the defined number of patches are selected. The patches with the highest positive energy are areas that are discriminative for this class. Figure 3.9 is a visualization of root filters applied on nine actions in the Weizmann dataset [5].



(a) Bend (b) Jack (c) Jump

(d) Jump on two legs (e) Run (f) Side

(g) Walk (h) Wave with one hand (i) Wave with two hands

Figure 3.9: Visualization of root filters applied on Weizmann dataset and their initialized patches. The bright area represents positive energy. The black squares are the patches found.

The tree structure among the patches could be defined manually. If a predefined tree is not available, it could also be found using the minimal spanning tree algorithm. We could use the distance between two patches as the weight of the edge connecting these two patches. Then we could use these weights to run a minimal spanning tree algorithm among these patches. In this way we obtain a tree structure in which the patches are connected to their closest neighbours.

3.6.2 Testing Patch Initialization

During testing, we can not use the patch initialization method described above because the class label of a test image is unknown, so we do not know which root filter to use. For a test image with global motion feature vector x_0, we could apply root filters of all classes on x_0 to obtain a set of $|\mathcal{Y}|$ feature vectors $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(|\mathcal{Y}|)}\}$, where each feature vector $\mathbf{x}^{(k)}$ is obtained by finding patches using the root filter η_k. We can calculate the probabilities of all possible initializations of patches using the learned model parameter and classify them by the class label that gives the maximum probabilities. In other words, the final class label y^* of x_0 is obtained as:

$$y^* = \arg\max_y \left[\max\{P(y|\mathbf{x}^{(1)};\theta), P(y|\mathbf{x}^{(2)};\theta), \ldots, P(y|\mathbf{x}^{(|\mathcal{Y}|)};\theta)\} \right]. \tag{3.56}$$

In this chapter we have described the HCRF model, including its graphical model, parameter estimation and inference. HCRF models an image as a constellation of part labels and it incorporates these part labels as hidden variables. It learns its model parameters using iterative gradient-based method. And the inference problem of computing marginal probabilities is solved by belief propagation. We have also described the root model, a baseline model of HCRF. The root model can be used to initialize the model parameter of HCRF, to find the salient patches and to evaluate the contribution of the root potential to HCRF.

Chapter 4

Max-Margin Hidden Conditional Random Fields

In Chapter 3, HCRF learns the model parameter in a maximum likelihood approach. That is, the model parameter of a HCRF model is learned by maximizing the conditional likelihood of the training data. During testing, a new image is classified as the action class that gives the highest conditional likelihood.

Wang and Mori [14] have proposed a different method to train the model parameter. Instead of using maximum likelihood, this method uses a max-margin criteria for training. A max-margin criteria [12, 16] aims to set the model parameter in a way that the margin between the correct label and the other labels is maximized. That is, the score of the correct label is higher than the scores of incorrect ones as much as possible. Using the max-margin approach to train the HCRF model parameter, this method is called *Max-Margin Hidden Conditional Random Fields* (MMHCRF). MMHCRF is very similar to the Latent SVM (LSVM) proposed by Felzenszwalb et al. [12]. The difference is that MMHCRF handles multi-class classification directly, whereas LSVM only handles binary classification. In this chapter we will explain the theory of MMHCRF in detail.

4.1 Model Formulation

MMHCRF uses the potential function and its parametrization in Eq.(3.1) as HCRF does. For a training image, assume its feature vector and action label pair $(\mathbf{x}, y)$ is scored by the potential function with the best assignment of hidden variables:

$$f_\theta(\mathbf{x}, y) = \max_{\mathbf{h}} \theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y), \tag{4.1}$$

where θ is a vector of the model parameters, $\mathbf{h}$ is a vector of hidden part labels, and $\theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y)$ is the potential defined by Eq.(3.1).

Assume the set of training samples are $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$, where T is the number of images. We want to find θ that can maximize the margin between the score of the correct

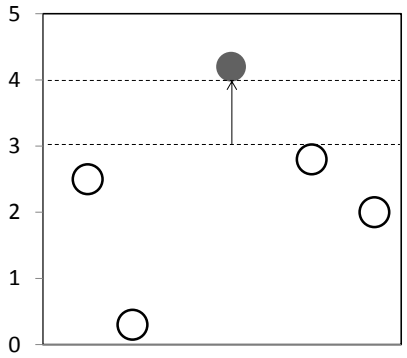


Figure 4.1: Illustration of the margin bound. The circles in the figure denote different labels. The correct labels is the filled circle and the rest ones are the empty circles. The height of each label designates its score. In this plot, the margins between the correct label and every wrong ones are larger than one.

label and the score of other labels. Similar to multi-class SVM [16], this training process can be formulated as an optimization problem:

$$\begin{aligned} &\min_{\theta,\xi} \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t \\ \text{s.t. } &f_\theta(\mathbf{x}_t, y) - f_\theta(\mathbf{x}_t, y_t) \leqslant \xi_t - 1, \forall t, \forall y \neq y_t \\ &\xi_t \geq 0, \forall t. \end{aligned} \tag{4.2}$$

In this optimization, we want to find θ whose L2-norm is as small as possible, and satisfies the constraints that the score for the correct label y_t is at least one larger than the scores of the other labels for each training sample. This margin bound is illustrated in Figure 4.1. ξ_t is the slack variable for the t-th training image to handle the soft margin when data is not fully linearly separable, and C controls the trade-off between the slack variable penalty and the size of the margin. Put Eq.(4.1) into Eq.(4.2) to get:

$$\begin{aligned} &\min_{\theta,\xi} \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t \\ \text{s.t. } &\max_{\mathbf{h}} \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y) - \max_{\mathbf{h}'} \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}', y_t) \leqslant \xi_t - \delta(y, y_t), \forall t, \forall y, \\ &\text{where } \delta(y, y_t) = \begin{cases} 1 \text{ if } y \neq y_t \\ 0 \text{ if otherwise} \end{cases}. \end{aligned} \tag{4.3}$$

Note that the constrains of this optimization problem is not convex. Therefore this method is not guaranteed to reach the global optimum. We will explain how to find a local optimum of the problem in Eq.(4.3).

$f_\theta(\mathbf{x}, y) = \max_{\mathbf{h}} \theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y)$ is convex in θ, because it is a maximum of a set of functions, each of which is linear (thus convex) in θ. If we fix $\mathbf{h}'$ in Eq.(4.3), that is, restrain the possible choice of $\mathbf{h}'$ to 1, $f_\theta(\mathbf{x}_t, y_t) = \max_{\mathbf{h}'} \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}', y_t)$ becomes a linear function, thus the constraints of Eq.(4.3) becomes convex. Using a coordinate descent algorithm similar to [12], a local optimum of Eq.(4.3) can be computed by iterating through these two steps:

1. Holding θ, ξ fixed, optimize the hidden part labels $\mathbf{h}^{'}$ for the training example $(\mathbf{x}_t, y_t)$:

$$\mathbf{h}_{t,y_t} = \arg\max_{\mathbf{h}^{'}} \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}^{'}, y_t). \tag{4.4}$$

2. Holding h_{t,y_t} fixed, optimize θ, ξ by solving this optimization problem:

$$\begin{aligned} &\min_{\theta,\xi} \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t \\ &\text{s.t.} \quad \max_{\mathbf{h}} \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y) - \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y_t}, y_t) \leqslant \xi_t - \delta(y, y_t), \forall t, \forall y. \end{aligned} \tag{4.5}$$

These two steps are performed repeatedly in iterations. Each step always improves or maintains the objective function [12]. And we can find the local optimum of Eq.(4.3) in iterations.

The first step of this algorithm can be solved using the decoding process of the BP algorithm explained in Section 3.5. The second step of this algorithm involves solving a quadratic program in Eq.(4.5). We notice that the constraints of this optimization problem require a maximization operation. But the standard quadratic program solver only accepts linear constraints. Therefore we wish to further decompose the optimization problem in Eq. (4.5) into a standard quadratic optimization problem with linear constraints. This decomposition can be done by replacing the maximum function with all possible assignments of hidden part labels:

$$\begin{aligned} &\min_{\theta,\xi} \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t \\ &\text{s.t.} \quad \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y) - \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y_t}, y_t) \leqslant \xi_t - \delta(y, y_t), \forall t, \forall \mathbf{h}, \forall y. \end{aligned} \tag{4.6}$$

But this decomposition has introduced another problem. The optimization problem in Eq.(4.6) has an exponential number of constraints, because there are an exponential number of possible $\mathbf{h}$ for every pair of image and possible label $(\mathbf{x}_t, y)$. To reduce the number of constraints, we consider only the "most violated" constrains. Define $\mathbf{h}_{t,y} = \arg\max_{\mathbf{h}} \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y_t)$, within a local neighbourhood of θ, Eq.(4.6) is equivalent to:

$$\begin{aligned} &\min_{\theta,\xi} \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t \\ &\text{s.t.} \quad \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y}, y) - \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y_t}, y_t) \leqslant \xi_t - \delta(y, y_t), \forall t, \forall y. \end{aligned} \tag{4.7}$$

Thus the learning algorithm is changed into the following two steps:

1. Fixing θ, ξ, optimize the hidden part labels $\mathbf{h}$ for each pair $(\mathbf{x}_t, y)$ of an example $\mathbf{x}_t$ and a possible labelling y:

$$\mathbf{h}_{t,y} = \arg\max_{\mathbf{h}} \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y). \tag{4.8}$$

2. Fixing $\mathbf{h}_{t,y}, \forall t, \forall y$, optimize θ, ξ by solving this optimization problem:

$$\begin{aligned} &\min_{\theta,\xi} \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t \\ &\text{s.t.} \quad \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y}, y) - \theta^{\intercal} \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y_t}, y_t) \leqslant \xi_t - \delta(y, y_t), \forall t, \forall y. \end{aligned} \tag{4.9}$$

Eq.(4.4) is an optimization problem to find the optimal $\mathbf{h}^*$ that $\mathbf{h}^* = \arg\max_{\mathbf{h}} \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}, y)$. Same as Eq.(4.4), this problem can also be solved using the decoding process of the BP algorithm. The difference between Step 1 of the above algorithm and Step 1 of Eq.(4.4) is that the decoding process of Eq.(4.8) needs to be performed for each example $\mathbf{x}_t$ and every possible class label y pair $(\mathbf{x}_t, y)$, while Eq.(4.4) only needs to be performed on each example $\mathbf{x}_t$ and its correct class label y_t pair $(\mathbf{x}_t, y_t)$.

For the optimization problem in step 2, we transform it into its dual form and solve it by dual optimization. We will explain how to do this in the next section.

4.2 Dual Optimization

We do not directly solve the optimization problem of Eq.(4.9). Instead, we first transform it into its dual form, and divide the dual problem into a set of smaller quadratic problems.

The quadratic optimization problem Eq.(4.9) is very similar to the primal problem of a multi-class SVM [16]. The only difference is that Eq.(4.9) has part labels $\mathbf{h}$. Analogously to the standard multi-class SVM, the primal problem of Eq.(4.9) can be transformed into its dual form.

To simplify the notation we define $\Psi(\mathbf{x}_t, y) = \Phi(\mathbf{x}_t, \mathbf{h}_{t,y}, y) - \Phi(\mathbf{x}_t, \mathbf{h}_{t,y_t}, y_t)$. We define a dual set of non-negative variables $\alpha = (\alpha_{1,1}, \ldots, \alpha_{T,\mathcal{Y}})$, one for each constraint in Eq.(4.9). Then the dual form of Eq.(4.9) is:

$$\begin{aligned} \max_{\alpha} \sum_{t=1}^{T} \sum_{y} \alpha_{t,y} \delta(y, y_t) - \frac{1}{2} || \sum_{t=1}^{T} \sum_{y} \alpha_{t,y} \Psi(\mathbf{x}_t, y) ||^2 \\ \text{s.t.} \quad \sum_{y} \alpha_{t,y} = C, \forall t \\ \alpha_{t,y} \geqslant 0, \forall t, \forall y. \end{aligned} \tag{4.10}$$

The proof of this transformation is attached in Appendix B. θ can be recovered from the dual variables α:

$$\theta = -\sum_{t=1}^{T} \sum_{y} \alpha_{t,y} \Psi(\mathbf{x}_t, y). \tag{4.11}$$

The optimization problem in Eq.(4.10) can also be written in the kernel form. We can define a $T \times |\mathcal{Y}|$ by $T \times |\mathcal{Y}|$ kernel function:

$$K(t, y; s, y') = \Psi(\mathbf{x}_t, y)^\intercal \Psi(\mathbf{x}_s, y'). \tag{4.12}$$

If we define δ as a vector of $\{\delta(y, y_t) : \forall t, \forall y\}$, then we can rewrite Eq.(4.10) in the kernel form:

$$\begin{aligned} \max_{\alpha} \alpha^\intercal \delta - \frac{1}{2} \alpha^\intercal K \alpha \\ \text{s.t.} \quad \sum_{y} \alpha_{t,y} = C, \forall t \\ \alpha_{t,y} \geqslant 0, \forall t, \forall y. \end{aligned} \tag{4.13}$$

The problem of Eq.(4.10) is that it has $T \cdot |\mathcal{Y}|$ dual variables α, where T is the number of training examples and $|\mathcal{Y}|$ is the number of all class labels. When the number of training samples is large, the scale of the problem becomes very large. We could decompose this optimization problem into small problems using Sequential Minimal Optimization (SMO) [16].

The intuitive idea of this algorithm is to separate the dual variables α into T disjoint sets, and each set of variables only involves one training sample. This algorithm works in iterations. On each iteration, the algorithm picks up an image t and improves the objective function in Eq.(4.10) by updating $|\mathcal{Y}|$ variables $\{\alpha_{t,y} : \forall y \in \mathcal{Y}\}$ and fix all other dual variables.

The objective function in Eq.(4.10) can be can be written only in terms of the set of dual variables only involves one image $\{\alpha_{t,y} : \forall y \in \mathcal{Y}\}$:

$$
\begin{aligned}
&\sum_{t=1}^{T}\sum_{y}\alpha_{t,y}\delta(y,y_t) - \frac{1}{2}||\sum_{t=1}^{T}\sum_{y}\alpha_{t,y}\Psi(\mathbf{x}_t,y)||^2 \\
&= \sum_{y}\alpha_{t,y}\delta(y,y_t) + \sum_{s:s\neq t}\sum_{y}\alpha_{s,y}\delta(y,y_s) - \frac{1}{2}||\sum_{y}\alpha_{t,y}\Psi(\mathbf{x}_t,y) + \sum_{s:s\neq t}\sum_{y}\alpha_{s,y}\Psi(\mathbf{x}_s,y)||^2 \\
&= \sum_{y}\alpha_{t,y}\delta(y,y_t) + \sum_{s:s\neq t}\sum_{y}\alpha_{s,y}\delta(y,y_s) \\
&- \frac{1}{2}\left[\|\sum_{y}\alpha_{t,y}\Psi(\mathbf{x}_t,y)\|^2 + 2\left(\sum_{y}\alpha_{t,y}\Psi(\mathbf{x}_t,y)\right)^{\intercal}\left(\sum_{s:s\neq t}\sum_{y'}\alpha_{s,y'}\Psi(\mathbf{x}_s,y')\right) + \|\sum_{s:s\neq t}\sum_{y}\alpha_{s,y}\delta(y,y_s)\|^2\right] \\
&= \sum_{y}\alpha_{t,y}\delta(y,y_t) - \frac{1}{2}\left[\|\sum_{y}\alpha_{t,y}\Psi(\mathbf{x}_t,y)\|^2 + 2\left(\sum_{y}\alpha_{t,y}\Psi(\mathbf{x}_t,y)\right)^{\intercal}\left(\sum_{s:s\neq t}\sum_{y'}\alpha_{s,y'}\Psi(\mathbf{x}_s,y')\right)\right] \\
&+ \text{other terms not involving} \{\alpha_{t,y} : \forall y \in \mathcal{Y}\}
\end{aligned}
\tag{4.14}
$$

Then (4.10) can be divided into small optimization problems that only have the parameters $\{\alpha_{t,y} : \forall y \in \mathcal{Y}\}$:

$$
\begin{aligned}
\max_{\alpha_{t,y}:\forall y} \sum_{y}\alpha_{t,y}\delta(y,y_t) - \frac{1}{2}\left[\|\sum_{y}\alpha_{t,y}\Psi(\mathbf{x}_t,y)\|^2 + 2\left(\sum_{y}\alpha_{t,y}\Psi(\mathbf{x}_t,y)\right)^{\intercal}\left(\sum_{s:s\neq t}\sum_{y'}\alpha_{s,y'}\Psi(\mathbf{x}_s,y')\right)\right] \\
\text{s.t.} \quad \sum_{y}\alpha_{t,y} = C, \\
\alpha_{t,y} \geqslant 0, \forall y.
\end{aligned}
\tag{4.15}
$$

When we solve this optimization problem, all other constraints considering variables $\alpha_{s,y}$ where $s \neq t$ are not affected. Appendix C contains the pseudo code of the MMHCRF training process.

During testing, when there comes a new image $\mathbf{x}$, we first calculate the optimal $\mathbf{h}$ for every possible class label y: $\mathbf{h}_y = \arg\max_{\mathbf{h}} \theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}, y)$. Then we calculate the score of each class label and pick the label with highest score: $y^* = \arg\max_y \theta^{\intercal} \cdot \Phi(\mathbf{x}, \mathbf{h}_y, y)$.

4.3 Comparison of HCRF and MMHCRF

HCRF and MMHCRF have their similarities. Both methods solve the problem of classification with structured hidden variables. Both of their objective functions are not convex, thus we can only find a local optimal solution for both models. Both of them are iterative methods. During each iteration, both of them require an inference or decoding on each training sample.

Their main difference is on their learning criteria: maximizing the conditional probabilities in HCRF, and maximizing the margin in MMHCRF. As a result, the training processes of HCRF and MMHCRF need to solve two types of problems: summing over $\mathbf{h}$ and maximizing over $\mathbf{h}$. We will explain these two differences below.

4.3.1 Max-Log Likelihood vs. Max-Margin

One difference between HCRF and MMHCRF is their training criteria. HCRF sets the model parameter in the way that it can give the maximum log likelihood, while MMHCRF sets the model parameter in the way that the margin between the score of the correct label and the scores of the other labels is maximized.

We give a simple example to explain the difference between these two criteria. To simplify the example, we assume there is a classification problem without hidden variables. Assume the set of possible class labels $\mathcal{Y} = \{1, 2, 3\}$ and there is only one sample in the training dataset: $(\mathbf{x}, y = 2)$. That is, the correct label of this sample is 2. Suppose we have two sets of model parameters θ_1 and θ_2. The conditional probabilities $P(y|\mathbf{x}; \theta_1)$ and $P(y|\mathbf{x}; \theta_2)$ are shown on Table 4.1.

	$y = 1$	$y = 2$	$y = 3$
$\theta = \theta_1$	0.03	0.49	0.48
$\theta = \theta_2$	0.13	0.45	0.42

Table 4.1: Max-log likelihood vs. max-margin.

If we train the model parameter with the maximizing log likelihood criteria, θ_1 is favoured over θ_2, because $P(y = 2|\mathbf{x}; \theta_1) > P(y = 2|\mathbf{x}; \theta_2)$. Let us define the margin as the difference between the score of the correct label y^* and the highest score of other labels: $m(\theta) = P(y^*|\mathbf{x}; \theta) - \max_{y \neq y^*} P(y|\mathbf{x}; \theta)$. If we train the model parameter with the maximizing margin criteria, θ_2 is favoured over θ_1 because $m(\theta_1) = 0.49 - 0.48 = 0.01$ and $m(\theta_2) = 0.45 - 0.42 = 0.03$. This simple example illustrates the difference between the two criteria of maximizing log likelihood and maximizing margin.

4.3.2 Summation vs. Maximization

Another difference between HCRF and MMHCRF is that they need to solve two types of problems: HCRF does an inference while MMHCRF does an decoding. This leads to two different approaches: summing over $\mathbf{h}$ in HCRF and maximizing over $\mathbf{h}$ in MMHCRF.

Recall that HCRF needs to compute the summation of all possible combinations of $\mathbf{h}$: $P(y|\mathbf{x},\theta) = \sum_{\mathbf{h}} P(y,\mathbf{h}|\mathbf{x},\theta)$. The intuition behind this is that a correct labelling of $\mathbf{h}$ leads to a higher conditional probability $P(y,\mathbf{h}|\mathbf{x},\theta)$ and an incorrect labelling of $\mathbf{h}$ has a lower conditional probability. Therefore the correct $\mathbf{h}$ will contribute more to $P(y|\mathbf{x},\theta)$.

However, the number of all possible assignments of part labels $\mathbf{h}$ is exponential. But only a few of them are the correct ones, i.e., the ones that are descriptive for this image. A well trained model parameter will assign the correct ones with higher probabilities (close to 1) and the incorrect ones with lower probabilities (close to 0). But since there are much more incorrect $\mathbf{h}$ than the correct ones, when they are summed together, the conditional probability $P(y|\mathbf{x},\theta)$ can still be overwhelmed with the incorrect $\mathbf{h}$'s. This problem will become more obvious when the dimension of $\mathbf{h}$ increases.

To avoid this problem, MMHCRF maximizes over all possible $\mathbf{h}$: $f_\theta(\mathbf{x},y) = \max_{\mathbf{h}} \theta^\intercal \cdot \Phi(\mathbf{x},\mathbf{h},y)$. It only considers the correct $\mathbf{h}$ and discards all other incorrect ones. It is equivalent to assign the correct $\mathbf{h}$ with probability of 1 and all other with 0. But this approach is not without problem. In the hidden part model, the correct assignment of $\mathbf{h}$ is not provided in the training data. The maximizing approach will have problem when the learned "correct" label is actually not quite correct. In this case, discarding all other possible assignments is too risky. Therefore, MMHCRF is less robust than HCRF.

In this chapter, we have explained a max-margin approach to learn the model parameter. An max-margin approach sets the model parameter in the way that the score of the correct label is higher than the scores of other labels as much as possible. The constraints of MMHCRF is not convex, so the method is not guaranteed to return a global optimum. We can search for the local optimum by iterating through two steps: finding the best assignment of part labels using the current model parameter, and optimizing the model parameter using the found part labels. The first step can be solved using the decoding process of the BP algorithm, and the second step can be transformed into its dual form and decomposed into to smaller quadratic programming problem.

Chapter 5

Part Labels and Global Features

In the previous chapter we have compared the summation approach adopted by HCRF with the maximization approach adopted by MMHCRF. The summation approach is a robust method, but it suffers from the problem that the large number of incorrect assignments of part labels might overwhelm the correct ones. The maximization approach solves this problem by only using the correct assignment of part labels, but with the price of becoming less robust. In this chapter we propose a new method to combine the advantages of both approaches.

Inspired by the concept of bag-of-words [27] and the root model described in Section 3.6, this model finds the best assignment of part labels for each image using the model parameter trained by HCRF, and uses them as local features for classification. Similar with the hidden part model, this method also combines the local features with the global feature. We call this method the Part Labels method.

5.1 Model Formulation

Given a set of training samples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$, the HCRF model trains the model parameter θ by summing over all possible assignments of part labels. Ideally, the trained θ should give the correct assignments of part labels higher probabilities, and the incorrect ones lower probabilities. In our method, we only use the correct assignment and discard all other incorrect ones to avoid the disturbance they might bring. Our method is different from MMHCRF in the way that MMHCRF adopts the maximization approach to pick the best assignment of part labels in every iteration during the training process, while the Part Labels method only does that once, after the HCRF training process.

The intuition behind the Part Labels method is that, after the HCRF training process, the learned model parameter θ should be able to assign the correct assignment of part labels with the highest probability, and the other ones with lower probabilities. Therefore we could safely use this θ to find the assignment of part labels with the highest probability $\mathbf{h}^* = \arg\max_{\mathbf{h}} \theta^\intercal \cdot \Phi(\mathbf{x}, \mathbf{h}, y)$ as the correct $\mathbf{h}$.

It is easy to obtain the best assignment of part labels $\mathbf{h}^*$ for each training sample. We can use the decoding process of the BP algorithm described in Section 3.5 to solve this problem.

The vector of part labels could be considered as a refined set of local features for the images, much as the "words" for the bag-of-words approach. For example, the part label for the patch on the head describes the movement pattern of the head. It is an abstraction of the patch features. We use these part labels as the local features of this image and combine them with the global feature vector by concatenation: $\mathbf{x}^{'} = (\mathbf{h}^{*}, x_0)$.

Next we train the new feature vector $\mathbf{x}^{'}$ in a similar way with the root model. For a training image $(\mathbf{x}^{'}, y)$, we define its potential function:

$$\eta^{'\intercal} \cdot \omega\left(y, \mathbf{x}^{'}\right) = \sum_{a \in \mathcal{Y}} \eta_a^{'\intercal} \cdot \mathbf{1}_{\{y=a\}} \cdot \mathbf{x}^{'}, \tag{5.1}$$

where $\eta^{'}$ is the model parameter, $\omega(\cdot)$ is the feature function, $\eta_a^{'}$ measures the compatibility between feature $\mathbf{x}^{'}$ and class label $y = a$. $\eta^{'}$ is the concatenation of $\eta_a^{'}$ for all $a \in \mathcal{Y}$. The length of vector $\eta^{'}$ is $|\mathcal{Y}||\mathbf{x}^{'}|$.

Using the potential function defined above, we could define the probability or likelihood of class label y given the feature vector $\mathbf{x}^{'}$:

$$P(y|\mathbf{x}^{'}; \eta^{'}) = \frac{\exp\left(\eta^{'\intercal} \cdot \omega\left(y, \mathbf{x}^{'}\right)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(\eta^{'\intercal} \cdot \omega\left(y', \mathbf{x}^{'}\right)\right)}. \tag{5.2}$$

The objective function for the set of all training samples can be formulated as the summation of log-likelihood of all samples:

$$L\left(\eta^{'}\right) = \sum_{t=1}^{T} L_t\left(\eta^{'}\right) = \sum_{t=1}^{T} \log P\left(y_t|\mathbf{x}_t^{'}; \eta^{'}\right) = \sum_{t=1}^{T} \log \frac{\exp\left(\eta^{'\intercal} \cdot \omega\left(y_t, \mathbf{x}_t^{'}\right)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(\eta^{'\intercal} \cdot \omega\left(y_t^{'}, \mathbf{x}_t^{'}\right)\right)}. \tag{5.3}$$

The training process can be formulated as an optimization problem to find the optimal $\eta^{'*}$ that gives the maximum of the objective function:

$$\eta^{'*} = \arg\max_{\eta^{'}} L\left(\eta^{'}\right). \tag{5.4}$$

We could use gradient ascend to search for the optimal $\eta^{'*}$. Similar to Eq.(3.21), the gradient of the log-likelihood with respect to the t-th training image can be calculated as:

$$\frac{\partial L^t\left(\eta^{'}\right)}{\partial \eta^{'}} = \omega\left(y_t, \mathbf{x}_t^{'}\right) - \sum_{y^{'} \in \mathcal{Y}} P\left(y^{'} \mid \mathbf{x}_t^{'}; \eta^{'}\right) \omega\left(y^{'}, \mathbf{x}_t^{'}\right). \tag{5.5}$$

5.2 Testing

Given a test image $\mathbf{x}$, we cannot calculate its part labels directly because its class label is unknown. Instead, we calculate the part labels for each class label to obtain a set of $|\mathcal{Y}|$ part labels $\{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(|\mathcal{Y}|)}\}$, where each part label vector $\mathbf{h}^{(k)}$ is obtained by finding patches using class label $y = k$. Then we concatenate them with global feature x_0 to form

a new set of feature vectors $\{\mathbf{x}^{'(1)}, \mathbf{x}^{'(2)}, \ldots, \mathbf{x}^{'(|\mathcal{Y}|)}\}$. We can calculate the probabilities of all possible assignments of the part labels using $\eta^{'}$ and classify it by the class label that gives the maximum probabilities. In other words, the final class label y^* of $\mathbf{x}$ is obtained as:

$$y^* = \arg\max_y \left[\max\{p(y|\mathbf{x}^{'(1)}; \eta^{'}), P(y|\mathbf{x}^{'(2)}; \eta^{'}), \ldots, P(y|\mathbf{x}^{'(|\mathcal{Y}|)}; \eta^{'})\}\right]. \tag{5.6}$$

5.3 Analysis

This method uses the learned part labels as a new set of features and sends them to the training process again. It uses the abstract information contained in the part labels explicitly. The model parameter is learned with a simple and fast method similar to the root filter learning method. Figure 5.1 shows the flow chart of the training and testing process. The output of the training process are two model parameters θ and $\eta^{'}$, which are learned using HCRF and gradient ascent respectively.

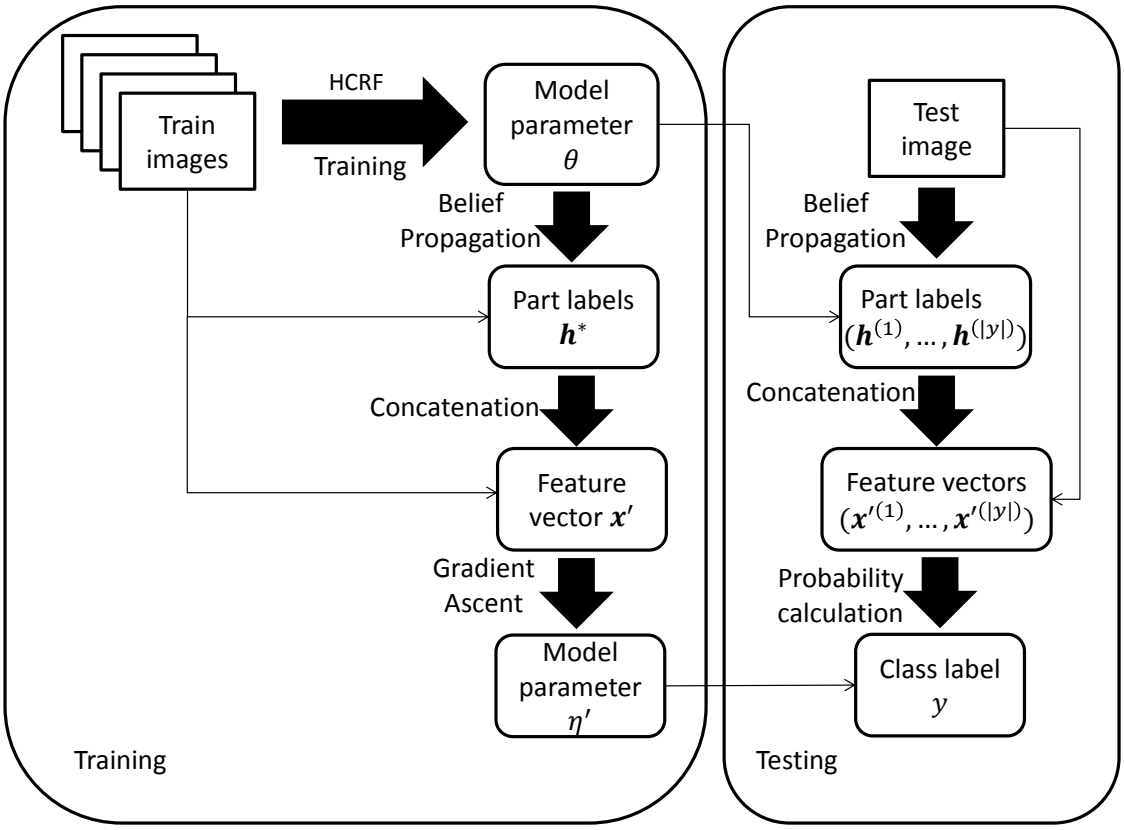


Figure 5.1: Flow chart of the Part Labels method.

This method is similar to the bag-of-words approach. The part labels can also be considered as words. But their ways to assign the part labels and words are different. This method uses the mode trained by HCRF to find the part labels, while the bag-of-words first computes a word vocabulary and next assigns words to patches by calculating the Euclidean distance. Another difference is that this method also combines both global and local features together.

The global features contain rich overall information for classification, and local part labels provide a higher level of abstraction from local patch features.

In fact, the Part Labels method can be considered as a hybrid of the root model and the HCRF model. It uses the part labels learned by HCRF and trains them using the root model. Comparing to the root model, it uses more information (the part labels) than the global feature alone. Comparing to the HCRF model, the Part Labels method has an extra maximization step, in addition to the summation approach in the HCRF model.

Chapter 6

Experimentation

In this chapter we evaluate the classification performance of the three algorithms: HCRF, MMHCRF and Part Labels method. We have implemented these models with Matlab R2011b. For the implementation of the BP algorithm, we used UGM[1], a Matlab library for undirected graphical models. We also used minFunc[2], a Matlab function for unconstrained optimization, for limited-memory BFGS. A Matlab pattern recognition toolbox PRTools[3] is used for data pre-processing and evaluation.

We evaluated these models on two datasets. The first dataset is the Weizmann human action dataset [5], which is a publicly available benchmark dataset. The second dataset is the Noldus ABR dataset [2], which is a private dataset for rodent behaviours. The structure of this chapter is organized as follows: first, we describe the datasets and explain how to calculate the global and local features on these datasets; next, experiments are done on the three algorithms to evaluate their performances; finally, an performance analysis of these methods is given.

6.1 Weizmann Dataset

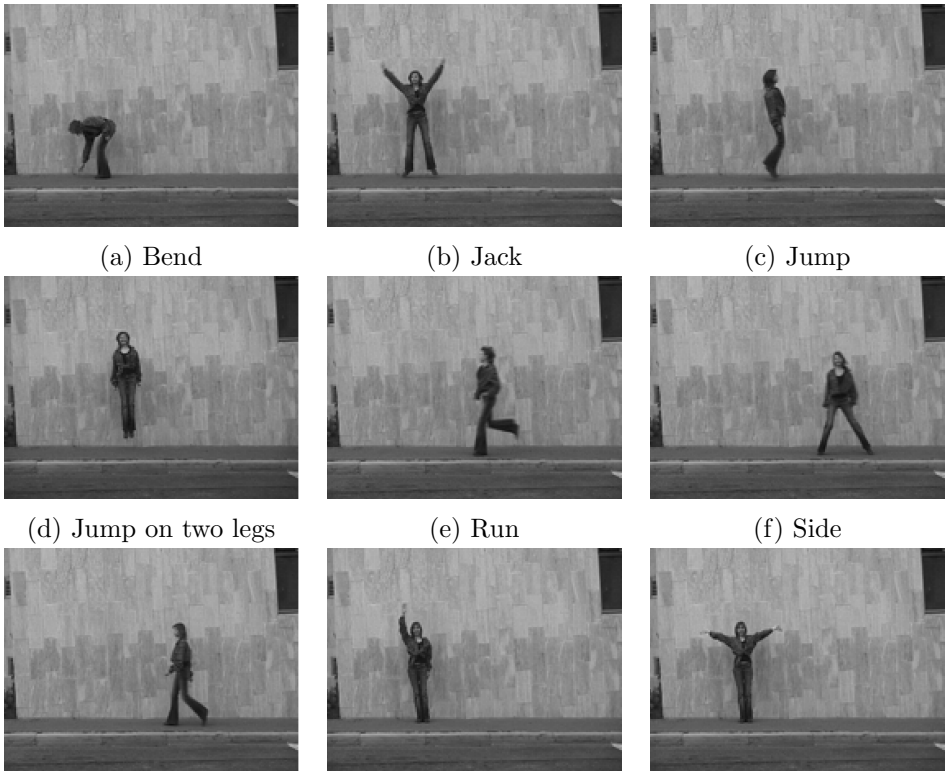
6.1.1 Dataset Description

The Weizmann dataset is a popular public benchmark used in many action recognition papers. It contains 83 video sequences at 180×144 pixel resolution and 25 frames per second. These video sequences record nine different people, each performing nine different natural actions: bend, jumping jack (or shortly "jack"), jump forward on two legs (or "jump"), jump in place on two legs (or "pjump"), run, gallop sideways (or "side"), walk, wave one hand (or "wave1"), wave two hands (or "wave2"). One of the subjects performs run and walk twice, one starts from the left of the screen to the right, and the other one starts from the opposite direction.

The dataset is captured under laboratory settings with fixed background and fixed camera

[1] Mark Schmidt. http://www.di.ens.fr/ mschmidt/Software/UGM.html. June 2013
[2] Mark Schmidt. http://www.di.ens.fr/ mschmidt/Software/minFunc.html. June 2013
[3] http://prtools.org/

location. It also provides a background subtraction mask for each video frame. Figure 6.1 shows an example frame of each action.



(a) Bend (b) Jack (c) Jump

(d) Jump on two legs (e) Run (f) Side

(g) Walk (h) Wave with one hand (i) Wave with two hands

Figure 6.1: Weizmann dataset

6.1.2 Feature Extraction

We calculate the motion features of these video sequences in the way similar to what has been proposed in [4]. This feature has been applied to action classification and is proved to perform reliably with low resolution images and imperfect video alignments.

This feature is based on pixel-wise optical flow as a natural technique to capture the motion information invariant to appearances. We compute the optical flow of each frame using the Lucas-Kanade algorithm [28] (see Figure 6.2b). The obtained optical flow vector F is split into two vectors corresponding to the horizontal and vertical components of the optical flow: F_x and F_y (see Figure 6.2c). Then F_x and F_y are further split into four non-negative channels: F_x^+, F_x^-, F_y^+ and F_y^-, so that $F_x = F_x^+ - F_x^-$ and $F_y = F_y^+ - F_y^-$ (see Figure 6.2d). This is to split the positive and negative components in the vectors. Each channels is blurred with a Gaussian kernel and normalized to obtain the four channels Fb_x^+, Fb_x^-, Fb_y^+ and Fb_y^- (see Figure 6.2e). This blurry technique is simple but powerful for capturing only the essential position information and discarding the minor variations.

Next, we use the background subtraction mask that is provided in the dataset to filter out the background pixels (see Figure 6.2f). Then move the salient region of the person to the

center of the view to obtain the final motion features (see Figure 6.2g). This procedure is different from the original feature in [4], which requires to track and stabilize the video first and compute the optical flow next. We do this adjustment because the computation of optical flow needs two consecutive frames; if we first track and stabilize the person in the video, the computed optical flow might not be accurate because the correspondence of pixels is lost.

The obtained motion feature vector is the global feature of a frame. We find the local patches on this frame from this global feature vector using the method described in section 3.6.1. The concatenation of the four channels $\left[Fb_x^+, Fb_x^-, Fb_y^+, Fb_y^-\right]$ within the salient region is the motion feature of this patch. But it is also important to describe the location of a patch. We divide the image into a grid of $w \times h$ bins. The bin where the patch is located is set to 1, all other bins are set to 0. This length $w \times h$ vector is the location feature of this patch. The motion feature vector and location feature vector are concatenated as the feature vector of a patch. The tree structure among the local patches are built by running a minimum spanning tree algorithm over these patches, using the distances between patches as edge weights. The result tree structure can be different from frame to frame.

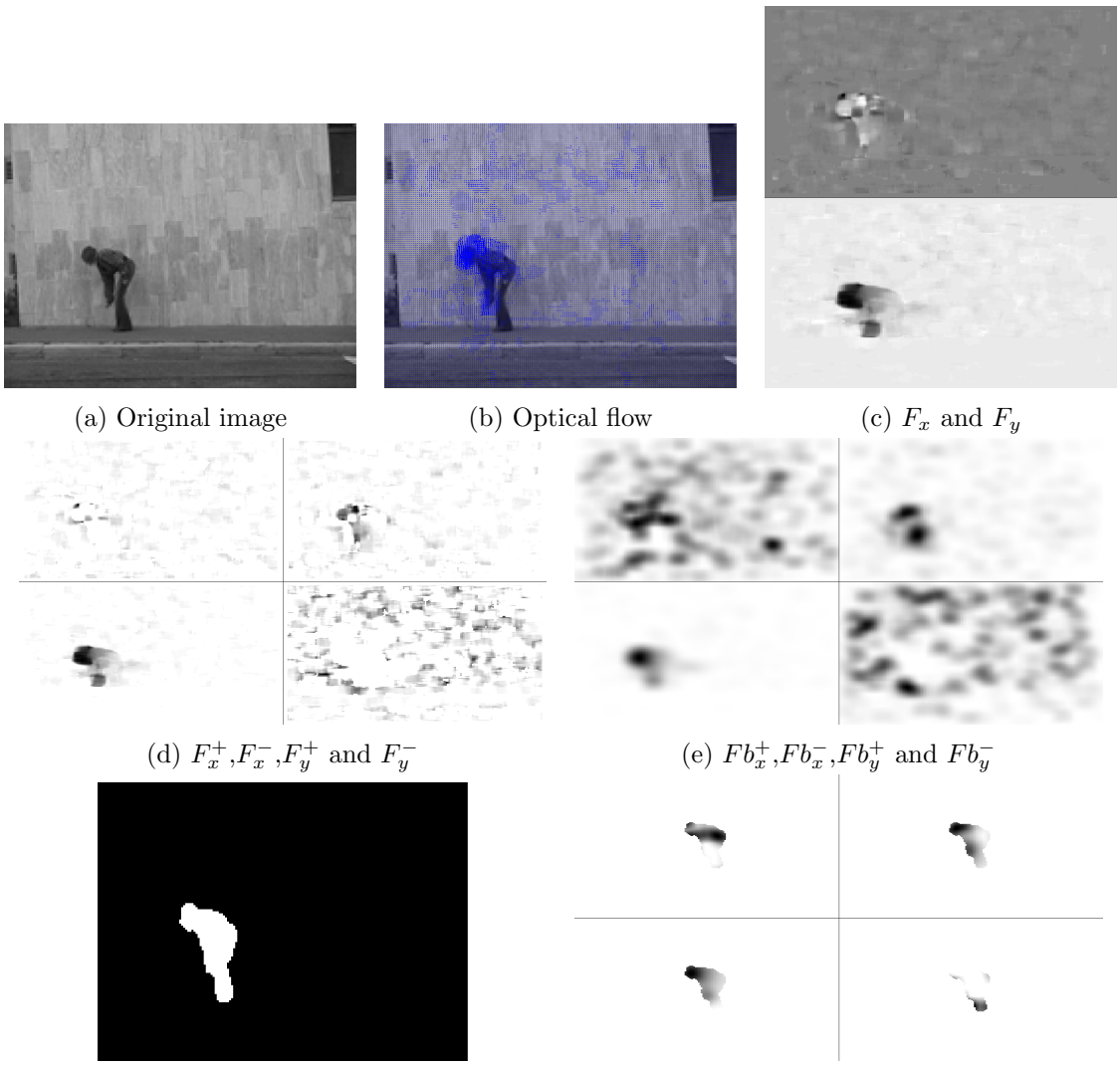
6.1.3 Experiment

We test the performances of HCRF, MMHCRF and the Part Labels method on the Weizmann dataset under the same experimental settings. We choose videos of five subjects as the training set, and the remaining videos performed by four other subjects as the testing set. All frames in the training set are randomly shuffled so that the training process converges faster. During testing, we classify frame-by-frame in a video (per-frame classification). We can obtain the action label for the whole video by majority voting of its frame labels (per-video classification).

Root Model Evaluation

We evaluate the root model as a baseline method. The root model only uses the global feature to train the root model parameter η. It only uses the root potential $\eta^\intercal \cdot \omega(y, x_0)$ as its potential function. Since the root model does not have the hidden part labels, it does not need to solve the inference problem for parameter estimation. This makes this method very efficient. In addition, it can give a global optimal solution because its objective function is convex when the hidden part labels are removed. We first evaluate its theoretical correctness. Then we use it to compare the motion features we proposed and the original motion features in [13].

Negative Log Likelihood The root model is trained with a maximum log likelihood approach. As the iterative method progresses, the sum of negative log likelihood for all training samples $-\sum_t L_{root}^t(\eta)$ should become smaller. Figure 6.3 gives the sum of the negative log likelihood over all training samples for each iteration. It shows the clear trend that the sum of negative log likelihood becomes smaller when the iterations becomes bigger. It allows us to see that after 20 iterations the model already converges towards a solution. Then it starts to slowly converge and eventually it oscillates around the global optimal.



(a) Original image (b) Optical flow (c) F_x and F_y

(d) F_x^+,F_x^-,F_y^+ and F_y^- (e) Fb_x^+,Fb_x^-,Fb_y^+ and Fb_y^-

(f) Background subtraction mask (g) Motion features

Figure 6.2: Calculation of motion features. (a) an sample image in action "bend"; (b) optical flow calculated from this image and its consecutive frame; (c) horizontal and vertical components of optical flow F_x and F_y; (d) half-wave rectified non-negative channels F_x^+,F_x^-,F_y^+ and F_y^-; (e) blurred four channels Fb_x^+,Fb_x^-,Fb_y^+ and Fb_y^-; (f) the background subtraction mask. (g) filter out background using mask and move the person to the center of view to obtain the final motion features.

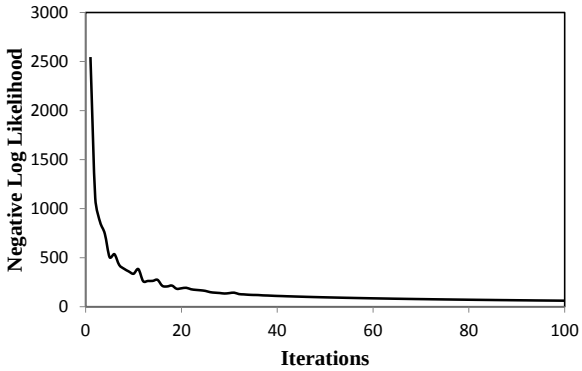


Figure 6.3: Negative log likelihood for each iteration in root model training.

Feature Evaluation In the previous subsection we have described how to calculate the motion features. The method we propose is slightly different from the original feature in [13]. The original feature needs to first track and stabilize the person and then calculate the optical flow, while our feature first calculates the optical flow and then move the person to the center of the view. We do this modification to avoid the optical flow calculation errors caused by the misalignment during stabilization.

	center-first	center-last
Per-frame	0.6711	**0.8659**
Per-video	0.8684	**0.9474**

Table 6.1: Comparison of the root model performance on two types of features.

We evaluate the performance of the root model on both types of features. Table 6.1 shows the classification result. The "center-first" column shows the per-frame and per-video classification results of the root model on the original feature which center the person first and calculate the optical flow subsequently. The "center-last" column shows the classification results on our proposed feature which calculates the optical flow first and centres the person last. The performance of our feature is much better than that of the original one on both per-frame and per-video classification. This is because video misalignment will cause errors in optical flow calculation. Our feature has avoid this error by using the original images for optical flow calculation. We will use our feature for experimentation in the rest of this section.

Root Model Performance Analysis Figure 6.4 and Figure 6.5 give the confusion matrices of the per-frame and per-video classification on our feature. For most actions the classification result is good, only "pjump" and "wave1" have video misclassification. One "wave with one hand" ("wave1") video is misclassified as "bend". This is because in "wave with one hand", the angle between arm and body is similar to the angle between upper body and lower body in "bend". Figure 6.6 is a comparison of "wave1" and "bend".

One "pjump" video is misclassified as "jack". Action "pjump" is to jump in place on two legs. After moving the person to the center of the view, the information about the whole body movement in the vertical direction is ignored. In this way, the body torso movements of

	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	0.9502	0.0000	0.0000	0.0000	0.0000	0.0083	0.0000	0.0415	0.0000
jack	0.0199	0.8458	0.0025	0.0771	0.0000	0.0000	0.0000	0.0025	0.0522
jump	0.1090	0.0128	0.7756	0.0000	0.0000	0.0897	0.0128	0.0000	0.0000
pjump	0.0051	0.2335	0.0051	0.7513	0.0000	0.0051	0.0000	0.0000	0.0000
run	0.0000	0.0058	0.0000	0.0000	0.8480	0.0526	0.0643	0.0000	0.0292
side	0.0000	0.0056	0.0056	0.0000	0.0167	0.9667	0.0056	0.0000	0.0000
walk	0.0000	0.0000	0.0318	0.0000	0.0058	0.1012	0.8613	0.0000	0.0000
wave1	0.1545	0.0091	0.0000	0.0000	0.0000	0.0000	0.0000	0.8318	0.0045
wave2	0.0080	0.0040	0.0000	0.0000	0.0161	0.0000	0.0000	0.0321	0.9398

Figure 6.4: Confusion matrix for the root model per-frame classification on the Weizmann dataset.

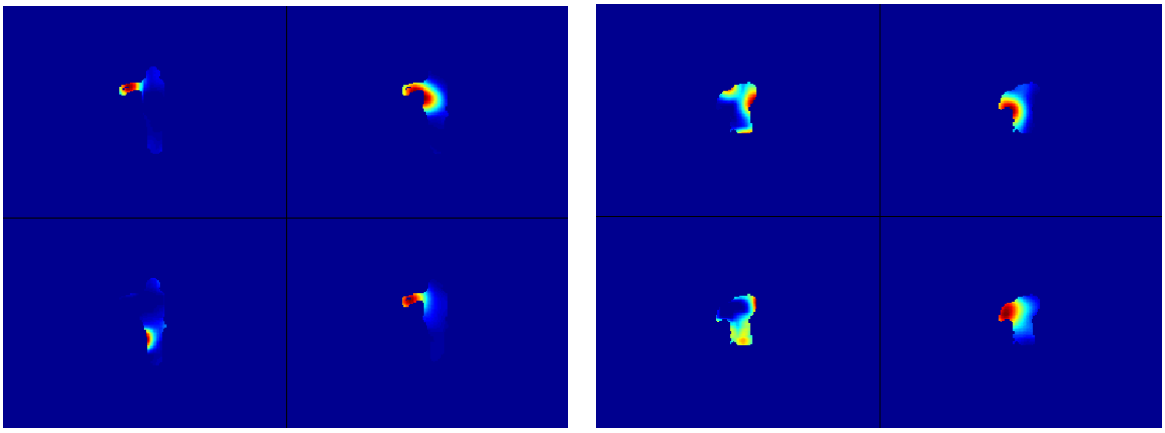
	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	1	0	0	0	0	0	0	0	0
jack	0	1	0	0	0	0	0	0	0
jump	0	0	1	0	0	0	0	0	0
pjump	0	0.25	0	0.75	0	0	0	0	0
run	0	0	0	0	1	0	0	0	0
side	0	0	0	0	0	1	0	0	0
walk	0	0	0	0	0	0	1	0	0
wave1	0.25	0	0	0	0	0	0	0.75	0
wave2	0	0	0	0	0	0	0	0	1

Figure 6.5: Confusion matrix for the root model per-video classification on the Weizmann dataset.

"pjump" and "jack" are similar to each other. Figure 6.7 shows an example that a "pjump" frame is mixed up with "jack". The root filter η for "pjump" has the characteristic that the whole body moves up, while η for "jack" shows that the limbs waving around and the torso moves up. After applying the feature on η for "jack", the movement of limbs is eliminated and only the movement of torso remains. Thus it is difficult to distinguish "pjump" from "jack".

There are some frames misclassified but their video classification is correct. This is because even though these actions are different from each other, like "bend" and "jump", some of their frames are similar.

The root model does not explicitly take into account the temporal information in a video sequence. It only uses the time information between two consecutive frames contained in the optical flow feature. As a result, movement patterns of all frames over time in an action are stacked up. As an example, Figure 6.8 is a visualization of the root filter η for action "jack". It is clear that the movement patterns of arms and legs are projected onto the root filter. When there comes a new image, it is classified as the action whose movement patterns it



(a) wave1 features (b) bend features

Figure 6.6: Features of wave1 and bend. The upper left is channel Fb_x^+, the upper right is channel Fb_x^-, the lower left is Fb_y^+ and the lower right is Fb_y^-. The angle between arm and body in wave1 is very similar to the angle between upper body and lower body in bend.

overlaps most.

This characteristic of the root model will cause confusion if two actions have similar frames. In addition, it causes the root model to prefer actions with more variations to actions with less variations. For an action with more variations, its root filter involves a broader set of features, whereas for an action with fewer variations, its root filter concentrates on a relatively small set of features. In the example of "jack" and "pjump", "pjump" is easy to be misclassified as "jack", but "jack" is rarely classified as "pjump". This is because the root filter of "jack" emphasises the actions of both limbs and torso, while the root filter of "pjump" focuses on the torso.

Overall, the root model is an efficient and powerful model. It has 0.8659 accuracy on per-frame classification, and 0.9474 accuracy on per-video classification. It assumes all features are independent and it does not model the temporal structure directly. It favours the actions with more variations over the actions with less variations.

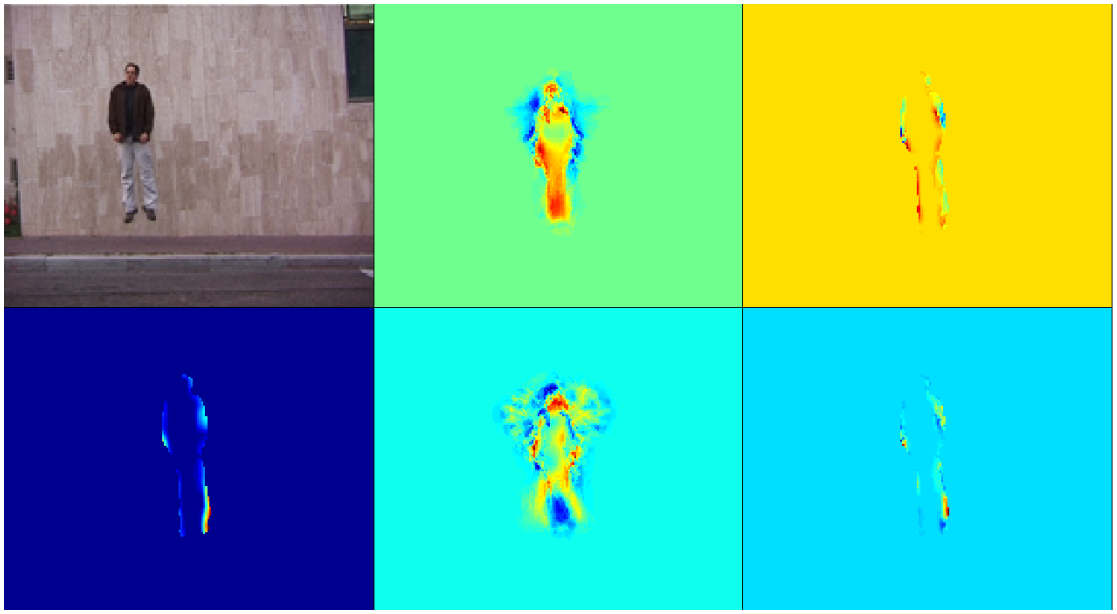


Figure 6.7: A "pjump" misclassification example. The upper left is the original frame of "pjump". The lower left is its motion feature in Fb_y^+ channel. We use this channel as an example because in the action "pjump", motion in the vertical direction is most important. The upper middle image is a visualization of η for "pjump" in Fb_y^+ channel. The lower middle image is a visualization of η for "jack" in Fb_y^+ channel. The upper right is a visualization of applying η for "pjump" on the feature, and the lower right is a visualization of applying η for "jack" on the feature. Note that upper right and the lower right are very similar with each other.

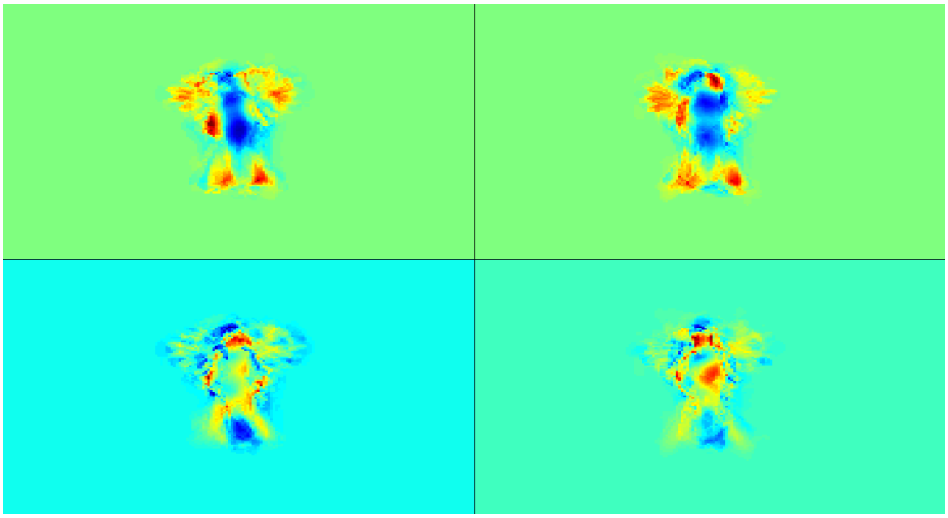


Figure 6.8: Visualization of the root filter for action "jack". The four grids in clockwise are Fb_x^+, Fb_x^-, Fb_y^- and Fb_y^+ channels.

HCRF Evaluation

To gain a better understanding of the different potential functions contributing to the performance of HCRF, we evaluate the HCRF model with two other baseline models. The first baseline (local HCRF) is a direct application of the original HCRF model [10]. This model does not use the root potential $\eta^\intercal \cdot \omega(y, x_0)$. That is, the local HCRF model only uses the root filter η to find the patches, but does not use the root potential in the objective function. Local HCRF only uses the local patch features, not the global features, for classification.

The second baseline (no pairwise HCRF) does not use the pairwise potential $\gamma^\intercal \cdot \psi(y, h_j, h_k)$. That is, there is no constraint between the part labels and there is no edge in graph G. The root filter η of this model is initialized using the root filter learned in the previous root model.

The full HCRF model is evaluated with two experiments. In the first experiment (HCRF random root), the root filter η is initialized randomly in the same way as other model parameters α, β and γ. In the second experiment (HCRF learned root), the root filter η is initialized using the root filter learned in the previous root model. This experiment is to evaluate the strength of root filter initialization.

The parameter settings in these experiments are kept the same with [13]. The size of possible part labels $\mathcal{H} = 10$. The number of salient patches on each frame is set to 10. The size of each patch is 5×5. The other parameters not specified in [13] are experimentally tuned. The grid division of each frame is set to 10×4. The model parameters α, β and γ are initialized randomly using a Gaussian distribution with mean 0 and standard deviation 0.01. All these parameters are tuned specificity for the Weizmann dataset.

Negative Log Likelihood The HCRF models are trained with a maximum log likelihood approach. As the iterative method continues, the sum of negative log likelihood for all training samples $-\sum_t P(y_t|\mathbf{x}_t, \theta)$ should become smaller. Figure 6.9 gives the sum of the negative log likelihood over all training samples for each iteration in four experiments. It shows the clear trend that in all experiments, the sum of negative log likelihood becomes smaller when the iterations becomes bigger.

It allows us to see that the local HCRF converges much slower than models with the root filter. Its local optimum is also much bigger than other models. This proves the strength of the root filter in speeding up the convergence speed and in searching for a better local optimum.

The three models with the root filter converges to the same local optimum, but the HCRF model whose root filter is initialized randomly starts with a bigger negative log likelihood comparing to the other two models whose root filters are initialized with the learned root model. It also has more fluctuation before convergence, which cause its outcome before convergence becoming less predictable. This proves that the learned root filter serves as a good starting point for η.

The HCRF model without the pairwise potential converges at the same pace with the full HCRF model, but its negative log likelihood is slightly smaller because it does not contain the pairwise potential in the objective function.

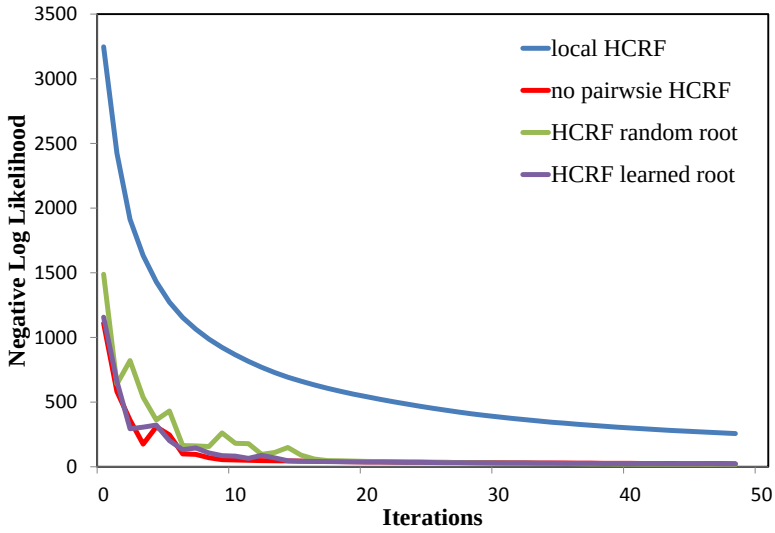


Figure 6.9: Negative log likelihood for each iteration in HCRF experiments.

HCRF Performance Comparison Table 6.2 shows the comparative performance of these HCRF models. The models including the root potential significantly outperform the local HCRF. The poor performance of local HCRF demonstrates that local patch features alone are not sufficiently informative for action classification under the HCRF framework. The contrast between the performance of the local HCRF and the other models with root potential proves the strength of combining the local patch features with the global feature.

Models	local HCRF	no pairwise HCRF	HCRF random root	HCRF learned root
Per-frame	0.4158	0.8409	0.8663	**0.8737**
Per-video	0.6053	0.8947	**0.9474**	**0.9474**

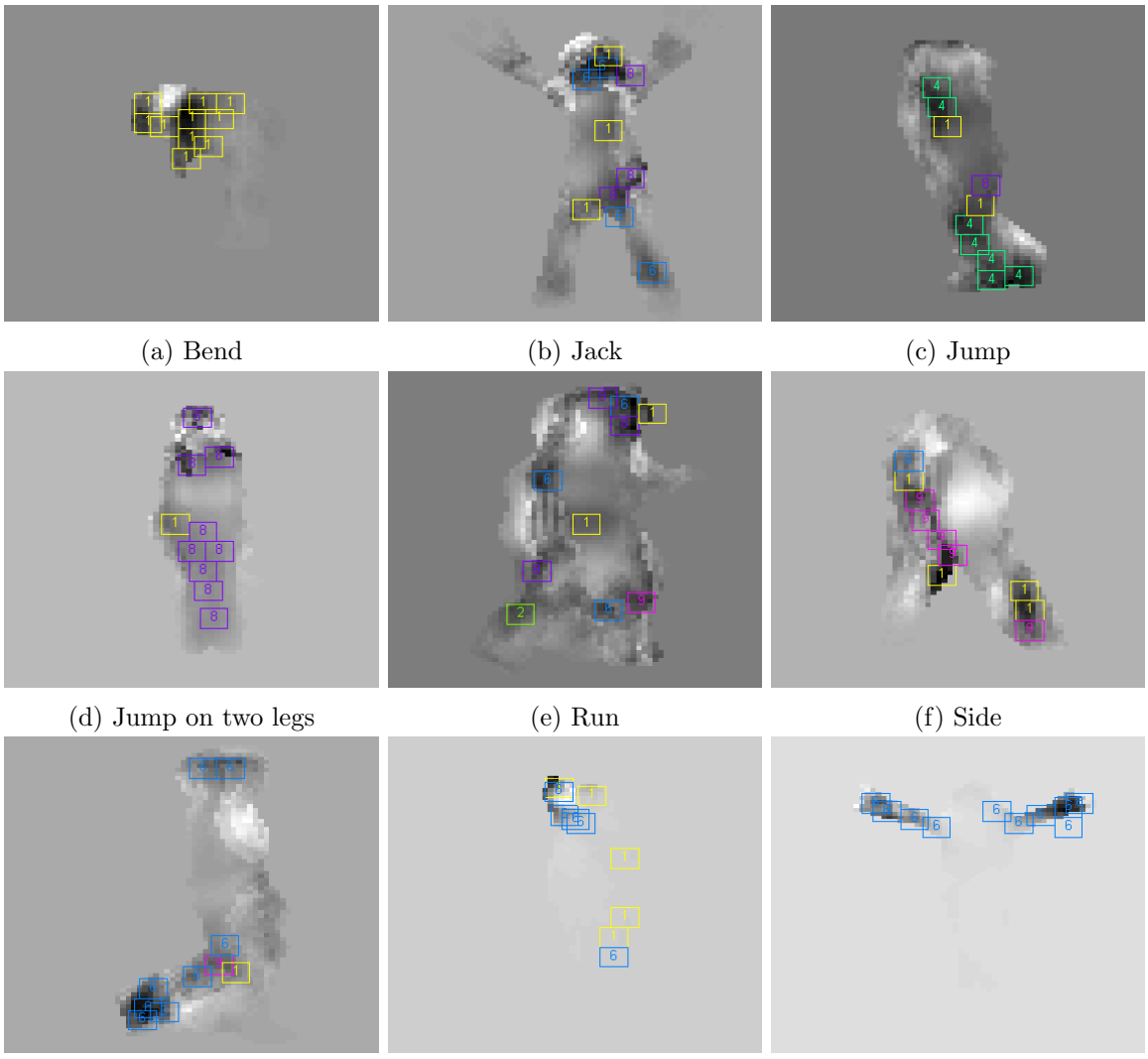
Table 6.2: Performance comparison of HCRF models on the Weizmann dataset.

The HCRF model without the pairwise potential performs worse than the full HCRF model. This proves that incorporating constraints between part labels improves the classification performance, because it captures the spatial structure among the part labels.

The performance of the two full HCRF model are comparable, although the HCRF model initialized with a learned root filter slightly outperforms the one with a randomly initialized root filter. This result shows that using the learned root filter for initialization only speeds up the process towards convergence, but does not have big influence on the final performance.

HCRF Performance Analysis We will evaluate the performance of the full HCRF model inialized by the learned root filter.

Figure 6.10 is a visualization of the learned part labels. The patches are assigned with their most likely part labels. From this visualization we can make observations about the meaning of the part labels. For example, the part label No.1 in yellow seems to represent the pattern "moving down" which occurs in "bend" as well as other actions. The part label No.8 in purple



(a) Bend (b) Jack (c) Jump

(d) Jump on two legs (e) Run (f) Side

(g) Walk (h) Wave with one hand (i) Wave with two hands

Figure 6.10: Learned part labels on the Weizmann dataset.

seems to present "moving up" which happens most in "pjump". The part label No.6 in blue seems to represent "rotating" which could happen in "walk" and "wave".

	root model	HCRF
Per-frame	0.8659	0.8737
Per-video	0.9474	0.9474

Table 6.3: Comparison of the root model with HCRF.

	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	0.9378	0.0083	0.0124	0.0000	0.0000	0.0000	0.0000	0.0415	0.0000
jack	0.0000	0.8930	0.0000	0.0796	0.0000	0.0000	0.0000	0.0025	0.0249
jump	0.0705	0.0385	0.8141	0.0000	0.0000	0.0577	0.0128	0.0000	0.0064
pjump	0.0051	0.2487	0.0000	0.7411	0.0000	0.0051	0.0000	0.0000	0.0000
run	0.0000	0.0585	0.0000	0.0000	0.8187	0.0409	0.0643	0.0058	0.0117
side	0.0000	0.0167	0.0167	0.0000	0.0111	0.9500	0.0056	0.0000	0.0000
walk	0.0000	0.0087	0.0202	0.0000	0.0116	0.0636	0.8960	0.0000	0.0000
wave1	0.1682	0.0000	0.0000	0.0000	0.0045	0.0000	0.0000	0.8227	0.0045
wave2	0.0080	0.0000	0.0000	0.0000	0.0482	0.0000	0.0000	0.0241	0.9197

Figure 6.11: Confusion matrix of per-frame classification results of HCRF learned root on Weizmann dataset.

	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	1	0	0	0	0	0	0	0	0
jack	0	1	0	0	0	0	0	0	0
jump	0	0	1	0	0	0	0	0	0
pjump	0	0.25	0	0.75	0	0	0	0	0
run	0	0	0	0	1	0	0	0	0
side	0	0	0	0	0	1	0	0	0
walk	0	0	0	0	0	0	1	0	0
wave1	0.25	0	0	0	0	0	0	0.75	0
wave2	0	0	0	0	0	0	0	0	1

Figure 6.12: Confusion matrix of per-video classification results of HCRF learned root on Weizmann dataset.

Figure 6.11 and Figure 6.12 show the confusion matrices of per-frame and per-video HCRF classification results. If we compare the classification results of the root model and the HCRF model (see Table 6.3) and their confusion matrices, surprisingly, we find their outputs are not significantly different from each other. This means the root filter has dominated the HCRF model and lowered the contributions of the other parts.

One possible reason of this result is that the global feature and local patch features are the same type of feature. The local patch feature is simply part of the global feature. Therefore the discriminative power of the global feature and the local patch feature is overlapping with

each other. In fact, among the frames correctly classified by the local HCRF model, 92.4% of them are also correctly classified by the root model.

Another reason is that the local patch features in 2D space are not informative enough for action recognition. Although this type of features work well on recognition tasks in the 2D domain, like object recognition, it is not sufficient for challenging tasks like action recognition. In Table 6.4 we have listed the classification results of the local HCRF and a few previous works which only use local patch features in 2D space. All of their performance are not satisfactory, regardless of their classification results.

Method	Classification result (%)
Scovanner et al. [17] (2D SIFT)	30.4
Niebles & Fei-Fei [27]	55.0
Local HCRF	41.6

Table 6.4: Classification results of works using only 2D patch features on Weizmann dataset.

The drawback of this HCRF model is that it neglects the temporal structure in an action, which is important for action recognition. It only models the spatial structure among the part labels, but the spatial structure of an action could differ from frame to frame. If we look at the confusion matrix of local HCRF, we can see that it performs best on the actions where the pose does not change significantly during the motion, such as "bend", "jump", "pjump" and "wave2". If we look at the learned part labels in Figure 6.10, it is clear that these actions need fewer part labels to describe them.

	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	0.7095	0.0249	0.0000	0.0124	0.0000	0.0000	0.0000	0.2116	0.0415
jack	0.1144	0.3284	0.0100	0.3159	0.0299	0.0299	0.0249	0.0149	0.1318
jump	0.0769	0.0449	0.5256	0.0000	0.1282	0.0577	0.1218	0.0321	0.0128
pjump	0.1421	0.2132	0.0000	0.6396	0.0000	0.0000	0.0000	0.0000	0.0051
run	0.1637	0.0175	0.4327	0.0000	0.2047	0.0702	0.0702	0.0234	0.0175
side	0.0167	0.0444	0.2611	0.0056	0.2833	0.1944	0.1833	0.0056	0.0056
walk	0.0231	0.0318	0.3757	0.0405	0.1445	0.0954	0.2572	0.0202	0.0116
wave1	0.4409	0.0409	0.0000	0.0136	0.0000	0.0000	0.0000	0.4727	0.0318
wave2	0.1807	0.1245	0.0000	0.0040	0.0161	0.0000	0.0000	0.1727	0.5020

Figure 6.13: Confusion matrix of per-frame classification results of local HCRF on Weizmann dataset.

Overall, the performance of the HCRF model is comparable to the root model. It uses the same kind of feature for both local and global representations. This leads to the domination of the root filter over other local parts of the model. The local patch feature in 2D alone is not sufficient for action recognition. The HCRF model does not capture the temporal structure in an action, thus it has difficulty on handling complex actions with more pose changes.

MMHCRF Evaluation

We have implemented the MMHCRF model for the Weizmann dataset. Unfortunately we are not able to get a satisfaction result. In order to prove that the failure is not caused by the dataset itself or the max-margin approach, we evaluated the Weizmann dataset on a simpler model which only has the root potential and trains its model parameter with a max-margin approach.

This model only uses the global features to train the model parameter η. It only uses the root potential $\eta^{\intercal} \cdot \omega(y, x_0)$ as its potential function. And it trains the model parameter with a max-margin approach, same as MMHCRF. This model and MMHCRF can be seen as analogies of the root model and HCRF. Because it does not contain the hidden part labels, it becomes a standard multi-class SVM [16]. Recall that in Eq.(4.2), if we replace $f_\theta(\mathbf{x}, y)$ with $\eta^{\intercal} \cdot \omega(y, x_0)$:

$$\begin{gathered} \min_{\theta,\xi} \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t \\ \text{s.t. } \eta^{\intercal} \cdot \omega(y, x_{t,0}) - \eta^{\intercal} \cdot \omega(y_t, x_{t,0}) \geqslant \xi_t - 1, \forall t, \forall y \neq y_t \\ \xi_t \geq 0, \forall t. \end{gathered} \tag{6.1}$$

It is a quadratic program with only linear constraints. This is in the same form as the standard multi-class SVM. Therefore we could evaluate this model with an off-the-shelf SVM solver, *SVM multiclass*[4].

	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	0.9544	0.0041	0.0000	0.0041	0.0000	0.0083	0.0000	0.0290	0.0000
jack	0.0050	0.8383	0.0000	0.1294	0.0000	0.0000	0.0000	0.0025	0.0249
jump	0.0769	0.0000	0.8654	0.0000	0.0064	0.0513	0.0000	0.0000	0.0000
pjump	0.0000	0.1726	0.0000	0.8223	0.0000	0.0051	0.0000	0.0000	0.0000
run	0.0000	0.0351	0.0058	0.0000	0.8304	0.0468	0.0702	0.0000	0.0117
side	0.0000	0.0056	0.0056	0.0000	0.0000	0.9833	0.0056	0.0000	0.0000
walk	0.0000	0.0058	0.0029	0.0000	0.0087	0.0751	0.9075	0.0000	0.0000
wave1	0.1455	0.0045	0.0000	0.0091	0.0000	0.0000	0.0000	0.8364	0.0045
wave2	0.0161	0.0201	0.0040	0.0080	0.0000	0.0000	0.0000	0.0040	0.9478

Figure 6.14: Confusion matrix of per-frame classification results of multi-class SVM on Weizmann dataset.

Figure 6.14 and 6.15 are the confusion matrices of the per-frame and per-video multi-class SVM classification results on the Weizmann dataset. The overall accuracy is 0.8867 for per-frame classification and 0.9737 for per-video classification. If we compare this model with the root model (see Table 6.5), we can see their performance are similar with each other, but the multi-class SVM is slightly better. This difference could be caused by the implementation, because *SVM multiclass* is a highly optimized SVM solver. But this experiment has proved the strength of the max-margin approach, because the multi-class SVM trains its model parameter with a max-margin approach. It has also proved that the reason of the failure

[4] http://svmlight.joachims.org/svm_multiclass.html

	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	1	0	0	0	0	0	0	0	0
jack	0	1	0	0	0	0	0	0	0
jump	0	0	1	0	0	0	0	0	0
pjump	0	0	0	1	0	0	0	0	0
run	0	0	0	0	1	0	0	0	0
side	0	0	0	0	0	1	0	0	0
walk	0	0	0	0	0	0	1	0	0
wave1	0.25	0	0	0	0	0	0	0.75	0
wave2	0	0	0	0	0	0	0	0	1

Figure 6.15: Confusion matrix of per-video classification results of multi-class SVM on Weizmann dataset.

of MMHCRF is not because the dataset, or the features are not suitable for an max-margin approach.

MMHCRF trains the model parameter in a similar way as the multi-class SVM. Their difference is that MMHCRF has introduced the hidden part labels. These hidden part labels cause the optimization problem to become not convex, thus MMHCRF can only search for a local optimum. Different ways of model parameter initialization lead to different local optimal solution. Therefore the performance of MMHCRF is heavily dependent on the way of model parameter initialization.

MMHCRF learns the hidden part labels in a maximization approach, which chooses an assignment of part labels with the highest score in every iteration. After the model parameter is initialized and its corresponding set of hidden part labels are learned, it is difficult to update these hidden part labels, because they have the highest probability (1) and all other possible assignments have the lowest probability (0). This leads the optimization process to be more dependent on the parameter initialization and easier to get stuck in local optimum. This approach is less predictable and less robust than HCRF, which adopted a summation approach to sum over all possible assignments of part labels.

In addition, this model is very sensitive to parameter settings, especially the trade-off parameter C. C is an important parameter for the max-margin approach. It controls the trade off between margin size and training error. The bigger the C, the less the system is tolerable to the training error.

Another problem of this model is that it is much slower compared to HCRF. This is because it needs to solve both inference problem and a quadratic program for every training sample, whereas HCRF only needs to do the inference.

Part Labels Evaluation

We evaluate the Part Labels method described in Chapter 5 on the Weizmann dataset. This method utilizes the model parameter trained by the HCRF framework to find the most likely part labels for each frame. It uses these part labels as local features and concatenates them

with the global feature for training. Table 6.5 shows the comparison of the root model, HCRF, multi-class SVM and this Part Labels method. Figure 6.16 and Figure 6.17 are the confusion matrices of the per-frame and per-video classification results of the Part Labels method.

	root model	HCRF	multi-class SVM	Part Labels
Per-frame	0.8659	0.8737	0.8867	0.8705
Per-video	0.9474	0.9474	0.9737	0.9737

Table 6.5: Comparison of the root model, HCRF, multi-class SVM and Part Labels.

	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	0.9046	0.0124	0.0124	0.0041	0.0000	0.0000	0.0000	0.0415	0.0249
jack	0.0075	0.9030	0.0000	0.0746	0.0000	0.0025	0.0000	0.0025	0.0100
jump	0.0449	0.0064	0.8269	0.0000	0.0256	0.0833	0.0128	0.0000	0.0000
pjump	0.0000	0.2487	0.0000	0.7360	0.0000	0.0000	0.0000	0.0051	0.0102
run	0.0000	0.0000	0.0000	0.0000	0.8363	0.0643	0.0994	0.0000	0.0000
side	0.0000	0.0111	0.0167	0.0000	0.0056	0.9611	0.0056	0.0000	0.0000
walk	0.0000	0.0000	0.0318	0.0000	0.0116	0.0925	0.8642	0.0000	0.0000
wave1	0.1045	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.8318	0.0636
wave2	0.0080	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0723	0.9197

Figure 6.16: Confusion matrix of per-frame classification results of Part Labels on the Weizmann dataset.

	bend	jack	jump	pjump	run	side	walk	wave1	wave2
bend	1	0	0	0	0	0	0	0	0
jack	0	1	0	0	0	0	0	0	0
jump	0	0	1	0	0	0	0	0	0
pjump	0	0.25	0	0.75	0	0	0	0	0
run	0	0	0	0	1	0	0	0	0
side	0	0	0	0	0	1	0	0	0
walk	0	0	0	0	0	0	1	0	0
wave1	0	0	0	0	0	0	0	1	0
wave2	0	0	0	0	0	0	0	0	1

Figure 6.17: Confusion matrix of per-video classification results of Part Labels on the Weizmann dataset.

The per-frame classification results of these four models are not significantly different from each other. This is because these methods essentially use the same information. The part labels are learned from the local patch features which are the same as the global features.

But the performance of the Part Labels method is still slightly better than the performance of the root model. This is because the Part Labels method has used the part labels in addition to the global feature. In the confusion matrix of per-video Part Labels classification, "wave1" is not misclassified as "bend". Even though the global features of these two actions are similar,

their part labels are different, as we can see from Figure 6.10. Using this information in the Part Labels model helps to distinguish them from each other.

6.2 Noldus ABR Dataset

6.2.1 Dataset Description

Noldus ABR dataset is a private dataset[5] of Noldus Information Technology to evaluate automatic behaviour recognition software for rodents. It consists of videos with 254,652 frames performed by 4 different rats. The videos are recorded using a top view camera, at 25 frames per second, and with a pixel resolution of 720×576. These recordings are annotated frame by frame with 15 action classes: drink, eat from feeder, eat from fist, groom, shake, jump, rest, rear unsupported down, rear unsupported up, rear wall down, rear wall up, scratch, sniff, walk and other. As the variations within the "other" action class is too big, if we train it together with other actions it might interfere the others. Therefore, we filter out the frames with the "other" action label in the dataset. In the end, there remain 157,332 frames in the dataset.

(a) Groom (b) Rear wall (c) Walk

Figure 6.18: Example frames of the Noldus ABR dataset.

During the training and testing process, all frames are classified using the remaining 14 action labels. But in the final result, some of the actions are grouped together after classification, because these are the actual target actions. "Eat from feeder" and "eat from fist" are grouped into action "eat"; "groom" and "scratch" are grouped into action "groom"; "rear unsupported down" and "rear unsupported up" are grouped into action "rear unsupported" (or "rear-u"); "rear wall down" and "rear wall up" are grouped into action "rear wall" (or "rear-w"). Table 6.6 shows the actions used for classification and the target actions after grouping. Figure 6.18 shows example frames for three target actions.

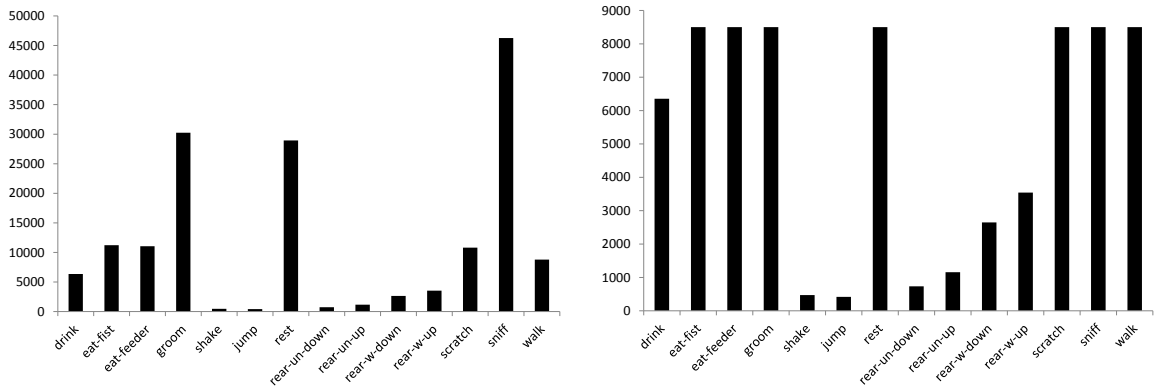
A characteristic of this dataset is that the distributions of frames among different actions are not equal. This is because the videos are recorded under non-intrusive settings where the rats move naturally, so they can do some actions more often and do other actions less frequently. This unbalance causes a problem for training: a frame is more likely to be labelled as an action with more samples than an action with fewer samples. In other words, the actions with less

[5]For more information about the dataset, please contact Nico van der Aa by email: n.vanderaa@noldus.nl.

classification actions	target actions	classification actions	target actions
drink	drink	rear unsupported down	rear-u
eat from feeder	eat	rear unsupported up	
eat from fist		rear wall down	rear-w
groom	groom	rear wall up	
scratch		shake	shake
jump	jump	sniff	sniff
rest	rest	walk	walk

Table 6.6: The grouping of actions. Classification actions are the actions actually used in classification. Target actions are the actions of interest.

training samples are "overshadowed" by the actions with more training samples. We use a simple down sampling method [29] to solve this problem. We restrict the number of training samples of each action class to be at most 8500. This threshold is experimentally tuned, so that it can ensure there are enough training samples for each action but the distribution becomes more balanced. We use this method because it is simple and it saves the training time. Figure 6.19a is the original distribution of frames over action classes. Figure 6.19b shows the distribution after setting the threshold as 8500 has become more balanced. But the actions such as shake, jump, rear unsupported and rear wall still have much fewer training samples than the other actions. This is because these actions are short actions which only last for a few frames.



(a) Original dataset distribution

(b) Dataset distribution after setting threshold

6.2.2 Feature Extraction

We use the features proposed by Van Dam et al. [2] in our experiments on the Noldus ABR dataset. The features have two parts: tracking features and motion features. Tracking features describe the contour, shape and velocity of rats. Motion features are extracted from optical flow statistics. We will describe how to extract these two types of features below.

A rat in a video is tracked using EthoVision XT 8.0 [6], a video tracking software developed

[6] http://www.noldus.com/animal-behavior-research/products/ethovision-xt

by Noldus Information Technology. The tracking features include body contour, shape model and velocity. The body contour features describe the shape and size of the rat. The shape model features include the center of gravity, nose point and tail base. The velocity feature describes the velocity of the nose and tail. The tracking system also identify different parts on animal body, such as head, middle and rear.

The motion features are based on optical flow features. The Lucas-Kanade algorithm [28] was used to calculate optical flow on different rat body parts on each frame. Then the motion statistics, such as mean, variance and motion intensity, are calculated on each body part at multiple sliding temporal windows. The motion features also include motion periodicity, which is calculated using log-Gabor filters [30].

Tracking features and motion features of a frame are concatenated to form a feature vector of length 169. This feature vector is used as global feature for this frame. We pick out the features related to different body parts as local features. There are three body parts of interest: head, middle and rear. Each body part has multiple sets of local features. These body parts and their feature sets together yield 6 nodes: head1, head2, head3, middle, rear1 and rear2. It is helpful to have three head nodes because the head movement is important for action detection on rats. Middle body part only has one node because the motion pattern of the middle body is closely connected to that of the rear, so it is redundant to include too many nodes for the middle body part. Each node has 26 features to describe its local action. These features include motion statistics, strength and periodicity.

6.2.3 Experiment

We test the performance of the root model, HCRF, MMHCRF and the Part Labels method on the Noldus ABR dataset. We choose the video frames performed by three rats as training set, and the remaining frames performed by the other rat as testing set. The videos in the dataset are not split into video clips with a single action class label, as the Weizmann dataset does. Therefore there will be no per-video classification. We will only classify the testing set frame-by-frame (per-frame classification).

For an unbalanced dataset like the Noldus ABR dataset, there are possibilities that a classifier has good classification results on certain actions with a lot testing samples, but bad results on the actions with only a few testing samples. In this case, the overall classification accuracy is high because the majority of the testing frames are classified correctly. But it does not reflect the fact that the results on the actions with less testing samples are not desirable. To make a fair comparison between different models, we evaluate the average classification accuracy of all actions (average precision). The average precision is the average of the percentage accuracies of all actions, in other words, the mean of the numbers on the diagonal of the confusion matrix.

Root Model Evaluation

Similar as the experiments on the Weizmann dataset, we evaluate the root model on the Noldus ABR dataset as a baseline method. The root model only uses the root potential $\eta^{\intercal} \cdot \omega(y, x_0)$ to train the root filter η. It does not have the hidden part labels, thus it does

not need to solve the inference problem for parameter estimation. This makes the training process very fast. In addition, when the hidden part labels are removed, its objective function is convex, therefore the root model can give a global optimal solution.

	drink	eat	groom	shake	jump	rest	rear-u	rear-w	sniff	walk
drink	0.7904	0.0889	0.0113	0.0000	0.0000	0.0000	0.0000	0.0000	0.1095	0.0000
eat	0.0840	0.7971	0.0059	0.0000	0.0000	0.0014	0.0000	0.0047	0.1066	0.0002
groom	0.0071	0.1710	0.7524	0.0018	0.0003	0.0056	0.0031	0.0027	0.0546	0.0013
shake	0.0167	0.0067	0.1333	0.4667	0.0367	0.0000	0.0367	0.0167	0.1100	0.1767
jump	0.0000	0.0000	0.0000	0.1316	0.3889	0.0000	0.0117	0.0175	0.0614	0.3889
rest	0.0001	0.0037	0.0000	0.0002	0.0000	0.9797	0.0000	0.0013	0.0146	0.0002
rear-u	0.0000	0.0000	0.0000	0.0083	0.0000	0.0000	0.4606	0.0871	0.0498	0.3942
rear-w	0.0590	0.0041	0.0000	0.0126	0.0026	0.0011	0.0326	0.5936	0.2462	0.0482
sniff	0.0953	0.1270	0.0161	0.0050	0.0006	0.0033	0.0203	0.0272	0.6526	0.0525
walk	0.0040	0.0034	0.0028	0.0042	0.0048	0.0000	0.0080	0.0024	0.1112	0.8591

Figure 6.20: Confusion matrix of the root model classification results on Noldus ABR dataset.

The classification rate of the root model on the 14 classification actions is 0.7388. After grouped into 10 target actions, the classification rate is 0.7646. The average precision of the target actions is 0.6741. Figure 6.20 is the confusion matrix of the classification result on the target actions. From the confusion matrix we can see that for most actions the classification rate is good. Only for actions "shake", "jump" and "rear unsupported", the classification results are notably worse than the other actions. If we look at the dataset distribution in Figure 6.19b again, it is clear that these three actions have the much less training samples than the other actions. The classification result of "rear-wall" is better than these three actions, but still worse than all others. This is because "rear-wall-down" and "rear-wall-up" are trained and classified separately, then their classification results are grouped together. In this way, each of the sub-actions has fewer training samples than other actions, as we can see from Figure 6.19b. Lacking of training samples causes the root model not be able to capture enough variations of these actions. Thus they are easily miss classified as other actions with large variations.

Another reason for the poor performance of these actions is that they are too short. The Noldus ABR features are calculated on sliding temporal windows. The sliding windows are too large for the short actions lasting only a few frames. Their features contain too much information from other behaviours and do not provide an accurate description of these actions.

The confusion of sniff with other actions drags down the overall accuracy. The original method [2] also suffers the same problem of mistakenly labelling other actions as "sniff". This is because "sniff" is an action with large variations. As explained in the Weizmann experiments section, the root model has the concept of projecting different frames of an action into a plane, and the root filter selects the features important for this action by assigning them different weights. This causes the root model favours actions with large variations over actions with small variations. A new image can easily be misclassified as another action with large intra-class variations, such as "sniff".

HCRF Evaluation

Same as in the experiments on the Weizmann dataset, we will evaluate the performance of HCRF on the Noldus ABR dataset with two baseline models: local HCRF and the HCRF model with no pairwise potential (no pairwise HCRF). Unlike in the Weizmann dataset experiments, we only evaluate the full HCRF model whose root filter is initialized with the learned root model, because we have already proved that the way of initialization only influence the convergence process, but not the final performance.

The parameter settings in these experiments are experimentally tuned. The size of possible part labels $\mathcal{H} = 15$. The model parameters are initialized randomly under the standard Gaussian distribution. The graph structure among the patches is manually defined. This is because the location information is not explicitly provided in the features. We can only define a fixed spatial structure from the relative locations of the body parts. Figure 6.21 is the defined graph structure. On one side, head, middle and rear are connected in a chain. On the other side, two head nodes are connected, and one of them is connected with the rear.

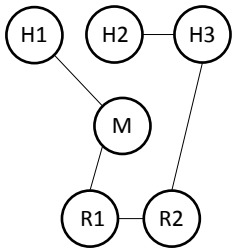


Figure 6.21: Graph structure of body parts in a frame. "H" is short for head, "M" is short for middle and "R" is short for rear.

HCRF Performance Comparison Table 6.7 shows the comparative performance of these HCRF models. Same as in the Weizmann dataset, the models with root potential significantly outperform the local HCRF. The poor performance of local HCRF demonstrates that local patch features alone are not sufficiently informative for action classification under the HCRF framework. The contrast between the performance of the local HCRF and the other models with root potential proves the strength of combining the local patch features with the global feature.

Models	local HCRF	no pairwise HCRF	HCRF
Classification actions	0.5936	0.7334	0.7322
Target actions	0.6121	0.7740	0.7567
Average precision	0.4862	0.6788	0.6854

Table 6.7: Performance comparison of HCRF models on the Noldus ABR dataset.

Surprisingly, the HCRF model without pairwise potential performs comparable, even slightly better than the full HCRF model. This shows that the manually defined graph is too rigid and does not capture much information about the spatial structure. For consistency reason we will still use the full HCRF model for later evaluation.

Models	root model	HCRF	Van Dam et al. [2]
Classification actions	0.7388	0.7322	-
Target actions	0.7646	0.7567	0.61
Average precision	0.6741	0.6854	0.57

Table 6.8: Performance comparison of the root model, HCRF and the method in Van Dam et al. [2] on the Noldus ABR dataset.

Table 6.8 also compares the full HCRF model with the root model. Like on the Weizmann dataset, the performance of the root model and the full HCRF model on the Noldus ABR dataset are comparable. The overall classification results of the root model are even slightly better than that of HCRF. Figure 6.22 is the confusion matrix of HCRF classification results. If we compare their confusion matrices, we can see the tendency of discrimination over actions lacking variances in the root model is softened in the HCRF model. The four actions without enough training frames ("shake", "jump", "rear-u" and "rear-w"), all have better classification results in the HCRF model than in the root model. The average precision of HCRF is also higher than that of the root model. This proves the discriminative power of the part labels. Even though without enough training samples, the HCRF model assigns part labels to these actions and differentiates them from other actions.

	drink	eat	groom	shake	jump	rest	rear-u	rear-w	sniff	walk
drink	0.7290	0.1001	0.0006	0.0056	0.0000	0.0188	0.0000	0.0000	0.1458	0.0000
eat	0.0667	0.7158	0.0127	0.0000	0.0000	0.0201	0.0002	0.0107	0.1723	0.0013
groom	0.0119	0.1187	0.7515	0.0128	0.0010	0.0182	0.0033	0.0047	0.0768	0.0011
shake	0.0100	0.0067	0.0433	0.5833	0.0400	0.0000	0.0200	0.0133	0.1067	0.1767
jump	0.0000	0.0000	0.0000	0.0702	0.4386	0.0000	0.0175	0.0322	0.1170	0.3246
rest	0.0011	0.0015	0.0000	0.0007	0.0000	0.9798	0.0000	0.0011	0.0153	0.0005
rear-u	0.0000	0.0000	0.0000	0.0373	0.0000	0.0000	0.5311	0.0581	0.0871	0.2863
rear-w	0.0286	0.0033	0.0000	0.0137	0.0059	0.0074	0.0337	0.6470	0.2258	0.0345
sniff	0.0615	0.1053	0.0109	0.0109	0.0014	0.0051	0.0193	0.0540	0.6936	0.0380
walk	0.0032	0.0040	0.0010	0.0050	0.0122	0.0026	0.0212	0.0112	0.1557	0.7838

Figure 6.22: Confusion matrix of HCRF classification results on the Noldus ABR dataset.

We also compare with the method in Van Dam et al. [2] in Table 6.8. Both the root model and the HCRF model outperform their method. But we admit that our experiment settings are not completely identical, because our ways to split the training and testing sets are different.

MMHCRF Evaluation

Same as for the Weizmann dataset, we did not manage to get a satisfactory result of MMHCRF for the Noldus ABR dataset. We also evaluated this dataset with the standard multi-class SVM classifier. The overall accuracy of this model is 0.7535 on the classification actions, and

0.7786 on the target actions. Its average precision is 0.6077. Figure 6.23 shows the confusion matrix of the classification result on the target actions.

	drink	eat	groom	shake	jump	rest	rear-u	rear-w	sniff	walk
drink	0.7685	0.1270	0.0081	0.0000	0.0000	0.0000	0.0000	0.0000	0.0964	0.0000
eat	0.0577	0.8204	0.0080	0.0000	0.0000	0.0013	0.0000	0.0021	0.1103	0.0001
groom	0.0028	0.1395	0.7901	0.0007	0.0000	0.0073	0.0009	0.0031	0.0529	0.0027
shake	0.0100	0.0100	0.2100	0.3300	0.0033	0.0000	0.0133	0.0533	0.0767	0.2933
jump	0.0000	0.0000	0.0000	0.1287	0.0673	0.0000	0.0029	0.0234	0.0380	0.7398
rest	0.0017	0.0040	0.0000	0.0000	0.0000	0.9799	0.0000	0.0002	0.0139	0.0002
rear-u	0.0000	0.0000	0.0000	0.0124	0.0000	0.0000	0.2531	0.1411	0.1037	0.4896
rear-w	0.0512	0.0248	0.0000	0.0085	0.0007	0.0000	0.0067	0.5358	0.3141	0.0582
sniff	0.0679	0.1458	0.0224	0.0025	0.0003	0.0018	0.0087	0.0251	0.6772	0.0482
walk	0.0042	0.0044	0.0024	0.0032	0.0000	0.0000	0.0010	0.0016	0.1287	0.8545

Figure 6.23: Confusion matrix of the multi-class SVM classification results on the Noldus ABR dataset.

Note that even though the overall accuracy of the multi-class SVM is slightly higher than the root model and the HCRF model (see Table 6.9), its average precision is much lower than that of the other models. This is because classification result on the four actions with fewer training samples (”shake”, ”jump”, ”rear-u” and ”rear-w”) are all much worse than the other two models. This result shows that the max-margin approach is more sensitive to the distribution of training data than the maximum likelihood approach. This is because the max-margin approach determines the hyperplane only from a few samples (support vectors), whereas the maximum likelihood summarizes the likelihood of all training samples.

Part Labels Evaluation

We evaluate the performance of the Part Labels method on the ABR Noldus dataset. Table 6.9 is a performance comparison of the root model, the HCRF model, multi-class SVM and the Part Labels model. Figure 6.24 is the confusion matrix of the Part Labels method classification result on the target classes.

	root model	HCRF	multi-class SVM	Part Labels
Classification actions	0.7388	0.7322	0.7535	0.7158
Target actions	0.7646	0.7567	0.7786	0.7610
Average precision	0.6741	0.6854	0.6077	0.6731

Table 6.9: Comparison of the root model, HCRF, multi-class SVM and Part Labels.

The overall performance of these models are comparable, this is still because the part labels are learned from the local patch features which are the same as the global features. But we notice that the Part Labels method performs slightly worse than the other two models. This is because in this case, the HCRF model parameters are not trained sufficiently to be able

	drink	eat	groom	shake	jump	rest	rear-u	rear-w	sniff	walk
drink	0.7459	0.0582	0.0426	0.0000	0.0000	0.0025	0.0000	0.0000	0.1508	0.0000
eat	0.0570	0.7249	0.0245	0.0000	0.0000	0.0015	0.0002	0.0160	0.1751	0.0008
groom	0.0066	0.1002	0.7940	0.0028	0.0014	0.0083	0.0030	0.0052	0.0780	0.0005
shake	0.0267	0.0067	0.1200	0.5133	0.0567	0.0033	0.0133	0.0000	0.1267	0.1333
jump	0.0000	0.0000	0.0058	0.0936	0.4591	0.0000	0.0029	0.0526	0.1170	0.2690
rest	0.0034	0.0110	0.0089	0.0004	0.0004	0.9610	0.0000	0.0011	0.0139	0.0000
rear-u	0.0000	0.0000	0.0000	0.0041	0.0000	0.0000	0.4606	0.1245	0.1535	0.2573
rear-w	0.0345	0.0030	0.0000	0.0182	0.0041	0.0133	0.0423	0.6110	0.2366	0.0371
sniff	0.0684	0.1003	0.0286	0.0066	0.0040	0.0117	0.0179	0.0555	0.6671	0.0399
walk	0.0014	0.0002	0.0018	0.0042	0.0269	0.0040	0.0140	0.0060	0.1471	0.7944

Figure 6.24: Confusion matrix of Part Labels method classification results on the Noldus ABR dataset.

to assign the best part labels with the highest probability. Therefore, using a maximization approach can not select the set of part labels descriptive for an image. There is no clear patterns among the part labels it found. It has been proved by the fact that the performance of the HCRF model is worse than the performance of the root model. If the model parameters are well trained, the HCRF model should at least be equal or better than the root model, because the HCRF model contains the root model.

The HCRF model can ease the problem with the poorly trained model parameter by summing over all assignments of part labels so that it does not omit any possible correct assignments. But the Part Labels method only uses the best assignment of the part labels. When the model parameters are not well trained, we can not be confident with the descriptive power of even the most likely part labels it found. Therefore, in this case, the learned part labels do not provide extra meaningful information, but mess up with the global feature and drag down the overall performance. This result demonstrates that the performance of the Part Labels model heavily depends on the training status of the model parameters. When the model parameters are not trained sufficiently, the HCRF model is more robust than the Part Labels model.

6.3 Discussion

In this chapter we have evaluated the performance of the root model, HCRF, MMHCRF and the Part Labels model on two datasets. The Weizmann dataset is a benchmark dataset on human actions recorded under laboratory settings. The Noldus ABR dataset is a private dataset to train and evaluate commercial behaviour recognition tools on rodent behaviours.

In the experiments, we have found the root model to be an efficient and powerful method. The root model only uses the global feature and treats the features as independent with each other. It can reach the global optimum because it does not use the hidden part labels. The performance of this model is comparable with other more complex models. But it does not

model the temporal structure explicitly. Instead, it adopts the simple concept of projecting the action patterns over time into a 2D plane. As a consequence, it can not distinguish between similar frames in different actions. In addition, it prefers the actions with more variations over the actions with less variations. The distribution of the dataset will influence the classification performance, because insufficient training samples lead to less variations. The actions with big inter-class variations are easily mixed up with other actions.

HCRF models an image as a constellation of part labels, and combines the local features with the global feature. It models the spatial structure among the local patch features. But its performance is not significantly better than the root model. One reason for this is that in the experiments, the local features are simply part of the global features, thus their discriminative power is overlapping with each other. Another reason is that the local patch features in 2D space are not informative enough for action recognition. The drawback of the HCRF model is that it only models the spatial structure among patches, but not the temporal structure across frames. The spatial structure of an action over time can change frame-per-frame. The model be confused if it only models the spatial structure. Therefore the HCRF model works best with the actions with consistent spatial structures over time.

MMHCRF also models the spatial structure of an image, but trains the model parameter with a max-margin criteria. It also learns the hidden part labels in a maximization approach, which only uses the best assignment of part labels in every iteration. This approach causes MMHCRF to be easily stuck on a local optimum and thus heavily relies on the initialization of the model parameter. The max-margin approach has been proved to be successful on action classification, but a good parameter setting is important for its performance. Furthermore, the max-margin approach handles unbalanced data poorly. MMHCRF is sensitive to the choice of the model parameter initialization method and parameter settings. It is less predictable and less robust than HCRF.

The Part Labels method is a hybrid of the root model and the HCRF model. It uses the model parameter trained by HCRF to find the best assignment of part labels for each image. It uses these part labels as a new set of local features and combines them with the global feature. It trains the new feature vector in the same way as the root model. The performance of this model depends on the status of the model parameter. If the model parameter is well trained, its learned part labels can be descriptive enough for the image, thus it can improve the performance based on the root model. If the model parameter is not trained sufficiently and the learned part labels can not serve as a good set of local features, its performance might be worse than the root model. Therefore, the Part Labels method can be an option when we are confident that the model parameters are sufficiently trained.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this project, we have implemented the root model, HCRF and MMHCRF for action recognition. We have also proposed a Part Labels method based on HCRF and the root model. We have evaluated these four models with a benchmark dataset for human actions and a private dataset on rodent behaviours.

The root model only uses the global feature and treats all features as independent of each other. It trains these features with an iterative gradient ascent method. It does not use hidden part labels, therefore it does not need to solve the inference problem during training. This makes its training process much more efficient than the other models. In addition, its iterative training process can reach the global optimum, because its objective function is convex.

The root model is a simple but powerful method. Its classification performance is comparable, if not better, than other more complex models. But this model has its limitations. It can not distinguish similar frames in different actions. It prefers actions with more variations than actions with less variations, because the more complex actions involve a larger set of features. If the inter-class variation within an action is too big, it can easily mix it up with other actions. The distribution of the training data also influences the classification result, because the root model is not able to capture enough variations for actions without enough training samples. Therefore this method is suitable for simpler datasets whose inter-class variation is not too big and intra-class variation is not too small.

The HCRF model uses both the local patch features and the global feature, and combines them under the unified framework of HCRF. It models an image as a constellation of parts conditioned on their local patch features. And it does not require the conditional independence assumption among the local patch features. HCRF models the spatial structure among the patches by using the structural hidden part labels. Each part label represents a motion pattern of a body part. Certain part labels have constraints between them. These constraints form a graph structure among the part labels. The HCRF model is trained with a maximum likelihood criteria, which tries to maximize the conditional log likelihood of all training samples. This involves solving an inference problem which needs to sum over all

possible assignments of part labels to the patches. The HCRF model is also trained with the gradient ascent method, but it can only reach a local optimum because the introduction of hidden part labels makes its objective function not convex.

The HCRF model is a robust method whose performance is stable and predictable. But its performance is just comparable to the performance of the root model in our experiments. One of the reasons of this result is that the local features and the global feature it uses are the same type of feature. Thus their discriminative power is overlapping with each other. Another reason is that the local patch features in 2D space is not informative enough for challenging tasks like action recognition. It only models the spatial structure among the local patches, and neglects the temporal structure across frames. But the spatial structure of an action could change from frame to frame. The model can be confused if the spatial structure is too complex and changes too often. Therefore, this HCRF model is suited better for actions with simpler spatial structures which is consistent over time. Nonetheless, HCRF is a robust model with much potential.

The MMHCRF model extends HCRF by training the model parameter an the max-margin criteria. It tries to set the model parameter in the way that the margin between the score of the correct label and the scores of the other labels is maximized. This criteria is the same as a standard multi-class SVM. The difference between MMHCRF and the multi-class SVM is that MMHCRF has hidden part labels in its model. These hidden part labels cause the optimization problem of MMHCRF to become not convex, thus it can only get a local optimal solution. MMHCRF searches for the local optimum in iterations. In every iteration, it first finds the best assignment of part labels with a maximization approach; next it uses these learned part labels to optimize the model parameter.

We have not been able to obtain a satisfactory result with MMHCRF. The maximization approach it adopted causes MMHCRF to be heavily dependent on the model parameter initialization method, thus less predictable and less robust than the HCRF model. We have also evaluated the max-margin approach using a standard multi-class SVM. We have found that it can obtain satisfactory results on both datasets, hence it proved the strength of the max-margin approach. But its performance is sensitive to the parameter settings and the dataset distribution. For this reason, we do not recommend MMHCRF for action recognition, but the standard multi-class SVM can be used when the dataset distribution is balanced.

The Part Labels model is a hybrid of the root model and the HCRF model. It finds the best assignment of part labels for each image using the model parameter trained by the HCRF model to. It uses these part labels as the local features of this image and combines them with the global feature for classification. It trains these features in the same way with the root model. Essentially, it extends the root model by including the information about the part labels.

The performance of the Part Labels model is also comparable to the root model and the HCRF model. Its performance depends on how well the model parameter is trained. If the model parameter is well trained, its learned part labels can be descriptive enough for the image, thus it can improve the performance based on the root model. If the model parameter is not trained sufficiently and the learned part labels can not serve as a good set of local features, its performance might even be worse than the root model. Therefore when the model parameters are not well trained, the HCRF model is more robust than the Part Labels

model.

In conclusion, we have found that the performance of simpler models such as the root model and the multi-class SVM, is comparable to the more complex models, such as HCRF. This is because in our work HCRF only models the spatial structure, and neglects the temporal structure over frames. It is sufficient to only model the spatial structure for tasks such as object recognition. But for challenging tasks such as action recognition, the spatial structure changes over time and becomes too complex to model. Thus HCRF is not be able to be more discriminative than the other simpler models. Furthermore, to exploit the potential of this model, the global and local features should be different from each other, thus their combination could be more powerful.

For Noldus ABR project, a clear tendency from all our experiments is that the short lasting actions have low classification accuracy and drag down the overall performance. This problem should be solved first, because inaccurate features would influence the output of any classification method. One possible solution is to omit the features with sliding temporal windows and use HCRF to model the temporal structure across frames. The nodes of the HCRF model can correspond to different stages of an action. In this case, HCRF is better than other generative models like HMM, because it does not assume the conditional independence of the features across frames.

In action recognition, the choice of features and the classification method should match with each other. HCRF is a structural modelling method. But the Noldus ABR feature is not designed for structural modelling in the first place. Information such as the patch locations, is important for the modelling, but not explicitly available from the features. We have to manually define a rigid graph structure based on the relative locations of the body parts. In addition, the feature lengths of different patches are different. For example, the head of a rat has three nodes, whereas the middle body only has one. This makes it difficult to model the body parts as a whole. Though we have made our efforts to fit this set of features into the HCRF model, it has not improved the performance comparing to other simpler models. It is recommended that the fellow researchers either modify the current set of features, or use direct classification models such as the root model, because this is the type of classifiers the Noldus ABR feature is designed for.

7.2 Future Work

A natural extension of our work is to include the temporal information in the HCRF framework. This could be done in two ways. The first approach is to include the temporal information in spatio-temporal features, such as 3D SIFT [17] and 3D HOG [31]. This type of features treat a video as a space-time cube and calculate features around interest points in 3D space. HCRF could model the structure among these interest points in 3D space. The second approach is to directly model the temporal structure among frames. HCRF can use the hidden variables to model different phases in the performance of an action. HCRF still has much potential to be developed for action recognition.

Appendix A

HCRF Inference Pseudo Code

Algorithm 1: Parameter Estimation

input : $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$,MaxIteration,stepSize
output: θ

initialize $\theta = \{\alpha, \beta, \gamma, \eta\}$ randomly; **while** *iteration* $<$ *MaxIteration* **do**
  $gradeint = \mathbf{0}$;
  for $t = 1 \cdots T$ **do**
    `// calculate beliefs`
    foreach $y \in \mathcal{Y}, j \in V, a \in \mathcal{H}$ **do**
      $P(h_j = a \mid y, \mathbf{x}_t, \theta) =$ `BeliefPropagation`$(y, j, a, \mathbf{x}_t, \theta)$;
    end
    foreach $y \in \mathcal{Y}, (j,k) \in E, a \in \mathcal{H}, b \in \mathcal{H}$ **do**
      $P(h_j = a, h_k = b \mid y, \mathbf{x}_t, \theta) =$ `BeliefPropagation` $(y, j, k, a, b, \mathbf{x}_t, \theta)$;
    end
    foreach $y \in \mathcal{Y}$ **do**
      $Z(y \mid \mathbf{x}_t, \theta) =$ `BeliefPropagation`$(\mathbf{x}_t, \theta)$;
    end
    `// calculate gradients for sample` t
    $gradeint_t = \mathbf{0}$;
    foreach $j \in V, a \in \mathcal{H}$ **do**
      $gradeint_t(\alpha) += P(h_j = a \mid y_t, \mathbf{x}_t, \theta) \cdot \phi(x_{t,j}, h_j)$;
      $gradeint_t(\beta) += P(h_j = a \mid y_t, \mathbf{x}_t, \theta) \cdot) \varphi(y_t, h_j)$;
    end
    foreach $y \in \mathcal{Y}, j \in V, a \in \mathcal{H}$ **do**
      $gradeint_t(\alpha) -= P(h_j = a \mid y, \mathbf{x}_t, \theta) \cdot \frac{Z(y \mid \mathbf{x}_t, \theta)}{\sum_{y'} Z(y' \mid \mathbf{x}_t, \theta)} \cdot \phi(x_{t,j}, h_j)$;
      $gradeint_t(\beta) -= P(h_j = a \mid y, \mathbf{x}_t, \theta) \cdot \frac{Z(y \mid \mathbf{x}_t, \theta)}{\sum_{y'} Z(y' \mid \mathbf{x}_t, \theta)} \cdot \varphi(y, h_j)$;
    end
    foreach $(j,k) \in E, a \in \mathcal{H}, b \in \mathcal{H}$ **do**
      $gradeint_t(\gamma) += P(h_j = a, h_k = b \mid y_t, \mathbf{x}_t, \theta) \cdot \psi(y_t, h_j, h_k)$;
    end
    foreach $y \in \mathcal{Y}, (j,k) \in E, a \in \mathcal{H}, b \in \mathcal{H}$ **do**
      $gradeint_t(\gamma) -= P(h_j = a, h_k = b \mid y, \mathbf{x}_t, \theta) \cdot \frac{Z(y \mid \mathbf{x}_t, \theta)}{\sum_{y'} Z(y' \mid \mathbf{x}_t, \theta)} \cdot \psi(y, h_j, h_k)$;
    end
    $gradeint_t(\eta) += \omega(y_t, x_{t,0})$;
    foreach $y \in \mathcal{Y}$ **do**
      $gradeint_t(\eta) -= P(y \mid \mathbf{x}_t, \theta) \cdot \omega(y, x_{t,0})$;
      $gradeint += gradeint_t$;
    end
  end
  `// L2-regularization`
  $gradient = gradient - \frac{\theta}{\sigma^2}$;
  `// update` θ `with gradient`
  $\theta = \theta + stepSize * gradient$;
end
return θ;

Appendix B

Proof of Dual Transformation

In this appendix, we will proof why the primal problem:

$$\begin{aligned} &\min_{\theta,\xi} \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t \\ &\text{s.t. } \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y}, y) - \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y_t}, y_t) \leqslant \xi_t - \delta(y, y_t), \forall t, \forall y. \end{aligned} \tag{B.1}$$

can be transformed into the dual problem:

$$\begin{aligned} &\max_{\alpha} \sum_{t=1}^{T}\sum_{y}\alpha_{t,y}\delta(y, y_t) - \frac{1}{2}||\sum_{t=1}^{T}\sum_{y}\alpha_{t,y}\Psi(\mathbf{x}_t, y)||^2 \\ &\text{s.t. } \sum_{y}\alpha_{t,y} = C, \forall t \\ &\alpha_{t,y} \geqslant 0, \forall t, \forall y, \end{aligned} \tag{B.2}$$

where $\Psi(\mathbf{x}_t, y) = \Phi(\mathbf{x}_t, \mathbf{h}_{t,y}, y) - \Phi(\mathbf{x}_t, \mathbf{h}_{t,y_t}, y_t)$.

First let us define the primal form:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\text{s.t. } C_k(\mathbf{x}) \geq 0, \forall k \\ &C_t(\mathbf{x}) - S_t = 0, \forall t \\ &S_t \geq 0, \forall t, \end{aligned} \tag{B.3}$$

and the dual form:

$$\begin{aligned} &\max_{\mathbf{x},\mathbf{S},\alpha} f(\mathbf{x}) - \sum_{k}\alpha_k C_k(\mathbf{x}) \\ &\text{s.t. } \nabla f(\mathbf{x}) - \sum_{k}\alpha_k \nabla C_k(\mathbf{x}) = 0. \end{aligned} \tag{B.4}$$

Every primal problem in the form of Eq.(B.3) can be converted into a dual problem in the form of Eq.(B.4) [32].

The problem in Eq.(B.1) is in the primal form of Eq.(B.3) if we consider:

$$\begin{aligned}
&\mathbf{x} = [\theta, \xi] \\
&f(\mathbf{x}) = \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t \\
&C_k(\mathbf{x}) = -(\theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y}, y) - \theta^\intercal \cdot \Phi(\mathbf{x}_t, \mathbf{h}_{t,y_t}, y_t)) + \xi_t - \delta(y, y_t) \\
&\qquad = -\theta^\intercal \cdot \Psi(\mathbf{x}_t, y) + \xi_t - \delta(y, y_t).
\end{aligned}$$

Note that the index transformation is $k = (t-1) \times T + y$, where T is the total number of training samples.

If we take derivative of $f(\mathbf{x})$ with respect to θ and ξ:

$$\begin{aligned}
\nabla_\theta f(\mathbf{x}) &= \theta; \\
\nabla_\xi f(\mathbf{x}) &= C;
\end{aligned} \tag{B.5}$$

We also take derivative of $C_k(\mathbf{x})$ with respect to θ and ξ:

$$\begin{aligned}
\nabla_\theta C_k(\mathbf{x}) &= -\Psi(\mathbf{x}_i, y); \\
\nabla_\xi C_k(\mathbf{x}) &= \mathbf{1}.
\end{aligned} \tag{B.6}$$

Therefore the constraints of the dual form can be written as:

$$\nabla_\theta f(\mathbf{x}) - \sum_k \alpha_k \nabla_\theta C_k(\mathbf{x}) = \theta - \sum_k \alpha_k(-\Psi(\mathbf{x}_i, y)) = 0 \tag{B.7}$$

$$\nabla_\xi f(\mathbf{x}) - \sum_k \alpha_k \nabla_\xi C_k(\mathbf{x}) = C - \sum_t \sum_y \alpha_{i,y} = 0 \tag{B.8}$$

From Eq.(B.7), we get:

$$\theta = -\sum_t \sum_y \alpha_{t,y} \Psi(\mathbf{x}_t, y) \tag{B.9}$$

From Eq.(B.8), we get the constraints for the dual problem:

$$\sum_t \sum_y \alpha_{t,y} = C \tag{B.10}$$

Next we address the objective function. We can get:

$$
\begin{aligned}
f(\mathbf{x}) - \sum_k \alpha_k C_k(\mathbf{x}) &= f(\mathbf{x}) - \sum_t \sum_y \alpha_k C_k(\mathbf{x}) \\
&= \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t - \sum_t \sum_y \alpha_{t,y} C_k(\mathbf{x}) \\
&= \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t - \sum_t \sum_y \alpha_{t,y}\left(-\theta^\intercal \cdot \Psi(\mathbf{x}_t, y) + \xi_t - \delta(y, y_t)\right) \\
&= \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t - \theta^\intercal \cdot \left(-\sum_t \sum_y \alpha_{t,y}\Psi(\mathbf{x}_t, y)\right) - \sum_t \left(\sum_y \alpha_{t,y}\right)\xi_t \\
&\quad + \sum_t \sum_y \alpha_{t,y}\delta(y, y_t) \\
&= \frac{1}{2}||\theta||^2 + C\sum_{t=1}^{T}\xi_t - \theta^\intercal \cdot \theta - C\sum_t \xi_t + \sum_t \sum_y \alpha_{t,y}\delta(y, y_t) \\
&= -\frac{1}{2}||\theta||^2 + \sum_t \sum_y \alpha_{t,y}\delta(y, y_t) \\
&= \sum_t \sum_y \alpha_{t,y}\delta(y, y_t) - \frac{1}{2}||\sum_t \sum_y \alpha_{t,y}\Psi(\mathbf{x}_t, y)||^2
\end{aligned}
\tag{B.11}
$$

This completes the derivation.

Appendix C

MMHCRF Training Pseudo Code

Algorithm 2: MMHCRF Training

input : $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$,MaxIteration
output: θ

initialize $\alpha = \{\alpha_{1,1} \cdots \alpha_{1,|\mathcal{Y}|} \cdots \alpha_{T,1} \cdots \alpha_{T,|\mathcal{Y}|}\}$ randomly;
initialize θ randomly; **while** *iteration* $<$ *MaxIteration* **do**
  for $t = 1 \cdots T$ **do**
    `// calculate optimal hidden part labels`
    foreach $y \in \mathcal{Y}, j \in V, a \in \mathcal{H}$ **do**
      $P(h_j = a \mid y, \mathbf{x}_t, \theta) =$ `BeliefPropagation`$(y, j, a, \mathbf{x}_t, \theta)$;
    end
    foreach $y \in \mathcal{Y}, j \in V$ **do**
      $\mathbf{h}_{t,y,j} = \arg\max_{a \in \mathcal{H}} P(h_j = a \mid y, \mathbf{x}_t, \theta)$;
    end
    `// solve optimization problem for sample` t
    foreach $y \in \mathcal{Y}$ **do**
      $\Psi(\mathbf{x}_t, y) = \Phi(\mathbf{x}_t, \mathbf{h}_{t,y}, y) - \Phi(\mathbf{x}_t, \mathbf{h}_{t,y_t}, y_t)$;
    end
    $\{\alpha_{t,y} : \forall y\} =$ `QuadraticProgram`$($*Optimization problem in* (4.15)$)$;
  end
  `// recover` θ `from` α
  $\theta = -\sum_{t=1}^{T} \sum_y \alpha_{t,y} \Psi(\mathbf{x}_t, y)$;
end
return θ;

Bibliography

[1] R. Poppe, "A survey on vision-based human action recognition," *Image Vision Comput.*, vol. 28, pp. 976–990, June 2010.

[2] E. A. v. Dam, J. E. v. d. Harst, C. F. J. t. Braak, R. A. J. Tegelenbosch, B. M. Spruijt, and L. P. J. J. Noldus, "An automated system for the recognition of various specific rat behaviors," *Journal of Neuroscience Methods*, 2013.

[3] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local svm approach," ICPR, (Washington, DC, USA), pp. 32–36, IEEE Computer Society, 2004.

[4] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," ICCV, (Washington, DC, USA), pp. 726–, IEEE Computer Society, 2003.

[5] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," ICCV, (Washington, DC, USA), pp. 1395–1402, IEEE Computer Society, 2005.

[6] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *ICCV*, pp. 1–8, 2007.

[7] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Second European Conference on Computational Learning Theory*, (London, UK, UK), pp. 23–37, Springer-Verlag, 1995.

[8] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Proceedings CVPR '92*, pp. 379–385, 1992.

[9] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," ICML, (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 2001.

[10] A. Quattoni, M. Collins, and T. Darrell, "Conditional random fields for object recognition," in *NIPS*, pp. 1097–1104, MIT Press, 2004.

[11] P. H. R.O. Duda and D. Stork, "Pattern classification," *J. Classif.*, 2001.

[12] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *CVPR 2008.*, pp. 1–8, 2008.

[13] Y. Wang and G. Mori, "Learning a discriminative hidden part model for human action recognition," in *NIPS*, pp. 1721–1728, 2008.

[14] Y. Wang and G. Mori, "Max-margin hidden conditional random fields for human action recognition," in *CVPR*, pp. 872–879, 2009.

[15] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," ICCV, (Washington, DC, USA), pp. 726–, IEEE Computer Society, 2003.

[16] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, Mar. 2002.

[17] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," MULTIMEDIA, (New York, NY, USA), pp. 357–360, ACM, 2007.

[18] S. Kumar and M. Hebert, "Discriminative random fields: A discriminative framework for contextual interaction in classification," ICCV, IEEE Computer Society, 2003.

[19] S. B. Wang, A. Quattoni, L.-P. Morency, and D. Demirdjian, "Hidden conditional random fields for gesture recognition," CVPR, (Washington, DC, USA), pp. 1521–1527, IEEE Computer Society, 2006.

[20] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," MIT Press, 2003.

[21] Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic versus max margin," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 7, pp. 1310–1323, 2011.

[22] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, pp. 1848–1852, Oct. 2007.

[23] J. Rennie, "On L2-norm Regularization and the Gaussian Prior," 2003.

[24] R. H. Byrd, J. Nocedal, and R. B. Schnabel, "Representations of quasi-newton matrices and their use in limited memory methods," *Math. Program.*, pp. 129–156, 1994.

[25] S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy, "Accelerated training of conditional random fields with stochastic gradient methods," ICML, (New York, NY, USA), pp. 969–976, ACM, 2006.

[26] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Exploring artificial intelligence in the new millennium," ch. Understanding belief propagation and its generalizations, pp. 239–269, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[27] J. Niebles and L. Fei-Fei, "A hierarchical model of shape and appearance for human action classification.," in *Proceedings of IEEE Intern. Conf. in Computer Vision and Pattern Recognition(CVPR).*, 2007.

[28] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," IJCAI, (San Francisco, CA, USA), pp. 674–679, Morgan Kaufmann Publishers Inc., 1981.

[29] F. Provost, "Machine learning from imbalanced data sets 101," *Proceedings of the AAAI-2000 Workshop on Imbalanced Data Sets*, 2000.

[30] D. J. Field, "Relations between the statistics of natural images and the response properties of cortical cells," *J. Opt. Soc. Am. A*, vol. 4, pp. 2379–2394, 1987.

[31] A. Kläser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *British Machine Vision Conference*, pp. 995–1004, sep 2008.

[32] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 2006.