OPTIMIZING NON-DECOMPOSABLE LOSS
FUNCTIONS IN
STRUCTURED PREDICTION

by

Mani Ranjbar
M.Sc., Sharif University of Technology, 2007
B.Sc., Sharif University of Technology, 2005

a THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

© Mani Ranjbar 2013
SIMON FRASER UNIVERSITY
Spring 2013

All rights reserved.
However, in accordance with the Copyright Act of Canada, this work may be reproduced without authorization under the conditions for “Fair Dealing.” Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.
APPROVAL

Name: Mani Ranjbar
Degree: Doctor of Philosophy
Title of Thesis: Optimizing Non-Decomposable Loss Functions in Structured Prediction

Examinig Committee: Dr. Brian Funt
Chair

_________________________________
Dr. Greg Mori, Co Senior Supervisor,
Associate Professor of Computing Science,
Simon Fraser University

_________________________________
Dr. Ze-Nian Li, Co Senior Supervisor,
Professor of Computing Science,
Simon Fraser University

_________________________________
Dr. Anoop Sarkar, SFU Examiner,
Associate Professor of Computing Science,
Simon Fraser University

_________________________________
Dr. Ben Taskar, External Examiner,
Associate Professor of Computer and Info. Science,
University of Pennsylvania

Date Approved: 10 April 2012
Partial Copyright Licence

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

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

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

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

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

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

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

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2011
Abstract

Learning functional dependencies (mapping) between arbitrary input and output spaces is one of the main challenges in computational intelligence. There have been two main threads in the literature for solving this problem – one focusing on designing more discriminative representation of the input and another one focusing on designing flexible mapping functions.

Interestingly, for many applications, the outputs follow a structure, which can be exploited to narrow down the space of possible (most likely) outputs and consequently boost the overall mapping performance. Applications with this property include object detection (computer vision), object category segmentation (computer vision), parsing (natural language processing), etc.

Current algorithms for learning the parameters of the model in structured prediction iteratively find the most confusing output configuration – the configuration that receives high score according to the model, but is very different from the ground truth output – and update the model parameters to suppress its score. Here, finding the most confusing configuration is the most expensive procedure in learning.

In this thesis we propose two algorithms for approximately finding the most confusing configuration when the model is a Markov network. Each algorithm works for a large group of non-decomposable performance measures that arise in many real-world applications. We first design a baseline that achieves state-of-the-art results in our main application of object category segmentation on person class by introducing fine and coarse clothing texture cues as a set of new features. Then, we propose our first algorithm that approximates the non-decomposable loss function in false positive and false negative space with a piecewise planar function and finds the most confusing output in each piece. Our second proposed algorithm decomposes the dual of the objective into a supermodular Markov random field and the loss function augmented with a linear term – both being efficient to optimize.

We empirically show the superiority of the two proposed algorithms over our baseline and another strong baseline – both used widely in the literature – on two main applications, object category segmentation (on PASCAL VOC 2009 and 2010 and H3D datasets) and action retrieval (on our nursing home dataset).
I would like to dedicate this thesis to my love and my wife, Zahra.
“There are two types of PhD thesis: perfect and submitted.”
Acknowledgments

I would like to thank my dear friends in Vision and Media Laboratory at Simon Fraser University – Mohammad Norouzi, Bahman Yari, Arash Vahdat, Yang Wang, Weilong Yang and Tian Lan for their assistance in various projects during my PhD.

My special thanks go to Dr. Greg Mori and Dr. Ze-Nian Li for their kindness, endless support and exceptional supervision.

Last but not least appreciations go to my wife and my parents for their supports and encouragements in every step of the way.
# Contents

Approval ii  
Partial Copyright License iii  
Abstract iv  
Dedication v  
Quotation vi  
Acknowledgments vii  
Contents viii  
List of Tables xi  
List of Figures xii  
Preface xv  

## 1 Introduction 1  
1.1 Model 3  
1.2 Accuracy measures 5  
1.3 Structural Support Vector Machine (SSVM) 6  
1.3.1 Support Vector Machine in brief 7  
1.3.2 Structural SVM 9  
1.4 Notation 12  
1.5 Contributions 12
4.5.1 Learning Method ........................................ 54
4.5.2 Mesh Creation ........................................ 55
4.5.3 Baseline Methods ..................................... 55
4.5.4 Object Category Segmentation ...................... 56
4.5.5 Pixels vs. Superpixels ................................. 57
4.5.6 Action Retrieval ....................................... 62

4.6 Summary .................................................... 68

5 Non-Decomposable Loss Optimization using Dual Decomposition 72
5.1 Overview and Related Works ............................ 73
5.2 Non-Decomposable Loss Optimization via Dual Decomposition ............. 75
5.3 Experimental Results .................................... 79
  5.3.1 Pixels vs. Superpixels ............................... 81
  5.3.2 Learning Method ..................................... 82
  5.3.3 Features ............................................. 82
  5.3.4 Pascal Visual Object Classes (VOC) 2009 and 2010 Segmentation
    Datasets ............................................. 83
  5.3.5 H3D Dataset ....................................... 85
  5.3.6 Convergence Analysis ............................... 85
5.4 Summary .................................................... 88

6 Conclusion .................................................. 89

Bibliography .................................................. 91
List of Tables

3.1 Errors as a function of different combinations of features and Grabcut errors with two different priors. Each '×' symbol indicates that the associated type of feature is used for learning the segmentation model. 37

3.2 Top segmentation accuracies (of 22 teams) on human class of PASCAL VOC 2009 challenge. The accuracies are reported as the intersection over union measure (Eq. 3.12). 39

4.1 Maximum achievable accuracy percentage in VOC 2009, VOC 2010 and H3D datasets due to superpixelization. 57

4.2 Background to foreground pixel ratio in Pascal VOC 2009 and 2010 60

4.3 Number of detected people in the training and test sets for each action 67

4.4 Precision percentage at $K = N_p$ 68

5.1 Background to foreground pixel ratio in Pascal VOC 2009 and 2010 datasets 83
List of Figures

1.1 Visualization of a Markov network. a) A simple Markov network on a grid. b) A sample Markov network for action retrieval. Each person in the frame is represented with a node in the network. Yellow and green edges represent the interaction between the actions of a person in two consecutive frames and the interaction between the action neighbouring people in the same frame, respectively. ................................. 4

1.2 Illustration of separable data. Both the dashed line and the solid line make no classification mistake, but the solid line has larger margin. ............... 8

2.1 Segmentation in different observation scales. a) original image, b) homogeneous region segmentation, c) object level segmentation. ....................... 18

3.1 Automatic human segmentation using the proposed model. (a) A test image. (b) Result of the full model (unary and pairwise cues). (c-e) Results of the models trained only using unary cues: coarse fold features, fine textures, and human shape prior respectively. ............................... 25

3.2 (a) Filters learned by a continuous CRBM from cropped clothing patches. (b) A sample of training cloth patches. ............................... 30

3.3 (a) Input image, (b) extracted superpixels, (c) edge-based prior, and (d) region-based prior. ............................... 33

3.4 A randomly chosen fold of results on the indoor dataset. From top: input images, Grabcut results based on the box prior, our results without the bounding box information. ............................... 38
3.5 Example human segmentations on VOC 09 test set. On the left the input image and the detected bounding boxes are shown and on the right the final segmentation is depicted. The final row illustrates some failures of our method.

4.1 Intersection over union loss surface in $FP$ and $FN$ space. a) Exact surface, b) a piecewise linear approximation with 40 subregions, c) a piecewise linear approximation with 15 subregions.

4.2 Visualization of the average root and part shapes in person category. Each row corresponds to shape models obtained from root and part appearance models of one object pose.

4.3 The process of computing top-down features for the head part. Instead of showing the center of the detected parts we depict the bounding box for visualization purposes in the second stage.

4.4 Intersection over union performance (%) comparison on VOC 2009 and 2010 datasets.

4.5 Segmentation for person category. Optimizing adjusted Hamming loss ("Hamming") against our proposed method. a) input image, b) segmentation considering adjusted Hamming loss ("Hamming"), c) our proposed method employing intersection over union. Intersection over union provides more true positives by possibly creating some false positives. Adjusted Hamming loss decreases false positive by sacrificing some true positives.

4.6 Intersection over union performance (%) comparison on H3D dataset.

4.7 Illustration of our model. A Markov random field with unary (red), intra-frame (green) and inter-frame (yellow) connections is used.

4.8 Illustration of the local spatio-temporal (LST) feature representation for describing a candidate region. $u$ is a vector of percentage of static foreground pixels, $v$ is a vector of percentage of moving foreground pixels. See text for details.

4.9 Visualization for some of the learned intra-frame (left) and inter-frame (right) interactions. Vertical labels are the query actions (walk (W), Stand(St), Sit(Si), Bend(B) and Fall(F)). The inter-frame interactions are asymmetric, which is shown as two weights one from query action to the other actions (left half) and from the other actions to the query action (right half).
4.10 Some segmentation results on Pascal VOC 2009 dataset. Each row corresponds to one object category. ................................. 70
4.11 Some segmentation results on H3D dataset. ................................. 71

5.1 Intersection over union performance (%) comparison on VOC 2009 and VOC 2010 datasets ............................................. 84
5.2 Intersection over union performance (%) comparison on H3D dataset .... 85
5.3 The improvement of the primal and the dual for solving the loss augmented inference versus the subgradient iterations. A typical case is shown. .... 86
5.4 The average percentage of disagreeing assignments versus the number of subgradient iterations. In average only 0.63% of assignments are disagreeing at the last iteration. ................................. 87
Preface

Here go all the interesting reasons why you decided to write this thesis.
Chapter 1

Introduction

Optimizing the parameters of complex models against non-decomposable loss performance measures is the goal of this thesis. Many real-world complex problems consist of numerous tightly coupled subproblems, in which the answer to a subproblem is affected by the results of its coupled subproblems. At the same time, evaluating the success rate in such problems requires simultaneous access to all the sub-results. This thesis is about superior ways of parameter tuning to perform better on such problems.

Humans can learn to perform such complicated tasks in such a fast pace that makes the task look easy. However, their precision on repeating the task is affected by many factors such as fatigue, anxiety, stress, anger, etc. On the other hand, machines can repeat the same task over and over again without getting bored or tired, but require a precise procedure to do the task. For simpler tasks humans can design a series of simple sub-tasks that if followed could accomplish the original large task. However, as the task gets more and more complicated demands for machines that can learn to perform the task increases. Like humans, we would like the machines to learn how to do the task by looking at an expert or by following a high level goal.

As a concrete example let’s focus on a classification problem – the task of figuring out which class an object belongs to. The classification problem has numerous real-world applications such as classifying someone’s blood type, classifying news, classifying an email to be a spam or not, classifying a hand-written digit, etc. In this thesis we are focused on problems involving classification of a group of objects with some relationship between their class labels.

One classification problem with this property is automated analysis of nursing home
footage – a video surveillance problem. The goal is to find actions of interest, for example someone falling down in long pre-recorded videos, in order to study the mechanism of the falls. One approach would be to have a person watching the video footage and label the falls. However, since the falls occur rarely this task becomes very boring and consequently may result in missed fall reports. As an alternative we can have an automated system that analyzes the video and retrieves the frames in which a fall has happened. Of course some falls might be harder to detect than the others due to occlusion or background clutter, and also some actions such as sitting might get confused as a fall. So there is a trade-off between what percentage of the actual falls the system can detect (recall) versus what percentage of the reported falls are real falls and not other actions (precision). Note also that the amount of time, $T$, that the user can spend to watch the result is limited. So a reasonable objective for the system is to maximize the number of real falls in the first $K$ returned results, where $K$ is the number of results that can be seen in $T$ time. This objective is called precision at $K$, which has been used widely for retrieval systems. Note that to evaluate this objective the label of all actions in the video along with system’s confidence on the label is required. We call this objective non-decomposable as opposed to objectives that could be computed without having access to the entire labeling at once, e.g. maximizing recall.

As in many other classification problems, there might be some information in the labels of the other objects that might help classifying the current object. In our previous fall retrieval system for example, detecting a fall might be very difficult due to occlusion or background clutter, but labeling neighbouring people as bending may suggest falling of the person, whom is being helped standing up. So, in many classification problems including action retrieval, object classification, object category segmentation, etc, having a model that can capture some interaction between the labels of objects is preferable.

This thesis is about tuning the parameters of such a model, when the task objective is non-decomposable. The previous alternatives are to ignore the interactions between the labels or to approximate the objective with a decomposable function, which simplifies parameter tuning. However, we show that the proposed approaches are superior to the previous alternatives in two computer vision applications – object category segmentation and action retrieval. In the rest of this chapter, we explain our model, describe the difference in accuracy measures, provide some background on our learning framework and express the contributions of the thesis.
CHAPTER 1. INTRODUCTION

1.1 Model

The choice of model is crucial since it defines which type of information could be employed to map the input to the output. In many classification applications, more than one object, region, action or activity is to be classified. The naive approach would be to classify each instance independently. However, classifying the instances all together simultaneously would allow considering the interaction between the instances. For example, deciding whether a person has fallen down or not might be challenging by itself due to occlusion. But, adding the information that there are other people bending close to this person, perhaps to help this person stand up, could make the decision much easier\(^1\).

One of the models that can benefit from interaction information, while considering individual information is the Markov network (Markov random field) [4]. A Markov network is a graphical model in which a set of random variables have a Markov property described by an undirected graph. Figure 1.1a depicts an example of a Markov network. The empty nodes correspond to the entities that need to be labeled, e.g., an image (for image classification), a pixel or a region (for object category or image segmentation), a person (for action or activity recognition). The shaded nodes correspond to their observations. The light gray connections between empty and shaded nodes hold the individual information and black edges carry the interaction information. A direct edge between two empty nodes models direct interaction between two entities, where as non-neighbour entities can interact indirectly through their neighbours. The classification problem given this model is basically finding the labeling of all variables (objects, image pixels, etc) which agrees the most with all individual and interaction (pairwise) information. Figure 1.1b visualizes how our nursing home fall retrieval scenario can be modeled using such network. A person in the video is represented as a node in the graph and the interaction between a person’s action in consecutive frames as well as the interaction between neighbouring people in the same frame are captured in yellow and green edges respectively.

In such network, the Markov property defines the conditional probability over the entities (variables) as follows.

- Any two non-adjacent variables are conditionally independent given all other variables

\(^1\)While being useful, trusting the interaction information too much could impair the decision as well. So, making right decisions requires setting the contribution of the interaction information versus the individual information precisely.
Figure 1.1: Visualization of a Markov network. a) A simple Markov network on a grid. b) A sample Markov network for action retrieval. Each person in the frame is represented with a node in the network. Yellow and green edges represent the interaction between the actions of a person in two consecutive frames and the interaction between the action neighbouring people in the same frame, respectively.

- Any variable is conditionally independent of all other variables given its neighbours
- Any two subsets \((A, B)\) of variables are conditionally independent given a separating subset \(C\), if all paths from a variable in \(A\) to a variable in \(B\) passes through \(C\)

The joint probability in this model is defined as

\[
p(x, y) = \frac{1}{Z} \prod_i \psi^{(u)}(x_i, y_i) \prod_{i} \prod_{j \in \mathcal{N}_i} \psi^{(p)}(y_i, y_j),
\]

\[
Z = \sum_X \sum_Y \prod_i \psi^{(u)}(x_i, y_i) \prod_{i} \prod_{j \in \mathcal{N}_i} \psi^{(p)}(y_i, y_j),
\]

where, \(Z\) is the partition function, \(\mathcal{N}_i\) is the set of all neighbors of the \(i^{th}\) node, \(\psi^{(u)}\) is the unary potential (individual information), \(\psi^{(p)}\) is the pairwise potential and \(X\) and \(Y\) are all possible input and label configurations, respectively. Note that computing the partition function is infeasible for most non-toy problems, because it involves iterating over all possible inputs and all possible labelings. For example in object category segmentation, the summation on \(X\) requires generating all possible images. Fortunately, in many problems (including the classification problem) the goal is to compute the labeling that maximizes the joint probability (the most likely labeling) also called maximum a posteriori or MAP for short. Solving the MAP problem does not require computing the partition function,
because $Z$ is just a constant.

In this thesis we assume that $\psi_{(u)}$ and $\psi_{(p)}$ are linear functions of the model parameters $\mathbf{w}$. Basically, they are defined as

$$
\psi_{(u)}(x_i, y_i) = \mathbf{w}_{(u)}^T \phi(x_i, y_i) \quad (1.3)
$$

$$
\psi_{(p)}(x_i, y_i) = \mathbf{w}_{(p)}^T \phi(x_i, y_i) \quad (1.4)
$$

where, $\mathbf{w} = [\mathbf{w}_{(u)}; \mathbf{w}_{(p)}]$ is the concatenation of the two. The goal of the thesis is to learn this $\mathbf{w}$ in a Markov network when the accuracy measure is non-decomposable. Next, we explain how the accuracy is measured and describe the difference between decomposable and non-decomposable accuracy measures.

### 1.2 Accuracy measures

Having defined the model formally, we need a performance measure to be able to find useful parameters and compare different approaches. There are two types of performance measure in general – qualitative and quantitative. Since the end-user of the results might be humans, one way of measuring the accuracy would be to ask people to rank the results of different parameter settings or methods. However, this measure suffers from two problems. First, it is expensive. People should spend hours to look at the results, compare them and rank them. Second, the result might be used as an intermediate task in a large system. Here, the accuracy should be measured based on the requirement of the system, which might not be in agreement with human preferences. To cope with these problems, quantitative results are employed.

Quantitative performance measures are defined based on the final goal of the task. For example, counting the number of correctly labeled inputs is a typical performance measure for many problems. Depending on how the performance measure is defined, finding the good set of parameters that achieve high performance could be computationally easy or hard.

In this thesis, we focus on a group of performance measures that comes naturally for many applications, but make finding the good set of parameters of Markov networks difficult. The shared characteristic of these performance measures are their non-decomposibility. Simultaneous access to all output labelings is required to be able to compute the performance as opposed to decomposable performance measures that are computable even if outputs are
accessed individually. Some examples of non-decomposable performance measures include \( F_\beta \) score (natural language processing), intersection over union (object category segmentation), Precision/Recall at \( k \) (search engines), average precision (AP) and ROC area (binary classifiers).

As an example, let us compare decomposable Hamming performance measure and non-decomposable F-measure on a binary output problem. Hamming performance measure is defined as

\[
Perf_{\text{Hamming}}(y, y') = \sum_{i=1}^{N} I[y_i = y'_i]
\]  

(1.5)

where, \( I[] \) is the indicator function and \( y \) and \( y' \) correspond to the ground truth labeling and the output labeling, respectively. The F-measure is defined as

\[
Perf_{F}(y, y') = \frac{2 \sum_{i=1}^{N} I[y_i = y'_i = 1]}{2 \sum_{i=1}^{N} I[y_i = y'_i = 1] + \sum_{i=1}^{N} I[y_i = 1 \land y'_i = 0] + I[y_i = 0 \land y'_i = 1]}
\]  

(1.6)

The F-measure is essentially the harmonic mean of precision and recall, which provides a better balance between the contribution of the two classes in the overall accuracy, especially for the unbalanced classes. The Hamming performance is called decomposable, because this measure can be decomposed to summation of sub-measures each involving a single output and therefore does not require access to all outputs at once. However, computing the F-measure needs access to all outputs as it cannot be decomposed to sub-measures each involving only one output.

The reason why finding a good set of parameters for Markov network when the loss function is non-decomposable becomes difficult is discussed in detail in Sec1.3.2.

1.3 Structural Support Vector Machine (SSVM)

We start this section with a brief introduction to the support vector machine (SVM [17][12]), which is the base for structural SVM [1][78] – the learning strategy considered in this thesis. Then, some preliminaries about structural SVM and the challenges in real applications is provided. We conclude this section with the contributions of this thesis.
1.3.1 Support Vector Machine in brief

Many real-life applications are concerned with classifying objects into a distinct known set of groups – called a classification problem. The object is first described with some of its discriminative features represented as a vector. The features are then represented with numbers to be understandable for computers. The output, which is the index of the group that the object belongs to is also represented as a number. For simplicity let’s assume that there are only two groups represented as class 1 or -1 (binary classification). The binary classification problem can be formally written as finding a mapping between the input features of the object to the output class index, also called the discriminant function, \( y = h(x) \), where \( x \) the input vector, \( y \) is the output class index and \( h \) is the discriminant (mapping) function. In machine learning, we are interested in finding \( h \) given some training instances, including some feature vectors and the true class labels. The function \( h \) could have several forms, but in the simple case the input is mapped to the output using a linear function \(^2\) as, \( y = w^T x \). The vector \( w \) represents a hyperplane in feature space that separates the output labels +1 and -1. Given the input instances, the learning is concerned with finding the \( w \) that makes few mistakes on the training instances and the unseen instances to come.

When the training instances are linearly separable, meaning that a hyperplane exists that can discriminate all input examples without any mistake, it is very likely that infinitely many hyperplanes achieve zero mistakes (see Fig. 1.2 for illustration). The main idea in support vector machines is to choose the hyperplane that has the largest margin. The margin is defined as the minimum distance between the hyperplane and all the points in the feature space.

The hyperplane with larger margin provides greater stability, because a small change in the features’ values is less likely to affect the classification result. Of course in real applications the input data are rarely linearly separable. Therefore, maximizing the margin is not the only concern and minimizing the number of mistakes should also be taken into account. It turns out that maximizing the margin is achieved by minimizing \( \|w\|^2 \) [17][12].

\(^2\)Note that \( x \) could be replaced by \( x' = \psi(x) \), where \( \psi \) is an arbitrary function, without affecting the following derivations.
So, the overall training can be expressed as

$$
\min_{w, \xi, b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i
$$

s.t. $y_i(w^T x_i - b) \geq 1 - \xi_i$, $\xi_i \geq 0$, $i = 1 \ldots N$

where, $\xi_i$ represents the degree of misclassification of the $i^{th}$ instance, $N$ is the number of instances, $b$ is the intercept of the hyperplane and $C$ is the contribution of maximizing the margin versus making fewer mistakes. The loss function that measures the amount of the mistakes is called the Hinge loss, and is zero as long as a point is outside of the margin in the correct side of the hyperplane and is linearly proportional to the distance of the point to the margin otherwise. Note that the total loss is the summation of the loss for individual points, or in other words, the loss function is decomposable. Furthermore, the output labels are assumed to be a function of the input features and independent of the other labels. This assumption is clearly unrealistic for problems involving interaction between the output labels, such as object category segmentation, object detection, action
recognition, parsing, etc. For example, knowing that a pixel in the image belongs to a car increases the chance of its neighbouring pixels to belong to the same car. Or knowing the role of a word in a sentence can help disambiguate the role of some other words in the sentence. These problems are called structured output problems, since the outputs are not independent, but structured. The aforementioned SVM formulation cannot benefit from the underlying structure of the problem and simply assumes independence between the outputs. To overcome this limitation, a new formulation with similar flavour called Structural SVM has been proposed. The next section briefly introduces this formulation and explains the pros and cons of the new formulation.

1.3.2 Structural SVM

The Support vector machine formulation [17][12] has been one of the most successful learning strategies for supervised classification due to good generalization and efficient computation. Maximizing the margin plays a significant role to achieve great generalization by keeping the decision boundary as far as possible from both classes, and the convexity of the objective with respect to the weight vector results in efficient learning for tens of thousands of training points. However, the output is restricted to a scalar from the set of possible labels or class identifiers. Structural SVM [1][78] tries to generalize the SVM formulation for problems involving structured objects such as sequences, strings, trees, graphs, etc. as their output.

For example, consider the problem of object classification in natural images. The task is to determine the existence of any instance from any of the $K$ possible object classes. So, the output is a vector of boolean variables each corresponding to an object class where 1 represents the existence of an instance and 0 represents the absence. One could naively solve this problem by treating the object classes independently and solve $K$ binary classification problems independently. However, some object classes are very likely to co-occur (e.g. chair and table) and some object classes are seen rarely together (e.g. boat and cow) in an image. Simply, there is a structure in the classification output.

The learning problem in this case involves finding a mapping from the input $x$ to the output object $y$, given some $(x, y)$ pairs. One way of finding such mapping is by learning a discriminant function $F: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$. The discriminant function assigns a real value to each $(x, y)$ pair corresponding to the compatibility of $x$ and $y$ given the function parameters.

$^3$Note that $y$ is a structured object unlike the scalar output in SVM formulation.
The goal is to learn the function parameters such that the most compatible output for an input is the ground truth. Assuming a linear discriminant function with respect to the parameters, the discriminant function can be written as,

\[ F(x, y; w) = w^T \Psi(x, y) \] (1.8)

where, \( \Psi(x, y) \) is the combined feature of inputs and outputs and \( w \) represents the parameters (weights) of the discriminant function. The learning in Structural SVM is concerned about finding \( w \) that satisfies

\[ w^T \Psi(x^{(i)}, y) \leq w^T \Psi(x^{(i)}, y^{(i)}), \quad y \in Y^{(i)} \] (1.9)

where \( y^{(i)} \) is the ground truth output for the \( i \)th input. Moreover, the weight vector \( w \) that maximizes the gap between the score of the ground truth output and the closest runner-up is preferable. This choice generalizes the maximum margin in the SVM formulation. The optimization that computes such \( w \), but with soft version of the constraints in Eq. 1.9 is as follows,

\[
\min_{\xi, w} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \\
\text{s.t. } \forall i, \forall y \in Y^{(i)} \setminus y^{(i)}: w^T \left( \Psi(x^{(i)}, y^{(i)}) - \Psi(x^{(i)}, y) \right) \geq 1 - \xi_i 
\] (1.10)

Note that in this formulation the margin violation penalty is independent of the similarity of the violating \( y \) to the ground truth. But, intuitively the margin violation for the vector \( y \) that is very different from the ground truth \( y^{(i)} \) should be penalized more than the vector \( y \) similar to \( y^{(i)} \). Given the loss function \( \Delta(y^{(i)}, y) \) that measures the dissimilarity of the two objects \( y^{(i)} \) and \( y \), two possible ways of modifying the penalty are as follows. The first option is to scale the slack variables based on the loss value [82],

\[
\min_{\xi, w} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \\
\text{s.t. } \forall i, \forall y \in Y^{(i)} \setminus y^{(i)}: w^T \left( \Psi(x^{(i)}, y^{(i)}) - \Psi(x^{(i)}, y) \right) \geq 1 - \frac{\xi_i}{\Delta(y^{(i)}, y)} 
\] (1.11)
The second option is to scale the margin based on the loss value \[76\],

\[
\min_{\xi, w} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i
\]

\[s.t. \quad \forall i, \forall y \in \mathcal{Y} \setminus \mathcal{y}^{(i)} : w^T \left( \Psi(x^{(i)}, y^{(i)}) - \Psi(x^{(i)}, y) \right) \geq \Delta(y^{(i)}, y) - \xi_i \tag{1.12}\]

Despite the theoretical advantages of slack rescaling over margin rescaling, the two formulations have shown to perform similarly in real applications \[82\]. Therefore throughout this thesis we consider the margin rescaling formulation for computational convenience.

Note that solving the optimization in Eq. 1.12 is challenging due to an enormous number of constraints. For a binary vector \(y\) of length \(n\) this optimization involves as many as \(2^n\) constraints. To solve this optimization Tsochantaridis et al. \[82\] proposed a cutting plane algorithm that requires only a polynomial number of constraints to converge to any \(\epsilon\) precision. The pseudocode of their algorithm is shown in Alg. 1.

**Algorithm 1** The algorithm for optimizing Eq. 1.12 taken from \[82\]-Alg. 1

1: Input: \((x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)}), C, \epsilon\)
2: \(S_i \leftarrow \emptyset\) for all \(i = 1, \ldots, N\)
3: repeat
4: for \(i = 1, \ldots, N\) do
5: \[H(y) = \Delta(y^{(i)}, y) - w^T \left( \Psi(x^{(i)}, y^{(i)}) - \Psi(x^{(i)}, y) \right)\]
6: where \(w = \sum_j \sum_{y' \in S_j} \alpha_{y'} \left( \Psi(x^{(i)}, y^{(i)}) - \Psi(x^{(i)}, y') \right)\)
7: compute \(\hat{y} = \arg \max_{y \in \mathcal{Y}} H(y)\)
8: compute \(\xi_i = \max \{0, \max_{y \in S_i} H(y)\}\)
9: if \(H(\hat{y}) > \xi_i + \epsilon\) then
10: \(S_i = S_i \cup \{\hat{y}\}\)
11: \(\alpha_S = \text{optimize dual over } S, S = \cup_i S_i\)
12: end if
13: end for
14: until no \(S_i\) has changed during iteration

The algorithm starts with an empty set of constraints. In each iteration, the most violated constraint – the one that has large scoring function value and large loss value is found. The most violated constraint is added to the set of constraints and the weight vector is updated by solving a quadratic program.

But is finding the most violated constraint (line 7) tractable?
The answer depends completely on the form of the scoring function and loss function. For example, in the case when maximizing the scoring function is tractable and the loss function decomposes as a part of the scoring function finding the most violated constraint is tractable. The other important question to ask when finding the most violated constraint is known to be hard is:

*Does an approximate solution for the most violated constraint have any value?*

Finley and Joachims [26] have studied this problem and showed that

1. There is a value in finding the most violated constraint approximately.
2. The solution of over-generating methods are more reliable than the under-generating methods solution.

The set of possible solutions in over-generating methods is a superset of the possible solutions in the original problem, whereas the set of solutions in the under-generating methods is a subset of the possible solutions for the original problem.

### 1.4 Notation

Throughout this thesis we use the following format. We write matrices with bold upper case letters (e.g. \( X \)), vectors with bold lower case letters (e.g. \( x \)) and scalars with normal lower case letters (e.g. \( x \)). In our notation, \( x_i \) represents the \( i^{th} \) column of matrix \( X \) and \( x_j \) represents the \( j^{th} \) element of vector \( x \). We overload the subscript notation (with additional brackets) to name vectors, matrices and functions, which define different aspects of the same concept. For example, we show the weight vector of unary terms with \( w_{(u)} \) and the weight vector of the pairwise terms as \( w_{(p)} \). Note that the bracket around the subscript clarifies the meaning. We also overload the superscript notation (with additional bracket) to index different vectors or matrices of possibly different sizes. For example to iterate over the matrix of images with different sizes.

### 1.5 Contributions

The candidate has been involved in numerous projects during his PhD study with various contributions. However, a subset of these projects that are closely related to each other and in which the candidate has contributed the most are explained in this thesis.
CHAPTER 1. INTRODUCTION

The three pieces of work, in which the candidate has contributed the most and appear in this thesis are as follows.

- In Chapter 3 we propose fine and coarse texture cues as novel features for human figure segmentation. We build a system using these features and employ a pairwise MRF as our model. We learn the parameters of the model using Structural SVM when the loss function is chosen to be Hamming loss. We show significant improvement by employing the proposed features on an indoor dataset and achieved state-of-the-art results on the PASCAL VOC 2009 dataset on person segmentation. The main problem with this system is the difference between training and testing loss functions. While the testing loss function is defined by user, the training loss function is decomposable Hamming loss to make the training tractable.

- The second contribution of this thesis is proposing an approach in Chapter 4 to allow optimizing against a large class of loss functions, those which are a function of false positive and false negative counts, when the model is a pairwise MRF. The proposed approach is based on approximating the loss surface in false negative and false positive space with a piecewise planar function. We then solve the modified optimization using LP relaxation. We show significant improvement over two strong baselines – one being the same model, but optimizing against decomposable Hamming loss and the other one being a decomposable model, but optimizing against the test loss function. This idea has been published in [67] and will appear in [66].

- In Chapter 5 we propose the third contribution of the thesis. We propose an alternative approach for optimizing against an even larger class of loss functions, for which finding the maximum of the loss function augmented with a linear term is tractable. Note that the loss functions covered in the previous proposed approach are a subset of the acceptable loss functions for this approach. This alternative is based on the idea of dual decomposition. While achieving comparable results to the previous approach of piecewise planar approximation, this approach is shown to be significantly faster. However, this approach is not parallelizable with the current implementation as opposed to the previous proposed approach. This work will appear in IEEE Conference in Computer Vision and Pattern Recognition (CVPR) 2012.

The list of the projects, in which the candidate has been involved, but won’t appear in this thesis are as follows.
• Proposing convolutional restricted Boltzman machines for feature learning in [61] in which the candidate has contributed in obtaining the results on INRIA dataset. This work has been used as a feature detector in our human figure segmentation project explained in Chapter 3. This work appeared in IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009.

• Proposing a unified pedestrian tracking system that learns the contribution of multiple cues jointly in a structured prediction framework published initially in [40] and later published in [41]. The candidate has contributed in many parts including the formulation, core optimization coding, and the result analysis. This work appeared in Canadian Conference on Computer and Robot Vision (CRV) 2010 and will appear in Computer Vision and Image Understanding journal.

• Proposing a discriminative key pose sequence model for recognizing human interactions presented in [84]. This work is also an example of structured prediction, which has been the main focus of the candidate. The contribution of the candidate in this work is mainly in formulation and implementation speed up. This work presented in IEEE International Workshop on Visual Surveillance (VS) 2011.
Chapter 2

Previous Work

In this chapter we review some of the previous works in the literature that are closely related to this thesis. Moreover, since most of the experiments in the thesis have been on object category segmentation, we will review alternative approaches for solving this problem later in this chapter.

2.1 Structured Prediction

Modeling dependencies between outputs while optimizing against a loss function has been a research topic for many years. Optimizing the expected loss in this scenario is a non-convex problem. Collins [16] has formulated structured prediction as a convex relaxation of the original non-convex problem, while optimizing against the zero-one loss. In this approach a perceptron-like update has been proposed to set the parameters of the model. It turns out that the most expensive step is to find the output that maximizes the scoring function, but is different from the ground-truth. However, given the choice of the loss function, this step is equivalent to the inference procedure and assumes to be tractable.

Altun et al. [1] employ the same formulations, but try to solve for the best parameters using quadratic programming by introducing a working set strategy. The most violated constraint is found in each iteration and added to the working set. The model parameters are then updated using the examples in the working set. The loss function is assumed to be zero-one loss similar to Collins [16] work.

Taskar et al. [78] proposed a similar formulation, but replaced the zero-one loss with a generic decomposable loss function as a scaling factor to the margin. Basically, examples
with larger loss are required to have larger margin than the examples with smaller loss. The model is assumed to be a Markov network, with unary and pairwise potentials. Unlike the previous formulations that required an exponential number of constraints, only a polynomial number of constraints is considered in this approach. To achieve this, the relaxation used by belief propagation algorithm [63] [93] is employed. Similar to the previous approaches, the most demanding step is to find the most violated constraint.

Tsochantaridis et al. [82] have proposed to use the cutting-plane algorithm and have proven relatively fast convergence despite dealing with an exponential number of constraints. Similar to [1] a working set strategy is employed, which provably achieves any fixed precision using only a polynomial size working set. In addition to margin rescaling formulation of Taskar et al. [78] the work of Tsochantaridis et al. [82] introduces slack rescaling that scales the slack variables based the examples loss. Despite theoretical advantages of slack rescaling over margin rescaling, the results are often comparable for both formulations in practice.

Teo et al. [80] presented a bundle method, which is basically the cutting-plane method stabilized with Moreau-Yosida regularizer and proved a tighter bound on the duality gap. Taskar et al. [79] solves the same problem using the extragradient method. Extragradient consists of a gradient descent followed by a projection to the feasible set. Shalev-Shwartz et al. [70] proposed Pegasos, which works solely in the primal space. Similar to [79], Pegasos consists of a gradient descent step followed by a projection step. The computational difficulty in all aforementioned structured prediction approaches is finding the subgradient, which requires solving the “most violated constraint” [82] or “loss augmented inference” [77]. It is shown that for decomposable performance measures learning is tractable when the model is a submodular MRF or a matching [78, 82, 79]. In contrast, in this thesis we focus on non-decomposable performance measures.

Joachims [39] proposed an approach to efficiently compute the most violated constraint for a large class of non-decomposable loss functions, a subset of those we consider in this paper. However, the underlying models were limited, and do not permit pairwise interactions between output labels. The method of Yue et al. [94] takes a similar approach to optimize against Mean Average Precision. Khanna et al. [13] present an algorithm in the same framework to optimize against normalized discounted cumulative gain (NDCG). Rather than solving a convex relaxation of the expected loss, McAllester et al. [56] proposed a perceptron-like training approach to directly optimize the original loss function, but still need to solve the loss augmented inference. For the problems in which the inference procedure is not
tractable, Finley et al. [26] compare under-generating and over-generating algorithms in structured prediction and conclude that “overgenerating methods [LP and graph cut] have theoretic advantages over undergenerating [LBP, greedy] methods”.

Three closely related pieces of work in the literature are the approaches of Meshi et al. [57], Komodakis [42] and Tarlow and Zemel [75]. The work by Meshi et al. [57] shares a similar formulation, and proposes to solve the loss augmented inference and parameter learning simultaneously. Our approach in Chapter 5 is similar in the sense that we work on the dual of the loss augmented inference. However, our goal is to optimize against non-decomposable losses, while [57] is only concerned with decomposable loss functions. Komodakis [42] addresses learning the parameters of a Markov network with higher order potential functions. The case of decomposable Hamming loss is developed in detail, and the applicability for general losses is alluded to. This thesis precisely shows how the parameters of a Markov network can be learned given non-decomposable loss functions that operate over all output variables. Tarlow and Zemel [75] also attempt to learn against complex losses, but with a very different approach. In that paper a message passing strategy is proposed and the results are shown on a subset of the PASCAL VOC images. Note that the ratio of positive to negative pixels is an order of magnitude larger in the selected subset, which makes the results incomparable with the results of this thesis.

### 2.2 Object Category Segmentation

Image segmentation in general is an ill-posed problem due to the scale of observation. Fig 2.1 shows two valid segmentations of an image, where the one in the center segments the image into homogeneous regions and right one segments the object. Although both results are valid, different applications may prefer one to the other. But because the ultimate goal of computer vision is scene understanding, the object level segmentation in Fig. 2.1c is more appealing.

Object level segmentation can be defined as a mapping from image pixels $X$ to object labels $y$, in which each pixel is assigned the label of the object category it belongs to. We assume there are only $K$ object categories and everything else is labeled as background (label 0). So we can write the object category segmentation as a function $S$,

$$ S : X \rightarrow y, X \in \{0, \ldots, 255\}^{3N}, y \in \{0, \ldots, K\}^N $$ (2.1)
where \( N \) is the number of pixels in the image and the input image is assumed to have three color channels. Note that the number of instances of an object in an image is not captured in Eq. 2.1. Also, note that founding the boundary between overlapping objects from one category requires some post-processings.

How the segmentation problem has been attacked in the previous works can be categorized into three main groups. The first group try to generate the segmentation using some building blocks (exemplars). The building blocks are usually the partial or full shape of the object in the training images. These methods include a voting scheme to decide which building block should contribute more to a test image. The second group, model the problem as an energy minimization problem (probability maximization problem). The energy function could be a combination of the local evidences in the image and the prior knowledge of the object’s shape. These energy minimizations approaches fall in the same grouped as the probabilistic approaches that consider the local image information and the object shape prior. The proposed approaches in this thesis fall into this group. The last group, take a reverse approach. Starting from a large set of possible segmentations for an image, the best candidate is chosen (the rest are trimmed) based on the local and global image features. All three groups belong to a higher category called supervised segmentation, because the ground-truth labeling of the training data is exploited by the methods in these categories. To have a more general taxonomy, we also include another group called unsupervised segmentation methods that do not have access to the ground-truth labeling of the training images.

It is worth mentioning that the first three groups are not disjoint and as will be shown later, there are several approaches that fall in the boundary of two or even all three groups.
2.2.1 Exemplar Based approaches

The method proposed by Borenstein et al. [6] takes a generative approach to segment the objects in the image. The method finds an ensemble of image patches (template) from the training data that have high likelihood ratio being part of the object of interest. Segmentation is achieved by maximizing the consistency, reliability and the coverage score between the pre-computed templates and the image in a greedy strategy. Leibe et al. [51] propose an approach to detect and segment the objects. First, object specific visual words are extracted around the interest points. Then, each visual word casts its vote both for the center of the object and the shape of the object. The voting result is thresholded to produce the final segmentation. Marszalek and Schmid [55] propose a similar framework, in which the localization and segmentation are done simultaneously and both benefit from the other one. It is in oppose to the Leibe et al. [51] approach that the segmentation result does not effect the localization. In Marszalek and Schmid [55] approach, the localization and segmentation are computed generatively. Interest points are found in the test image and the corresponding patches are assigned to the learned visual vocabularies. Using a classifier trained in the learning phase, all possible shape masks for a patch are collected. Patches with similar shape masks are clustered together and vote for the final results.

The approach of Pantofaru et al. [62] have the same flavor, but the building blocks are now segmentation results from different segmenters. The method compares four different strategies for object segmentation. The first method computes many superpixels in different scales, classifies them independently based on some features and combines their results uniformly. In the second method the labels of superpixels are aggregated proportional to their likelihood given the image. The third strategy formulates a MRF with classification scores as unary terms and a pairwise smoothing term. The last method adds partially labeled data such as bounding boxes from the detection set and classification labels of images. The result show that all methods perform not worse than the best single scale superpixel configuration. Moreover, adding partially labeled data improves the base results the most.

Recently, Yang et al. [92] proposed a method to model the occlusion for multi-class segmentation given the detection results and the shape prior for each object class. An ordering is assumed between detections and image pixels are uniquely assigned to detections. Based on the ordering between the detections the occluded pixels are ignored for appearance
model estimation. All possible orderings of the detections are explored to find the one with the maximum likelihood. The reason for categorizing this method in this group is because the final segmentation is just a function of the shape priors, which can be seen as a building block, and the ordering of the detections. However, this method is certainly on the boundary of the exemplar based and probabilistic approaches.

2.2.2 Energy based and probabilistic approaches

Most of the previous works in object category segmentation belong to this group. Majority of the methods in this group exploit MRF or CRF models. Note that the methods that formulate the segmentation problem as a multi-class classification problem are basically a MRF model with only unary potentials. To emphasize this property, we break the energy-based approaches into two subgroups. The method in the first subgroup only includes unary potentials in their model and the second subgroup benefit from unary, pairwise and maybe higher order potentials.

Classification-based methods

Hoiem et al. [37] try to segment the image into sky, ground and vertical. Multiscale superpixels are extracted by a greedy growing strategy. The likelihood of each superpixel being from all classes are computed using a logistic regression form of Adaboost given the features of that superpixel. Each pixel is then labeled by aggregating the labels of the superpixels including that pixel. Yang et al. [91] categorizes the objects into two groups, the one with discriminant shapes and the one without. If the shape of an object class is not discriminant, the segmentation is achieved using only bottom-up scores. Otherwise, the method optimizes the sum of the bottom-up and top-down scores to obtained the final segmentation. A new global feature is proposed that can model the context by computing the spatial arrangement of bag-of-word features in the image. More precisely, the probability of possible spatial configurations between all pairs of visual words are approximated as the occurrence count in quantized spatial regions. The bottom-up scores are defined as the score of a nonlinear multi-class SVM trained on these novel features. Annotated objects in the training set form the shape templates for each object class that are matched to the test image to produce the top-down score. Tu [83] proposes a method that can benefit from long range connections between the labeling sites in oppose to MRF. In this method, every pixel has two distinct sets of features, appearance feature of the pixel and posterior probability of
its long-range neighbors. Starting with uniform posterior probabilities and image features the first classifier is learned. The posteriors are updated and the method iterates. At the test time, the same iterative approach is employed to classify each pixel.

**Higher order methods**

He et al. [33] propose a multi-scale CRF for image segmentation. In addition to unary and pairwise potentials, the method incorporates some regional features which model the interaction between labels in a larger region. By varying the size of the region, multi-scale features are constructed. Each regional feature is associated with a hidden binary variable that models the existence of that regional feature. The graphical model for regional features are of Restricted Boltzmann Machine form. Contrastive divergence learning is employed to approximate the gradient for the overall gradient descent learning of the parameters. The model is similar to product of experts, where local classifiers and regional features are the experts.

The method proposed by Kumar et al. [45] exploits the advantages of MRF bottom-up configuration and pictorial structure (PS) [27] top-down model. The pictorial structure model is employed to localize the object by considering the local appearance scores of the object’s parts and their spatial configuration. Here, an extra unary potential is added to the MRF, which gives the pixels scores proportional to their distance to the object prior given by PS. Shotton et al. [71] employ the same CRF structure with different unary and pairwise potentials. This method includes shape, texture, color and location features as unary potentials and edge features as pairwise potentials. The method models the shape, texture and contextual information jointly as follows. The textons are extracted in an image given a learned visual vocabulary. A large collection of Harr-like features similar to the one used in Viola and Jones [87] face detectors are created on the textonized image. A multi-class boosting algorithm is then utilized to form a final strong classifier. Learning the parameters of the CRF is achieved by piecewise training. Basically, the best parameter setting for each potential is computed independently with hope to have a good overall performance.

The method of Galleguillos et al. [30] employs the co-occurrence and relative location of the objects as pairwise terms. The co-occurrence and relative locations probabilities between all object pairs are pre-computed from the ground-truth. Batra et al. [3] propose to have class-specific pairwise potentials. In this approach, the pairwise potentials between two regions are not just a function of image features inside those regions, but also a function
of their labels. The problem is modeled as a CRF with unary and pairwise potentials and approximate inference is achieved using Loopy Belief Propagation (LBP). Kohli et al. [64] propose a principled way to incorporate multiple quantizations of the image space for segmentation. The method augments a higher order potential (higher than 2) to the common CRF with unary and pairwise potentials. The new potential measures the consistency of the labels inside each subregion (superpixel) in oppose to previous approaches that assume a single label for each superpixel. A move making algorithm called $\alpha$-expansion [9] is used for approximate inference. Ladicky et al. [46] extend this work by assuming pairwise potentials on this higher order potentials. Basically, the new pairwise term is a function of all pixels inside two neighboring superpixels, which encourages consistency between neighboring cliques (pixels inside superpixels). Again, a graph-cut-based move making strategy is used for inference.

Szummer et al. [74] formulate the segmentation problem as a structured prediction problem. Their model is a CRF with unary and pairwise potentials, where the corresponding energy is defined as a linear function of unary and pairwise features. To learn the parameters of the model, Structural SVM [81] formulation is employed. Structural SVM tries to assign parameters such that the energy of the ground truth segmentation be lower than the energy of any other segmentation. The similarity between two segmentations are measures by Hamming loss. The MAP estimate at inference time is computed efficiently by graph cut. This method can only optimize against separable loss function.

Larlus et al. [48] proposed a generative approach for object segmentation. The image is assumed to have been generated using an unknown number of blobs (each corresponding to an object) and a background region. Each blob is responsible for generating its own image patches based on its parameters. The final pixel segmentation is achieved by aggregating the information of all patches intersecting with a pixel. Each image patch is described with BOW features. A MRF is employed to model the interaction between neighbouring patches. The assignment of patches to blobs follows a Dirichlet process mixture, which prefers to assign a new patch to the blobs with more patch members. The method uses Gibbs sampling for inference.

Borenstein et al. [7] try to combine bottom-up and top-down cues in a Bayesian framework. The bottom-up cues are appearance features, such as texture, color, boundary properties of superpixels of an image and the similarity between the overlapping superpixels
in different scales of a hierarchy of superpixels. The top-down features include some pre-computed templates that are matched to the image in different locations and scales. The inference, which is achieved by a message passing strategy on a simplified form of the model, approximately computes the segmentation that respects the bottom-up and top-down cues the most.

In this thesis we propose two algorithms that can optimize against non-decomposable loss functions. Our first approach can optimize against non-decomposable loss function which are a function of false positive (FP) and false negative (FN) counts. This method approximates the loss surface in the space spanned by FP and FN with many piecewise planar surfaces, which are separable. The energy minimization problem is then solved using linear programming relaxation. At inference time, graph cut is exploited to efficiently compute the MAP estimate. Our second approach, can optimize the loss augmented inference for a larger class of loss functions. Basically, any loss function that can be optimized when augmented with a linear term (not quadratic) can be optimized using our second approach.

2.2.3 Trim based approach

The method presented in [52] creates many independent segmentations using random foreground seeds and smooth the labeling with a pairwise smoothing function. Then, the method ranks these segmentations by a regressor that maps the shape and appearance features of foreground to a score value for each object. Top rank segmentations vote for the label of the pixels they cover.

2.3 Summary

In this chapter we reviewed the related works in structured prediction, which were mostly concerned with decomposable loss functions. We also reviewed different methods for object category segmentation. We now proceed to describe the main contributions of the thesis, methods for optimizing against non-decomposable loss functions in structured prediction, and show they can be effective for tasks such as object category segmentation.

\footnote{Please see Chapter 4 for details}
\footnote{Please see Chapter 5 for details}
Chapter 3

Human Segmentation

Before introducing the proposed algorithms for handling non-decomposable loss functions in structured prediction, we show experimentally why comparing our results against the same model except optimizing against decomposable loss functions such as Hamming loss is a reasonable comparison. In this chapter, we introduce fine and coarse texture cues as a set of new features for human figure segmentation. Employing the new features, but optimizing against decomposable Hamming loss, we achieve state-of-the-art results on one of the most challenging datasets – VOC Pascal 2009 – for person class. In the next chapters, we use this method as one of our baselines and significantly improve it by optimizing against the target non-decomposable loss function.

3.1 Overview

In this chapter we focus on the development of an automatic algorithm for segmenting human figures in images. Humans are a common subject of photographs and the focus of attention in many applications (e.g. Sketch2Photo [15]). Their segmentation is a challenging computer vision problem. A number of factors contribute toward this difficulty – but of particular concern are (1) variation in shape due to articulated pose and (2) the large amount of variation in object texture and presence of substantial internal texture gradients due to clothing. For this reason, generic segmentation routines are unlikely to be successful for this task. In this chapter we focus on addressing this challenge, and on the use of clothing texture cues for segmentation of human figures. We demonstrate that these clothing textures can be used to automatically produce accurate segmentations of human figures superior to those
CHAPTER 3. HUMAN SEGMENTATION

Figure 3.1: Automatic human segmentation using the proposed model. (a) A test image. (b) Result of the full model (unary and pairwise cues). (c-e) Results of the models trained only using unary cues: coarse fold features, fine textures, and human shape prior respectively.

of generic segmentation methods.

When designing an algorithm for this problem, one must attempt to construct features that are invariant to clothing appearance, while being discriminating about human form. One such property that has been profitably exploited in the past is the shape of a human—the boundaries of a human figure, if they can be reliably detected, are invariant to lighting and clothing variations. In this chapter we instead follow a less common path, attempting to use material properties, particularly fine textures such as the fabric of clothing, and coarse textures, the folding of this clothing. It can be argued that these are also robust cues—while the colour and appearance of clothing have significant variation, clothing is typically made from materials with a certain class of fine texture and coarse folding properties.

Consider the human subject in Fig. 3.1. This figure shows the results of our fully automatic segmentation method. Our method uses a Markov Random Field (MRF) segmentation model and combines unary cues from a number of sources, including a top-down human body prior using shape cues and our fine and coarse clothing texture cues. Each of these cues has its successes and failures. Considering each cue in isolation, the coarse-scale texture labels as human figure those regions of the image resembling folds of clothing (c). The fine-scale texture labels those of material similar in appearance to clothing (d). The top-down prior labels regions with extended, roughly parallel edges (e). We learn a weighted combination of these cues (b) which resolves many of the failures of each independent cue.

The main contributions of this chapter center around the development of an MRF segmentation algorithm incorporating the aforementioned texture and top-down shape prior.
cues. We define a novel set of texture features for discriminating clothing. The MRF model contains a large number of parameters for combining these cues with shape prior and binary segmentation cues. We describe a novel application of the max-margin framework [81] for learning the parameters of a fully automatic segmentation algorithm over a large set of features. We show that each individual feature aids in segmentation accuracy and this segmentation model outperforms competing methods on two challenging datasets of human figures, in particular achieving state-of-the-art segmentation performance on the PASCAL 2009 Segmentation Competition.

3.2 Related Works

The literature on “looking at people” is vast, and contains much relevant work in areas such as pose estimation, tracking, shape and appearance features. Forsyth et al. [28] provides a comprehensive survey of this area. In this section we review a subset of closely related work.

Our work is inspired by that of Haddon et al. [32] who developed physics-based models of folds and grooves, and used these to discern body grooves (e.g. along the spine) and clothing folds. We push this line of work further, and represent these cues with non-parametric models that seem to capture more variation than the physics-based models and SVM learning in previous methods. Other recent clothing-related work includes Chen et al. [14] who develop an and-or graph representation used for exemplar matching, but not segmentation, based on clothing texture. Gallagher and Chen [29] recognize people using facial and clothing features. Torso segmentations are obtained by initializing a colour model under face detections. However, it isn’t clear this method will handle articulated figures nor full body segmentations.

There is related work on multi-camera tracking and body shape estimation from silhouette data. Rosenhahn et al. [68] build a tracking model that includes in its state cloth draping parameters for modeling free-flowing clothing such as skirts. Balan and Black [2] estimate the parameters of a detailed and accurate model of body shape under clothing from multi-camera data. In this chapter we concentrate on discriminating the material properties of clothing rather than estimating its parameters.

In this chapter we use a Markov random field (MRF) segmentation model defined over superpixels. Winn and Shotton’s LayoutCRF [90] defines a probabilistic segmentation model
over possibly occluding object instances. He et al. [34] perform image labeling (animals, snow, water, etc.) on superpixels while exploiting context cues. The model is mixture of conditional random fields, trained using conditional likelihood. Our model focuses on texture cues for human figures, and learns the parameters of a related MRF model using the max-margin criterion. Recently, Szummer et al. [74] used the same max-margin criterion for learning MRF parameters for semi-automatic segmentation, and for geometry estimation. Our work uses the same learning framework, but with an order of magnitude more features, focuses on cues for human segmentation, and is automatic via the use of a pose prior.

For segmenting human figures, Mori et al. [59] also label superpixels in a similar fashion. However, the texture features used in that work rely on the fact the human figure was often in focus, a useful cue for segmentation. The PoseCut work of Bray et al. [11] jointly estimates human pose and segmentation from multi-camera video data. Our method focuses on monocular image data and learning of segmentation parameters, though it would be interesting to consider joint pose estimation and segmentation as in PoseCut. Bourdev and Malik [8] detect and segment human figures using a max-margin voting scheme on a part-based representation termed poselets. The preliminary segmentations produced seem to be based on a voting scheme which does not re-analyze low-level image cues. However, excellent detection results are obtained, which could be used in our method in place of our current top-down pose prior.

### 3.3 Human Segmentation Model

The problem that we are dealing with in this chapter is segmenting people in images. The desired output is a binary labeling of image pixels, marking all pixels corresponding to all people in the input image. Our approach is to define informative features to distinguish between person and non-person objects in the scene. We propose to use clothing texture cues for human segmentation. Our intuition is that the type of fine material texture that can be found in clothing, and shape of folds, which we term coarse clothing fold texture, are strong cues for human segmentation. We use these novel clothing features along with a top-down shape prior and standard low-level features in a segmentation model. The combination of these features can form an effective segmenter, and we employ a learning procedure to learn the appropriate weights for each feature.

We formulate the segmentation problem using a Markov Random Field model, in which
the background pixels should get label 0 and foreground pixels label 1. For computational reasons and to allow for computing features over larger spatial areas, we group similar pixels in superpixels and then label each superpixel rather than individual pixels.

The segmentation problem is to decide on binary output labels of superpixels $y \in \mathcal{Y} = \{0, 1\}^N$, based on inputs $X \in \mathcal{X}$, the features of $N$ superpixels.

Following the standard MRF formulation of segmentation, we employ two types of cues: unary features that are defined on a single superpixel $\nu \in \mathcal{V}$, and pairwise features that are defined on a pair of superpixels $(\epsilon, \epsilon') \in \mathcal{E}$. In this case, the scoring function specifying good values for superpixel segmentation labels $y$ given inputs $X$ is:

$$
\sum_{\nu \in \mathcal{V}} w_{(u)} T \phi_{(u)}(x_{\nu}, y_{\nu}) + \sum_{\epsilon, \epsilon' \in \mathcal{E}} w_{(p)} T \phi_{(p)}(x_{\epsilon}, x_{\epsilon'}, y_{\epsilon}, y_{\epsilon'}), \tag{3.1}
$$

where $w_{(u)}$ and $w_{(p)}$, are weight parameters corresponding to unary and pairwise features, and $\phi_{(u)}$ and $\phi_{(p)}$ denote the unary and pairwise feature functions, respectively. The global maximum to this function can be found using graph cuts since the pairwise features meet the requisite sub-modularity condition.

Of crucial importance to the effectiveness of such an MRF segmentation scheme are the set of features used in $\phi_{(u)}$ and $\phi_{(p)}$, and the weights $w_{(u)}$ and $w_{(p)}$ placed on them. In the following sections we elaborate on these components of this model. In particular, Section 3.3.2 describes our novel clothing texture features, top-down shape prior, and pairwise features. The segmentation model contains a large number of weights. Section 3.3.3 formalizes the learning of these parameters via the max-margin criterion. Section 3.3.1 describes the computation of superpixels upon which the model is built.

### 3.3.1 Superpixel Extraction

In our pre-processing step, we group pixels into superpixels, and perform person/background labeling on superpixels rather than the original pixels. The reason for grouping pixels in superpixels is that in the latter case one would only label a few hundred superpixels (around 600 here) rather than labeling the roughly 1 million pixels in our images. Finding superpixels is not very time consuming so this modification decreases both training and testing time (as described in Sec. 3.3.3, learning requires multiple graph cut optimizations per training image). In addition, pixel level segmentation can be noisy in the absence of a strong smoothness constraint, which can be reduced by aggregating the features inside a
superpixel.

Among previously proposed segmentation approaches, we choose the graph-based segmentation [25] for extracting superpixels. The main advantage of this approach is its low computational cost compared to other segmentation methods. Fig. 3.3(b) shows some example results of superpixel extraction.

### 3.3.2 Features

As argued above, clothing texture can be an informative cue for human figure segmentation. To capture this information, we define two types of texture for clothing: fine texture and coarse texture. Fine texture is responsible for capturing the information regarding the material of an object, while coarse texture extracts the folding, or wrinkles of clothing in the input image. But these unary features alone are insufficient to accurately segment humans – lacking top-down body shape information and local smoothness they will not necessarily mark coherent regions of images as person. Hence we combine these with top-down information in the form of a human figure shape prior, and with bottom-up pairwise cues.

Below we describe the unary cues (fine, coarse, and prior features on a superpixel) whose concatenation is used to form the unary features \(\phi(u)\), and the bottom-up pairwise color cues used in \(\phi(p)\) above.

**Fine Material Texture**

We desire a set of features that can capture the fine texture of clothing – i.e. the appearance of fabrics typically used in clothing. We use a standard non-parametric texton-based texture representation [54] in which a vocabulary of textons is obtained via clustering on filter responses. The fine material texture in a superpixel is represented by the histogram of these textons.

This representation requires a collection of images for different materials for building textons. One way of generating this collection is to randomly sample patches from our human dataset. The problem that leads us to use another strategy is that the type of object materials in our dataset might be limited. Therefore, our overall method might not generalize well for arbitrary images.

Thus, instead we use a standard texture dataset, Columbia-Utrecht [19]. We choose
56 materials to generate our texture model. We then apply a rotation-invariant Maximum Response-8 filter bank [86]. We select 30 images for each texture of size $200 \times 200$ pixels, and use K-Means to produce 10 textons for each texture. Via this procedure we have 560 (10 × 56) textons to describe an arbitrary input texture.

We define the fine texture of superpixel $\nu$, $f_{(\text{fine})\nu}$, to be the histogram of these texton IDs for its pixels.

**Coarse Clothing Fold Features**

To discriminate coarse-scale clothing folds, one approach is to use physics-based models of cloth, as in [32]. Instead, we directly model the appearance of folds by learning a generative model of cropped clothing patches. This model learns a set of filters that are good at reconstructing images of clothing folds. The set of filters learned by this model are depicted in Fig. 3.2a. Notice the seemingly 3d structure of these filters and their resemblance to actual cloth folds. The responses of input images to these filters will be used as a non-parametric representation of coarse clothing fold texture. The details of this model are described in this section.

The probabilistic model we use is called a Convolutional Restricted Boltzman Machine (CRBM) [61, 49]. We choose this model since it has been shown to be effective and efficient to compute in learning shape features for object recognition tasks. Here we demonstrate its effectiveness for texture modeling for segmentation. The CRBM defines the density of images as a product of overlapping patch densities. For each image patch a density is defined
according to the dot product of a filter bank and the patch pixels – each patch and filter are of the same size. A key property of the CRBM is that it leads to learning translation invariant filters. Because each filter is multiplied by all the overlapping patches of an image, adding translations of a filter to the filter bank is unhelpful. This property is useful for texture modeling since the precise spatial location of cloth folds is not relevant, only their presence is. We learn a set of filters by (approximately) maximizing the likelihood of training examples under the CRBM density model. These filters will learn image patches that are frequent among the training examples (cloth fold images) and rare in other settings.

The continuous CRBM is a two layer partially observed MRF that consists of a layer of continuous visible units associated with image pixels and a layer of binary hidden units related with filter responses. Each hidden unit represents presence of a particular feature at a specific location of the image. The probability of hidden units given the image pixels is computed as a sigmoid non-linearity over the filter responses. We use these non-linear filter responses as the features in our segmentation model.

Maximum likelihood learning of the CRBM filters is intractable due to exponential number of terms involved in its partition function. Instead we use contrastive divergence learning [35] that approximates the gradient of the log likelihood by Gibbs sampling. In addition, a L2 regularizer and sparsity constraints are added to the objective to obtain less noisy filters.

Details. We collected a set of 564 grayscale clothing patches of size $31 \times 31$ pixels (Fig. 3.2b) from a set of people images, none of which were included in the test datasets. The mean of each patch was subtracted, and all the pixels multiplied by a constant to obtain fixed variance over the whole set of pixels. From the preprocessed dataset 50 filters of size $9 \times 9$ were learned (Fig. 3.2a). For contrastive divergence learning only one step of Gibbs sampling was performed, the L2 regularization parameter was set to 0.03, and the hidden nodes were forced to activate one tenth of the time. For the gradient descent, we subdivided the images into 6 mini-batches of 94 images. Learning was performed in 2000 epochs with initial learning rate that was decreased after each epoch.

For feature extraction, we filtered the images with the learned filter bank. We passed the filter responses through a sigmoid non-linearity. Finally, the coarse clothing fold feature of a superpixel $\nu$, $f_{(\text{coarse})\nu}$, is assigned to the sum of the non-linear filter response over pixels within this superpixel.
CHAPTER 3. HUMAN SEGMENTATION

Human Figure Prior

Employing only bottom-up features for segmentation results in blob-ish segments that have the correct low-level texture properties. Top-down features are required to produce a coherent segmentation. We develop two strategies to provide top-down features that provide this knowledge based on body pose. The first uses a parametric body model and infers the locations of parts. The second uses a non-parametric template-based representation. We then learn weights on these top-down features and our bottom-up features simultaneously.

The parametric body model top-down feature uses the pictorial structure iterative parsing model [65]. The pictorial structure model produces a marginal probability density for each body part assuming a tree structure over the parts configuration. Iterative parsing refines these probabilities exploiting color information iteratively. Fig. 3.3(c,d) illustrates the probabilities for pictorial structure (generic edge templates) and iterative parsing models (after color information).

For each pixel, we collect the probability of each limb obtained before and after iterative parsing in a 20 dimensional vector. We compute the top-down feature \( f_{(prior)\nu} \) by accumulating these vectors for all pixels inside superpixel \( \nu \). We then learn weights on the parts to figure out which parts are more useful as a prior.

The second template-based top-down features use a grid-based non-parametric pose prior. We use a resizable grid with a fixed number of bins, \( N_b \), over an image. The top-down feature for an image is defined as the number of foreground pixels lying in each bin. We assign these features to superpixels by generating a vector of size \( N_b \) that the \( i^{th} \) entry measures the intersection of this superpixel and bin \( i \).

Comparing these two approaches, the pictorial structure model is better at adapting to people in a variety of poses. However, it assumes that all 10 body parts exist in the image. This is problematic when the person is not visible entirely in the image. In this case, not only the non-visible parts are found mistakenly in the image, but the visible parts are also pushed to incorrect locations, because of the constraints between parts locations. We choose the appropriate one for each experiment based on the types of poses present.
CHAPTER 3. HUMAN SEGMENTATION

Figure 3.3: (a) Input image, (b) extracted superpixels, (c) edge-based prior, and (d) region-based prior.

Pair-wise Feature

In order to enforce smoothness of the segmentation results, we also incorporate pair-wise features between adjacent superpixels. We employ the mean intensity and chromaticity differences of neighboring superpixels as our pairwise feature. Formally, we define the pairwise feature between superpixel $\epsilon$ and $\epsilon'$ as

$$ f_{(\text{pairwise})\epsilon\epsilon'} = \mathcal{P}_{\epsilon\epsilon'}, \exp \left[ -\alpha L_{\epsilon\epsilon'}, -\beta a_{\epsilon\epsilon'}, -\beta b_{\epsilon\epsilon'} \right] $$

(3.2)

where, $L_{\epsilon\epsilon'} = |L_\epsilon - L_{\epsilon'}|$, and similarly for $a_{\epsilon\epsilon'}$ and $b_{\epsilon\epsilon'}$. Here, $L$, $a$ and $b$ are intensity and chromaticity values in $L_a^*b^*$ color space, and $\mathcal{P}_{\epsilon\epsilon'}$ is the length of shared boundary between superpixels $\epsilon$ and $\epsilon'$, which is used to measure how much these two superpixels are neighboring.

3.3.3 Learning Segmentation Model Parameters

The previous sections describe a set of features that can be used to segment human figures from background. We would like to embed these features in our MRF segmentation model. However, it is crucial to decide how much emphasis to place on each of these features. Rather than setting these feature weights by hand, we employ a learning method to decide their values in a principled fashion. The learning method we exploit is Structural SVM [81],
in which one chooses weights to push the model’s score on correct segmentations away from that for incorrect ones.

As noted above, the MRF formulation of segmentation employs two types of cues: unary features that are defined on a single superpixel $\nu \in \mathcal{V}$, and pairwise features which are defined on a pair of superpixels $(\epsilon, \epsilon') \in \mathcal{E}$. We write the scoring function of Eq. 3.1 as the dot product between the combined (pairwise and unary) weight vector $\mathbf{w}$ and the combined feature representation $\Psi$:

$$\langle \mathbf{w}, \Psi(x, y) \rangle = \mathbf{w}_u^T \sum_{\nu \in \mathcal{V}} \phi_u(x_\nu, y_\nu) + \mathbf{w}_p^T \sum_{\epsilon, \epsilon' \in \mathcal{E}} \phi_p(x_\epsilon, x_{\epsilon'}, y_\epsilon, y_{\epsilon'}). \quad (3.3)$$

For the binary variables in segmentation, we only have two choices for $y_\nu$, labeling of superpixel $\nu$, so we define the feature functions as

$$\phi_u(x_\nu, y_\nu) = \mathbb{I}_{[y_\nu = 1]} F_u(x_\nu), \quad (3.4)$$

$$\phi_p(x_\epsilon, x_{\epsilon'}, y_\epsilon, y_{\epsilon'}) = \mathbb{I}_{[y_\epsilon \neq y_{\epsilon'}]} F_p(x_\epsilon, x_{\epsilon'}), \quad (3.5)$$

where $\mathbb{I}_{[\cdot]}$ represents the indicator function, and $F_u$ and $F_p$ denote the unary and pairwise features, respectively. We define these features, using the component features defined previously, as

$$F_u(x_\nu) = [f_{(\text{fine})\nu}, f_{(\text{coarse})\nu}, f_{(\text{prior})\nu}] \quad (3.6)$$

$$F_p(x_\epsilon, x_{\epsilon'}) = [f_{(\text{pairwise})\epsilon\epsilon'}] \quad (3.7)$$

where $[\ldots]$ denotes concatenation.

Given training data of image and ground-truth segmentation pairs $\{x^{(i)}, y^{(i)}\}$, the Structural SVM sets model parameters $\mathbf{w}$ according to an objective similar to SVM:

$$\min_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{M} \sum_{i=1}^{M} \xi_i, \quad s.t. \forall i, \xi_i \geq 0, \quad (3.8)$$

subject to margin constraints, where $M$ is the number of training examples.

In the structural SVM the margin is defined according to a loss function $\Delta(y, y')$. For the case of segmentation this loss specifies the penalty to be paid for predicting incorrect
CHAPTER 3. HUMAN SEGMENTATION

segmentation \(y'\) when the ground truth is \(y\). One choice for this loss is

\[
\Delta(y, y') = \sum_{\nu \in \mathcal{V}} 1[y_{\nu} \neq y'_{\nu}] \gamma_{\nu},
\]

(3.9)

where \(\gamma_{\nu}\) is the area of superpixel \(\nu\). This loss function measures the hamming distance between the labelings \(y\) and \(y'\). The loss is used to define the margin constraints that push the scores on ground truth segmentations \(y_i\) away from others \(y\), modulated by how poor the others are:

\[
\langle w, \Psi(x^{(i)}, y^{(i)}) \rangle - \Psi(x^{(i)}, y) \rangle \geq \Delta(y^{(i)}, y) - \xi_i,
\]

\[
\forall i, \forall y \in \mathcal{Y}^{(i)} \setminus y^{(i)}
\]

(3.10)

Our implementation is based on the SVM\textsc{Struct} framework [81]. In this framework, for margin constraints of Eq. 3.10, one needs to compute

\[
\hat{y} = \arg \max_{y \in \mathcal{Y}} \Delta(y^{(i)}, y) + \langle w, \Psi(x^{(i)}, y) \rangle.
\]

(3.11)

Note that the loss function of Eq. 3.9 is decomposable to the sum of loss values, each depends on one superpixel, and therefore can be considered as a unary term. Thus, the arg max of Eq. 3.11 is computed over a sum of unary terms plus a sum of pairwise terms. In this case, the global optimum can be found efficiently using graph cut. We employ the Fast-PD [43] framework to find this optimal cut.

3.4 Experimental Results

We conducted two sets of experiments to evaluate our human segmentation framework. In our first experiment images contain a single person in a cluttered indoor scene where both the person and background are in focus. In this experiment we investigate the contributions of different features employed in our model and we compare against Grabcut [69]. In a second experiment we evaluate our method on the segmentation competition of the PASCAL VOC 2009 challenge. We compare our method against other segmentation models.

\[\text{Using the implementation from http://www1.idc.ac.il/tokyo/CompPhoto-09/Projects/Stud_projects/IrenaAviad/Web/index.htm}\]
CHAPTER 3. HUMAN SEGMENTATION

Evaluation. In our experiments the segmentation accuracy is measured as the standard ratio of intersection over union between the ground truth and the results. This metric is defined as

\[
\text{Accuracy} = \frac{\cap}{\cup} = \frac{TP}{TP + FN + FP}.
\]

(3.12)

To optimize this accuracy we minimize a related loss within the max-margin framework which is defined as

\[
\text{Loss} = FN + \kappa FP.
\]

(3.13)

The parameter \( \kappa \) determines the penalty ratio of \( FN \) over \( FP \). When \( \kappa \) is one then the loss becomes the hamming distance between ground truth and the reported labeling.

3.4.1 Controlled indoor dataset

We have collected a dataset of 120 indoor images (60 images of 36 different people, along with their flipped versions), in which every image includes one person that roughly occupies 20% of the image. People in this dataset have a variety of poses and appear in scenes with complex backgrounds.

We conduct experiments using 12-fold cross-validation on these images. The same person does not appear in training and test. For this dataset, we use pictorial structure and iterative parsing as the top-down features. These features give reasonable responses in this dataset, since it only includes one person per image, who is entirely inside the image. The baseline Grabcut method also requires a prior over the location of the person. We provide this prior either as a tight bounding box around the person or as the pose prior. The pose prior is a zero-one mask obtained by thresholding the marginal probabilities given by pictorial structure (Fig. 3.3(c)).

We train the system with different combinations of features and report the result in Table 3.1. This table shows that substantial reductions in accuracy are incurred by removing any of our features. We use \( \kappa = 1 \) and normalize the loss values with the person area. Each unary feature includes additional information that has not been captured by the other features. Moreover, the pose prior feature is shown to be the strongest cue (as expected), but augmenting with textural cues improves the result significantly. Furthermore, adding the pairwise features reduces the error significantly. Finally, employing all unary and pairwise features, we have achieved high accuracy in this dataset. In comparison, Grabcut performs noticeably worse than our method because it only uses color information, which might be
Table 3.1: Errors as a function of different combinations of features and Grabcut errors with two different priors. Each ‘×’ symbol indicates that the associated type of feature is used for learning the segmentation model.

<table>
<thead>
<tr>
<th>Pairwise</th>
<th>Fine</th>
<th>Prior</th>
<th>Coarse</th>
<th>Loss%</th>
<th>( \cap )/( \cup )%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td>68.9</td>
<td>43.3</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>60.9</td>
<td>49.8</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>83.7</td>
<td>30.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td>36.3</td>
<td>69.4</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>39.8</td>
<td>66.5</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>60.9</td>
<td>51.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td>32.0</td>
<td>72.6</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>46.8</td>
<td>61.8</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>54.8</td>
<td>54.7</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>63.9</td>
<td>48.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td>29.5</td>
<td>74.9</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>33.8</td>
<td>71.1</td>
</tr>
<tr>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>40.9</td>
<td>67.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td>25.0</td>
<td>78.3</td>
</tr>
<tr>
<td>Grabcut</td>
<td>pose prior</td>
<td></td>
<td></td>
<td>58.0</td>
<td>59.1</td>
</tr>
<tr>
<td></td>
<td>box prior</td>
<td></td>
<td></td>
<td>61.3</td>
<td>62.2</td>
</tr>
</tbody>
</table>

confusing, especially for clothing. Also, note that our method segments the entire image, while Grabcut uses the entire image, but just decides on foreground values for the region inside the given prior that is tight to the person. Figure 3.4 shows some results on this dataset.

### 3.4.2 VOC 2009 Human Segmentation

PASCAL VOC 2009 [22] is a dataset for evaluating segmentation models. The results of 23 methods are available in the segmentation category of VOC 09 (comp5 and comp6 competitions), and a server\(^2\) is made accessible that receives submissions and automatically scores them on the VOC 09 test set. We obtain state-of-the-art accuracy in the person class segmentation task. This highlights the effectiveness of our texture features and our learning framework (see Table 3.2). Some examples from our automatic human segmenter on VOC 09 Test set are pictured in Fig. 3.5. Note that the server limits the number of submissions to prevent tuning. Hence Table 3.2 only shows results for all features. Table 3.1 demonstrates

\(^2\)http://vch.cs.uiuc.edu/
each feature we use contributes positively to segmentation accuracy, and it is likely the same would hold for the VOC dataset.

A wide range of human figures in different scales, diverse poses, with significant occlusions, and under varying lighting conditions appear in the VOC 09 dataset. To handle variability of scales and images with multiple people, we localize individuals in images using the human detector of [24] and separately process each positive bounding box. The sizes of the detection windows give us a good estimate on the scale of people, so we can normalize for scale by resizing the bounding boxes to similar sizes. We train our model and segment people within the scaled bounding boxes. After segmenting figures within each bounding box, in a post-processing stage we rescale the detection windows back to their original sizes and combine overlapping ones by simply taking their foreground union.

Human detectors benefit from having two distinct templates for full body and upper body [24] mainly because of usual occlusions (see the detections in Fig. 3.5). For doing segmentation we also trained two MRFs to segment full and half human figures separately. The scale ratio of the detection windows determine whether they should be processed by the
Table 3.2: Top segmentation accuracies (of 22 teams) on human class of PASCAL VOC 2009 challenge. The accuracies are reported as the intersection over union measure (Eq. 3.12).

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCI–LayeredShape</td>
<td>38.9</td>
</tr>
<tr>
<td><strong>Our method</strong></td>
<td>37.8</td>
</tr>
<tr>
<td>BERKELEY–Poseletes [8]</td>
<td>36.3</td>
</tr>
<tr>
<td>NECUIUC–CLS-DTCT</td>
<td>34.8</td>
</tr>
</tbody>
</table>

Figure 3.5: Example human segmentations on VOC 09 test set. On the left the input image and the detected bounding boxes are shown and on the right the final segmentation is depicted. The final row illustrates some failures of our method.

...full or upper body segmenters. After scaling the detections, we extract a set of superpixels from each bounding box using [25]. We aggregated the fine material texture and the coarse clothing fold features over the superpixels. In our experiments on VOC 09, the parametric pose-prior features proved to be not beneficial for segmenting people, likely due to cluttered backgrounds and occlusions. Instead we employed a non-parametric template-based shape prior discriminatively. We learned a set of weights for different locations within the human detection window that assigns specific bias scores to different superpixels based on their locations. We learned two separate $12 \times 12$ grids of weights for full and upper body...
The evaluation criterion in the PASCAL segmentation challenge is the standard intersection over union (Eq. 3.12) computed for each object class over the whole test images. We ideally want to learn the MRF weights by maximizing this accuracy on the training set (minimizing the corresponding loss). Exact optimization of this loss within the structural SVM using graph-cut is not easy. Instead we train the weights to minimize a related loss (Eq. 3.13).

Details. Because we reject some of the detection windows and only process the bounding boxes above a certain threshold, we require a precise detector threshold so that we do not dismiss many detections or process only a few of them. We found the threshold by cross-validation to minimize the segmentation error on the validation set. The $\kappa$ parameter of the loss was set to 0.4 in our experiments.

3.5 Summary

In this chapter we presented an automatic method for segmenting human figures in images. Novel texture features for this problem that examine clothing texture were defined – analyzing fine scale material texture and coarse scale clothing folds. We combined these with a top-down human figure shape prior in a max-margin learning framework. Quantitative experiments on challenging datasets of human figures demonstrated the effectiveness of these cues over competing methods such as GrabCut. In particular, the method achieves state-of-the-art performance on the PASCAL VOC 2009 Segmentation Challenge for person category. Note that the loss function considered in this chapter is decomposable Hamming loss, despite focusing the thesis around non-decomposable loss functions. However, this chapter confirms that MRF models can achieve state-of-the-art results when backed up by discriminative features and appropriate set of parameters and forms a strong baseline to further improve. Still, when the desired loss function is non-decomposable, e.g. in PASCAL VOC challenge, even higher accuracy is achievable as will be explained in the next chapters.
Chapter 4

Non-Decomposable Loss Optimization using Planar Approximation

In the previous chapter we showed state-of-the-art results in human figure segmentation by employing the proposed set of discriminative features in a Markov network. However, to be able to learn the parameters of the model we approximated the original non-decomposable intersection over union loss with decomposable Hamming loss. The objective of this chapter is to alternatively optimize (approximately) against the original non-decomposable loss function, which will be shown to be superior. This work was published as [67] and will appear in detail in [66].

The main idea in this chapter is to replace the original non-decomposable loss function in false positive and false negative space with a piecewise planar function. Each piece forms a decomposable function, but introduces some constraints on the space it is valid on. We employ LP relaxation techniques to solve the resulting QP. We show significant improvement over two strong baselines – one being the approach presented in the previous chapter – on object category segmentation on Pascal VOC 2009, Pascal VOC 2010 and H3D datasets. We also show significant improvement on action retrieval on our own nursing home dataset.
CHAPTER 4. PLANAR APPROXIMATION

4.1 Overview

Solving challenging vision problems such as image understanding, object category segmentation, and video retrieval arguably requires the use of structured models – those incorporating relationships between multiple input and output entities. Evidence for this comes from state-of-the-art approaches to the aforementioned problems. For example, Hoiem et al. [38] formulate image understanding models that tie together object locations, camera parameters, and surfaces. Blaschko and Lampert [5] localize objects using an efficient solution to a structured output regression model. Desai et al. [20] learn models for simultaneously detecting all objects in an image. Non-max suppression and contextual object co-occurrence statistics are learned in a discriminative fashion. Object category segmentation is a canonical example of structured labeling problem – individual pixel labels are not obtained independently, but by considering structured relationships over groups of pixels (e.g. [54, 9, 74]).

For many of these problems the natural performance measures are also “non-decomposable” – ones that do not decompose into a simple sum of individual terms measured over each output entity. Examples of such measures are object detection scores that penalize for multiple detections on a single true positive (e.g PASCAL VOC [23]) and region labeling or object segmentation scores that penalize for over and under labeling or segmentation (e.g. intersection / union score). Typical methods for solving these problems learn parameters against other performance measures, e.g. Hamming loss for segmentation, and then apply post-processing techniques (e.g. non-maximum suppression in object detection) to address the structure in the performance measure. Instead, in this chapter we develop an algorithm for linking these two together and formulate learning as jointly considering the complex, structured relationships between output variables in the model and in the learning objective.

The main contribution of this chapter is developing a general algorithm for addressing this type of learning problem with non-decomposable models and those non-decomposable loss functions which are a function of false positive and false negative counts. We specifically apply it to two problems, object category segmentation and human action retrieval, but note that the algorithm can be applied more broadly. We experiment with Markov Random Field (MRF) models. For segmentation, this is a standard model that contains both unary terms for labeling pixels and pairwise terms on the labels of neighbouring pixels. For action retrieval, we formulate a novel MRF that can capture contextual relationships between the
actions of the people in a scene. In both cases, we show that learning the parameters to the
model under an objective directly tied to the performance measure significantly improves
performance relative to baseline algorithms.

This chapter builds on our preliminary work [67]. In this chapter we formulate a multi-
label version of the method, with different inference scheme, and new experiments on object
category segmentation and action retrieval.

The structure of this chapter is as follows. We review related learning work in Sec. 4.2.
Sec. 4.3 provides background on structured prediction. Sec. 4.4 describes our proposed
method for learning. Sec. 4.5 presents the details of our segmentation and action retrieval
models, and experimental comparisons to baseline methods. We provide the summary of
the chapter in Sec. 4.6.

4.2 Related Works

A wide range of learning algorithms exist. Despite technical differences, all of these ap-
proaches rely on a performance measure to define what is a “good” result. Based on the
complexity of the performance measure, two general approaches to optimize it are imagin-
able, formulate the learning problem to directly optimize this measure, or approximate this
measure with a simpler one and try to optimize it aiming to indirectly optimize the original
non-decomposable performance measure. We will call the former “direct optimization” and
the latter “indirect optimization”.

Due to the complexity of some performance measures, e.g., average precision and in-
tersection over union, many state-of-the-art approaches in different challenges exploit an
indirect optimization. Looking at the PASCAL VOC challenge 2010 [23], for example, aver-
age precision and intersection over union are defined as performance measures for detection
and segmentation tasks respectively, but methods for both tasks use indirect optimizations
for solving these problems.

Structured models are arguably a requirement for robust solutions to learning problems
in a variety of application domains. Tasks such as machine translation, object category seg-
mentation, and scene understanding involve reasoning about relationships between words in
a document, pixels in an image, and objects in a scene respectively. In addition, the perfor-
ance measures for these applications often are non-decomposable and are not a simple sum
of terms measured over individual output entities. Instead, they measure performance as a
function of the entire, structured output. The focus of this chapter is developing a learning approach that can handle these together, handling structured prediction while optimizing against certain non-decomposable performance measures.

Modeling dependencies between outputs while optimizing against a loss function has been a research topic for many years. Optimizing the expected loss in this scenario is a non-convex problem. However, Taskar et al. [78] and Tsochantaridis et al. [82] have proposed rather to optimize a convex relaxation of the expected loss. The cutting-plane algorithm has been shown to be efficient for solving this optimization [82]. Teo et al. [80] presented a bundle method, which is basically the cutting-plane method stabilized with Moreau-Yosida regularizer and prove a tighter bound on the duality gap. Taskar et al. [79] solves the same problem using the extragradient method. Extragradient consists of a gradient descent followed by a projection to the feasible set. Shalev-Shwartz et al. [70] proposed Pegasos, which works solely in the primal space. Similar to [79], Pegasos consists of a gradient descent step followed by a projection step. The computational difficulty in all aforementioned structured prediction approaches is finding the subgradient, which requires solving the “most violated constraint” [82] or “loss augmented inference” [77]. It is shown that for decomposable performance measures learning is tractable when the model is a submodular MRF or a matching [78, 82, 79]. In contrast, in this chapter we focus on non-decomposable performance measures.

Joachims [39] proposed an approach to efficiently compute the most violated constraint for a large class of non-decomposable loss functions, a subset of those we consider in this chapter. However, the underlying models were limited, and do not permit pairwise interactions between output labels. The method of Yue et al. [94] takes a similar approach to optimize against Mean Average Precision. Khanna et al. [13] present an algorithm in the same framework to optimize against normalized discounted cumulative gain (NDCG). Rather than solving a convex relaxation of the expected loss, McAllester et al. [56] proposed a perceptron-like training approach to directly optimize the original loss function, but still need to solve the loss augmented inference. For the problems in which the inference procedure is not tractable, Finley et al. [26] compare under-generating and over-generating algorithms in structured prediction and conclude that “overgenerating methods [LP and graph cut] have theoretic advantages over undergenerating [LBP, greedy] methods”.

In this chapter we provide an algorithm for structured prediction with a non-decomposable scoring function that optimizes against non-decomposable performance measures, those
which are a function of false positive and false negative counts.

4.3 Background

To create a foundation for the proposed approach, we start with an overview of our learning formulation. Next, we discuss the two common approaches, one based on decomposable loss functions with non-decomposable scoring functions and the others with non-decomposable loss functions and decomposable scoring functions. We call a loss function simple if it can be decomposed into loss on individual training samples. Likewise, a scoring function is called simple if it only depends on a single sample point and its ground-truth label. Finally, we propose a framework to incorporate certain non-decomposable loss functions and non-decomposable scoring functions in structured prediction.

4.3.1 Problem Formulation

The goal of our learning problem is defined as finding a function \( h \in \mathcal{H} \) from the hypothesis space \( \mathcal{H} \) given training samples \( S = ((x_1, y_1), \ldots, (x_N, y_N)) \) that optimizes the expected prediction performance on the new samples \( S' \) of size \( N' \).

\[
R(\Delta)(h) = \int \Delta \left( \left[ h(x'_1), h(x'_2), \ldots, h(x'_{N'}) \right], \left[ y'_1, y'_2, \ldots, y'_{N'} \right] \right) d\Pr(S').
\]  

(4.1)

In general, the loss function \( \Delta \) cannot be decomposed into a linear combination of a loss function \( \delta \) over individual samples. But, for simplicity, most discriminative learning algorithms (e.g. SVM) assume decomposability and i.i.d. samples, which allows for rewriting Eq. 4.1 as

\[
R(\Delta)(h) = R(\delta)(h) = \int \delta(h(x'), y') d\Pr(x', y').
\]  

(4.2)

Instead of solving the estimated risk in Eq. 4.2, learning algorithms approximate that with empirical risk \( \hat{R}(\delta) \) defined as

\[
\hat{R}(\delta)(h) = \frac{1}{N} \sum_{i=1}^{N} \delta(h(x_i), y_i).
\]  

(4.3)
For non-decomposable loss functions, such as $F_1$ score or intersection over union, optimizing Eq. 4.2 does not provide the desired answer. Rather, we are interested in finding an algorithm that can directly optimize the empirical risk based on the sample loss,

$$
\hat{R}_S^\Delta(h) = \Delta ((h(x_1), \ldots, h(x_N)), (y_1, \ldots, y_N)).
$$

(4.4)

Note that finding an $h \in \mathcal{H}$ that optimizes Eq. 4.4 for an arbitrary loss function $\Delta$ can be computationally challenging.

### 4.3.2 Structured Prediction Learning

For non-decomposable loss functions, one can reformulate the SVM model based on the idea of multivariate prediction [39]. Instead of having a mapping function $h : \mathcal{X} \rightarrow \mathcal{Y}$ from a single example $x$ to its label $y$, where $x \in \mathcal{X}$ and $y \in \{-1, +1\}$, we look at all examples at once and try to learn a mapping function $\tilde{h} : \mathcal{X} \times \cdots \times \mathcal{X} \rightarrow \tilde{\mathcal{Y}}$, where $\tilde{\mathcal{Y}} = \{-1, +1\}^N$. We define $X = [x_1, \ldots, x_N]$, and $y = [y_1, \ldots, y_N]$.

We can define the best labeling using a linear discriminant function (scoring function)

$$
\tilde{h}(X) = \arg \max_{y' \in \tilde{\mathcal{Y}}} \mathbf{w}^T \Psi(X, y').
$$

(4.5)

Here, function $\Psi$ measures the compatibility of the data points and their assigned labels. If we define the $\Psi$ function as a simple form

$$
\Psi(X, y') = \sum_{i=1}^{N} y'_i x_i,
$$

(4.6)

that only depends on individual training points and their labels, the optimal labeling sequence would be

$$
\arg \max_{y' \in \tilde{\mathcal{Y}}} \mathbf{w}^T \Psi(X, y') = \arg \max_{y' \in \tilde{\mathcal{Y}}} \sum_{i=1}^{N} y'_i \mathbf{w}^T x_i
$$

$$
= (h(x_1), \ldots, h(x_N)),
$$

(4.7)

which is exactly the same as the optimal labeling in SVM.
One way of incorporating a loss function $\Delta$ in SVM formulation is *Margin Rescaling* [78],

$$\min_{w, \xi \geq 0} \|w\|^2 + C\xi$$

s.t. $\forall y' \in \hat{\mathcal{Y}} \setminus y$, $w^T [\Psi(X, y) - \Psi(X, y')] \geq \Delta(y, y') - \xi$

Similar to the original SVM formulation, $\xi$ in Eq. 4.8 is an upper bound on $\Delta(\hat{h}(X), y)$ [39].

The guarantee for convergence in polynomial time, the potential for incorporating complex loss functions in the objective and good performance in practice are the most important reasons why structured prediction has garnered much attention in computer vision recently.

In the standard approaches for solving Eq. 4.8, the output vector, $\tilde{y}$, corresponding to the most violated constraint should be found repeatedly [81],

$$\tilde{y} = \arg \max_{y' \in \hat{\mathcal{Y}}} \Delta(y, y') + w^T \Psi(X, y').$$

Finding $\tilde{y}$ is computationally challenging given an arbitrary loss function, $\Delta(y, y')$, and compatibility function, $\Psi(X, y')$. However, solving Eq. 4.9 in two special cases has been shown to be efficient. We categorize these approaches based on the simplicity of their $\Delta$ and $\Psi$ functions. We call a loss function simple if it can be decomposed into individual training samples. Likewise, a compatibility function is called simple if it decomposes over single sample points and their ground-truth labels.

### 4.3.3 Decomposable $\Delta$, Complex $\Psi$

Optimizing the parameters of a MRF structure when the loss function can be decomposed into the loss of individual samples falls into this category [78]. One popular application in this category is foreground-background segmentation with Hamming loss, which is defined as

$$\Delta_H = \sum_i I_{[y_i \neq y'_i]}.$$  \hspace{1cm} (4.10)

where, $I[]$ is the indicator function. Szummer et al. [74] have employed this formulation and reported promising results for interactive segmentation.

Decomposability of the loss function results in a MRF form for Eq. 4.9, because the loss function can be treated as another unary term that adds up to the unary terms of the compatibility function. Assuming binary labels, this MRF can be solved efficiently using
The advantage of this approach is to exploit pairwise connections, but it is only tractable for decomposable loss functions.

### 4.3.4 Non-decomposable \( \Delta \), Simple \( \Psi \)

The other special case presented by Joachims [39], is when the \( \Psi \) function has a simple form of

\[
\Psi(X, y') = \sum_{i=1}^{N} y'_i x_i.
\]

If the loss function, \( \Delta \), is just a function of true positive (TP), false positive (FP) and false negative (FN), then there are at most \( N_p \times N_n \) distinct loss values, where \( N_p \) and \( N_n \) represent the number of positive and negative training examples, respectively. Hence, Eq. 4.9 can be solved by iterating over all loss values and maximizing \( w^T \Psi(X, y') \) subject to the value of TP, FP and FN [39].

Unlike the approach of Taskar et al. [78], many standard accuracy measures that lead to non-decomposable loss functions, such as \( F_\beta \) score (natural language processing), intersection over union (object category segmentation), Precision/Recall at k (web search engines) and ROC area (binary classifiers) can be directly optimized by this approach. However, this method cannot benefit from the pairwise interactions of training samples, which are shown to be advantageous in many applications, such as object detection [20] and scene interpretation [38].

### 4.4 Solving Non-decomposable \( \Delta \), Complex \( \Psi \)

Discussing the advantages and shortcomings of the previous methods, we now propose an approach to directly optimize certain complex loss functions in a MRF. Here, we can optimize non-decomposable accuracy measures, such as \( F_\beta \) and intersection over union and still be able to benefit from pairwise interactions between training points. So far, we have considered only binary output problems for simplicity, but for the rest of the chapter the output is assumed to be multilabel. For notational convenience, we encode the output label, \( y \) in one-of-\( M \) format, where \( M = |\mathcal{L}| \) and \( \mathcal{L} \) is the set of all possible labels. In this encoding, the assigned label of \( y_i \) is represented using a binary vector of size \( M \) such that
its $j^{th}$ element is 1, when the $j^{th}$ label is assigned to this output, and the rest of its elements are 0.

We choose to follow the general framework of Structural SVM [81], shown in Eq. 4.8. Solving Eq. 4.8 requires finding the most violated constraint (Eq. 4.9) at each iteration and modifying the parameter vector $w$ accordingly. We propose a novel method to efficiently solve for an approximate most violated constraint for certain non-decomposable loss functions in presence of pairwise terms in the compatibility function, $\Psi$.

We can summarize the proposed approach as

1. Replacing the original non-decomposable loss function with a piecewise linear approximation,

2. Writing the problem of finding the most violated constraint as a quadratic program,

3. Converting the quadratic program to a linear program and solve the relaxed problem.

### 4.4.1 Piecewise Linear Approximation

Many standard accuracy measures, including the one presented in the previous section, share the property that they can be computed from the contingency table\(^1\). Given the number of positive and negative examples, $N_p$ and $N_n$, the loss function corresponding to these accuracy measures is just a function of $FP$ and $FN$. Using piecewise linear approximation, we can write

$$\Delta(FP, FN) \approx \hat{\Delta}(FP, FN)$$

$$= \sum_{r=1}^{Q} 1_{[(FP, FN) \in \mathcal{R}_r]} \{\alpha_r FP + \beta_r FN + \gamma_r\}$$

(4.12)

where, $Q$ is the number of subregions (pieces), $\alpha_r$, $\beta_r$ and $\gamma_r$ represent the $r^{th}$ plane coefficients and $\mathcal{R}_r$s are the subregions that partition the space spanned by $FP$ and $FN$.

As an example, Figure 4.1 illustrates the intersection over union loss function,

$$\Delta_\cap(FP, FN) = \frac{FN + FP}{N_p + FP},$$

(4.13)

along with its piecewise linear approximations using 15 and 40 pieces.

---

\(^1\)Are just a function of $TP$, $FP$, $TN$ and $FN$. 

CHAPTER 4. PLANAR APPROXIMATION

Figure 4.1: Intersection over union loss surface in $FP$ and $FN$ space. a) Exact surface, b) a piecewise linear approximation with 40 subregions, c) a piecewise linear approximation with 15 subregions.

Given the subregion $R_r$, the original non-linear loss function is a linear function of $FP$ and $FN$. The next step is to substitute the approximated loss function, $\tilde{\Delta}$ into Eq. 4.9 and solve for the most violated constraint.

4.4.2 Forming the Quadratic Program

Capturing the structure of the output requires a model that is rich enough to absorb the dependencies between the outputs. At the same time, a preferred model candidate offers tractable inference procedure. A choice that satisfies both requirements are MRFs, which are commonly used for modeling interdependent inputs and outputs in many applications.

We assume that we are given a MRF represented by a graph $G = (V, E)$ where $V$ is the set of nodes with $N = |V|$, and $E$ is the set of edges. The output label takes value from the set $\mathcal{L}$, which has $M$ members. We define our $\Psi$ with unary and pairwise terms as

$$\Psi(X, Y) = \sum_{i=1}^{N} \sum_{k=1}^{M} y_{ik} \phi_u(x_i) + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \sum_{k=1}^{M} \sum_{l=1}^{M} y_{ik} y_{jl} \phi_p(x_i, x_j).$$

(4.14)

Here $\mathcal{N}_i$ is the set of neighbors of sample $i$. We later explain how we define the unary and pairwise features ($\phi_u$ and $\phi_p$) in our experiments. We rewrite Eq. 4.9 with approximated
loss function, $\tilde{\Delta}$ as

$$
\hat{Y}^* = \arg \max_{Y'} \tilde{\Delta}(Y, Y') + w^T \Psi(X, Y')
$$  \hspace{1cm} (4.15)

$$
= \arg \max_{Y'} \tilde{\Delta}(Y, Y') + w^T_{(u)} \sum_{i=1}^{N} \sum_{k=1}^{M} y'_{ik} \phi_{(u)}(x_i)
+ w^T_{(p)} \sum_{i=1}^{N} \sum_{j \in N_i} \sum_{k=1}^{M} \sum_{l=1}^{M} y'_{ik} y'_{jl} \phi_{(p)}(x_i, x_j)
$$  \hspace{1cm} (4.16)

where $w = [w_{(u)}; w_{(p)}]$ (concatenation of unary (u) and pairwise (p) parameters).

The group of non-decomposable loss functions that are considered in the proposed approach are a function of false positive and false negative counts. Although the definition of false positive and false negative counts are straightforward in binary output problems, for multi-label problems such definitions are task-dependent. In this chapter, we assume that the loss is defined for one label (label $p$) versus the rest and therefore, define the false positive and false negative counts as

$$
FP_{Y, Y'} = \sum_{i=1}^{N} y'_{ip} \sum_{k \in \{1, \ldots, M\} \backslash p} y_{ik},
$$  \hspace{1cm} (4.17)

$$
FN_{Y, Y'} = \sum_{i=1}^{N} y_{ip} \sum_{k \in \{1, \ldots, M\} \backslash p} y'_{ik}.
$$  \hspace{1cm} (4.18)

Assuming that the loss values fall in subregion $R_r$, we can write Eq. 4.16 as

$$
\hat{Y}^* = \arg \max_{Y'} \left( \alpha_r \sum_{i=1}^{N} y'_{ip} \sum_{k \in \{1, \ldots, M\} \backslash p} y_{ik} + \beta_r \sum_{i=1}^{N} y_{ip} \sum_{k \in \{1, \ldots, M\} \backslash p} y'_{ik} + \gamma_r +
\begin{align*}
& w^T_{(u)} \sum_{i=1}^{N} \sum_{k=1}^{M} y'_{ik} \phi_{(u)}(x_i) + \\
& w^T_{(p)} \sum_{i=1}^{N} \sum_{j \in N_i} \sum_{k=1}^{M} \sum_{l=1}^{M} y'_{ik} y'_{jl} \phi_{(p)}(x_i, x_j) \right).
$$  \hspace{1cm} (4.19)

Note that Eq. 4.19 only includes the predicted label $y'$ in linear and quadratic forms. Hence,
we can write a quadratic program based on Eq. 4.19 subject to the loss values being in subregion $R_r$.

$$\left( \sum_{i=1}^{N} y'_{ip} \sum_{k \in \{1, \ldots, M\} \setminus p} y_{ik}, \sum_{i=1}^{N} y_{ip} \sum_{k \in \{1, \ldots, M\} \setminus p} y'_{ik} \right) \in R_r$$

In order to have linear constraints in Eq. 4.20, the boundary of all subregions should be definable as a linear function of $y'$. One way is to separate the subregions by straight lines. If for example, we partition the space spanned by $FP$ and $FN$ into triangles (Fig. 4.1-b,c) then Eq. 4.20 will be substituted by three linear constraints corresponding to the three sides of the triangle.

### 4.4.3 Converting Quadratic Program to Linear Program

The quadratic function in Eq. 4.19 is potentially non-convex, since there is no constraint on the coefficients of this function. So, instead of looking for a local optima of this non-convex function, we relax the problem (MAP-MRF LP relaxation [89]) by introducing some variables that substitute the quadratic terms in the this function and form a linear program, which is convex. In detail, we introduce $y'_{ij} = y'_{ik}y'_{jl}$. To relate these new variables to the output variables $y'$, we augment some linear inequality constraints in the form, $y'_{ij} \leq y'_{ik}$, $y'_{ij} \leq y'_{jl}$ and $\sum_{k,l} y'_{kl} = 1$. The final linear program that needs to be solved for subregion...
CHAPTER 4. PLANAR APPROXIMATION

Maximize:

\[
\alpha_r \sum_{i=1}^{N} y_{ip} \sum_{k \in \{1, \ldots, M\} \setminus p} y_{ik} + \beta_r \sum_{i=1}^{N} y_{ip} \sum_{k \in \{1, \ldots, M\} \setminus p} y_{ik} + \gamma_r + \\
\sum_{i=1}^{N} \sum_{k=1}^{M} y_{ik} \phi(u_i) + \\
\sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \sum_{k=1}^{M} \sum_{l=1}^{M} y_{ij}^{ kl} \phi(p)(x_i, x_j)
\]

Subject to:

\[
\left( \sum_{i=1}^{N} y_{ip} \sum_{k \in \{1, \ldots, M\} \setminus p} y_{ik}, \sum_{i=1}^{N} y_{ip} \sum_{k \in \{1, \ldots, M\} \setminus p} y_{ik} \right) \in \mathcal{R}_r \\
\sum_{k, l} y_{ij}^{ kl} \leq y_{ik}, \quad y_{ij}^{ kl} \leq y_{jl} \\
\sum_{k} y_{ij}^{ kl} = 1, \quad \sum_{k} y_{ik}^{ kl} = 1 \\
y_{ij}^{ kl}, y_{ij}^{ kl} \in \{0, 1\}, \quad i \in \{1, \ldots, N\}, \quad j \in \mathcal{N}_i, \quad k, l \in \mathcal{L}
\]

Solving this LP for thousands of binary variables (labels), is not computationally tractable. So instead we relax the label values to real numbers between zero and one and solve for optimal labeling. Later, we map the optimal labels to integer values, if necessary, by rounding the results. We solve Eq. 4.21 for each subregion separately, and return the labeling of the one with the maximum objective value as the most violated constraint.

### 4.4.4 Inference

The inference procedure concerns about maximizing the assignment score \(w^T \Psi(X, Y)\) over different assignments and is MAP-MRF problem. For general multilabel problems this task is shown to be NP hard (see [9]), but many approximate inference approaches has been proposed to solve this problem [89, 44]. However, for a supermodular binary problem efficient min-cut/max-flow algorithms exist that can solve the inference exactly [10].

For many applications, not only the maximizing assignment, but also the confidence score associated to each element’s assignment is required. Given the confidence scores,
compromising FP versus FN or vice versa is possible.

One way of computing a confidence score for the $i^{th}$ element of the output is by looking at the difference in the scores, when the $i^{th}$ output is assigned to the positive label $p$ comparing to when it is assigned to any other label, while the rest of the output elements get their best assignments [20]. Formally,

$$s_i = \max_{Y', y_p=1} w^T \Psi(X, Y) - \max_{Y', y_p \neq 1} w^T \Psi(X, Y)$$  \hspace{1cm} (4.22)

Each maximization in Eq. 4.22 is a MAP estimate in the MRF. For our segmentation experiments, we only require the best labeling, but for our action retrieval experiment the score is needed to compare different approaches.

### 4.5 Experiments

To highlight the superiority of the proposed approach over two existing alternatives – keeping the model non-decomposable, but optimizing against a decomposable loss function, or keeping the loss function non-decomposable, but forgetting about the dependency between outputs and employing decomposable models – we design a set of experiments. We compare the methods on two applications – object category segmentation and action retrieval. The rest of this section explains the details of our experiments.

#### 4.5.1 Learning Method

We utilize Non-convex Regularized Bundle Method (NRBM) [21] – an instance of a bundle method – as the core of our learning and solve the loss augmented inference based on the proposed approach. NRBM solves the unconstrained form of Eq. 4.8

$$\min_w \frac{\lambda}{2} \|w\|^2 + \max_{Y'} (w^T \Psi(X, Y') + \Delta(Y', Y)) - w^T \Psi(X, Y)$$  \hspace{1cm} (4.23)

Note that other structured prediction formulations such as Pegasos [70] or the formulation proposed by Meshi et al. [57] could easily replace the bundle method. We chose NRBM due to implementation simplicity knowing that it has the same bound of $O(1/\epsilon)$ like afore-mentioned alternatives to obtain a solution of accuracy $\epsilon$. We do cross validation to set the $\lambda$ parameter in Eq. 4.23. To be able to solve the inference as well as the loss augmented
inference exactly, for the first set of experiments involving binary output labelings, we add $w_{(p)} \preceq 0$ constraints similar to Szummer et al. [74] to make the function supermodular.

### 4.5.2 Mesh Creation

The main idea of this chapter is to approximate the loss function with a piecewise planar function in false positive and false negative space, in which the loss function is assumed to live. The process of computing the piecewise planar approximation is offline and could be performed using many approaches. We choose to start with a dense mesh in false positive and false negative space and employ mesh simplification methods to reduce the number of pieces to the desired number. To keep the domain of the loss function intact, the boundary of the original mesh and the simplified version should be the same. The minimum number of pieces that a mesh can be simplified to before violating this property is dictated by the simplification method and the primitive shapes that create the mesh.

The other alternative for creating the approximate mesh is to fix the number of vertices and initialize the approximate surface, e.g. on a grid. Then, try to minimize the distance between the original and approximated meshes. The distance function is non-convex in most cases and it is hard to find its global minimum. Instead, general techniques such as gradient descent could be employed to find a local minimum of this function.

In our experiments, we exploit MeshLab [58] and set the number of pieces to 15, which is the lowest that respects the boundary condition for all of the loss surfaces in the experiments. We tried higher numbers of pieces, but did not notice significant improvement in the overall accuracy. An example of an original densely created loss function along with its approximated versions are shown in Fig 4.1. We use “quadric edge collapse” technique in MeshLab, which simplifies the mesh based on the method of Gerland et al. [31].

### 4.5.3 Baseline Methods

We implement two baseline approaches to compare against, each including one aspect of our proposed method. The first baseline, which we name “Hamming”, consists of our model, but optimized against Hamming loss, a decomposable loss function that is used widely for structured prediction [78, 82, 74]. Hamming loss is defined as

$$\Delta_{\text{Hamming}} = \pi_1 FP + \pi_2 FN$$  \hspace{1cm} (4.24)
where, $\pi_1$ and $\pi_2$ adjust the contribution of FP and FN in the overall loss. We set $\pi_1 = \frac{1}{2N_n}$ and $\pi_2 = \frac{1}{2N_p}$ to equalize the contribution of FP and FN in our highly imbalanced dataset. Here, $N_p$ and $N_n$ represent the number of positive and negative examples in the training set. Solving the loss augmented inference given this loss function is as hard as solving the inference problem, because the loss is augmented to each node in the graph as a unary term. Comparison to this baseline reveals the importance of the proposed learning framework, which lets us optimize against non-decomposable loss functions.

To show the importance of the structure in the model (smoothing in segmentation and intra-frame and inter-frame interactions in action retrieval), we implement the approach of Joachims [39]. This approach can exactly optimize against multivariate non-decomposable performance measures, the ones that can be approximately optimized using the proposed approach, but only for decomposable models. We remove the pairwise interactions from the model and train the model parameters using only the unary features. We call this approach “Unary” in the results.

### 4.5.4 Object Category Segmentation

We employ object category segmentation as an example of a structured output problem with binary outputs. The task is to label the pixels of an image as being part of a known object (foreground) or not (background). We set the label of foreground to one and the label of background to zero. One example of a non-decomposable performance measure in the object category segmentation task is intersection over union, defined as

$$\text{Acc}_\cap(FP, FN) = \frac{N_p - FN}{N_p + FP} \iff \Delta_\cap(FP, FN) = \frac{FP + FN}{N_p + FP},$$

which has been used to compare segmentation accuracies on the Pascal VOC challenge [23]. For object category segmentation we optimize against this loss function while comparing to the baselines on three datasets – Pascal VOC 2009, Pascal VOC 2010 and H3D. Solving the MAP inference and the loss augmented inference exactly requires a supermodular scoring function. So, for this experiment we guarantee supermodularity by forcing the weights corresponding to pairwise features to be negative, knowing that the pairwise features are always positive.
Table 4.1: Maximum achievable accuracy percentage in VOC 2009, VOC 2010 and H3D datasets due to superpixelization.

<table>
<thead>
<tr>
<th></th>
<th>Aeroplane</th>
<th>Bus</th>
<th>Car</th>
<th>Horse</th>
<th>Person</th>
<th>TV/monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOC09</td>
<td>73.17</td>
<td>85.34</td>
<td>77.2</td>
<td>66.48</td>
<td>74.88</td>
<td>86.2</td>
</tr>
<tr>
<td>VOC10</td>
<td>72.63</td>
<td>82.24</td>
<td>77.67</td>
<td>68.37</td>
<td>74.06</td>
<td>85.78</td>
</tr>
<tr>
<td>H3D</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>79.11</td>
<td>-</td>
</tr>
</tbody>
</table>

4.5.5 Pixels vs. Superpixels

If we decide to perform segmentation on the pixel level, meaning that the input be the set of all features extracted from all pixels in the dataset and the output be the binary label of each pixel, then for Pascal VOC 2009 dataset we would have 133,567,772 pixels and the same number of nodes in our MRF. On average each node in our graph has about 4 neighbors, which would create around $535 \times 10^6$ edges in the graph. Learning the parameters on this huge graph is intractable both for the baseline methods and the proposed approach. Moreover, the features extracted from a group of neighbouring similar pixels are perhaps more robust comparing to the features extracted from single pixels. As an alternative, neighbouring pixels that share similar appearance features could form a group (superpixel) and share the same label. The downside of moving from pixel to superpixel is the possibility that the pixel of a superpixel come from both foreground and background. In this case, the maximum achievable accuracy drops.

In our experiments we employ the superpixel extractor of Felzenszwalb et al. [25] and set its parameters to $\text{MinArea} = 2000$, $k = 200$, $\sigma = 0.01$. This setting of parameters result in an average of 50 superpixels per image of size $300 \times 500$ pixels. Using this parameters the number of nodes in the graph decreases from about $134 \times 10^6$ to approximately $27 \times 10^3$ nodes in Pascal VOC 2009 dataset and from about $190 \times 10^6$ to approximately $24 \times 10^3$ nodes. However, as explained before, the maximum possible accuracy drops from 100% to the numbers reported in Table 4.1.

The second baseline assumes that all positive examples (superpixels of the foreground) contribute equally in the loss function, which is not true if the areas of the superpixels are different\(^2\). On the other hand, this approach has $O(N^2)$ complexity, when $N$ is the

\(^2\)The other alternative is to force the superpixels to have the same size, but then large flat regions such as sky would be broken into many small superpixels and regions of small objects could be grouped with background regions.
number of nodes in the graph and clearly is not tractable if working on pixels. Assuming the same features for each pixel of a superpixel, we have modified Joachims [39] algorithm to work on superpixels as follows. Instead of sorting the superpixels based on their scores, we sort superpixels by their scores divided by their areas. We also adjust the value of the loss function based on the area of the superpixels. This approach is guaranteed to produce correct labelings for all superpixels except possibly one foreground and one background superpixels.\footnote{Algorithm 2 of [39] has been proven to find the optimal assignment if all positive examples contribute equally in the loss function and also do all negative examples. Based on Algorithm 2 of [39], the first $a$ positive examples get value 1 and the rest get value 0 at the optimal $v$. Knowing that the pixels of a superpixel are sorted sequentially, the only superpixel that may have inconsistent labels is the one that its pixel is located at position $a$. The same argument holds for background superpixels. So, all pixels of other superpixels get the same labels as they would get if we could afford to run the algorithm on pixels.}

Features

We define an MRF segmentation model with unary and pairwise features, for which the exact inference is performed using min-cut/max-flow algorithm [10].

The structure in the model comes from pairwise interactions between neighboring superpixels $i$ and $j$. We define a set of pairwise features that represent $\phi_p(x_i, x_j)$ in Eq. 4.21. We first convert the image from $RGB$ to $La^*b^*$ color space. We define $L_i$, $a_i$ and $b_i$ to be the average $L$, $a$ and $b$ values inside superpixel $i$, respectively and assign the length of the common boundary between superpixel $i$ and $j$ to $P_{ij}$. We then compute the pairwise features as

$$\phi_p(x_i, x_j) = P_{ij} \exp \left[ -\tau_1 (L_i - L_j)^2, -\tau_2 (a_i - a_j)^2, -\tau_2 (b_i - b_j)^2 \right]$$

In our experiments the values of $\tau_1$ and $\tau_2$ are set to $2 \times 10^{-2}$ and $5 \times 10^{-3}$, respectively.

To represent each superpixel, we use a set of bottom-up and top-down features, which form the unary features $\phi_u(x_i)$ for superpixel $i$ in Eq. 4.21. To create the bottom-up features, we compute Color SIFT features [85] on a dense grid with 6 pixel spacing in horizontal and vertical directions. We then turn this into a bag-of-words representation using a codebook of 1000 visual words.

For top-down features, we take a similar approach to the implicit shape model [50]. We first learn two appearance models for each of the 6 object categories using the detector of
Felzenszwalb et al. [24]. The result includes two root filters and $6 \times 2$ part filters, where each root filter and 6 corresponding part filters model the object appearance in one pose. We run this detector on the training set and collect all bounding boxes that have positive scores. We then crop the ground-truth images on the bounding box locations and compute the average shape for the roots and parts, Fig 4.2.

We explain the rest of the process for one part, but the same process is applied to all parts and both roots. We find the potential part locations and their confidences by running the detector on the image in different scales. We call the result at each scale a confidence map, Fig. 4.3-b. Each potential part location casts its vote for the shape of that part proportional to its confidence. We implement this by convolving the confidence maps (different scales) with the average shape for that particular part. We call the convolution result in each scale a potential mask, Fig. 4.3-c. To merge the potential masks, we rescale them to the original image size and get the maximum of the masks, Fig. 4.3-d. We accumulate the mask values inside each superpixel to form the top-down feature corresponding to the part. Fig. 4.3 depicts the entire process for one part.

**Pascal VOC 2009 and 2010 Segmentation Datasets**

The Pascal VOC 2009 dataset includes 749 pixel-level labeled training images and 750 validation images. The Pascal VOC 2010 dataset includes 964 training and 964 validation images. We decide to train our method on the training set and test on the validation set, because the ground-truth for the test set is not publicly available and our focus is on comparison to baseline methods using a different model or learning criterion. We present the
results on 6 object categories, Aeroplane, Bus, Car, Horse, Person, and TV/Monitor. We select these categories because the top-down unary features obtained from the Felzenszwalb et al. object detector [24] provide reasonable detection on them. Without the top-down features, the overall accuracy would be so low as to make the comparison between different learning methods uninformative. Note that we perform the experiments on these objects independently. For example, when we segment object class car, any other object is taken as background. This is different from the VOC segmentation challenge in which the segmentation result should contain all object classes simultaneously. One of the most challenging aspects of these datasets is the ratio of foreground to background pixels for all categories (Table 4.2). Moreover, the images in these datasets are not taken in a controlled environment and include severe illumination and occlusion. We compare the proposed approach to the baselines on the 6 object categories in Figure 4.4. As illustrated, the proposed approach significantly outperforms the baselines on this dataset.

Moreover, the results of “Unary” in most cases except for “aeroplane” class is superior to “Hamming”, which suggests optimizing against the right performance measure is more important than smoothing the assignments in this dataset.

Table 4.2: Background to foreground pixel ratio in Pascal VOC 2009 and 2010

<table>
<thead>
<tr>
<th></th>
<th>Aeroplane</th>
<th>Bus</th>
<th>Car</th>
<th>Horse</th>
<th>Person</th>
<th>TV/Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOC 2009</td>
<td>163</td>
<td>69</td>
<td>86</td>
<td>130</td>
<td>24</td>
<td>96</td>
</tr>
<tr>
<td>VOC 2010</td>
<td>168</td>
<td>64</td>
<td>70</td>
<td>119</td>
<td>26</td>
<td>105</td>
</tr>
</tbody>
</table>
We compare the effect of optimizing against adjusted Hamming loss versus intersection over union in Fig. 4.5. Adjusted Hamming loss tends to return fewer false positives, but with the cost of missing many true positives. In fact, it often marks all pixels as background, while intersection over union actually produces segmentations. That is because when the entire image is labeled as background the adjusted Hamming loss results 1/2 loss while intersection over union loss results 1.

**H3D Dataset**

We also compare the results on the H3D dataset [8]. This dataset includes 273 training and 107 testing images along with three types of annotations – keypoint annotations, 3d pose annotation and region annotation. The keypoint annotation includes the location of joints and other keypoints such as eyes, nose, elbows, etc. The 3d pose annotation has been inferred from the keypoints. The region annotation, which we use in this chapter, provides detailed annotation of people, such as face, neck, lower and upper cloth, etc. For our experiments we compute the union of all region annotations that are part of a person.
Chapter 4. Planar Approximation

Figure 4.5: Segmentation for person category. Optimizing adjusted Hamming loss (“Hamming”) against our proposed method. a) input image, b) segmentation considering adjusted Hamming loss (“Hamming”), c) our proposed method employing intersection over union. Intersection over union provides more true positives by possibly creating some false positives. Adjusted Hamming loss decreases false positive by sacrificing some true positives.

Figure 4.6: Intersection over union performance (%) comparison on H3D dataset

(bags, occluder and hat are not considered as parts of a person) as foreground and the rest as background. The ratio of background to foreground pixels in this dataset is 3.9, which is significantly lower than the ratio in Pascal VOC datasets. The reason is that all images in H3D dataset include at least some foreground pixels, which is not the case in Pascal VOC datasets. The comparison result in Figure 4.6 shows that the proposed approach outperforms the baselines significantly on this dataset. We also show some segmentation results on H3D dataset in Figure 4.11.

4.5.6 Action Retrieval

The second application that we consider in this chapter is action retrieval. The task is to find actions that are similar to the query action in video frames. In this experiment we are
interested in detection and localization of the query action. Action retrieval is an important
problem with numerous real-world applications such as multimedia content analysis and
surveillance and security systems. In the experiments we explore a surveillance application,
automated analysis of nursing home video footage. We would like to find actions of interest
– for instance residents falling down, sitting, or standing up. The offline batch processing
setting is of interest to clinicians studying the behaviours of nursing home residents. For
instance, this setting is useful for gathering data on the circumstances of injurious falls by
residents, or mobility measures for residents.

The choice of loss function is arbitrary in our learning framework as long as it remains
a function of false positive and false negative counts. A widely used performance measure
for retrieval tasks is precision on the first $K$ retrieved elements, termed precision at $K$. This
measure represents what we care about when performing retrieval in many applications
– one wants to maximize the number of relevant events of interest in a fixed number of
retrieved videos. The loss associated with the precision is defined as

$$\Delta_{\text{Prec}} = 1 - \text{Precision} = \frac{FP}{N_p + FP - FN},$$

(4.26)

Here, $N_p$ is the number of positive examples (people with ground-truth label equal to the
query label). In our action retrieval task, all detections that have the same label as the
query action are considered positive and all other detections are negative. So, false positive
and false negative counts are computed using Eq. 4.17 and 4.18, respectively.

**Model**

We describe our model for action retrieval in a sequence of video frames. We assume that
a set of person locations in each video frame has been provided via a human detection
algorithm. The goal is to automatically retrieve the people in a video who perform a query
action. We believe there are correlations between the actions of different people in a scene
and try to capture these interactions in our action retrieval model.

The model we develop is depicted in Fig. 4.7. Our model is a Markov Random Field
(MRF), where each detection corresponds to a node (site) in the graph (shown in blue).
There are three types of edges in the graph, shown in red, green and yellow. Red edges
denote the relationship between assigning different labels to each node given the video
features describing the corresponding detection. These edges form the unary potentials in
our MRF.

The other two edge types model intra-frame and inter-frame correlations between actions. The types of interaction between people in one frame and people in consecutive frames are different. Intra-frame interactions are about which actions are likely to co-occur. On the other hand, inter-frame interactions model the smoothness of people’s actions over time. To differentiate the two types of interactions, two groups of pairwise interactions are included in the model, intra-frame interactions and inter-frame interactions, shown in green and yellow, respectively. An edge between two nodes holds a vector of scores corresponding to every possible combination of action labels for its nodes.

Let \( \phi(x_i) \) be the feature vector for \( i^{th} \) detection and \( \mathcal{L} \) be the set of all possible action labels, with \( M \) elements. For notational convenience, we encode the action label in a one-of-\( M \) format.

**Action Appearance Potential \( \theta \):** The appearance score for the \( i^{th} \) node in the graph is formulated as:

\[
\theta(x_i, y_i, w_{(u)}) = \sum_{k=1}^{M} w_{(u)k} y_{ip} \phi(x_i),
\]

(4.27)
CHAPTER 4. PLANAR APPROXIMATION

Later, in Section 4.5.6 we describe how we compute the appearance features $\phi(x)$.

**Intra-frame Action Potential $\rho$:** The pairwise action-action scores are only a function of the action labels at two neighbouring nodes with no ordering (symmetric). Under these assumptions, there will be $M(M + 1)/2$ parameters. The intra-frame interaction scores between nodes $i$ and $j$ can be written as

$$
\rho(y_i, y_j, w_{(p_1)}) = \sum_{k=1}^{M} \sum_{l=k}^{M} w_{(p_1)k+(l-k)M} y_{ik} y_{jl}
$$

(4.28)

Essentially, $w_{(p_1)}$ parameters encode which actions are likely to appear together in a frame.

**Inter-frame Action Potential $\mu$:** Similarly, inter-frame interaction scores can be formulated as

$$
\mu(y_i, y_j, w_{(p_2)}) = \sum_{k=1}^{M} \sum_{l=1}^{M} w_{(p_2)k+lM} y_{ip} y_{jq}
$$

(4.29)

with a different set of parameters $w_{\mu}$ scoring pairs of action labels in consecutive video frames. Note that the transition between actions of a person is not symmetric (walking to falling vs. falling to walking), which results in $M^2$ parameters for inter-frame potentials in our model.

The overall model score aggregates these cues over a video sequence, defined as

$$
S(X, Y, w) = \sum_{i=1}^{N} \theta(x_i, y_i, w_{(u)}) + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \rho(y_i, y_j, w_{(p_1)}) + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \mu(y_i, y_j, w_{(p_2)}),
$$

(4.30)

where $N$ is the number of nodes in the model (number of detections), $\mathcal{N}_1$ and $\mathcal{N}_2$ are the set of pairs of neighbouring nodes in intra-frame and inter-frame connections, respectively.

**Person Detection and Description**

We implement a simple method for person detection that proves to be reasonably effective for our dataset. We extract moving regions from the videos using the OpenCV implementation.
Figure 4.8: Illustration of the local spatio-temporal (LST) feature representation for describing a candidate region. \( \mathbf{u} \) is a vector of percentage of static foreground pixels, \( \mathbf{v} \) is a vector of percentage of moving foreground pixels. See text for details.

of the standard Gaussian Mixture Model (GMM) [73]. Moving regions with area less than a threshold (500 pixels in our experiments) are deemed unreliable and therefore ignored. In the training set we manually label the output of the detection process from the set of possible actions, which includes the “unknown” action to label the false positives. At test time, we detect people using the same process, extract their features and then recognize their actions.

In our surveillance dataset, widely used features such as optical flow or HOG [18] are typically not reliable due to low video quality. Instead, we use the local spatio-temporal (LST) descriptor [53], which has been shown to be reliable for low spatial and temporal resolution videos (Figure 4.8). The feature descriptor is computed as follows. We first divide the bounding box of a detected person into \( N \) blocks. In the experiments we use a 10×10 grid to obtain 100 blocks for each detection. Foreground pixels are detected using background subtraction. Each foreground pixel is classified as either static or moving by frame differencing. Each block is represented as a vector composed of two components: \( \mathbf{u} = [u_1, \ldots, u_t, \ldots, u_\tau] \) and \( \mathbf{v} = [v_1, \ldots, v_t, \ldots, v_\tau] \), where \( u_t \) and \( v_t \) are the percentage of static and moving foreground pixels at time \( t \) respectively. \( \tau \) is the temporal extent used to represent each moving person, which has been set to 5 frames in our experiments.
Table 4.3: Number of detected people in the training and test sets for each action

<table>
<thead>
<tr>
<th></th>
<th>unknown</th>
<th>walk</th>
<th>stand</th>
<th>sit</th>
<th>bend</th>
<th>fall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>626</td>
<td>331</td>
<td>454</td>
<td>291</td>
<td>38</td>
<td>20</td>
</tr>
<tr>
<td>Test</td>
<td>877</td>
<td>330</td>
<td>163</td>
<td>199</td>
<td>13</td>
<td>15</td>
</tr>
</tbody>
</table>

Nursing Home Dataset

We have collected a dataset of 13 video clips from a surveillance camera in a nursing home recorded at 3 frames per second and spatial resolution of 640×480 pixels [47]. The size of the clips in the dataset varies from 94 to 234 frames. The action label set includes 6 actions, unknown, walk, stand, sit, bend and fall. We use 7 clips for training and the remaining 6 clips for testing. After running the detector on all video clips, we manually label all detected bounding boxes. These bounding boxes are employed for training and testing. The summary of the number of detections for each action in the dataset is presented in Table 4.3. Note that the actions are highly imbalanced and there are only a few detected people with fall, bend actions. We choose the action query label from a subset of these actions – walk, stand, sit, bend and fall.

We fix the value of $K$ to $N_p$ in the experiments and compare three approaches based on precision at $K$ retrieved items. The results are shown in Table 4.4. The proposed approach outperforms the other approaches for all the actions except bending, which has the fewest instances in the test set.

We visualize the intra-frame and inter-frame interaction weights in Fig. 4.9. One interesting observation is the positive intra-frame weight between bending and walking while looking for the walking action. The bending action usually happens in the nursing home dataset when a nurse is helping an elderly resident who has fallen. In this scenario another nurse is very likely to come to help, who performs the walking action. As another example, a person is very unlikely to switch his action from walking to sitting and vice-versa (Fig 4.9 right). Also, repeatedly performing the same action over time is likely for all actions except abrupt actions like falling.
Figure 4.9: Visualization for some of the learned intra-frame (left) and inter-frame (right) interactions. Vertical labels are the query actions (walk (W), Stand(St), Sit(Si), Bend(B) and Fall(F)). The inter-frame interactions are asymmetric, which is shown as two weights one from query action to the other actions (left half) and from the other actions to the query action (right half).

Table 4.4: Precision percentage at $K = N_p$

<table>
<thead>
<tr>
<th></th>
<th>walk</th>
<th>stand</th>
<th>sit</th>
<th>bend</th>
<th>fall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamming</td>
<td>49.8</td>
<td>17.1</td>
<td>34.2</td>
<td>0.0</td>
<td>11.1</td>
</tr>
<tr>
<td>Unary</td>
<td>53.5</td>
<td>16.3</td>
<td>38.0</td>
<td>8.8</td>
<td>11.1</td>
</tr>
<tr>
<td>Our</td>
<td><strong>55.2</strong></td>
<td><strong>17.8</strong></td>
<td><strong>38.4</strong></td>
<td>2.9</td>
<td><strong>22.2</strong></td>
</tr>
</tbody>
</table>

4.6 Summary

In this chapter we developed a general algorithm for addressing learning problems with complex models and complex loss functions, those which are a function of false positive and false negative counts. We replace the original non-decomposable loss function with a piecewise linear approximation, and solve it using a linear programming relaxation of the original quadratic program.

In future work it would be interesting to analyze the quality of these approximations. However, in this work we have provided experimental evidence of their effectiveness. In particular we apply this method to learning an object category segmentation model that contains both unary terms for labeling pixels and pairwise terms on the labels of neighbouring pixels. We show that learning the parameters to this model under an objective directly
tied to the performance measure significantly improves performance relative to baseline algorithms on the PASCAL VOC Segmentation Challenges and H3D datasets. Moreover, we proposed a new model for action retrieval that can capture three sources of information: body motion, intra-frame action interaction and inter-frame action interaction. We showed empirically that the proposed approach can significantly improve on two strong baselines, one including our structured model of all actions in a scene, but optimizing decomposable Hamming loss (similar to the one explained in the previous chapter); and the other one optimizing the desired loss function, but without any interaction between different people’s actions.
Figure 4.10: Some segmentation results on Pascal VOC 2009 dataset. Each row corresponds to one object category.
Figure 4.11: Some segmentation results on H3D dataset.
Chapter 5

Non-Decomposable Loss Optimization using Dual Decomposition

In the previous chapter we proposed a planar approximation strategy to decompose the original non-decomposable loss function. In this chapter, we propose an alternative approach for solving the same problem – optimizing against non-decomposable loss functions in structured prediction when the model is a Markov network. The main idea is based on dual decomposition, in which we break the dual into two pieces – a supermodular MRF and the loss function augmented by a linear term. We then optimally solve both subproblems and force their results to agree by introducing some Lagrange multipliers. We show empirically that this approach produces comparable results to our previous method of planar approximation, but is significantly faster. It is worth mentioning that our planar approximation strategy could handle linear constraints in the loss function and is fully parallelizable unlike this fast dual decomposition method. This will appear in IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2012.
5.1 Overview and Related Works

Max-margin structured prediction is a common framework for a variety of vision problems. Tasks such as action recognition [88, 36], image segmentation [74], and scene understanding [20] involve reasoning about relationships between locations in a video, pixels in an image, and objects in a scene respectively. In addition, the performance measures for these applications often are non-decomposable and are not a simple sum of terms measured over individual output entities. Instead, they measure performance as a function of the entire, structured output. The focus of this paper is developing a learning approach that can handle these together, formulating a learning algorithm based on dual decomposition for handling structured prediction while optimizing against certain non-decomposable performance measures.

Capturing the structure of the output requires a model that is rich enough to absorb the dependencies between the outputs. In this chapter we employ Markov networks, which are commonly used for modeling interdependent inputs and outputs in many applications.

We focus on a subset of non-decomposable loss functions, namely, multivariate non-linear performance measures\(^1\), such as \(F_1\) (image retrieval), area under ROC (binary classification), or intersection over union (image segmentation). However, our approach can be applied more generally, for optimizing against any loss function for which the maximum value of the loss plus a linear function can be computed efficiently. One example of such a loss is mean average precision [94].

Modeling dependencies between outputs while optimizing against a loss function has been a research topic for many years. Optimizing the expected loss in this scenario is a non-convex problem. However, Taskar et al. [78] and Tsochantaridis et al. [82] have proposed rather to optimize a convex relaxation of the expected loss. The cutting-plane algorithm has been shown to be efficient for solving this optimization [82]. Teo et al. [80] presented a bundle method, which is basically the cutting-plane method stabilized with Moreau-Yosida regularizer and prove a tighter bound on the duality gap. Taskar et al. [79] solves the same problem using the extragradient method. Extragradient consists of a gradient descent followed by a projection to the feasible set. Shalev-Shwartz et al. [70] proposed Pegasos, which works solely in the primal space. Similar to [79], Pegasos consists of a gradient descent step followed by a projection step. The computational difficulty in all aforementioned structured

\(^1\)losses that are a function of false positive and false negative counts
prediction approaches is finding the subgradient, which requires solving the “most violated constraint” [82] or “loss augmented inference” [77]. It is shown that for decomposable performance measures learning is tractable when the model is a submodular Markov network or a matching [78, 82, 79]. In contrast, in this chapter we focus on non-decomposable performance measures.

Joachims [39] proposed an approach to efficiently compute the most violated constraint for a large class of non-decomposable loss functions, a subset of those we consider in this chapter. However, the underlying models were limited, and do not permit pairwise interactions between output labels. The method of Yue et al. [94] takes a similar approach to optimize against Mean Average Precision. Khanna et al. [13] present an algorithm in the same framework to optimize against normalized discounted cumulative gain (NDCG). Rather than solving a convex relaxation of the expected loss, McAllester et al. [56] proposed a perceptron-like training approach to directly optimize the original loss function, but still need to solve the loss augmented inference. For the problems in which the inference procedure is not tractable, Finley et al. [26] compare under-generating and over-generating algorithms in structured prediction and conclude that “overgenerating methods [LP and graph cut] have theoretic advantages over undergenerating [LBP, greedy] methods”.

Four closely related pieces of work in the literature are the approaches of Meshi et al. [57], Komodakis [42], Tarlow and Zemel [75] and Ranjbar et al. [67]. The work by Meshi et al. [57] shares a similar formulation, and proposes to solve the loss augmented inference and parameter learning simultaneously. Our approach is similar in the sense that we work on the dual of the loss augmented inference. However, our goal is to optimize against non-decomposable losses, while the method of Meshi et al. [57] is only concerned with decomposable loss functions. Komodakis [42] addresses learning the parameters of a Markov network with higher order potential functions. The case of decomposable Hamming loss is developed in detail, and the applicability for general losses is alluded to. This chapter precisely shows how the parameters of a Markov network can be learned given non-decomposable loss functions that operate over all output variables. Tarlow and Zemel [75] also attempt to learn against complex losses, but with a very different approach. In that paper a message passing strategy is proposed and the results are shown on a subset of the Pascal VOC images. Note that the ratio of positive to negative pixels is an order of magnitude larger in the selected subset, which makes the results incomparable. Ranjbar et al. [67] take a different approach to optimize against non-decomposable loss functions. There, the loss function is approximated
with a piecewise planar surface and the loss augmented inference is solved independently for each piece by solving a linear program. The drawbacks of this approach are two-fold. First, its complexity increases linearly with the number of planes in the approximation. Moreover, this approach is computationally very expensive in practice since it requires solving a linear program for each plane in each iteration of the cutting plane approach. Second, this method optimizes against the approximated version of the loss function as opposed to our approach that optimizes against the actual loss.

5.2 Non-Decomposable Loss Optimization via Dual Decomposition

Our goal in this chapter is to solve the loss augmented inference problem for Markov networks and a subset of non-decomposable loss functions, those that can be maximized when augmented with a linear term. Multivariate non-linear performance measures (e.g. $F_1$ measure, intersection over union, and area under ROC) and mean average precision are some examples of such losses.

We assume that we are given a Markov network represented by a graph $G = (V, E)$ where $V$ is the set of nodes, and $E$ is the set of edges. The score of an assignment $y'$ is defined as a linear function of unary and pairwise scoring functions. Although several methods have been proposed to find the best assignment by maximizing the scoring function of a Markov network, $w^T \Psi(x, y')$, the optimization becomes challenging when the score is augmented with a non-decomposable loss function. In this section we propose to use the dual decomposition technique to divide the loss augmented inference into simpler components that can be solved independently.

For simplicity, let’s rewrite the structured SVM with margin rescaling in Eq. 4.8 as an unconstraint optimization problem [21],

$$\min_w \frac{\rho}{2} \|w\|^2 + \max_{y' \in \bar{Y}} (w^T \Psi(x, y') + \Delta(y', y)) - w^T \psi(x, y).$$

(5.1)

We can modify the loss augmented inference problem (maximization in Eq. 5.1) by duplicating the $y'_i$ variables, and then forcing the duplicates to be equal. So, the equivalent
optimization is:
\[
\max_{y', y''} \quad w^T \Psi(\bar{x}, y') + \Delta(y'', y) \\
\text{s.t.} \quad y'_i = y''_i, \forall i
\] (5.2)

Without the constraints, the maximization in Eq. 5.2 could be decomposed into independent maximization for each term which are easier to solve. In order to remove the constraints, we use the Lagrangian Relaxation technique. We define the Lagrangian as:
\[
L(\lambda, y', y'') = w^T \Psi(\bar{x}, y') + \Delta(y'', y) + \sum_{i=1}^{N} \sum_{\hat{y}_i \in \mathcal{Y}} \lambda_i (\hat{y}_i) \left( 1_{[y'_i = \hat{y}_i]} - 1_{[y''_i = \hat{y}_i]} \right)
\] (5.3)

where \( \lambda = [\lambda_1, \lambda_2, \ldots, \lambda_N] \) and \( \lambda_i \) is a vector including all Lagrange multipliers for the \( i^{th} \) node of Markov network for all \( y_i \in \mathcal{Y} \). Then the following optimization is equivalent to the loss augmented inference optimization in Eq. 5.2:
\[
\max_{y', y''} \quad L(\lambda, y', y'') \\
\text{s.t.} \quad y'_i = y''_i, \forall i
\] (5.4)

The optimization in Eq. 5.4 is as hard as the original optimization. But we can omit the constraints and define \( L(\lambda) \) as:
\[
L(\lambda) = \max_{y', y''} L(\lambda, y', y'') = \max_{y'} \left( w^T \Psi(\bar{x}, y') + \sum_{i=1}^{N} \lambda_i (y_i) \right) + \max_{y''} \left( \Delta(y'', y) - \sum_{i=1}^{N} \lambda_i (y''_i) \right)
\] (5.5)

The optimization subproblems in Eq. 5.5 are independent and can be solved separately. It can be easily shown that \( L(\lambda) \) is an upper bound for the original optimization problem (Sontag et al. [72]). So, we can find an approximate solution to the loss augmented inference by finding the tightest upper bound. We solve the dual problem, \( \min_{\lambda} L(\lambda) \) to find such an upper bound, and use its solution to recover an assignment for the primal optimization.

Note that \( L(\lambda) \) is non-differentiable at all points \( \lambda \) that either of the two terms in Eq. 5.5 has multiple local optima. So, similar to [72] the subgradient method is used to solve the dual problem. In each step of this method, the maximization problems in Eq. 5.5 are solved for the current \( \lambda^t \), then the \( \lambda^{t+1} \) is updated toward the direction of subgradient using \( \lambda^{t+1} = \lambda^t - \alpha^t \mathcal{G}, \) where \( \mathcal{G} \) is the subgradient of \( L(\lambda) \) at \( \lambda^t \) and \( \alpha^t \) is a step-size that is set to \( \alpha^t = \frac{1}{\sqrt{t}} \) in our experiments. In the remainder of this section we discuss how we
solve the subproblems in Eq. 5.5 and calculate the subgradient.

We need to solve two maximization problems in each iteration of the subgradient method. The first maximization problem in Eq. 5.5 is maximizing a Markov network with additional unary terms from the Lagrange variables. So, for solving the first subproblem in Eq. 5.5 we modify the Markov network such that the pairwise scores are the same, but the unary scores are added with $\lambda_i$ at each node. Hence, the first maximization can be solved (exactly or approximately) using any MAP inference technique (e.g. see [89]).

The second optimization in Eq. 5.5 involves maximizing the loss function plus a linear term, which is challenging, because an exhaustive search over all $y'' \in \hat{Y}$ is not tractable. However, several methods have been proposed to solve this maximization. For example, Joachims method [39] solves the maximization for multivariate non-linear performance measures and Yue et al. method [94] solves the maximization for mean average precision. Here, we show how to solve this maximization for multivariate non-linear performance measures using the general form of Joachims approach. However, any other loss function, for which this maximization is tractable could be considered.

Note that for multivariate performance measures there are only $O(N^2)$ different contingency cases when the network has $N$ nodes. This is due to the fact that the number of false positives can change from zero to the size of the negative set, and number of false negatives can change from zero to the size of the positive set. So, overall, loss function can take at most $O(N^2)$ different values. Joachims’s approach [39] takes advantage of this fact and proposes an algorithm to solve the loss augmented inference efficiently in $O(N^2)$ time. However, the limiting assumption in this work is that the model only involves unary terms. So, instead of using this algorithm to solve the entire loss augmented inference, which involves unary and pairwise potentials, we employ this approach to solve the second term in Eq. 5.5. Note that the second maximization in Eq. 5.5 is a summation of the loss and unary terms that score the assignments individually. We further generalize the Joachims method for multi-label problems in Algorithm 2.

In Algorithm 2, we assume that the loss function is defined on the number of false positives and false negatives even for multi-class labeling. In other words, the loss is defined for a particular class versus other classes. That is the reason our algorithm accepts the positive label $p$ as an input. For more complex loss functions defined as a function of false positives for each class, we still could generalize the Algorithm 2 by repeating the for loops for each class and finding the labeling that maximizes the summation of loss and Lagrange
Having maximized the subproblems in Eq. 5.5, we calculate the subgradient of \( L(\lambda) \) to update the \( \lambda \). Let \( \mathbf{y}^* \) and \( \mathbf{y}^{**} \) be the optimal assignments for the first and the second subproblems of Eq. 5.5 respectively. We calculate the subgradient of \( L(\lambda) \) at \( \lambda^{'} \) using Algorithm 3. In this algorithm, if a node is assigned to the same label in both subproblems, the subgradient will be zero for all labels; Otherwise, the subgradient is set to the +1 and −1 for the labels assigned by the first and the second optimization in Eq. 5.5 respectively. In each iteration, if a node is labeled differently by the subproblems, the subgradient method will decrease the value of \( \lambda_i(y^*_i) \), and increase the value of \( \lambda_i(y^{**}_i) \). So, intuitively, in the next iteration, both \( y^*_i \) and \( y^{**}_i \) will have less chance to be assigned to their previous labels and the subproblems are more likely to produce agreeing assignments.

To find the tightest upper bound, we iterate the subgradient descent until \( L(\lambda) \) stops changing significantly or a desired maximum number of iterations (100 in our experiments) is reached. If at the final iteration subgradient becomes zero, we can conclude that both

---

**Algorithm 2** Finding the argmax value of \( \lambda(\mathbf{y}') + \Delta(\mathbf{y}, \mathbf{y}') \) over \( \mathbf{y}' \)

1: Input : \( \lambda = \{\lambda_1, \ldots, \lambda_N\} \), \( \mathbf{y} = \{y_1, \ldots, y_N\} \), \( \mathcal{Y} \) and positive label \( p \)
2: \#pos \( \leftarrow \sum_j I[y_j = p] \)
3: \#neg \( \leftarrow \sum_j I[y_j \neq p] \)
4: \((r^p_1, \ldots, r^p_{\#pos}) \leftarrow \text{Sort} \downarrow \{i : y_i = p\} \) by \( \max_{j 
eq i}(\lambda_i(j)) - \lambda_i(p) \)
5: \((r^p_n, r^p_{#neg}) \leftarrow \text{Sort} \downarrow \{i : y_i \neq p\} \) by \( \lambda_i(p) - \max_{j 
eq i}(\lambda_i(j)) \)
6: \((r^p_{#pos}, \ldots, r^p_{#neg}) \leftarrow \{\arg \max_r \lambda_i(r) : i \in \{j : y_j = p\}, r \in \mathcal{Y} \setminus \{p\}\} \)
7: \((r^p_{#pos}, \ldots, r^p_{#neg}) \leftarrow \{\arg \max_r \lambda_i(r) : i \in \{j : y_j \neq p\}, r \in \mathcal{Y} \setminus \{p\}\} \)
8: for \( fn \leftarrow 0 \) to \#pos do
9: set \( \{y^p_{1,1}, \ldots, y^p_{1,n}\} \) to \( \{r^p_1, \ldots, r^p_p\} \) and set \( \{y^p_{n+1,1}, \ldots, y^p_{n,n}\} \) to \( p \)
10: for \( fp \leftarrow 0 \) to \#neg do
11: set \( \{y^p_{1,1}, \ldots, y^p_{1,n}\} \) to \( p \) and set \( \{y^p_{n+1,1}, \ldots, y^p_{n,n}\} \) to \( \{r^p_{n+1,1}, \ldots, r^p_{n,n}\} \)
12: \( v \leftarrow \sum_j \lambda_j(\mathbf{y}_j) + \Delta(fp, fn) \)
13: if \( v \) is the largest so far then
14: \( \mathbf{y}^{**} \leftarrow (y_1, \ldots, y_N) \)
15: end if
16: end for
17: end for
assignments, $y^*$ and $y^{**}$ are in agreement for all nodes, and the assignment is the global solution to the loss augmented inference problem. However, there is no guarantee to reach such an assignment, and in practice, we often have a few nodes that have different assignments in the final $y^*$ and $y^{**}$. In this case we need to recover a primal solution from the assignments proposed by each subproblem. A simple heuristic to recover the solution to the primal equation, which works very well in practice, is to count the number of times that a node is assigned to a label in the iterations of the subgradient method, and assign to the nodes with disagreeing assignments the label that has occurred the most for each node [44].

Algorithm 3 Calculating $G^t$, the subgradient of $L(\lambda)$ at $\lambda^t$

1: Input : $y^*$ and $y^{**}$
2: $g_i^t(y) = 0$, $\forall i, \forall y \in \mathcal{Y}$
3: for $i \leftarrow 1$ to $N$ do
4: if $y_i^* \neq y_i^{**}$ then
5: $g_i^t(y_i^*) = +1$
6: $g_i^t(y_i^{**}) = -1$
7: end if
8: end for

5.3 Experimental Results

We employ object category segmentation as an example of a structured output problem. The task is to label the pixels of an image as being part of a known object (foreground) or not (background). We set the label of foreground to one and the label of background to zero. One example of a non-decomposable performance measure in image segmentation task is intersection over union defined as

$$\text{Acc}_{\cap \cup}(FP, FN) = \frac{N_p - FN}{N_p + FP} \iff \Delta_{\cap \cup}(FP, FN) = \frac{FP + FN}{N_p + FP},$$

(5.6)

which has been used to compare the segmentation accuracies on the Pascal VOC challenge [23]. In our experiments we optimize against this loss function while comparing to two strong baselines.

The first baseline called Hamming approximates the intersection over union loss with

\footnote{In our experiments, in more than 95% of the cases less than 1% of the nodes had different assignments}
normalized Hamming loss – a decomposable loss defined as

\[ \Delta_{\text{Hamming}} = \alpha FP + \beta FN \]  

(5.7)

where the value of \( \alpha \) and \( \beta \) define the importance of the \( FP \) counts versus \( FN \) counts. The two typical approaches for setting these coefficients are either assuming the same importance for \( FP \) and \( FN \) counts and set \( \alpha = 1/(2N_n) \) and \( \beta = 1/(2N_p) \). Here, \( N_p \) and \( N_n \) represent the number of positive and negative samples in the dataset, respectively. This is the same as the first baseline in the previous chapter. The second common approach for setting \( \alpha \) and \( \beta \) coefficients is by using cross validation. Here, we fix \( \beta \) at \( 1/(2N_p) \) and cross validate \( \beta/\alpha \) on the set \( \{1, 2, 5, 10, 50, 100, N_n/N_p\} \). We label this baseline as HammingCV in the results.

Note that Hamming loss decomposes as an extra unary potential in the overall Markov network. Solving for the loss augmented inference in this case is computed by performing MAP inference on the network. For some special cases, e.g. when the graph is submodular, MAP inference has been shown to be exact. However, in a general graph MAP inference has been shown to be NP hard and therefore approximate inference techniques could be employed to find an approximate solution [9, 89]. Note that solving the loss augmented inference for Hamming loss is similar to solve the first maximization in Eq. 5.5. This baseline shares the same model as we have in the proposed approach, but optimizes against a decomposable loss as opposed to the original non-decomposable loss.

The third baseline is based on the idea of Joachims [39], which can optimize against a large set of loss functions, those which can be computed from the contingency table, a subset of loss functions that can be optimized using the proposed approach. However, this approach assumes a decomposable model and cannot benefit from the pairwise interactions on the outputs. It can only capture the dependency between the outputs through the loss function and not through the model. Note that the second maximization in Eq. 5.5 has exactly the same form as explained here – a non-decomposable loss computed from the contingency table plus a linear term.

The fourth method that we compare the results with is the planar approximation approach presented in the previous chapter. The planar approximation method takes a very different approach for solving the loss augmented inference. Briefly, the loss function that is assumed to be a function of false positive and false negative counts is approximated with
a piecewise linear surface in false positive and false negative space. Then a linear program is solved for each piece independently and the piece with maximum loss augmented value is picked as the answer. The drawback of this method is the need to solve many linear programs in each iteration of the parameter learning. We use 15 pieces to approximate the loss function, which generates almost identical results to the original choice of 40 pieces in [67] for computational efficiency. We employ Mosek [60] the same off-the-shelf LP solver to solve the LPs. Note that solving each LP takes more than finding the loss augmented inference in the proposed method with 100 subgradient steps, which makes the proposed approach more than 15 times faster than the approach in [67], yet with comparable results.

5.3.1 Pixels vs. Superpixels

If we decide to perform segmentation on the pixel level, meaning that the input be the set of all features extracted from all pixels in the dataset and the output be the binary label of each pixel, then for a dataset with 750 images each of size $500 \times 300$ pixels, we will have $112,500,000$ variables (nodes in the Markov network). Solving the second term of Eq. 5.5 (Algorithm 2) as well as computing the loss augmented inference in Joachims’ method [39] have time complexity $O(N^2)$, where $N$ is the number of nodes. Clearly, such computation is intractable.

To solve this problem, we group neighbouring pixels into superpixels (of possibly different sizes\(^3\)) based on their color distance and assign one label to all pixels of a superpixel. However, the assumption in Algorithm 2 is that all positive examples (superpixels of the foreground) contribute equally in the loss function, which is not true if the area of the superpixels are different. To solve this problem, one could look at the equivalent pixel level problem. Assuming a similar feature vector for all pixels of a superpixel, each pixel receives $1/m$ of the unary score of its superpixel, where $m$ is the number of pixels in the superpixel. Therefore, sorting the scores (lines 4 and 5 of Algorithm 2) would rank all pixels of a superpixel sequentially. So the equivalent pixel level computation would never create inconsistent labels for pixels of a superpixel except for possibly one foreground superpixel and one background superpixel\(^4\). So instead of iterating over the pixels, we sort the unary score of

---

\(^3\)The other alternative is to force the superpixels to have the same size, but then large flat regions such as sky would be broken into many small superpixels and regions of small objects could be grouped with background.

\(^4\)Let’s first consider the simple case of binary labels($|Y| = 2$). When the optimal labeling $y^{**}$ is achieved,
the superpixels divided by their area and correct the false positive and false negative counts based on the area of the chosen superpixels. With superpixels of size at least 2000 pixels, which still can potentially provide very accurate segmentations, we achieved a speedup of more than 4,000,000 times. In our experiments we employ the superpixel extractor of Felzenszwalb et al. [25] and set its parameters to MinArea = 2000, k = 200, σ = 0.01.

5.3.2 Learning Method

We utilize NRBM [21] – an instance of a bundle method – as the core of our learning and solve the loss augmented inference based on the proposed approach. Note that other structured prediction formulations such as Pegasos [70] or the formulation proposed by Meshi et al. [57] could easily replace the bundle method. We chose NRBM due to implementation simplicity knowing that it has the same bound of $O(1/\epsilon)$ like aforementioned alternatives to obtain a solution of accuracy $\epsilon$. We do cross validation to set the $\rho$ parameter in Eq. 5.1. Although we conduct the experiments on binary label problems, the first term in Eq. 5.5 could still be NP hard to solve given an arbitrary vector $w$. To make the graph supermodular, we add constraints to force the weights corresponding to pairwise features (which are always positive) to be smaller than zero.

5.3.3 Features

In our experiments we use a set of features that represent each superpixel and the pairwise interaction between neighbouring superpixels. Our first feature is a bag-of-words representation over color SIFT [85] with 1000 visual words. Our second set of features are the part filter responses obtained from the object detector of Felzenszwalb et al. [24]. We aggregate the filter responses in different scales and sum the responses inside each superpixel. The pairwise features between neighbouring features have three components each representing the average absolute distance between pixels’ colors of the two superpixels on one color channel. The cost of assigning the same labels to neighbouring superpixels is set to zero.

---

the first $fn$ pixels of foreground pixels in the sorted order get value 0 and the rest get value 1. Knowing that the pixels of a superpixel are sorted sequentially, the only superpixel that may have inconsistent labels is the one that its pixel is located at position $fn$. The same argument holds for background superpixels. One could easily verify that the same property holds for the multi-label case as well.
Table 5.1: Background to foreground pixel ratio in Pascal VOC 2009 and 2010 datasets

<table>
<thead>
<tr>
<th></th>
<th>Aeroplane</th>
<th>Bus</th>
<th>Car</th>
<th>Horse</th>
<th>Person</th>
<th>TV/Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOC 2009</td>
<td>163</td>
<td>69</td>
<td>86</td>
<td>130</td>
<td>24</td>
<td>96</td>
</tr>
<tr>
<td>VOC 2010</td>
<td>168</td>
<td>64</td>
<td>70</td>
<td>119</td>
<td>26</td>
<td>105</td>
</tr>
</tbody>
</table>

5.3.4 Pascal Visual Object Classes (VOC) 2009 and 2010 Segmentation Datasets

The Pascal VOC 2009 and 2010 datasets include 749 and 964 pixel-level labeled training images, respectively. They also include 750 and 964 validation images, respectively. We decide to train our method on the training set and test on the validation set, because the ground-truth for the test set is not publicly available and our focus is on comparison to baseline methods using a different model or learning criterion. We present the results on 6 object categories, Aeroplane, Bus, Car, Horse, Person, and TV/Monitor. We select these categories because the top-down unary features obtained from the Felzenszwalb et al. object detector provide reasonable detection on them. Without the top-down features, the overall accuracy would be so low as to make the comparison between different learning methods uninformative. One of the most challenging aspects of these datasets is the ratio of foreground to background pixels for all categories (Table 5.1). Moreover, the images in these datasets are not taken in a controlled environment and include severe illumination and occlusion. We compare the proposed approach to the baselines on the 6 object categories in Figure 5.1. As illustrated, the proposed approach significantly outperforms the baselines in both datasets. also compare the results with [67] in Fig. 4.4. The results of the proposed approach are slightly better in all classes except in class “Aeroplane” in VOC 2009 and “Car” in VOC 2010 dataset, but is comparable overall. However, the proposed approach is more than 15 times faster.

Experiments on class “Aeroplane” in VOC 2010 dataset show that our Matlab implementation takes 104.4 seconds to optimize the loss augmented inference for a given $w$ on an Intel Core i7-860 2.80GHz with 8GB RAM. This running time is averaged over all inferences computed for NRBM algorithm given a $\rho$ parameter. Consequently, training a model using 500 iterations of NRBM takes about 14.5 hours.
### Figure 5.1: Intersection over union performance (%) comparison on VOC 2009 and VOC 2010 datasets
5.3.5 H3D Dataset

We also compare the results on the H3D dataset [8]. This dataset includes 273 training and 107 testing images along with three types of annotations – keypoint annotations, 3d pose annotation and region annotation. The keypoint annotation includes the location of joints and other keypoints such as eyes, nose, elbows, etc. The 3d pose annotation has been inferred from the keypoints. The region annotation, which we use in this chapter, provides detailed annotation of people, such as face, neck, lower and upper cloth, etc. For our experiments we compute the union of all region annotations that are part of a person (bags, occluder and hat are not considered as parts of a person) as foreground and the rest as background. The ratio of background to foreground pixels in this dataset is 3.9, which is significantly lower than the ratio in Pascal VOC datasets. The reason is that all images in H3D dataset include at least some foreground pixels, which is not the case in Pascal VOC datasets. The comparison result in Figure 5.2 shows that the proposed approach outperforms the baselines significantly on this dataset.

5.3.6 Convergence Analysis

The proposed approach provides an efficient way of decomposing the loss augmented inference into two subproblems – one involving a supermodular function and the other one that can be exhaustively searched over to find the maximum value. At the same time, the proposed method inherits all the convergence properties of Lagrangian relaxation solved by the subgradient method. Precisely, the subgradient method is guaranteed to converge to the optimal of the dual function if the sequence of the multipliers $\alpha^t$ satisfy

$$\alpha^t \geq 0, \lim_{t \to \infty} \alpha^t = \infty, \sum_{t=1}^{\infty} \alpha^t = \infty$$  (5.8)
which is the case for our choice of $\alpha^t = 1/\sqrt{t}$ [44]. Although the solution to the Lagrangian relaxation is as good as the solution of the LP relaxation, there is no guarantee that the LP relaxation is tight. Fig. 5.3 shows the improvement of the primal and the dual values in loss augmented inference for a given $w$. The gap between the primal and the dual functions decreases as subgradient descent algorithm proceeds. Fig. 5.4 shows the average percentage of disagreeing assignments versus the number of subgradient iterations. The average is taken over all inferences computed for the NRBM algorithm in training a model for the class ‘Aeroplane” in VOC 2010 dataset. Number of disagreeing assignments decreases as the loss augmented model is being optimized and in average only 0.63% of assignments are disagreeing at the last iteration. Note, of course, that amounts of disagreement do not necessarily translate into solution quality. However, the small amount of disagreement is suggestive of reasonable dual decomposition performance.
Figure 5.4: The average percentage of disagreeing assignments versus the number of subgradient iterations. In average only 0.63% of assignments are disagreeing at the last iteration.
5.4 Summary

This chapter presents a novel approach for optimizing against a large class of non-decomposable performance measures with a Markov network model. The approach can be used for performance measures which can maximized when augmented with a linear term, when the model is a Markov network with unary and pairwise potentials. Observing that the computationally challenging part of most structured prediction approaches is to find the subgradient, which requires solving the loss augmented inference, we proposed an approach to solve the loss augmented inference efficiently. We adopt the idea of dual decomposition and factorize the loss augmented inference into two factors and show how to solve each factor efficiently. The proposed approach is applicable to a large range of structured prediction methods. We compared the proposed approach to three strong baselines – two with the same model as ours, but optimizing against decomposable Hamming loss, and the other one optimizing against the same non-decomposable loss function, but using a decomposable model. We showed significant improvements over the baselines on three object segmentation datasets – Pascal VOC 2009, Pascal VOC 2010 and H3D. We also compare the results with our previous approach in Chapter 4 for solving the same problem and show comparable results with more than 15 times speedup.
Chapter 6

Conclusion

In this thesis we attacked the problem of optimizing against non-decomposable loss functions in structured prediction when the model is a Markov network. We proposed two approaches for approximately solving the loss augmented inference each with some pros and cons. In the first approach (planar approximation) we assume that the loss function is a function of false positive and false negative counts. We approximate the loss surface in false positive and false negative space with a piecewise planar surface. Loss augmented inference on the approximated surface can be decomposed into each piece where a decomposable planar surface is used to estimate the loss inside that piece. The loss augmented inference for each piece can be written as a QP. We then solve each QP using LP relaxation.

The alternative approach proposed in Chapter 5 is based on the idea of dual decomposition. There we break the dual into two pieces – a supermodular MRF and the loss function augmented by a linear term. We then optimally solve both subproblems and force their results to agree by introducing some Lagrange multipliers. We show empirically that this approach produces comparable results to our previous method of planar approximation, but is significantly faster. It is worth mentioning that our planar approximation strategy could handle linear constraints in the loss function and is fully parallelizable unlike this fast dual decomposition method.

We also proposed fine and coarse texture cues in Chapter 3 and achieved state-of-the-art results for human segmentation on Pascal VOC 2009 dataset.

We showed empirically that both proposed approaches can significantly improve two strong baselines – one employing the same model but optimizing against decomposable Hamming loss and the other one exploiting only the unary potentials, but optimizing against
the same non-decomposable loss function.

However, there are limitations in the proposed methods that require further research. The main limitation of the planar approximation approach is its scalability to large datasets. The current formulation would solve a LP for each subregion which involves all the variables in the entire dataset. A variable could be a pixel or superpixel for segmentation, a person’s bounding box for action retrieval, etc. A simple yet effective fix would be to partition the training set into some smaller subsets, compute the optimal weight vector for each subset and find a weight vector that agrees the most with all of these weight vectors.

The idea of dual decomposition is more scalable since minimization of the scoring function is decoupled for each training example. However, the loss term is still computed over the entire dataset with the complexity of $O(N_p \log(N_p)) + O(N_n \log(N_n)) + O(N_p \times N_n)$. Here, $N_p$ and $N_n$ represent the number of positive and negative examples, respectively. Moreover, we showed empirically that the primal and dual objectives converge in few iterations. However, analyzing the convergence rate and proving some upper bounds would be a possible future work.

In sum, this thesis provides two superior algorithms for solving the loss augmented inference, when the model is a Markov network and the loss function is non-decomposable. The superiority of the results over the baselines demonstrates the potential of the proposed algorithms as a reliable candidate for optimizing structured prediction problems against non-decomposable loss functions, especially when the model is a Markov network.
Bibliography


