Inference and learning algorithms with applications to 3D indoor scene understanding

Author(s):
Schwing, Alexander G.

Publication Date:
2013

Permanent Link:
https://doi.org/10.3929/ethz-a-010117523

Rights / License:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection. For more information please consult the Terms of use.
Inference and Learning Algorithms with Applications to 3D Indoor Scene Understanding

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH Zurich

(Dr. sc. ETH Zurich)

presented by

ALEXANDER GERHARD SCHWING
Dipl.-Ing. Univ., Technische Universität München
born 16. November 1982, citizen of Germany and

accepted on the recommendation of
Marc Pollefeys
Raquel Urtasun
Tamir Hazan
Andreas Krause

2013
# Contents

Acknowledgements v  
Abstract vii  
Zusammenfassung ix  

1. Introduction 1  
1.1. Applications for Inference and Learning 2  
1.2. Problem Statement 4  
1.3. Contributions 5  
1.4. Thesis Outline 6  

2. Background and Related Work 7  
2.1. Formal Problem Definition 7  
2.2. Inference in Structured Models 8  
2.2.1. Computation of the Partition Function 10  
2.2.2. Computation of the Most Likely Labeling 12  
2.3. Learning 14  
2.3.1. Learning Decision Trees 14  
2.3.2. Learning Parameters of Structured Models 14  
2.3.3. Learning Parameters of Latent Structured Models 16  

3. Adaptive Random Forest 17  
3.1. Review 17  
3.2. Faster Inference via Adaptive Tree Selection 19  
3.2.1. Binomial Prediction 19  
3.2.2. Multinomial Prediction 22  
3.3. Experimental Evaluation 24  
3.3.1. Evaluation on Small Scale Data Sets 24  
3.3.2. Experiments on Large Scale Data Sets 26  
3.4. Conclusion 29  

4. Inference in Structured Models 31  
4.1. Review 31  
4.2. Distributed Message Passing Algorithm 37  
4.2.1. Intuition 37  
4.2.2. Experimental Results 40
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.3.</td>
<td>Detailed Derivation</td>
<td>43</td>
</tr>
<tr>
<td>4.2.4.</td>
<td>Summary</td>
<td>44</td>
</tr>
<tr>
<td>4.3.</td>
<td>Globally Convergent LP Relaxation Solver</td>
<td>44</td>
</tr>
<tr>
<td>4.3.1.</td>
<td>Subdifferentials</td>
<td>46</td>
</tr>
<tr>
<td>4.3.2.</td>
<td>$\epsilon$-Subgradients and Steepest $\epsilon$-Descent</td>
<td>48</td>
</tr>
<tr>
<td>4.3.3.</td>
<td>Conditional Gradient for $\epsilon$-descent Directions</td>
<td>51</td>
</tr>
<tr>
<td>4.3.4.</td>
<td>Experiments</td>
<td>58</td>
</tr>
<tr>
<td>4.3.5.</td>
<td>Summary</td>
<td>62</td>
</tr>
<tr>
<td>4.4.</td>
<td>Conclusion</td>
<td>62</td>
</tr>
<tr>
<td>5.</td>
<td>Learning in Structured Models</td>
<td>63</td>
</tr>
<tr>
<td>5.1.</td>
<td>Review</td>
<td>63</td>
</tr>
<tr>
<td>5.2.</td>
<td>Distributed Message Passing Algorithm</td>
<td>68</td>
</tr>
<tr>
<td>5.2.1.</td>
<td>Intuition</td>
<td>68</td>
</tr>
<tr>
<td>5.2.2.</td>
<td>Experimental Results</td>
<td>71</td>
</tr>
<tr>
<td>5.2.3.</td>
<td>Detailed Derivation</td>
<td>72</td>
</tr>
<tr>
<td>5.3.</td>
<td>Notes on a Globally Convergent Algorithm</td>
<td>74</td>
</tr>
<tr>
<td>5.4.</td>
<td>Conclusion</td>
<td>75</td>
</tr>
<tr>
<td>6.</td>
<td>3D Indoor Scene Understanding</td>
<td>77</td>
</tr>
<tr>
<td>6.1.</td>
<td>Review</td>
<td>78</td>
</tr>
<tr>
<td>6.2.</td>
<td>Parameterization and Scoring Functions</td>
<td>79</td>
</tr>
<tr>
<td>6.2.1.</td>
<td>Parameterization and Score for Layout Prediction</td>
<td>80</td>
</tr>
<tr>
<td>6.2.2.</td>
<td>Parameterization and Score for Layout and Object Prediction</td>
<td>81</td>
</tr>
<tr>
<td>6.3.</td>
<td>Features</td>
<td>84</td>
</tr>
<tr>
<td>6.3.1.</td>
<td>Orientation Maps</td>
<td>84</td>
</tr>
<tr>
<td>6.3.2.</td>
<td>Geometric Context</td>
<td>85</td>
</tr>
<tr>
<td>6.3.3.</td>
<td>Object Features</td>
<td>85</td>
</tr>
<tr>
<td>6.4.</td>
<td>Globally Optimal Inference</td>
<td>86</td>
</tr>
<tr>
<td>6.4.1.</td>
<td>Layout Inference</td>
<td>86</td>
</tr>
<tr>
<td>6.4.2.</td>
<td>Layout and Object Inference</td>
<td>89</td>
</tr>
<tr>
<td>6.5.</td>
<td>Efficient Score Computation via Integral Geometry</td>
<td>92</td>
</tr>
<tr>
<td>6.6.</td>
<td>Learning</td>
<td>95</td>
</tr>
<tr>
<td>6.7.</td>
<td>Experimental Evaluation</td>
<td>95</td>
</tr>
<tr>
<td>6.7.1.</td>
<td>Layout</td>
<td>95</td>
</tr>
<tr>
<td>6.7.2.</td>
<td>Layout and Object</td>
<td>100</td>
</tr>
<tr>
<td>6.8.</td>
<td>Conclusion</td>
<td>104</td>
</tr>
<tr>
<td>7.</td>
<td>Learning with Latent Variables</td>
<td>107</td>
</tr>
<tr>
<td>7.1.</td>
<td>Loss Minimization with Latent Variables</td>
<td>107</td>
</tr>
<tr>
<td>7.2.</td>
<td>Efficient Message Passing Algorithm</td>
<td>109</td>
</tr>
<tr>
<td>7.2.1.</td>
<td>Approximate Latent Structured Loss Minimization</td>
<td>109</td>
</tr>
<tr>
<td>7.2.2.</td>
<td>Message Passing Algorithm</td>
<td>111</td>
</tr>
<tr>
<td>7.2.3.</td>
<td>Experiments</td>
<td>112</td>
</tr>
<tr>
<td>7.2.4.</td>
<td>Detailed Derivation</td>
<td>116</td>
</tr>
</tbody>
</table>
Acknowledgements

The present thesis would not have been possible without support from a multiplicity of people. Thank you for your continuous encouragement and your help in many challenging situations but also for sharing the precious moments.

I am particularly grateful that I was able to work with three great scientists during my PhD studies. Thanks to Marc Pollefeys for the discussions, the continuous support and the scientific freedom. In the same manner, I am very thankful to Raquel Urtasun for the many scientific discussions via phone or in person, many joint projects that were fun to investigate, numerous invitations and most importantly for sharing vast personal advice. Just as important for my graduate studies was Tamir Hazan for sharing his insights, insisting in mathematical rigorousness, the desire to unify derivations, lengthy phone conversations and the invitations.

Without all of your support in providing such an amazing environment, this thesis, its contributions in form of applicable algorithms and my skills would not exist in its current form. Thank you so much! I hope I will ever be able to return this favor!

Naturally there are a lot more friends behind the scenes that contribute to a greater or lesser extent which I want to mention in alphabetic order subsequently.

I thank current and former team members at Eidgenössische Technische Hochschule Zurich and in particular Christopher and my office mates Bernhard, Jens and Roland as well as the administrative team around Beatrice, Susanne and Thorsten.

Similarly I am grateful to the valuable moments during my stays at Toyota Technological Institute at Chicago. Let me particularly recall the scientific discussions with David, Jian, Sanja, Vikas and Wenjie as well as the administrative support from Liv, Gary and Adam.

Having spent some time at Siemens Corporate Research at the beginning of my PhD studies, I am particularly thankful to Alexey, Bogdan, David, Dorin, Kevin, Terrence and Yefeng.

During my studies, many friends provided immense support. Let me mention the discussions and support from Andreas, Anja and Zhizhen and the assistance from Philip and Sybille.

Many more people provided encouragement in numerous ways. Be assured that I am aware of your gestures and let me apologize for not being able to mention everyone in this short one page list.

This thesis is dedicated to my parents Egon and Gerlinde as well as my brother Joachim and my sister Julia for providing a carefree environment during my studies and immeasurable assistance to meet many bureaucratic and organizational challenges.

Zurich, November 8, 2013

Alexander Schwing
Abstract

Inference and learning algorithms are a common tool for analyzing data and for making decisions. They are applied in many fields from social sciences and computational biology to natural language processing and particularly computer vision, e.g., for 3D scene understanding of monocular imagery.

Three trends for future developments of algorithms are clearly observable. Firstly, the amount of data that is processed with current inference and learning algorithms is growing rapidly. In fact, oftentimes the available data exceeds the processing capacity. Secondly, the amount of information we aim to extract from data is increasing. While expected, since larger data sets most likely contain more information, reasoning within higher dimensional spaces increases complexity significantly. Thirdly, inference and learning algorithms are employed more and more frequently.

The requirement of dealing with more data, and more information, more frequently emphasizes the need for efficiency. It is however understood that we are not interested in harming performance or optimality guarantees. Within this thesis we therefore propose extensions to existing algorithms that either increase efficiency or that better utilize available computational resources to decrease wall clock time.

More specifically, to increase computational efficiency for prediction with a Random Forest classifier, we propose to augment inference by a confidence analysis which enables early termination if the algorithm is sufficiently confident about the result. Since the added computation is performed ahead of time, prediction is effectively augmented by a table lookup. We show that such a procedure yields up to 5-times faster prediction performance.

To decrease the computation time for algorithms operating on models with explicitly encoded dependencies between multiple variables, we propose to distribute both learning and inference by taking advantage of multiple computers. In order not to slow down computation by frequently accessing non-local variables, we rephrase the task via dual decomposition.

The most effective algorithms for inference with explicitly modeled correlations between variables are unfortunately not globally convergent when used for predicting the most likely result. We address the challenge via an $\epsilon$-steepest descent procedure and show that the proposed method efficiently retrieves the global optimum.

Oftentimes the available data is incomplete. To derive a learning algorithm for the latent variable setting we interleave prediction of the missing variables and the step of extracting the model parameters. We show that the proposed approach is able to gain information even if some variables are never observed.

Predicting the 3D box which best describes the observed room layout given a single image is an important problem for 3D scene understanding with applications to surveillance and robotics. Further, layout prediction is the baseline for more complex
Abstract

tasks like automatic floor plan recovery and prediction of interactions between humans and objects. Within this thesis we show that it is possible to design an inference algorithm that is able to efficiently retrieve the global optimum for this task.
Zusammenfassung


Die Anforderung mit mehr Daten und mehr Information immer öfter klarzukommen unterstreicht die Notwendigkeit von effizienten Methoden. Es ist jedoch verständlich, dass wir nicht daran interessiert sind die Qualität der Lösung oder Garantien der Algorithmen aufzugeben. Diese Arbeit schlägt deshalb Erweiterungen zu existierenden Algorithmen vor die entweder die Effizienz erhöhen oder die die heutzutage zur Verfügung stehende Rechenkapazität besser ausnutzen um die Berechnungszeit zu reduzieren.

Im Besonderen schlagen wir vor die Effizienz der Klassifikation eines ‘Random Forest’ zu steigern indem wir den Vorgang der Schätzung um eine Konfidenzanalyse erweitern. Diese erlaubt es die Klassifikation verfrüht abzubrechen falls die nötige Sicherheit über das zu erwartende Ergebnis erreicht wurde. Da die zusätzlich eingeführten Berechnungen unabhängig von den Daten sind ist während der Klassifikation nur ein kurzer Tabellenzugriff notwendig und wir zeigen, dass diese zu fünf mal schnellerer Klassifikation führen kann.

Um die Berechnungszeit von Algorithmen für Modelle mit explizit modellierten Abhängigkeiten zwischen mehreren Variablen zu senken, schlagen wir vor die notwendigen Berechnungen sowohl beim maschinen Lernen der Modelle als auch beim Schätzen mit Hilfe der Modelle auf mehrere Rechner zu verteilen. Um die Einbußen durch sehr häufige Zugriffe auf Variablen die nicht lokal gespeichert sind zu minimieren nutzen wir die mathematische Technik der dualen Zerlegung.

Die effektivsten Algorithmen zur Berechnung der wahrscheinlichsten Konfiguration bieten leider keine Garantie bezüglich globaler Konvergenz. Um diese Herausforderung zu lösen benutzen wir eine Methode zur Berechnung des $\epsilon$-steilsten Abstiegs und zeigen, dass es möglich ist das globale Optimum effizient zu finden.
Zusammenfassung

Oftmals sind die zur Verfügung gestellten Daten unvollständig. Um Algorithmen zum maschinellen Lernen für solche Fälle zu entwickeln, verschachteln wir die Schätzung des Zustands für die fehlenden Daten mit der Berechnung der Parameter. Wir zeigen, dass der vorgeschlagene Ansatz in der Lage ist Informationen sogar dann aus den Daten zu extrahieren wenn einige Variablen nie explizit angegeben wurden.

Diejenige 3-dimensionale Box zu berechnen, die ein auf einem einzigen Bild festgehaltenen Raum am Besten beschreibt ist ein wichtiges Teilproblem um dargestellte Szenen zu verstehen und findet Anwendung bei der Überwachung und in der Robotik. Des Weiteren ist die Schätzung der Anordnung die Basis für komplexere Aufgaben wie die automatische Rekonstruktion des Gebäudeplans und die Darstellung möglicher Interaktionen zwischen Menschen und Objekten. Innerhalb dieser Arbeit zeigen wir, dass es möglich ist einen Algorithmus zu entwickeln der das globale Optimum einer häufig verwendeten Kostenfunktion effizient findet.
1. Introduction

For years we construct “intelligent” systems that provide answers to specific questions given some form of data. Think about recommender systems for e-commerce applications where the data is the products currently inside the shopping cart as well as a list of co-purchased articles, and the question can be as simple as “is the customer interested in a particular product?” Suppose we are given a list of people and their relationship among each other, while we are wondering: “how likely is a person going to vote for a certain party?” Let the given data be a sentence while a possible question reads as “how are the words and sentence parts related to each other?” or “what is the part of speech of a certain word?” Given a sequence of amino acids we are wondering “what is the structure of the resulting protein?” or when considering given arrays of genes we aim at answering: “what is the gene expression given chemical influences?” Taking a single image illustrating an indoor scene we ask: “what is the 3D configuration of the observation?” or, as frequently implemented in digital consumer cameras these days, we are given a single image and wonder: “where are the faces?”

We subsequently refer to answering questions like these as solving a statistical inference problem. Note that inference tasks are commonly employed in a diversity of fields from social sciences, where we consider recommender systems or dependency networks, to natural language processing, with tasks like sentence parsing or part of speech tagging. Or from computational biology, which considers protein design problems or gene expression analysis, to computer vision, with tasks like 3D scene understanding or object detection, to just name a few.

At the heart of the aforementioned illustrative inference tasks is a model or more generally an inference algorithm that is applied to newly observed data. While the e-commerce application requires the algorithm to distinguish between interesting and irrelevant products, prediction within social networks aims at leveraging the influence of friendships, and speech parsing reasons about the sentence structure or the different parts of speech. Protein design tasks infer angular orientations of amino acid structures to improve stability, gene expression analysis investigates regulatory influences, 3D indoor scene understanding is concerned about semantically reconstructing a scene while object detection targets outlining of objects or humans. Designing such a model or algorithm by using data that corresponds to input and the desired output for the considered question is commonly referred to as the learning problem. We emphasize that we extract knowledge by analyzing given data being, e.g., the browsing history, friendships, sentences, biological structures or images.

As illustrated, many fields apply inference and learning in one or another way. Besides general inference and learning algorithms we particularly address a 3D indoor scene understanding application within this thesis. More concretely: we answer the question on how to find a 3D parametric box that best describes the observed
1. Introduction

Figure 1.1.: (a) A graph linking Amazon products if they were frequently purchased together. (Data from Leskovec et al. (2007)) (b) A possible parsing for the given sentence. (c) A protein structural motif called beta hairpin with backbone and hydrogen atoms (gray) as well as side-chains (cyan). (d) A possible input image and a newly synthesized view generated by 3D scene understanding algorithms.

room layout given a single image and design extensions to also obtain object estimates. Such a task is useful for static surveillance cameras and also robotics.

1.1. Applications for Inference and Learning

To illustrate the applicability of inference and learning algorithms described within this thesis we subsequently provide more details regarding the above mentioned introductory questions for four arbitrarily chosen, but according to our opinion publicly very relevant areas.

Social and Behavioral Sciences

One of the most widely used, but probably rarely publicly realized examples for inference applications within social sciences are recommender systems. Almost every
well known e-commerce website employs techniques to suggest products based on many more metrics than just the articles currently within the shopping cart, e.g., purchases commonly co-occurring as illustrated in Fig. 1.1(a) for Amazon products frequently bought jointly. Models to retrieve increasingly accurate estimates for user preferences are being designed on a daily basis with the Netflix challenge certainly increasing their popularity. See, e.g., the work by Stern et al. (2009) and references for examples.

Another vivid area of research is the influence of social interactions to predict user behavior. Not only is it important to infer the user behavior but rather are we interested in using the prediction to change the result to our favor. It is therefore not astonishing that even presidential elections have been ‘influenced’ by running inference algorithms to predict where to place ads and how to convince voters\(^1\). Algorithms to maximize profit in social networks are, e.g., presented by Lu & Lakshmanan (2012).

### Natural Language Processing

Just like social and behavioral sciences, it is also the natural language processing community that largely relies on inference and learning algorithms. Depending on the task, i.e., for example sentence parsing or part of speech tagging, the target is extraction of the sentence structure in order to find the correct dependencies of its constituents or extraction of the parts of speech. Significant success has been presented to accurately parse the sentence structure, which is useful for fully automatic translations or adequate interactive voice response in call centers. Resolving ambiguities which arise from individual words like “book,” or from entire sentence structures, are the main challenge. Considering the example visualized in Fig. 1.1(b). It is not clear whether “I” looked through the telescope to see “the man” or whether the man “I” saw carried a telescope as indicated by the given parse tree. See, e.g., (Collins, 2003; Gormley & Eisner, 2013) for further references and examples as well as notes on the inside-outside algorithm by Baker (1979). See also reports by Collins (retrieved 2013) and Eisner (retrieved 2013).

### Computational Biology

Within the field of computational biology, inference techniques are employed to protein design tasks, important for improving stability of drugs. A protein consists of an amino acid structure to form the backbone with protruding amino acid side-chains determining the final conformation. Figuring out the side-chain placement and type as well as the angular orientation, as illustrated in Fig. 1.1(c), are important quantities to be inferred in order to construct the most stable protein configuration (Yanover et al., 2008).

To give another example, gene expression profiling analyzes the activity, i.e., the expression of genes. The activity of a particular cell in transcribing proteins from the genome depends on factors like the time a day, the surroundings and the received chemical signals. To investigate the reaction of a cell on specific settings, the amount of produced proteins is measured to obtain information regarding the type of cell and

\(^1\) [http://www.computerworld.com/s/article/9235587/Barack_Obama_39_s_Big_Data_won_the_US_election](http://www.computerworld.com/s/article/9235587/Barack_Obama_39_s_Big_Data_won_the_US_election)
1. Introduction

its task (Friedman, 2004). This information is then used for drug development, with tolerability being improved by careful data analysis.

Computer Vision

A popular topic within the computer vision community making significant use of inference and learning techniques is 3D scene understanding. Besides outdoor tasks important, e.g., for autonomous driving and traffic scenario reasoning, indoor scenes are employed for surveillance, indoor navigation and robotics. An example task, originally proposed by Hoiem et al. (2007a) and Hedau et al. (2009), reasons about a parametric 3D room layout and a containing object. The retrieved information is employed to synthesize novel views as visualized in Fig. 1.1(d).

Pixel-wise object detection heavily relies on inference as well, and is used frequently in consumer cameras. To adjust auto-focus, exposure and color balance, cameras detect faces which boils down to running an inference algorithm to answer whether a pixel is likely to be part of a face (Viola & Jones, 2001).

1.2. Problem Statement

Inference tasks for problems in spirit similar to the aforementioned examples are automatically solved millions of times per day by computers. Efficiency and suitability of respective algorithms for current computing architectures is therefore an important topic, addressed within this thesis: How can we decrease computation time and increase accuracy of current inference methods?

Contrasting the inference procedure, the learning problem, i.e., the construction of an inference procedure, still requires a lot of human interaction and creativity in general. Despite the existence of many fairly standard algorithm types, known via melodic names such as “Boosting,” (Schapire & Freund, 2012) “Random Forests,” (Breiman, 2001) “Conditional Random Fields,” (Lafferty et al., 2001) “Max-Margin Markov Networks,” (Taskar et al., 2003) or “(Structured) Support Vector Machines,” (Cortes & Vapnik, 1995; Tsochantaridis et al., 2005) and corresponding algorithms for finding their parameters, a lot of savvy and manual trial and error is generally required. We are therefore interested in performing many experiments rather quickly. The second topic within this thesis is subsumed by the following question: How can we parallelize the three last mentioned learning methods in order to increase the number of experiments per time without affecting accuracy when considering large models?

One of the key question to be answered manually and specifically for every application when designing a learning and inference procedure these days is related to representation of the data and representation of the answer. For an important problem within the 3D indoor scene understanding community, concerned with finding a parametric box that best describes the observed room layout given a single image (Hedau et al., 2009), we answer as a third topic: What is a favorable representation of the data and the parametric box for the 3D indoor scene understanding task such that inference and learning is efficiently possible?
As mentioned before, learning methods make use of labeled data, more specifically, input data and an answer to the considered question. It is however time consuming, costly and sometimes not even possible to annotate all the data, i.e., to answer the question of interest for a large number of data samples manually. As a fourth topic we are wondering: How can we design algorithms that extract information from data that is only partly labeled, i.e., only part of the answer is specified?

1.3. Contributions

Guided by the aforementioned questions, it is the goal of this thesis to present extensions to existing algorithms for both learning and inference. More specifically we suggest the following:

- We propose to augment the inference procedure of a Random Forest classifier to include a confidence analysis. Although computational overhead is introduced at first, this addition allows to terminate classification early on. It therefore depends on the specifics of the data and the computational complexity of the features whether this early termination procedure increases inference efficiency. Using synthetic examples and a medical imaging data set we show that such an addition increases efficiency, sometimes significantly, without impacting the prediction performance.

- Computational efficiency is traded for the ability to work more explicitly with dependencies between different sub-parts of the inference question, i.e., different variables, if we change from Random Forest classifiers to more structured models. To decrease inference time of structured models while retaining its convergence properties, we propose to leverage multiple computers each having multiple cores. In order not to suffer from low-latency access to storage partitioned onto different computers, we propose to augment the optimization to reduce transmission of information between machines connected only via networks which are relatively slow compared to direct memory access.

- For inference in structured models it was empirically shown that finding the optimal answer of a particular problem relaxation, i.e., an approximation, yields good results if the original problem is computationally intractable. This approximation is however challenging since the fastest methods are applicable, but not guaranteed to converge to the global optimum. We therefore present how to design a globally convergent algorithm.

- It is not only the inference task for structured models that is computationally demanding. Finding the parameters of our models and algorithms, i.e., solving the learning task, is at least as challenging. We therefore extend our mechanism of partitioning the inference task to the optimization problem targeted when learning the parameters of structured models, and investigate the resulting performance improvements.
1. Introduction

- For the learning task to succeed in general, we provide data samples and desired answers, which is known as complete supervision. It is often time-consuming, costly and sometimes not even possible to obtain a complete answer. We therefore design an algorithm to learn parameters given partial answers and show that our approach is indeed capable of extract information from weak supervision.

- All of the above contributions assume that a domain expert has designed a favorable representation for the data and for the answers considered to be a valid response. For the questions considered during inference, we illustrate the importance of designing representations for the data, and representations for the valid answers. To this end, we focus on the 3D indoor scene understanding task of finding a parametric box that best fits the scene visualized within a single image. We show that we can derive globally optimal inference methods that achieve state-of-the-art performance in a fraction of the time required for existing methods and present extensions towards reasoning about objects commonly observed in a scene.

1.4. Thesis Outline

After having reviewed related work in Chapter 2 we subsequently present the aforementioned contributions.

Firstly, discussing inference, we detail in Chapter 3 how to augment Random Forest prediction via a confidence analysis. In Sec. 4.2 of Chapter 4 we derive how to distribute optimization in structured problems such that the task is suitable to be solved in parallel even when machines are connected only via high-latency networks. Afterwards we address global convergence of a specific inference algorithm in Sec. 4.3.

Secondly, we turn our attention to distributing existing learning algorithms in Chapter 5 such that we can leverage multiple computers commonly available these days.

Having a set of inference and learning algorithms at our hands we subsequently consider in Chapter 6 the application of estimating the 3D parametric box given a single image displaying an indoor scene. We will use this example to discuss important details regarding representation of data and answers.

Afterwards we describe in Chapter 7 how to extend our general algorithms to also cope with missing data.

We conclude with some discussion regarding topics for future work in Chapter 8. Since we implemented all the presented algorithms as libraries in the programming language C++, we briefly describe the features of the individual packages in Appendix A.
2. Background and Related Work

Having highlighted the importance and applicability of efficient inference and learning in the previous chapter, we detail the notation as well as the relationship between our contributions and work developed before, during and after having published our thoughts on specific topics in the following.

2.1. Formal Problem Definition

Throughout this thesis we refer to $\mathcal{X}$ as the input space which describes our data without regards to a specific representation. Thinking about recommender systems, an object $x \in \mathcal{X}$ from the input space might denote the timestamp including browsing history of a particular user and co-occurring purchases, while $x$ denotes the words of a sentence when working in the part-of-speech tagging domain. For protein design problems $x$ refers to the backbone structure and its properties. Considering an image, an object $x \in \mathcal{X}$ from the input space denotes the color values for each pixel without specifying the particular form of $x$ being a vector, matrix or tensor.

While the input space is commonly defined by the considered application, i.e., all the data available to us, and in general not to be changed, it is the domain for the possibly valid answers that constitutes a first crucial choice. In general we let $\mathcal{S}$ denote the output space of interest also known as the space of configurations, i.e., the space of answers to be considered as valid. A magnitude of possibilities exist for every application and some output spaces are likely more tractable than others. For the 3D indoor scene understanding example, $\mathcal{S}$ could refer to all possible projected 3D boxes in an enumerative manner by listing them one after the other. Alternatively, we might consider every pixel of an image to be assigned to a particular wall of a 3D box, while ‘somehow’ enforcing consistency. Having different image sizes in mind, we note that the output space $\mathcal{S}$ can vary from data sample to data sample. For clarity of the presentation we neglect this dependence throughout the thesis while noting that the provided implementations do not possess this restriction.

For many applications it is suitable to decompose the possible answers $s \in \mathcal{S}$ into sub-parts, i.e., smaller sub-spaces. We let such a sub-space object be referred to via $s_i \in \mathcal{S}_i$ with $i \in \{1, \ldots, N\}$ and we further let a tuple $s = (s_1, \ldots, s_N)$ denote the structured answer or configuration. Note that the space of configurations $\mathcal{S}$ is hence a product space $\mathcal{S} = \prod_{i=1}^{N} \mathcal{S}_i$. Throughout this thesis $\mathcal{S}$ and $\mathcal{S}_i$ are assumed to be discrete spaces.

Recall the introduction, where we abstractly defined inference to be concerned with answering questions. We therefore want to compare different answers, i.e., different output space objects $s \in \mathcal{S}$. Given a data sample $x$ we most generally want inference
2. Background and Related Work

to retrieve a probability distribution $p(s \mid x)$ over the output space $\mathcal{S}$. Finding the most likely answer amounts to finding the state maximizing the distribution $p(s \mid x)$.

Oftentimes it is hard to define a distribution $p(s \mid x)$ over the output space $\mathcal{S}$ manually. Hence the distribution is not directly computed using the data but rather employs some specific measurements, commonly referred to as features, which originate from the data. In its most general form we describe a feature vector as a mapping from the product of input and output space to an $F$-dimensional real valued vector, i.e., more formally, $\phi(x, s) : \mathcal{X} \times \mathcal{S} \mapsto \mathbb{R}^F$. Besides defining an appropriately parameterized output space $\mathcal{S}$ it is also the features $\phi$ that are of crucial importance for the success of an application. Considering the 3D indoor scene understanding task, a feature vector may for example contain elements that measure the color intensity of a pixel at the location of interest.

To define the learning task considered within this thesis in a general manner we augment the probability distribution $p(s \mid x)$ to depend on a set of parameters subsumed within a $W$-dimensional vector $w \in \mathbb{R}^W$. The learning task considered within this thesis hence generally amounts to choosing the weight vector $w$ of our parametric distribution $p(s \mid x, w)$.

Instead of manually choosing the weights $w$, a learning algorithm is employed to find the parameters in a data driven way, based on available observations, i.e., annotated data. Therefore we subsume annotations within a data set $\mathcal{D}$. It contains $|\mathcal{D}|$ pairs $(x, s)$, each consisting of an input space object $x \in \mathcal{X}$ and a corresponding ground truth annotation – an answer to the question considered for inference – $s \in \mathcal{S}$ being an element from the output space, i.e., $\mathcal{D} = \{(x^{(i)}, s^{(i)})\}_{i=1}^{\mathcal{D}}$.

For particular tasks we imagine answering the question of interest during inference to be time consuming, even for a domain expert. Think about choosing the type of side chain, its angular orientation and placement which represents the most stable configuration of a protein. This might in general not even be possible and we are therefore interested in designing learning algorithms that only require part of the question to be answered. We divide the output space $\mathcal{S} = \mathcal{Y} \times \mathcal{H}$ into two disjunct parts, the annotated space $\mathcal{Y}$ and the unobserved or latent variables subsumed within the hidden space $\mathcal{H}$. Note again that this disjunct partitioning is commonly data dependent. Similar to the output space $\mathcal{S}$ we neglect this dependence for clarity of the presentation.

The aforementioned general definitions of the input space $\mathcal{X}$, the output space $\mathcal{S}$, the annotated space $\mathcal{Y}$, the hidden space $\mathcal{H}$, inference to find a distribution $p(s \mid x)$ over the output space, features $\phi(x, s) \in \mathbb{R}^F$ which map to a vector space, learning to obtain the parameters $w \in \mathbb{R}^W$ and the data set $\mathcal{D}$ are used through this thesis and more concretely defined in respective sections.

2.2. Inference in Structured Models

Having formally defined the problem and its ingredients, we first provide a historical overview of topics related to statistical inference and learning. This is by no means
complete and we apologize for any important contributions that slipped our minds when writing this part.

A formal definition of probability and its fundamental principles wasn’t available until the 17th century when French mathematicians Blaise Pascal and Pierre de Fermat discussed problems on games of chance in a famous correspondence (Apostol, 1969)\textsuperscript{1}.

Many well known mathematicians from Christiaan Huygens and Jacob Bernoulli to Thomas Bayes, Carl Friedrich Gauss, Adrien-Marie Legendre and Pierre-Simon Laplace have since contributed to the evolution of the field until it was based on axiomatic grounds by Andrey Kolmogorov in 1933 (available in English as (Kolmogorov, 1956)).

Most notably however, inference in structured models has originally been considered by statistical physicists. Of particular interest and related to approaches considered till date are Ising models and generalizations known as spin-glasses. Good books on the topic have been written by Mézard et al. (1987), Mézard & Montanari (2009) and Talagrand (2011). The Ising model was invented by physicist Wilhelm Lenz in 1920 and the one-dimensional case was solved by his student Ernst Ising (1924) in his thesis. To relate Ising models to the notation within this work, let the N ‘spin-1/2’ particles or sites $s_i$, $i \in \{1, \ldots, N\}$ be variables of the output space, i.e., $s = (s_1, \ldots, s_N) \in S$. The sites are arranged in a chain or a lattice, and every particle $s_i \in S_i = \{1, -1\}$ takes either an up or a down spin. The configuration probability is referred to via $p(s \mid x)$ where $x$ denotes an external magnetic field and interactions for possibly any pair of particles.

The two dimensional case was however not accessible until Bethe (1935) proposed an approximation known as “Bethe lattice” and frequently referred to as “Bethe approximation” these days. Rather than describing a 2D lattice of particles via a grid like structure, i.e., a graph with loops, Bethe proposed to use an infinitely connected, cycle-free tree-structured graph. As a consequence of using a tree-structured graph, the approximate models are exactly solvable. Later, Onsager (1944) derived an exact solution for the Ising model on a 2D lattice without external magnetic field and periodic boundary conditions. A tractable solution for the 2D lattice of the more general spin-glasses, also without an external magnetic field, was given by Kasteleyn (1963); Fisher (1966). Exact solutions for other special cases have been found till date, but their discussion is beyond the scope of this thesis.

In an attempt to facilitate exchange between statistical physics and probability theory, Kindermann & Snell (1980) relate Ising models to Markov random fields. A particular representation of a probability distribution was referred to as Markov networks (Geman & Geman, 1984) which became later known more generally as undirected graphical models (Lauritzen, 1996; Koller & Friedman, 2009).

A graphical model consists of nodes which represent random variables $s_i$, $i \in \{1, \ldots, N\}$ arranged in an arbitrary configuration and connected via an edge if dependencies between two variables exist. Every random variable $s_i$ takes one of the states specified via the configuration space $S_i$, i.e., $s_i \in S_i$. The Markov property within such a general network is stated by taking any two subsets of variables. Those subsets

\textsuperscript{1}excerpt available from http://homepages.wmich.edu/~mackey/Teaching/145/probHist.html on Aug. 11, 2013
are conditionally independent given a separating subset if any path between the two subsets passes through the separating set.

Graphical models as visual illustration of the factorization properties of probability distributions have since been extended and generalized. Instead of connecting two variables via edges to illustrate the dependencies, bipartite factor graphs allow to visualize relationships between any number of variables (Kschischang et al., 2001). Every factor of the probability distribution is connected to the variables being its functional arguments. Even greater visualization flexibility of the distributions properties is obtained with Hasse diagrams which capture approximation properties (Yedidia et al., 2005). We employ the latter throughout this thesis and provide details in respective chapters.

These days, graphical models are frequently employed in many domains, for example for image segmentation (Levin & Weiss, 2006), image completion (Levin et al., 2003), stereo matching (Sun et al., 2003) and super-resolution (Freeman et al., 2002) when considering computer vision. Independent of the application, two tasks are commonly of interest. Those are computation of the ground states of the distribution, i.e., its modes or most likely configurations, and computation of the partition function which is equivalent to computation of the marginals. Although related by a temperature parameter, we subsequently detail both tasks separately in order to better capture algorithmic and historic relationships.

2.2.1. Computation of the Partition Function

Similar to Ising (1924), Onsager (1944) originally employed geometric series and smart counting strategies to obtain the value of the normalizing constant of \( p(s \mid x) \) being known as the partition function. More formal approaches have since been developed, often following the dynamic programing paradigm.

For inference on tree-structured graphs, and more specifically to compute the marginals \( p(s_i \mid x) = \sum_{s \setminus s_i} p(s \mid x) \), Pearl (1982) proposed an algorithm known today as belief propagation or more precisely as sum-product message passing. Soon afterwards the approach was extended to polytrees by Kim & Pearl (1983). On generally structured graphs, the algorithm was also shown, e.g., in (Pearl, 1988; MacKay & Neal, 1995; Weiss, 1996; Frey 1997; Murphy & Weiss, 1999), to be a useful approximation now known as “loopy belief propagation.” It was however noted by Pearl (1988) that loopy belief propagation has to avoid double counting, i.e., evidence passed around in the network should not be considered as new evidence by mistake.

This double counting is best observed by unwrapping a loopy graphical model as described in (MacKay & Neal, 1995; Weiss, 1996; Frey et al., 1998; Weiss, 2000) to obtain a computation tree which corresponds to the “Bethe lattice” introduced as an approximation by Bethe (1935). Due to double counting, convergence of the sum-product algorithm is not guaranteed for loopy graphical models and there has been a large body of work to understand and fix convergence issues.

For graphical models with a single loop, Weiss (2000) showed how to analytically relate the marginals obtained by the sum-product algorithm to the true ones. Tatikonda & Jordan (2002) show that the sum-product algorithm converges uniformly if the Gibbs
measure on the unwrapped computation tree is unique, which is however non-trivial to proof in general. Ihler et al. (2004) answered the question regarding convergence by considering contractive mappings for each iteration which were improved in Ihler et al. (2005) and strengthened in independent work by Mooij & Kappen (2007).

A different methodology to proof uniqueness of fixed-points for the sum-product algorithm is presented by Heskes (2004) and earlier work in (Heskes, 2002), which made use of the important connection between the Bethe free energy and the sum-product algorithm (Yedidia et al., 2001). On a high-level, the idea is to consider inference as a constrained optimization problem with the generally non-convex cost function being the Bethe free energy. Showing that the energy for specific types of graphs is convex, is key for Heskes (2004) to proof uniqueness of fixed-points.

It was shown by Kschischang et al. (2001) that the sum-product algorithm corresponds to sending messages on a factor graph. This insight is important since it illustrates the effectiveness of sum-product message passing compared to standard solvers: sum-product message passing takes the graphical model structure of the problem into account.

Double-loop algorithms (Yuille, 2002; Teh & Welling, 2002; Heskes et al., 2003) have been proposed to guarantee convergence for non-convex optimization problems arising from loopy graphical models, but they are an order of magnitude slower than the sum-product algorithm consisting of a single loop.

Still considering the cost function, generalizations to other types of energies have been proposed by Yedidia et al. (2001); Welling (2004); Yedidia et al. (2005); Pakzad & Anantharam (2005). Due to convergence guarantees, convex approximations (Heskes, 2006; Yanover et al., 2006; Weiss et al., 2007; Meltzer et al., 2009; Hazan & Shashua, 2010), and “tree-reweighted” free energies (Wainwright et al., 2003, 2005a,b) mostly based on the fractional belief propagation idea of Wiegerinck & Heskes (2003) are of particular interest. Fixing the convergence of the original tree-reweighted algorithms by suggesting a sequential version was presented by Kolmogorov (2006). Replacing the exactly computable tree object by planar graph decompositions following Kasteleyn (1963); Fisher (1966) was considered by Globerson & Jaakkola (2006). The accuracy of different approximations was, e.g., investigated by Meshi et al. (2009).

Convergent message passing algorithms for convex free energies based on block-coordinate descent algorithms following Tseng & Bertsekas (1987, 1991); Tseng (1990, 1993, 2001) were derived by Globerson & Jaakkola (2007) taking the edges of two connecting variables as a block. Increasing the number of variables in a block was discussed by Hazan & Shashua (2008, 2010). More general updates involving tree-like objects were proposed by Komodakis et al. (2007); Sontag & Jaakkola (2009) and Tarlow et al. (2011). Importantly all those solvers make use of the graphical model structure.

Besides the aforementioned free energy approximations related to the cost function of the optimization problem, the approximated constraint set is of importance. A local polytope is commonly considered to outer-bound the marginal polytope (Wainwright & Jordan, 2008), also discussed in earlier work, e.g., (Wainwright & Jordan, 2003a). Tighter relaxations were considered in work by Sontag & Jaakkola (2007); Sontag et al. (2008); Batra et al. (2011); Sontag et al. (2012). Importantly it is the work
by Wainwright & Jordan (2003a) based on variational methods as, *e.g.*, discussed by Jordan *et al.* (1999); Neal & Hinton (1999), which emphasizes a connection between computation of the marginals via the sum-product algorithm and finding the best scoring configuration via what is known as the max-product algorithm and LP relaxations.

### 2.2.2. Computation of the Most Likely Labeling

The problem of finding the most likely assignment $s^* = \arg \max_{s \in S} p(s \mid x)$, *i.e.*, the maximum a-posteriori (MAP) labeling has strong relations to dynamic programming (Bellman, 1954) with application, *e.g.*, in computer vision by Amini *et al.* (1990). For tree-structured models the problem is solvable in polynomial time and algorithms have been discovered independently in different fields, *e.g.*, as Viterbi (1967) algorithm or forward-backward algorithm as well as the inside-outside algorithm by Baker (1979).

In the late 90s and early 2000s it was trendy within the computer vision community to phrase energy minimization problems on graphs and utilize the duality between max-flow and min-cut (Ford & Fulkerson, 1956; Elias *et al.*, 1956; Ford & Fulkerson, 1962) to obtain either globally optimal results as in (Boykov *et al.*, 1998) or local optima as, *e.g.*, in (Boykov *et al.*, 2001).

A general graph construction suitable for many problems was illustrated by Kolmogorov & Zabih (2004) and it was shown to extend cut functions (Cunningham, 1985), a class of sub-modular functions that can be minimized more efficiently than the general, still polynomial-time minimizable sub-modular function (Grötschel *et al.*, 1988; Schrijver, 2000; Iwata *et al.*, 2001).

Approximations to optimize non-sub-modular functions, among them multi-label problems, have subsequently been constructed. Fusion moves proposed by Lempitsky *et al.* (2007) as well as $\alpha$-expansion and $\alpha\beta$-swaps described in (Boykov *et al.*, 2001) are algorithms for which individual steps are solved to optimality.

Even before employing max-flow algorithms to Markov random field problems and as detailed by Werner (2007), it was proposed by Schlesinger (1976) to relax a resulting integer programming formulation. The proposed linear programming (LP) relaxation is similar in spirit to work by Koster *et al.* (1998); Koster (1999); Chekuri *et al.* (2001); Wainwright *et al.* (2002, 2005a). Note that NP-completeness of the general MAP task was also established by Shimony (1994).

Equivalence between different relaxations were shown in (Hammer *et al.*, 1984; Kolmogorov & Wainwright, 2005) and different approaches, *e.g.*, quadratic programming relaxations (Ravikumar & Lafferty, 2006) and semidefinite programming (Wainwright & Jordan, 2003b) have also been considered.

The max-product algorithm as the zero-temperature limit of the sum-product algorithm has been revitalized by Wainwright & Jordan (2003a) as a very important connection to unify the inference tasks for computation of the marginals and the MAP configuration.

Many of the proposed algorithms for solving the linear programming (LP) relaxation of the MAP inference task are guaranteed to converge (Schlesinger, 1976; Koster *et al.*, 1998; Wainwright *et al.*, 2005a; Kolmogorov, 2006; Globerson & Jaakkola, 2007;
2.2. Inference in Structured Models

Meltzer et al., 2009; Sontag & Jaakkola, 2009; Hazan & Shashua, 2010; Tarlow et al., 2011) but convergence to the global optimum of the LP relaxation is only guaranteed upon augmentation to yield strictly convex problems, e.g., by smoothing the objective or by augmented Lagrangian methods (Johnson, 2008; Jojic et al., 2010; Ravikumar et al., 2010; Werner, 2010; Hazan & Shashua, 2010; Martins et al., 2011; Meshi & Globerson, 2011; Savchynskyy et al., 2012). Subgradient methods (Komodakis et al., 2010) or bundle approaches (Kappes et al., 2012), which construct polyhedral approximations based on subgradients, directly consider the original function and converge to the global optimum.

We introduce the optimization problems for computation of marginals and the most likely labeling more formally in Chapter 4. We will also detail our suggestion on how to distribute computation onto multiple computers commonly available these days by extending (Schwing et al., 2011a). Further we show how to obtain an efficient algorithm for solving the LP relaxation of the MAP inference task which converges to the global optimum without introducing artificial modifications and by improving upon subgradients. To this end we make use of what is known as the set of $\epsilon$-subdifferentials and replace the standard solver employed by Schwing et al. (2012c) by a conditional gradient approach to compute the steepest descent direction.

Several approaches have also been developed to parallelize and distribute graph cuts algorithms. Strandmark & Kahl (2010) and Shekhovtsov & Hlavac (2011) proposed a parallel and distributed graph cut using dual decomposition. Their method facilitates computation of larger problems by partitioning the model onto multiple machines. Similar to their intention we aim at assigning the inference task to multiple computers. Contrasting their work, we will derive a decomposition method for message-passing algorithms to ensure applicability for non-sub-modular potentials. Although some applications, e.g., observation selection, can be phrased to optimize a sub-modular energy (Krause, 2008), non-sub-modular potentials arise in many more applications such as image editing (Pritch et al., 2010) and segmentation (Kohli et al., 2009).

To allow for faster computation of message-passing solvers, Felzenszwalb & Huttenlocher (2006) proposed a red-black graph coloring algorithm for parallelizing loopy belief propagation (BP) on grid structures. Here, we extend this strategy to general graphs and Hasse diagrams using a greedy graph coloring algorithm, and employ this extension for local computations within each machine. The main drawback of the BP algorithm is that it is not guaranteed to converge in many cases of interest.

Hazan & Shashua (2008) introduced a parallel convex sum-product algorithm, which provably gives the optimal solution for strictly concave entropies. This algorithm differs from our proposed approach in important aspects: it propagates only a fraction of information through its messages. It therefore converges slowly even for small graphs, and it is numerically unstable for large graphs which are the focus of this work.

Recently, Low et al. (2010) presented GraphLab, a framework for efficient and provably convergent parallel algorithms. They show impressive results on typical machine learning tasks such as belief propagation by improving on the MapReduce abstraction. Unfortunately, their implementation assumes that all the data is stored in shared-memory, which makes it infeasible to apply GraphLab to large scale problems. The shared memory assumption is severe as it does not allow efficient distribution
of the task at hand to multiple machines. In contrast, we distribute the memory requirements such that this is no longer a burden.

More recently Low et al. (2012) and Gonzalez et al. (2012) also presented algorithms for distributed memory environments. Contrasting their methodology we employ dual decomposition to minimize the required transmissions between different computers while maintaining convergence guarantees.

2.3. Learning

Learning the parameters of a distribution has its origins in a variety of fields. In statistics, researchers are interested, for example, in modeling distributions given some observations. Trying to model the human brain lead to the study of non-linear elements and their connection to memorize more complicated functions, i.e., structured models. In contrast, artificial intelligence researchers were frequently interested in studying games and related interpretable models for decisions. We subsequently review the resulting learning approaches briefly.

2.3.1. Learning Decision Trees

To obtain interpretable decision models, trees were utilized. Starting at the root node, a test determines which leaf node to progress to. Learning discriminative tree structures lead to models like the iterative dichotomiser 3 (ID3) (Quinlan, 1986) and its improvement C4.5 (Quinlan, 1993) as well as classification and regression trees (CARTs) described by Breiman et al. (1984).

By combining the bagging idea (Breiman, 1996), i.e., model averaging, with CARTs and random selection of features (Amit & Geman, 1997; Ho, 1998), Breiman (2001) invented an ensemble classifier named Random Forest. A variety of subsequent work improved upon the original idea as, e.g., summarized in Criminisi et al. (2012). The algorithm is frequently used in computer vision these days, e.g., (Shotton et al., 2008).

The description in Chapter 3 follows Schwing et al. (2011b) and reduces the amount of models employed for averaging during Random Forest classification in case the confidence of the obtained result is sufficiently high. Note that a similar approach for a subset of our results was independently developed by Hernández-Lobato et al. (2009).

2.3.2. Learning Parameters of Structured Models

In addition to tree-like models, it is the structure of the neurons within the human brain that has lead researchers to construct early learning algorithms based on neural networks, e.g., a network of perceptrons (Rosenblatt, 1957). The perceptron is commonly viewed as a linear and binary classifier. Clearly, single- and multi-layer perceptrons/neural networks are able to represent significantly more complex functions than a single element.

Importantly, the perceptron can also be seen as the building block of support vector networks (Cortes & Vapnik, 1995) nowadays known as support vector machines.
Similarly, Boltzmann machines (Hinton et al., 1984) as well as their restrictions and generalizations employ building blocks in spirit similar to a perceptron combined with a non-linearity.

Importantly, the original support vector machine formulation considered only a single output space variable and aims at maximizing the margin between the decision boundary and the samples taking different states. Extensions to structured output spaces, i.e., spaces where the answer is composed of multiple but distinct sub-parts, were considered by Taskar et al. (2003) in work on max-margin Markov networks and by Tschantzaridis et al. (2005) on structured support vector machines. To solve the respective optimization algorithmically, early approaches focused on problems with inference being exactly solvable via dynamic programming (Taskar et al., 2003) or combinatorial matchings (Taskar et al., 2006). Further, primal methods like stochastic subgradient (Ratliff et al., 2007) and cutting plane methods (Joachims et al., 2009) have been suggested among others, while a dual approach is presented by Taskar et al. (2005) employing a standard quadratic programming solver and Lacoste-Julien et al. (2013) proposing a Frank-Wolfe method.

Instead of maximizing the margin, work on conditional random fields learns a log-linear combination of features to maximize the likelihood of the training data set (Lafferty et al., 2001). Again, early work focused on tasks where inference is exactly solvable, e.g., via Viterbi decoding combined with a perceptron update (Collins, 2002). Afterwards more general models and methods have been considered, e.g., exponentiated gradients by Collins et al. (2008) and approximations of the gradient via loopy belief propagation. Relationship of conditional random fields with maximum entropy approaches are illustrated by Pietra et al. (1997, 2001) and Dudík et al. (2004); Dudík & Schapire (2006) and entropy approximations were investigated, e.g., by Ganapathi et al. (2008).

Algorithmically, all previous approaches required to solve an inference problem before performing an update step w.r.t. the parameters. For max-margin methods it was first recognized by Meshi et al. (2010) and Hazan & Urtasun (2010) that inference and parameter updates can be blended/interleaved.

Moreover, combining the two seemingly different algorithms, i.e., max-margin methods and maximum likelihood approaches, via a temperature parameter was first proposed by Pletscher et al. (2010); Hazan & Urtasun (2010). It was the work of Hazan & Urtasun (2010) which also suggested to blend the inference step and the parameter update whenever the entropy approximation is convex.


As detailed in Chapter 5 we describe a method to distribute computation of structured predictors and extend the approach described by Schwing et al. (2012d).
2.3.3. Learning Parameters of Latent Structured Models

The previously mentioned learning algorithms usually require the entire output space \( S \) to be annotated. In case we obtain data with unobserved parts and hence latent variables, expectation maximization (EM) (Dempster et al., 1977; McLachlan & Krishnan, 1997; Borman, 2004) is a frequently employed technique which was later generalized to the concave-convex procedure (Yuille & Rangarajan, 2003; Sriperumbudur & Lanckriet, 2009).

Computing the mean and covariance parameters of a Gaussian mixture model, or obtaining the cluster centers of a k-means algorithm (Lloyd, 1982) are prominent examples derived via the EM framework.

Learning in structured spaces with latent variables was first proposed by generalizing the hidden Markov model to hidden conditional random fields (HCRFs) (Quattoni et al., 2007). Following the two-step EM procedure, HCRFs compute the expectation of the missing data by solving a ‘latent variable prediction problem’ to augment the empirical observations of the measurements by estimates of the latent variables. In a subsequent step the parameters are updated. Since the update of the parameters is an algorithm required to converge on its own, we can think of HCRFs as a double-loop algorithm.

Following the maximum likelihood approach, learning a max-margin model using a ‘latent structured support vector machine’ was suggested by Yu & Joachims (2009).

Extensions to higher order functions (Komodakis, 2011), successive introduction of difficult data via self-paced learning (Kumar et al., 2010), hidden units between data and output space variables (v.d.Maaten et al., 2011) and more general entropies (Miller et al., 2012) have subsequently been proposed.

To learn the parameters of structured models with missing data by interleaving the optimization in case of convex approximations, we extend the approach presented by Schwing et al. (2012b) to the more general Hasse diagrams in Chapter 7.
3. Adaptive Random Forest

3.1. Review

For discussing inference with Random Forest classifiers, let the output space contain a single variable \( s \). This variable takes on discrete values with all its possible states being subsumed within the set of output space configurations \( \mathcal{S} \). In the classification context, elements within \( \mathcal{S} \) are often also referred to as class labels.

To make the formalism more concrete, think about the application of detecting faces in an image. We consider a single pixel at a time and the possible answers for the inference task are \( \mathcal{S} = \{ \text{face, no-face} \} = \{1, 2\} \). Choosing the face-state to be represented by 1 is arbitrary and of no importance. To classify the entire image we process all pixels in any given ordering, independent of one another. Since we are however interested in classifying every pixel of the image we say that the individual variables are independent and not related to each other.

Carrying the output space representation to the other extreme we define the configuration space of our variable to contain all possible labellings of the entire image. Hence every element within the set of possible output space objects describes a complete labeling of the image. While we are able to jointly reason about all the pixels in an image, we note that the size of the output space \( |\mathcal{S}| \) grows quickly even when increasing the size of the image only moderately. Taking computational resources into account, it is therefore beneficial to go for an approach somewhere between the two extremes even though we continue our subsequent considerations with a prediction operating on a pixel by pixel basis for simplicity.

The feature vector \( \phi \) of a Random Forest classifier often depends only on the considered data sample \( x \) and not necessarily on the output space, i.e., we let \( \phi(x) \in \mathbb{R}^F \) denote the features for the input \( x \). When detecting faces in an image we let the input space object \( x \) denote the considered image and a particular pixel location. An element of the feature vector \( \phi(x) \) represents a measurement of, e.g., intensity, gradients and possibly many other values on and around the pixel location subsumed by sample \( x \). Note that the computational complexity of the elements within the feature vector varies significantly in general.

A Random Forest consists of a set of classification trees with inference processing tree after tree, starting from a distinct root node and terminating at one leaf node. Many variants of trees exist and in the following we consider binary trees for simplicity. During inference we compute at every node a particular feature and compare it to a threshold to determine whether the sample is passed to the right or left child. Classification with tree \( t \) terminates once we reach a leaf node. Each leaf node contains a normalized histogram over the possible class labels, i.e., the elements within the
output space $S$. We refer to the distribution within the leaf node of tree $t$ reached by sample $x$ via $p_t(s \mid x)$.

Having passed the sample $x$ through all the trees, inference computes the probability distribution over the output space $S$ by averaging the obtained leaf node histograms, i.e.,

$$p(s \mid x) = \frac{1}{T} \sum_{t=1}^{T} p_t(s \mid x),$$

with $T$ denoting the number of trees.

Learning from a given set of annotated data samples is concerned with constructing the distributions $p_t(s \mid x)$ and the trees, i.e., the parameters $w$ denoting which feature to compare with which threshold at which node. It is beyond the scope of this thesis to describe the learning procedure. We refer the interested reader to the respective literature for details.

Instead we investigate the inference methodology more carefully. The number of trees $t$ is often as large as several tens or hundreds. Passing a sample $x$ through a tree amounts to comparing elements of the feature vector to thresholds. Since the number of features easily exceeds ten-thousand, while the order of magnitude of the tree-depth is around several tens, it is common for real-world applications to compute an entry of the feature vector only if it is required.

Assume that we are interested in finding the output space configuration which maximizes the probability distribution $p(s \mid x)$. If we are sufficiently confident about the prediction result after having evaluated a subset of the available trees, we may terminate classification of this sample by neglecting the remaining trees and instead progress to classify the next input space object $x$. We subsequently show that such an early termination strategy does not harm performance while it potentially increases classification speed.

To give an example, suppose we employ 100 trees for a binary classification task where each leaf node casts a vote for one of the two classes. We refer to $v_1$ as the
number of votes for the first class and $v_2$ stores the number of votes for the second class. When successively progressing through the 100 trees for a set of samples, we visualize in Fig. 3.1 how frequently a proportion $(v_1, v_2)$ was hit. Note that to the best of our knowledge there is no evidence favoring a distribution $p_t(s \mid x)$ rather than a unit vote, i.e., $p_t(s \mid x)$ being zero for all but one output space configuration. However and as detailed later, full distributions are easily included into the presented methods.

We propose an analysis based on the assumptions that 1) each “expert” makes an independent prediction and 2) the “experts” are approximately equally knowledgeable. The latter assumption holds well for Random Forest classifiers. However, the first assumption does not follow from the injected randomness which strives for uncorrelated trees (Breiman, 2001). Consequently, higher order moments do not necessarily drop out and the trees are not independent. Experiments nevertheless show, that we obtain encouraging results.

To achieve improved classification efficiency, several additional methods for Random Forest classifiers were proposed. The parallelized architecture of the graphics processing unit was leveraged by Sharp (2008). Another approach named Random Fern was proposed by Özuysal et al. (2007). Similar to Random Forest classifiers, the sample is passed down the set of ferns. Each node within a fern provides a result for the binary test which is used to access the leaf node containing the posterior probability $p_t(s \mid x)$.

Many more methods besides Random Forests were examined w.r.t. efficiency improvements. Among those are random sample consensus (RANSAC) (Fischler & Bolles, 1981) and other classifiers like Boosting (Freund & Schapire, 1996; Schapire et al., 1998) or neural networks. Methods applied there are training of a rejection trace (Bourdev & Brandt, 2005), genetic algorithms (Zhou et al., 2002) and the sequential probability-ratio test (SPRT) (Wald, 1947) used, e.g., in (Šochman & Matas, 2005; Chum & Matas, 2008) for Boosting and RANSAC. SPRT is a likelihood-ratio test which distinguishes two hypotheses. Thus it is applicable to binary classification problems but contrasting our approach, extensions to multiple classes are not straightforward.

### 3.2. Faster Inference via Adaptive Tree Selection

To detail our early termination which leverages confidence intervals, we distinguish between binomial and multinomial classification in the following. The multinomial formulation is of course applicable to a binomial problem. For both cases all the early termination criteria are pre-computed. During inference we hence just retrieve a value from the respective position in a table and our online pruning technique hardly adds any computational complexity to the classification task.

#### 3.2.1. Binomial Prediction

Given a sample, we consult $V$ trees and $v_1$ cast a vote for the first label being also referred to as positive class. An intuitive estimate of the probability for this sample to be positive is $p = \frac{v_1}{V}$. Is this an unbiased estimate? How reliable is this estimate?
3. Adaptive Random Forest

Figure 3.2.: The binomial case: (a) illustrates the probability $P(p \geq 0.5 \mid v_1, V)$ for the current votes $(v_1, v_2)$ and (b) shows the region of early termination obtained with a confidence of $1 - \alpha = 0.9$ when observing different votes $(v_1, v_2)$. If we stop consulting additional classifiers and draw the conclusion that this sample is of positive class 1, what is the probability that we make a correct decision given the proportion $v_1$ and the number of consulted trees $V$? In this section, we will answer all these questions.

Consider $V$ trees and suppose each one independently casts a vote for the first object class with probability $p$. The probability of observing $v_1$ (where $0 \leq v_1 \leq V$) positive tests follows the binomial distribution

$$b(v_1 \mid p, V) = \binom{V}{v_1} p^{v_1} (1 - p)^{V - v_1}.$$ (3.1)

Note that the following derivation is analogous for the number of votes $v_2$ for the negative class.

Claim 1 Assuming no a priori knowledge, the distribution for the classification probability $p \in [0, 1]$ given $v_1$ out of $V$ votes is

$$P(p \mid v_1, V) = \frac{(V + 1)!}{v_1! (V - v_1)!} p^{v_1} (1 - p)^{V - v_1}.$$ (3.2)

Proof: Using Bayes’ rule we compute the distribution of $p$ given the object votes $v_1$ and the number of trials $V$ as

$$P(p \mid v_1, V) = \frac{b(v_1 \mid p, V) P(p \mid V)}{\int_0^1 b(v_1 \mid p, V) P(p \mid V) \, dp}.$$ (3.3)

Without a priori knowledge about the distribution of $p$ we assume it to be least informative, i.e., we let $P(p \mid V) = P(p)$ be a uniform distribution independent of
3.2. Faster Inference via Adaptive Tree Selection

Figure 3.3.: A comparison of SPRT and our approach is shown in (a). The boundary in the multinomial case with proportions \((v_1, v_2, v_3)\) and confidence \(1 - \alpha = 0.9\) is given in (b).

the number of trees \(V\). Simplifying Eq. (3.3) we obtain

\[
P(p \mid v_1, V) = \frac{b(v_1 \mid p, V)}{\int_0^1 b(v_1 \mid p, V) \, dp}.
\] (3.4)

Plugging the binomial distribution given in Eq. (3.1) into Eq. (3.4) yields the conjugate prior of the binomial, i.e., the beta distribution given in Eq. (3.2) which concludes the proof.

By computing the mode of the distribution given in Eq. (3.2), it is easy to verify that \(\hat{p} = v_1/V\) is a maximum likelihood estimate. But the unbiased estimate for a beta distribution is given as

\[
\mathbb{E}_{P(p \mid v_1, V)}[p] = \frac{v_1 + 1}{V + 2}.
\] (3.5)

The unbiased estimate is slightly smaller than the maximum likelihood result. We will indicate the difference between this estimator and the maximum likelihood result during our large scale experiments.

Having computed the distribution for our random variable \(p\) as given in Eq. (3.2), we calculate the probability to make a correct decision for a positive label, i.e.,

\[
P(p \geq 0.5 \mid v_1, V) = \int_{0.5}^1 P(p \mid v_1, V) \, dp
\]

to be

\[
P(p \geq 0.5 \mid v_1, V) = 1 - \frac{(V + 1)!0.5^{v_1+1}}{(v_1 + 1)!(V - v_1)!} 2F_1(v_1 + 1, v_1 - V; v_1 + 2; 0.5)
\] (3.6)

with \(2F_1(v_1 + 1, v_1 - V; v_1 + 2; 0.5)\) being the hypergeometric function.

The probability \(P(p \geq 0.5 \mid v_1, V)\) is illustrated in Fig. 3.2(a) for the voting results \((v_1, v_2)\) with \(v_1 + v_2 = V\). If it exceeds a confidence \(1 - \alpha\), we stop consulting additional trees.
Assume we obtain the current votes \((v_1, v_2)\) to decide whether to classify using additional trees. If either the probability for object samples or non-object samples exceeds a certain threshold \(1 - \alpha\) we stop. The level curves for some thresholds are depicted in Fig. 3.2(a). Thus we get the region of early termination marked with red color in Fig. 3.2(b). Note that the confidence \(1 - \alpha\) does not necessarily need to be equal for different classes. A sample given to a binary classifier will “walk” in the \((v_1, v_2)\) space until it either hits the area highlighted with red color or until no more trees are available. Interestingly our method inherently forces us to consult at least four trees to achieve a confidence of \(1 - \alpha = 0.9\). This is intuitive and automatically incorporated.

In Fig. 3.3(a) we compare the “termination boundary” of our proposed approach with the one obtained by a sequential probability ratio test (Wald, 1947), which is \(A \leq v_2 - v_1 \leq B\). The constants \(A (\xi_1, \xi_2, \sigma_1, \sigma_2)\) and \(B (\xi_1, \xi_2, \sigma_1, \sigma_2)\) depend on the error probabilities \(\xi_1, \xi_2\) and the hypothesis probabilities \(\sigma_1 > 0.5, \sigma_2\). Assuming equal treatment of object and non-object samples we obtain at least two parameters with \(\sigma_2 = 1 - \sigma_1\) and \(\xi_1 = \xi_2\), contrasting one confidence threshold \(\alpha\) necessary for our proposed approach. In favor of experiments, we refer the interested reader to, e.g., (Govindarajulu, 2004) for an in-depth discussion and examples regarding the sequential probability ratio test. Investigating Fig. 3.3(a) \((\alpha = \xi_1 = \xi_2 = 0.1, \sigma_1 = 0.6)\), the alternatively derived rule is more aggressive in making a decision when considering few trees and more cautious if the ensemble members cannot find a quick consensus. We emphasize however that this obviously depends on the chosen parameters.

Note that Eq. (3.6) depends only on the discrete values for the number of positive tests \(v_1\) and total number of tests \(V\). Additionally we know that the maximum amount of tests is bounded by the maximum amount of available trees \((V \leq T)\) and that the number of positive tests \(v_1\) is bounded by the total number of classifiers consulted so far. Summarizing those facts, we just need to store a polynomial amount of \((\frac{(T+1)(T+2)}{2}) - 1\) values which is tractable even for a large number of available base classifiers. Consequently the computation of Eq. (3.6) is replaced by a fast table lookup and almost no computational complexity is added. Further extending the computed table to \((\frac{(T+|S|+1)(T+|S|+2)}{2}) - 1\) entries, we incorporate distributions stored within the leaf nodes of each tree. We highlight that many values of the enlarged table are close to 0 or 1 which can be used to decrease its size.

### 3.2.2. Multinomial Prediction

Oftentimes we are interested in multi-class scenarios and our approach is applicable with few modifications. Keep in mind that we do not want to add significant computational complexity to the classification process.

Let the tuple \(v = (v_1, \ldots, v_{|S|})\) denote votes for classes with \(\sum_{i=1}^{|S|} v_i = V \leq T\) being the number of base classifiers consulted so far. Note the normalization constraint \(\sum_{i=1}^{|S|} p_i = 1\) governing the distribution \(p = (p_1, \ldots, p_{|S|})\) of interest. For the binomial formulation we derived usage of its conjugate prior, i.e., the beta distribution. In the following, we apply the Dirichlet distribution \(P (p \mid v, V) = \frac{1}{Z(v)} \prod_{i=1}^{|S|} v_i^{p_i-1}\) as the
3.2. Faster Inference via Adaptive Tree Selection

The conjugate prior of the multinomial without showing the derivation. The multinomial beta function \( Z(v) \) is used for normalization of the Dirichlet distribution. Contrasting the binomial case shown in Eq. (3.6), exact computation of the probability to make a correct decision for \( p_i \) is hard for \(|S| > 2\). The reason is the required marginalization over all variables except \( p_i \) and the one probability replaced via the normalization constraint. This step, not necessary for the binomial formulation, results in an integral that cannot be expressed analytically. We therefore turn to the maximum likelihood estimator of \( p = (p_1, \ldots, p_{|S|}) \). For a Dirichlet distribution it is \( \hat{p} = (\hat{p}_1, \ldots, \hat{p}_{|S|}) \) with \( \hat{p}_i = \frac{v_i}{V} \).

Contrasting the binomial case and as mentioned above, it is hard to derive a closed-form solution for the probability to make a correct decision, using a Dirichlet distribution for \( p \). Instead, we compute the confidence range for all the variables of the multinomial distribution of the probability \( p \in [0, 1]^{\mid S\mid} \). Thus, we want the probability that \( p_i \) is within the interval \([l_i, u_i]\) to be higher than the confidence \( 1 - \alpha \), i.e.,

\[
P(l_i \leq p_i \leq u_i) \geq 1 - \alpha.
\]

(3.7)

Given the bounds, it is easy to judge whether to stop the classification process or consult further trees. It is sufficient to compare the lower bound of the variable having the highest expected value \( \hat{p}_i \) with the upper bound of the others. Formally, we first find \( \gamma = \arg \max_{i \in S} \hat{p}_i \) and stop the classification if \( l_\gamma - u_i > 0 \ \forall i \in S \setminus \gamma \). Thus we end up with \((|S| - 1)\) comparisons.

To facilitate the comparison we need to compute confidence intervals for multinomial distributions in a fast manner. Moreover, we need a method providing reasonable results for a statistically small number of observations being at most the number of available trees \( T \). This problem has a long history and it is well known that the confidence regions built from asymptotic statistics do not have good coverage for few observations.

The method proposed by Chafaï & Concordet (2009) suggests conducting a comparison between two functions for each value of \( p \). Based on the result of the comparison they obtain the confidence region. It is generally not necessary in our case to compute the entire confidence region. Hence the number of comparisons used for this method can be reduced, but a simple computation is not possible. Another approach to construct simultaneous confidence intervals for multinomial proportions was described by Genz & Kwong (1999). Their method requires numerical solution of equations which will be too time consuming for our purposes. To really leverage the gain we prefer an analytic solution for the bounds. A number of different alternatives were proposed for asymptotic simultaneous confidence intervals for \( p \) (Gold, 1963; Quesenberry & Hurst, 1964; Bailey, 1980). Goodman (1965) modified the methods which were later further improved by Kwong & Iglewicz (1996). We found the method proposed by Quesenberry & Hurst (1964) to perform best for our purpose. Let the current proportions be
3. Adaptive Random Forest

\((v_1, \ldots, v_{|S|})\) with \(\sum_{i=1}^{|S|} v_i = V\). The bounds \(l_i(\alpha)\) and \(u_i(\alpha)\) are then given by

\[
\begin{align*}
    u_i(\alpha) &= \frac{\chi^2 + 2v_i + \chi(\chi^2 + 4\frac{v_i}{V}(V - v_i))^{\frac{1}{2}}}{2(V + \chi^2)} \\
    l_i(\alpha) &= \frac{\chi^2 + 2v_i - \chi(\chi^2 + 4\frac{v_i}{V}(V - v_i))^{\frac{1}{2}}}{2(V + \chi^2)}
\end{align*}
\]  

(3.8)

with \(\chi^2 = \chi^2_1\left(\frac{\alpha}{|S|}\right)\) defined as the 100\((1 - \frac{\alpha}{|S|})\) percentage point of the chi-squared distribution with one degree of freedom.

We illustrate the boundary between further consulting base classifiers and stopping for a three-class classification problem in Fig. 3.3(b). Investigating the graph closely, we encounter that trees are consulted as long as we cannot clearly distinguish between any two or all three of the classes. Again the approach forces us to consult at least a certain number of base classifiers before concluding with a decision. It is inherently built into our method without any heuristics.

Similar to the binomial case we pre-compute the bounds \(l_i(\alpha)\) and \(u_i(\alpha)\) necessary for Eq. (3.7) using the formula provided in Eq. (3.8) such that computation reduces to a simple table lookup given \(\alpha, |S|, v_i\) and \(V\). The additional complexity introduced are \((|S| - 1)\) comparisons and finding the most likely class. It remains to be shown in the following sections that we indeed achieve an improvement w.r.t. speed without degrading the accuracy.

3.3. Experimental Evaluation

We investigate the proposed termination criteria on synthetic small scale examples in Sec. 3.3.1 before turning our attention to medical imaging data in Sec. 3.3.2.

3.3.1. Evaluation on Small Scale Data Sets

Due to the saturation nicely shown by Hastie et al. (2009) and observed in our own results (e.g., Fig. 3.4), we decide to use 100 classification trees as a base, similar to Breiman (2001). The work presented in (Lempitsky et al., 2009) shows that nowadays the number of classifiers is decreased to a pre-determined threshold to obtain faster classification. We refer to such an approach as “Fixed” expert set in the following. Before targeting large scale data sets we evaluate our approach on a total of seven diverse data sets from the UCI repository (Bache & Lichman, 2013) as detailed in Tab. 3.1.

As those data sets are fairly small we test using the leave-one-out technique. Hence, we use all but one sample for training, and classify the left out sample during testing. For each leave-one-out experiment we train \(T = 100\) trees. Due to the randomness in the classifier we repeat each experiment 100 times. To study the trade off between speed and accuracy for the binomial and multinomial formulation, we tune \(\alpha \in [0, 1]\). Hereby, \(\alpha = 0\) means we always use all 100 base classifiers. For \(\alpha > 0\), we adaptively decide for each sample how many of the available trees to consult. For binary classification
3.3. Experimental Evaluation

problems we further apply the SPRT procedure and tune $\xi = \xi_1 = \xi_2 \in [0.5^{26}, 0.5]$ with the second specifiable parameter $\sigma_1 = 1 - \sigma_2 = 0.55$ being fixed. As $\xi$ sweeps through an interval, the particular choice for $\sigma_1$ and $\sigma_2$ is irrelevant for the results to be presented. Given a specific value of $\alpha$ or $\xi$, we obtain the average number of “experts” $E$ applied, which is a direct measure for the time spent during classification. We also compare our method with the approach of using a fixed number of trees $E_X$ for all samples. The error is measured in deviation from the result obtained with all 100 classifiers. The representative results for the Tic-Tac-Toe and Ecoli data sets are shown in Fig. 3.4 together with the standard deviation. All three methods proposed to be applied for early termination, i.e., SPRT, the binomial and the multinomial formulation, clearly outperform the simple but very common approach of using a fixed number of trees. The difference between the three proposed methods on binary data sets is minor. But most importantly, the standard deviation obtained with those approaches is far less than the one resulting from just decreasing the number of base classifiers. This shows the adaption to the difficulty of the data and makes testing results more reliable.

To provide the full picture, we summarize all results in Table 3.1. By inspection of graphs like the ones shown in Fig. 3.4, we find that value $\alpha_{100}$ for Eq. (3.7) and Eq. (3.8) or Eq. (3.6) as well as $\xi_{100}$ for SPRT such that we don’t worsen the classification result compared to using all 100 trees. Consequently we obtain the average number of “experts” $E$ that were consulted after having classified all the samples. Next, we require consulting a fixed number of trees $E_X$ that is approximately equal to the average number $E$. This will result in an approximately identical classification time. We read off the percentage of additional errors we make compared to using 100 classifiers. We also provide the minimal number of classifiers $E_{min}$ necessary for the method named “Fixed” expert set, such that we are not worse compared to working with 100 trees.

Investigating Tab. 3.1, we observe that the average number of considered classifiers $E$ necessary to achieve a detection rate equivalent to the one obtained with 100 trees is smaller. We realize that we never manage to get the same accuracy by just asking a fixed number of “experts” $E_X \approx E$. The minimum number of “experts” $E_{min}$ which

![Figure 3.4.: Comparison of fixed number of trees and our adaptive multinomial approach on the binary Tic-Tac-Toe and the eight class Ecoli data set plotted with identically scaled axis.](image)
is necessary to achieve the same accuracy as the one we obtain when applying the proposed approach is always larger. Savings depend on the particularities of the data, i.e., our method is only $1.4 \approx \frac{100}{71}$ times faster for the Yeast data but 6.3 times faster on the Iris set. The absolute error rates of our Random Forest settings (choose one among $\lceil \sqrt{F} \rceil$ features maximizing Gini entropy) for the Ionosphere, Glass and Ecoli set are 6.6%, 20.3% and 12.3%, consistent with the 7.1%, 20.6% and 12.8% provided by Breiman (2001).

Above results encourage to look at larger data sets, we obtain when considering typical detection problems in computer vision. Especially as the class distributions for above data sets are fairly uniform. Due to the skewed data, the possibilities to achieve the same accuracy by asking less “experts” should be given for imaging data sets as well.

### 3.3.2. Experiments on Large Scale Data Sets

As the usual computer vision data is too large for leave-one-out techniques we restrict ourselves to dividing the images/volumes into two sets, a training part and a test set. We look at medical image classification as the volumes are usually fairly noisy. Specifically, we decided to evaluate our method on 3D magnetic resonance images from

### Table 3.1.: The UCI data set name together with $\alpha_{100}$ or $\xi_{100}$ achieving the same performance as 100 trees. The average number of “experts” ($\bar{E}$) that were evaluate when applying $\alpha_{100}$ or $\xi_{100}$ and how many additional misclassification errors (Error) made when requiring to consult a fixed number of classifiers ($E_X \approx \bar{E}$). The minimum number of classifiers $E_{\text{min}}$ necessary to achieve the same accuracy as with 100 trees. We compare binomial (b), multinomial (m) and SPRT (s) approach using the Ionosphere and the Tic-Tac-Toe data set. Further results are presented in (Schwing et al., 2011b).

<table>
<thead>
<tr>
<th>Data set</th>
<th>$\alpha_{100}$ or $\xi_{100}$ ($\bar{E}$)</th>
<th>Error ($E_X$)</th>
<th>$E_{\text{min}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.0002 (40.3)</td>
<td>16.5% (41)</td>
<td>100</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.0018 (38.4)</td>
<td>18.0% (39)</td>
<td>100</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.0185 (39.4)</td>
<td>17.5% (40)</td>
<td>100</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.0017 (19.6)</td>
<td>2.7% (20)</td>
<td>60</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.0065 (19.7)</td>
<td>2.7% (20)</td>
<td>60</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.0372 (25.1)</td>
<td>2.5% (26)</td>
<td>60</td>
</tr>
<tr>
<td>Iris</td>
<td>0.0046 (15.4)</td>
<td>10.9% (16)</td>
<td>97</td>
</tr>
<tr>
<td>Wine</td>
<td>0.0200 (15.2)</td>
<td>14.3% (16)</td>
<td>37</td>
</tr>
<tr>
<td>Glass</td>
<td>0.0062 (49.7)</td>
<td>2.4% (50)</td>
<td>87</td>
</tr>
<tr>
<td>Ecoli</td>
<td>0.0028 (35.6)</td>
<td>6.5% (36)</td>
<td>95</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.0019 (71.0)</td>
<td>0.5% (71)</td>
<td>100</td>
</tr>
</tbody>
</table>
3.3. Experimental Evaluation

Figure 3.5.: Three zoomed in examples (size $100 \times 100$) of our wrist bone training database together with the overlaid annotations.

Table 3.2.: The number of “experts” considered for different values of $\alpha$ together with the true positive rate (TPR) and the false positive rate (FPR) for biased (B) maximum likelihood estimate and for the unbiased estimate (UB) given in Eq. (3.5).

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>E</th>
<th>TPR$_B$</th>
<th>FPR$_B$</th>
<th>TPR$_{UB}$</th>
<th>FPR$_{UB}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100.0</td>
<td>0.948</td>
<td>0.045</td>
<td>0.948</td>
<td>0.045</td>
</tr>
<tr>
<td>0.0001</td>
<td>25.1</td>
<td>0.929</td>
<td>0.042</td>
<td>0.952</td>
<td>0.050</td>
</tr>
<tr>
<td>0.0057</td>
<td>15.0</td>
<td>0.930</td>
<td>0.042</td>
<td>0.960</td>
<td>0.056</td>
</tr>
<tr>
<td>0.03</td>
<td>10.2</td>
<td>0.929</td>
<td>0.042</td>
<td>0.954</td>
<td>0.053</td>
</tr>
</tbody>
</table>

We show three sample images from a total of 23 3D training volumes together with annotations in Fig. 3.5. The volumes of left and right wrists are approximately parallel or anti-parallel to one of the two planar coordinate axes. The average size of the volumes is $240 \times 240 \times 30$ voxel and the number of Haar-like features is about $F \approx 100,000$. The five 3D volumes of the test set consist of approximately 1.5 million samples each. Having trained 100 base classifiers, we apply them to test data and count how often proportion $(v_1, v_2)$ was hit during classification. We show the result in Fig. 3.1. Considering the logarithmic scale in Fig. 3.1 and recalling the region for termination, exemplarily shown for a confidence value $\alpha = 0.1$ in Fig. 3.2(b), indicates the potential to improve speed if we adaptively decide how many trees to employ for classification of a pixel/voxel.

We apply a Random Forest classifier together with the proposed methods for different $\alpha$ and $\xi$ on the test set. The error rates are compared in Fig. 3.6(a). The three approaches outperform the common method of reducing the number of trees to a fixed number. This time, even 100 ensemble members cannot achieve the accuracy obtained with the SPRT, the binomial or the multinomial formulation. To measure the decrease in performance of the binomial formulation for a varying object classification threshold, we observe the receiver operator characteristic (ROC). Fig. 3.6(b) to (d) and Table 3.2
Figure 3.6.: The three proposed methods are compared in (a). Comparison of biased and unbiased estimator in (b). Cutout of test set ROC curves using the binomial approach (Eq. (3.5)) for different $\alpha$ resulting in $\bar{E}$ shown in (c) and for roughly equal number of fixed “experts” $E_X \approx \bar{E}$ in (d).

summarize the results. For the other methods, ROCs are provided in (Schwing et al., 2011b).

The result for varying $\alpha$ is shown in Fig. 3.6(c) and the effect introduced by simply decreasing the number of “experts” to a fixed number $E_X$ is illustrated in Fig. 3.6(d). We observe that the true positive rate around a reasonable operating point (5% false positive rate) drops by about 5% if we reduce $E_X$ to one tenth. Applying our proposed approach, we roughly maintain the true positive rate while decreasing the consulted number of “experts” to almost one tenth. Closer investigations (Schwing et al., 2011b) reveal that a fixed number of about 50 trees is necessary to achieve the same accuracy.

Besides the exact numbers for a usual operating point in Fig. 3.6(c), we indicate in Tab. 3.2 and via the markers in Fig. 3.6(b) ($\alpha = 0.0057$) the difference when using the unbiased estimator given in Eq. (3.5). The curves themselves are of course supposed to lie ontop of each other. The rates are provided for a threshold of 0.5, i.e., every sample having probability higher than 0.5 is considered an object sample. If we don’t correct the probability, we move on the ROC curve towards slightly lower false positive rates which is avoided with the unbiased estimate. Of course we can alternatively tune the threshold to achieve the same operating point in ROC space. Applying the estimator given in Eq. (3.5) circumvents tuning.
3.4. Conclusion

The number of “experts” that were asked for each voxel of a test volume is shown color coded in Fig. 3.7. We observe the regions that are “easy” for the classifier, i.e., the interior of the bones and the exterior of the wrist.

3.4. Conclusion

For small scale and large scale data we emphasize that the common procedure of reducing the number of trees results in a drop of performance which is minimized by adaptively deciding how many “experts” to ask before making a decision. The detailed, parallelizable approach is based on a statistical formulation and we show in numerous leave-one-out experiments on publicly available data sets as well as on 3D images not used for training that we achieve faster classification without severely degrading the accuracy.
3. Adaptive Random Forest
4. Inference in Structured Models

We observe that a Random Forest classifier as introduced in the previous chapter does not allow to model the relationship between different output space variables more carefully. Indeed, every leaf node contains a full joint distribution over the entire output space $S$ and every sample is considered independently. It is the structured models introduced subsequently that trade this property with computational complexity.

4.1. Review

To provide more details on structured models let us extend the output space to $N$ discrete variables $s_i, i \in \{1, \ldots, N\}$. Let the domain of a variable $s_i$ be denoted by the set of discrete values referred to via $S_i = \{1, \ldots, |S_i|\}$. The complete output space $S$ is consequently a product space, i.e., $S = \prod_{i=1}^{N} S_i$.

Given an input space object $x$ we assume a scoring function $\theta(s, x)$ often also referred to as a potential function, negative energy or negative Hamiltonian $\theta(s)$ to evaluate the different output space configurations according to $\exp (\beta \theta(s))$ where the scalar inverse-temperature parameter $\beta \geq 0$ controls the smoothness. Finding the distribution of interest $p(s | x)$ is effectively equivalent to finding the normalizing constant $Z(\beta, \theta)$, known as the partition function, such that

$$p(s) = \frac{1}{Z} \exp (\beta \theta(s))$$

is a valid probability distribution that scores all the output space configurations between zero and one while the sum of all scores equals one. We subsequently refer to a valid probability distribution as being a member of the simplex set $\Delta = \{p(s) : \sum_{s \in S} p(s) = 1 \land p(s) \geq 0 \ \forall s \in S\}$. For notational convenience we neglect dependence on the input object $x$ for now. We compute the partition function $Z = \sum_{s \in S} \exp (\beta \theta(s))$ and note that the normalizing constant is equivalently given by the following variational program

$$\frac{1}{\beta} \ln Z = \max_{p \in \Delta} \sum_{s \in S} p(s)\theta(s) + \frac{1}{\beta} H(p), \quad (4.1)$$

where $H(p) = -\sum_{s \in S} p(s) \ln p(s)$ measures the entropy of distribution $p$ in nats. This program provides another intuition for the fact that the temperature $1/\beta$ controls the smoothness. Strictly speaking, we should consider the limits $\beta \to \infty$ and $\beta \to 0$ more carefully, but for the discrete output spaces discussed within this thesis, the following simplification shall suffice. We find the set of highest scoring configurations within
4. Inference in Structured Models

the output space, i.e., the ground states, by setting $\beta = \infty$, while $\beta = 0$ yields the uniform distribution $p(s) = 1/|S| \forall s \in S$. We therefore don’t differentiate between both tasks for now.

One way to derive this variational equivalence is obtained by minimizing the Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951) between our distribution of interest $p$ and the scoring distribution, while noting that the minimal KL divergence is zero, i.e., $\min_{p \in \Delta} D_{KL}(p \mid \frac{1}{Z} \exp(\beta \theta(s)))$. Keep in mind that the partition function $Z$ is not known. Hence, inference in structured models is also known as I-projection.

Moreover, if we employ conjugate duality we obtain another derivation, as illustrated in the following: we compute the conjugate dual of the negative free energy $\frac{1}{\beta} \ln Z$, convex in $\theta(s)$, via

$$\max_\theta \sum_s p(s)\theta(s) - \frac{1}{\beta} \ln \sum_s \exp \beta \theta(s).$$

Taking the derivative of the cost function w.r.t. $\theta(s)$ results in

$$\frac{\partial}{\partial \theta(s)} \left( \sum_s p(s)\theta(s) - \frac{1}{\beta} \ln \sum_s \exp \beta \theta(s) \right) = p(s) - \frac{\exp \beta \theta(s)}{\sum_\hat{s} \exp \beta \theta(\hat{s})} = 0,$$

which yields $p(s) = \exp(\beta \theta(s)) / \sum_s \exp(\beta \theta(s)) \geq 0$, i.e., the non-negativity requirement $p(s) \geq 0 \forall s \in S$. Summing both sides of this relationship over the entire output space $S$ yields

$$\left( \sum_s p(s) \right) \left( \sum_s \exp \beta \theta(s) \right) = \sum_s \exp \beta \theta(s)$$

and subsequently $\sum_s p(s) = 1$ which hence ensures that $p(s)$ is a valid probability distribution when taking into account above non-negativity constraint. Considering the cost function, we obtain the conjugate dual

$$\sum_s p(s)\theta(s) - \frac{1}{\beta} \ln \sum_s \exp \beta \theta(s) = \sum_s p(s) \left( \frac{1}{\beta} \ln \exp \beta \theta(s) - \frac{1}{\beta} \ln \sum_\hat{s} \exp \beta \theta(\hat{s}) \right)$$

$$= \sum_s \frac{1}{\beta} p(s) \ln p(s)$$

$$= -\frac{1}{\beta} H(p)$$

to be the negative entropy. We retrieve the relationship given in Eq. (4.1) when noting the equivalence between a convex function and the conjugate dual of the functions conjugate dual.
4.1. Review

In addition, application of the Jensen (1906) inequality to the Boltzmann factor \( \exp(\beta \theta(s)) \), convex in \( \theta(s) \), via

\[
\frac{1}{\beta} \ln Z = \frac{1}{\beta} \ln \sum_{s \in S} p(s) \exp(\beta \theta(s)) \geq \sum_{s \in S} p(s) \theta(s) + \frac{1}{\beta} H(p)
\]

yields a lower bound, valid for all \( p(s) \in \Delta \).

Computation of the distribution \( p(s \mid x) \), i.e., the partition function \( Z \), given the scores \( \theta(s \mid x) \) via either exhaustive summation or maximization is tractable only if the output space is not too large. For many applications of interest, approximations to compute the partition function are however required.

To our advantage the scoring function for many practical problems is structured. To keep a general and application independent notation we introduce index sets \( r \subseteq \{1, \ldots, N\} \) which refer to a subset of the \( N \) variables that span the output space. The space spanned by such a subset of variables is indicated by \( S_r \) and a possible configuration of this subspace is denoted by the restriction \( s_r = (s_i)_{i \in r} \in S_r \). The index sets \( r \) generalize the way we refer to individual variables \( i \in \{1, \ldots, N\} \) and elements of their domain \( s_i \in S_i \) to arbitrary subsets of variables. Introducing some terminology, we refer to the index set \( r \) as a region and subsume all regions that a potential function \( \theta(s) \) depends on in the set \( R \). All in all we therefore let the potential function decompose as follows:

\[
\theta(s) = \sum_{r \in R} \theta_r(s_r).
\]

To give a concrete example let the potential function \( \theta(s_1, s_2, s_3) = \theta_1(s_1) + \theta_{\{1,2\}}(s_1, s_2) + \theta_{\{2,3\}}(s_2, s_3) \) when using the regions subsumed within the set \( R = \{\{1\}, \{1, 2\}, \{2, 3\}\} \).

Such a decomposition is useful when taking into account the resources required to store the potential function. Considering a general potential function, \( \theta(s) \) amounts to storing a value for every possible configuration \( s \) within the output space \( S \). The output space size \( |S| = \prod_{i=1}^{N} |S_i| \) therefore grows rapidly. In general it grows much faster than the values necessary for describing the decomposition, i.e., \( \sum_{r \in R} |S_r| = \sum_{r \in R} \prod_{i \in r} |S_i| \). More formally we observe that \( \sum_{r \in R} \prod_{i \in r} |S_i| \ll \prod_{i=1}^{N} |S_i| \).

Employing the decomposition, we note that

\[
\sum_{s \in S} p(s) \theta(s) = \sum_{r \in R, s_r \in S_r} b_r(s_r) \theta_r(s_r), \tag{4.2}
\]

with \( b_r(s_r) = \sum_{s \setminus s_r} p(s) \) indicating the marginal distribution of \( p(s) \) w.r.t. the domain \( S_r \), i.e., we sum over all elements except the ones within the considered domain.

Towards a tractable program for the task given in Eq. (4.1) we also need to decompose the entropy over the joint distribution \( H(p) \) via entropies over smaller marginals.
4. Inference in Structured Models

The previously employed marginals $b_r(s_r)$ lend themselves to such an approximation via

$$H(p) \approx \sum_r c_r H(b_r),$$

where $c_r$ is referred to as a counting number, ideally chosen or computed such that the approximation induces a small error.

Importantly, we required the marginals $b_r(s_r)$ to arise from a single distribution $p(s)$ via $b_r(s_r) = \sum_{s \setminus s_r} p(s)$. Since this constraint involves summing over almost the entire domain of the output space it is an intractable constraint in general and there are a variety of approximations. Let the set of all parents of region $r$, i.e., the set of ancestors, be defined by $A(r) = \{p \in \mathcal{R} : p \supset r\}$ to contain all those index sets $p \in \mathcal{R}$ that are strict supersets of $r$, i.e., the region $p$ refers to additional variable indices besides the ones indicated within the index set $r$. More generally we let the set of considered parents $P(r) \subseteq A(r)$ be a subset of all the ancestors. Conversely to the set of parents we refer to the set of children of region $r$ via $C(r) = \{c : P(c) = r\}$.

This parent-child relationship is best visualized in Hasse diagrams, a generalization of common factor graphs. In factor graphs we differentiate between two types of regions, i.e., variables with $|r| = 1$ and cliques or factors with $|r| > 1$. As visualized in Fig. 4.1(a), a factor graph is a bipartite graph with connecting edges only between variables visualized by blue circles and factors depicted by green rectangles. The corresponding Hasse diagram is illustrated in Fig. 4.1(b) with undirected edges indicating the parent-child relationship, i.e., the sets $P(r)$ and $C(r)$. A more general Hasse diagram is depicted in Fig. 4.1(c) and due to the relationships between regions of size larger than one, no factor graph is able to capture the parent-child relationships visualized via edges in the Hasse diagram. Also note that a region $r$ is not necessarily connected to all the elements within $A(r)$ but only the subset referred to by the set of parents $P(r)$.

Those sparse relationships increase tractability of the optimization problem since the parent-child relationship is involved in approximating the generally intractable marginalization constraint $b_r(s_r) = \sum_{s \setminus s_r} p(s)$. More formally we require that the

![Figure 4.1.](image-url)
marginalization of all parents of region \( r \) agrees with the belief \( b_r(s_r) \) of region \( r \), i.e.,
\[
\forall r \in \mathcal{R}, p \in \mathcal{P}(r), s_r \in \mathcal{S}_r \sum_{s_p \setminus s_r} b_p(s_p) = b_r(s_r). \tag{4.4}
\]

For the derivation of an algorithm it is important to note that this constraint is equivalently specified via
\[
\sum_{s_r \setminus s_c} b_r(s_r) = b_c(s_c) \forall r \in \mathcal{R}, c \in \mathcal{C}(r), s_c.
\]

Although the resulting marginals \( b_r(s_r) \) are consistent amongst each other, those constraints don’t generally require the marginals to arise from a single joint probability distribution \( p(s) \). Choosing the sets of parents and children hence represents the second approximation besides the entropy considerations.

Instead of the intractable computation of the partition function via the inference program
\[
\max_{p \in \Delta} \sum_{s \in \mathcal{S}} p(s) \theta(s) + \frac{1}{\beta} H(p),
\]
we obtain the following approximation when taking into account Eq. (4.2), Eq. (4.3) and Eq. (4.4):
\[
\max_{b_r} \sum_{r,s} b_r(s_r) \theta_r(s_r) + \sum_r \frac{c_r}{\beta} H(b_r) \tag{4.5}
\]
\[
\text{s.t.} \quad \left\{ \begin{array}{l}
\forall r, p \in \mathcal{P}(r), s_r \quad b_r \in \Delta \sum_{s_p \setminus s_r} b_p(s_p) = b_r(s_r).
\end{array} \right.
\]

Note that we employ the set of valid probability distributions \( \Delta \) without indicating the appropriate dimension which should nonetheless be clear from the context.

Importantly, convergence guarantees depend on the choice of counting numbers. Letting \( c_r \geq 0 \) ensures concavity of the problem and therefore convergence. This is known as convex belief propagation (Weiss et al., 2007; Hazan & Shashua, 2008; Meltzer et al., 2009; Hazan & Shashua, 2010). Standard loopy belief propagation (Pearl, 1988) or generalized loopy belief propagation (Yedidia et al., 2005) employs counting numbers \( c_r = 1 - \sum_{a \in A(r)} c_a \) where \( A(r) \) is the set of ancestors of region \( r \). Due to neither concavity nor convexity of the primal problem and due to their local block-coordinate descent like updates, convergence is not guaranteed for general Hasse diagrams and a general choice of counting numbers.

To leverage the structure of the program encoded in the marginalization constraints which are visualized via edges in the Hasse diagram, it is beneficial to optimize the dual problem. The cost function of the dual is obtained by introducing Lagrange multipliers \( \lambda_{r \rightarrow p}(s_r) \) for the marginalization constraints of the beliefs. Conjugate duality between entropy and the logarithm of a partition function, i.e., the soft-max function, yields the following unconstrained dual objective:
\[
\sum_r \frac{c_r}{\beta} \ln \sum_{s_r} \exp \left( \frac{\beta}{c_r} \left( \theta_r(s_r) - \sum_{p \in \mathcal{P}(r)} \lambda_{r \rightarrow p}(s_r) + \sum_{c \in \mathcal{C}(r)} \lambda_{c \rightarrow r}(s_c) \right) \right). \tag{4.6}
\]

In order to optimize this objective, we follow Hazan & Shashua (2010); Hazan et al. (2012) to derive a block-coordinate descent algorithm, where a region \( r \) is chosen.
4. Inference in Structured Models

**Algorithm: Message Passing Inference**

Repeat until convergence

Iterate over $r$:

$$\forall p \in P(r), s_r$$

$$\mu_{p \rightarrow r}(s_r) = \frac{c_p}{\beta} \ln \sum_{s_p \in S} \exp \left( \frac{\theta_p(s_p) - \sum_{p' \in P(p)} \lambda_{p \rightarrow p'}(s_p) + \sum_{r' \in C(p) \setminus r} \lambda_{r' \rightarrow p}(s_{r'})}{c_p / \beta} \right)$$

$$\forall p \in P(r), s_r$$

$$\lambda_{r \rightarrow p}(s_r) \propto \frac{c_p}{c_r + \sum_{p \in P(r)} c_p} \left( \theta_{r}(s_r) + \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c) + \sum_{p \in P(r)} \mu_{p \rightarrow r}(s_r) \right) - \mu_{p \rightarrow r}(s_r)$$

Figure 4.2.: A block-coordinate descent algorithm for the inference task.

iteratively, and the Lagrange multipliers $\lambda_{r \rightarrow p}(s_r)$ \(\forall p \in P(r), s_r\) are updated via a minimization computable in closed form. The required update rules for the resulting message passing algorithm are summarized in Fig. 4.2.

Given the Lagrange multipliers $\lambda$ we recover the beliefs

$$b_r(s_r) \propto \exp \left( \frac{\theta_r(s_r) - \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_r) + \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c)}{c_r / \beta} \right)$$

if $c_r / \beta \neq 0$. For $c_r / \beta = 0$ the beliefs nonzero domain is determined by the maximizing elements $S_r^* = \arg \max_{s_r} \left( \theta_r(s_r) - \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_r) + \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c) \right) \subseteq \prod_{i \in r} S_i$ (cf. Danskin’s theorem in, e.g., Bertsekas (1999)).

In this chapter we address two limitations of existing inference algorithms. Firstly, the computational complexity of message passing algorithms is known to scale linearly with the number of regions. Given multiple computers we are interested in distributing the computation and the memory requirements while maintaining convergence guarantees of convex message passing algorithms. The approach follows a dual decomposition paradigm and is detailed in Sec. 4.2. Secondly, we observe that computation of the maximum a-posteriori (MAP) configuration requires to solve a non-smooth dual program as $\beta \rightarrow \infty$. Since message passing algorithms perform block-coordinate descent like updates, convergence to the global optimum is not guaranteed if the cost function is not strictly convex. In addition, the optimal configuration cannot necessarily be decoded. The approach described in Sec. 4.3 employs steepest $\epsilon$-descent directions to yield a globally convergent procedure.
4.2. Distributed Message Passing Algorithm

Subsequently we first provide the intuition to design a distributed inference algorithm (Sec. 4.2.1) and some experimental results (Sec. 4.2.2) before considering mathematical details (Sec. 4.2.3). The reader may skip or catch a glimpse of the latter section without missing information required in subsequent chapters.

4.2.1. Intuition

The aforementioned message passing inference algorithms do not scale well when utilized for large problems, as they require to store both the graphical model and the messages in memory. Furthermore, solving the approximate inference problem is computationally demanding, as it scales with the number of dimensions involved in the largest region, i.e., they are already inherently quadratic for graphical models involving pairwise connections. This is a tremendous limitation for a wide variety of applications, where one is faced with very large graphical models with millions of regions having thousands of states.

If we consider shared memory environments, e.g., current standard multi-core computers, one strategy is application of a graph coloring of the Hasse diagram by making
4. Inference in Structured Models

Algorithm: Distributed Message Passing Inference

Repeat until convergence

1. For every $\kappa$ in parallel: iterate over $r$ and compute

$$
\forall p \in P(r), s_r \quad \mu_{p \rightarrow r}(s_r) = \frac{\hat{c}_p}{\beta} \ln \sum_{s_p \setminus s_r} \exp \left( \hat{\theta}_p(s_p) - \sum_{p' \in P(p)} \lambda_{p' \rightarrow p}(s_p) + \sum_{r' \in C(p) \cap \kappa \setminus r} \lambda_{r' \rightarrow p}(s_{r'}) + \nu_{\kappa \rightarrow p}(s_p) \right) / (\hat{c}_p / \beta) \quad (4.7)
$$

$$
\forall p \in P(r), s_r \quad \lambda_{r \rightarrow p}(s_r) \propto \frac{\hat{c}_p}{\beta_r} + \sum_{p \in P(r)} \frac{\hat{c}_p}{\beta} \left( \hat{\theta}_p(s_r) + \sum_{c \in C(r) \cap \kappa} \lambda_{c \rightarrow r}(s_c) + \nu_{\kappa \rightarrow r}(s_r) + \sum_{p \in P(r)} \mu_{p \rightarrow r}(s_r) \right) - \mu_{p \rightarrow r}(s_r) \quad (4.8)
$$

2. Iterate over $r \in G_P$

$$
\forall \kappa \in M(r), s_r \quad \nu_{\kappa \rightarrow r}(s_r) = \frac{1}{|M(r)|} \left( \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c) - \sum_{\kappa \in M(r), p \in P(r)} \lambda_{r \rightarrow p}(s_r) \right) + \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_r) - \sum_{c \in C(r) \cap \kappa} \lambda_{c \rightarrow r}(s_c) \quad (4.9)
$$

Figure 4.5.: A block-coordinate descent algorithm for the distributed inference task.

sure that regions within the Markov blanket are assigned different colors. Messages of all nodes having the same color are then computable in parallel. Considering large graphs, we hence minimize the number of idle cores while directly optimizing the considered problem. For densely coupled graphical models this strategy results in a large number of colors with few regions per color.

High performance compute clusters are a cheap alternative for large-scale computation when memory and computational complexity are the main limiting factors. In this setting, we leverage the computational resources by distributing the problem onto all the available machines. Since we deal with distributed memory that is connected
via local area networks, relatively slow compared to direct memory accesses, we cannot
neglect the latency for transmitting messages between different computers. Therefore,
we want to minimize the communication between computers while maintaining con-
vergence guarantees particularly for strictly convex problems (e.g., strictly positive
counting numbers).

Towards this goal, we partition the output space $S$. Note that this induces a
partitioning of the region graph. We highlight that the likelihood of a region being
assigned to multiple computers increases with the degree of the region, i.e., the number
of involved variables. It is therefore crucial to choose a suitable problem dependent
partitioning. Since most of the considered tasks are not decoupled by nature, there
are always regions assigned to multiple resources. In order to guarantee convergence
to the optimum of the approximated problem, we are required to enforce that the
regions divided onto multiple machines are consistent upon convergence.

Our intuition is illustrated in Fig. 4.3 for a graphical model with 8 random variables
and pairwise regions depicted by edges. The output space is partitioned onto two com-
puters $\kappa_1$ and $\kappa_2$, and the factors/regions divided onto two machines are highlighted
via two parallel lines close to each other.

More formally let $M(r)$ be the set of machines that region $r$ is assigned to, and let
$R(\kappa)$ be the set of regions that are partitioned on machine $\kappa$. A region is assigned
to a machine if at least one of its variables is assigned to computer $\kappa$. Let the
potentials and counting numbers be distributed equally across machines, i.e., $\tilde{\theta}_r(s_r) = \theta_r(s_r)/|M(r)|$ and $\tilde{c}_r = c_r/|M(r)|$. In addition we introduce a factor-graph $G_{\mathcal{P}}$ with
nodes representing the computers and a factor representing the regions shared between
at least two machines. The factor-graph for the example in Fig. 4.3 is illustrated in
Fig. 4.4.

The inference task presented in Eq. (4.5) is then equivalently given by the following
program:

$$
\max_{b^\kappa_r \in \Delta} \sum_{\kappa, r \in R(\kappa), s_r} b^\kappa_r(s_r) \hat{\theta}_r(s_r) + \sum_{\kappa, r \in R(\kappa)} \frac{\tilde{c}_r}{\beta} H(b^\kappa_r) \\
\text{s.t.} \forall \kappa, r \in R(\kappa), s_r, p \in P(r) \sum_{s_p \not\subseteq s_r} b^\kappa_p(s_p) = b^\kappa_r(s_r) \\
\forall \kappa, r \in R(\kappa), s_r \\
\quad b^\kappa_r(s_r) = b_r(s_r).
$$

Importantly we only store a belief $b^\kappa_r$ on machine $\kappa$ if the region $r$ is assigned to
this computer, i.e., if $r \in R(\kappa)$. The latter constraint is introduced to enforce con-
sistency between the regions assigned to multiple computers. Note that this is the
only constraint that couples the individual sub-problems partitioned onto different
machines.

To leverage the structure within the constraint set we change to the dual domain by
introducing Lagrange multipliers $\lambda_{r \rightarrow p}(s_r)$ and $\nu_{\kappa \rightarrow r}(s_r)$ for the marginalization con-
straints and consistency requirements respectively. We then derive a block-coordinate
descent algorithm which updates the messages $\lambda$ and $\nu$. We refer the reader to
Sec. 4.2.3 for the detailed derivation and a formal statement regarding the convergence
properties. The resulting algorithm is provided in Fig. 4.5.
4. Inference in Structured Models

![Graphs showing time vs iterations for different beta values](image)

Figure 4.6.: Comparing the time of non-distributed inference (dotted magenta) with the distributed inference algorithm exchanging information at different frequencies when considering a given number of iterations.

<table>
<thead>
<tr>
<th></th>
<th>50</th>
<th>25</th>
<th>10</th>
<th>5</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta = \infty$</td>
<td>16.60</td>
<td>15.74</td>
<td>14.95</td>
<td>13.51</td>
<td>10.48</td>
<td>7.67</td>
</tr>
<tr>
<td>$\beta = 100$</td>
<td>15.57</td>
<td>15.27</td>
<td>14.34</td>
<td>13.19</td>
<td>10.87</td>
<td>8.32</td>
</tr>
<tr>
<td>$\beta = 10$</td>
<td>15.38</td>
<td>15.10</td>
<td>14.23</td>
<td>13.24</td>
<td>10.76</td>
<td>8.24</td>
</tr>
</tbody>
</table>

Table 4.1.: Inference speedup after 200 iterations for different $\beta$ (rows) and information exchange rates (columns).

The algorithm of our distributed program is divided into two parts: (1) a standard message passing, local on every machine and hence easily parallelizable using graph-coloring, and (2) an exchange of information between different computers enforcing the consistency constraints. This exchange of information corresponds to message passing on the factor-graph $G_P$.

Our distributed algorithm maintains the convergence guarantees irrespective of how frequently we exchange information. The frequency of the exchange only has an impact on the speed of convergence. We expect faster convergence in terms of number of iterations if we exchange information at every single iteration. This will however not lead to the best convergence speed in terms of time, due to the communication overhead between computers as transmission latency will significantly slow down the algorithm.

The best trade off depends both on the network latency and the graph structure, as the later determines the number of messages and their size required to be exchanged.

4.2.2. Experimental Results

We illustrate the advantages of the proposed approach in the problem of stereo estimation. Towards this goal, we employ the Tsukuba image from the Middlebury data set (Scharstein & Szeliski, 2002). The respective Hasse diagram is composed of $\approx 315000$ regions. Subsequently we investigate the time required for the iterations
4.2. Distributed Message Passing Algorithm

before providing results regarding the frequency of transmission of messages between computers. Afterwards we also illustrate the primal-dual gap and visually investigate the results of distributing a task without exchanging messages between computers. We conclude the experiments for distributed inference with a large scale example.

**Time per iteration:** We investigate the time required for a certain number of iterations. Fig. 4.6(a) – (c) show timings for different values of $\beta$. The magenta dotted line illustrated the baseline algorithm that parallelizes the inference task only onto the available eight cores of a single computer, while the solid lines illustrate the time for a certain number of iterations for the distributed algorithm when exchanging information only every 1, 2, 5, 10, 25 or 50 iterations. We observe the distributed algorithm to be significantly faster than the single machine shared memory counterpart. The speedups are given in Tab. 4.1. Due to computational benefits of finding the maximum within a vector rather than computing the soft-max, i.e., the logarithm of the partition function of a region, we also note that inference with $\beta = \infty$ is slightly faster.

**Frequency of transmission:** We next divide the graph into 16 equally sized partitions and investigate how often to transmit information between the different computers. Fig. 4.7(a) – (c) illustrates the course of the dual energy w.r.t. time. Depending
4. Inference in Structured Models

Figure 4.9.: (a) Disparity map when solving nine independent sub-problems. Frame boundaries are depicted. (b) Disparity map when exchanging messages every 10 iterations. (c)-(f) zoomed view of the highlighted parts in (a) (bottom) and (b) (top).

on $\beta$ and for the investigated data it is beneficial to transfer messages between different computers only every two or only every five iterations. This obviously depends largely on the problem at hand and on the connectivity between the computers being in our case a 4-connected grid graph with a standard local area network connection.

**Primal-dual gap:** The primal and dual scores for different $\beta \in \{10, 100, \infty\}$ are illustrated in Fig. 4.8(a) – (c). Our distributed block-coordinate descent algorithm guarantees the monotonicity of the dual energy, which is generally not the case for the primal energy. We emphasize that due to only 200 iterations the primal-dual gap is not yet necessarily 0, even for $\beta \neq \infty$.

**Simple distributed baseline:** We refer the reader to Fig. 4.9 for visual results comparing the exchange of information with an approach that distributes the problem without exchanging information. For this experiment we divided a disparity computation task for the image of a tree onto nine computers. We observe clear artifacts when not exchanging messages between the sub-problems.

**Large scale graphs:** To illustrate the large scale behavior we perform disparity map
4.2. Distributed Message Passing Algorithm

Figure 4.10.: 283 label disparity map computed from a 12 MPixel stereo pair.

estimation with 283 labels on the 12 MPixel image illustrated in Fig. 4.10. The Hasse diagram consists of more than 36,000,000 regions with 12 million regions having 283 disparity levels while 24 million regions have more than 80,000 states.

4.2.3. Detailed Derivation

To derive the distributed inference algorithm presented in Fig. 4.5 we repeat the primal problem motivated in Sec. 4.2.1:

\[
\max_{b^\kappa_r \in \Delta} \sum_{\kappa, r, s_r} b^\kappa_r(s_r) \hat{\theta}_r(s_r) + \sum_{\kappa, r} \frac{\hat{c}_r}{\beta} H(b^\kappa_r) \tag{4.10}
\]

\[\text{s.t.} \quad \forall \kappa, r \in R(\kappa), s_r, p \in P(r) \quad \sum_{s_p \in s_r \setminus s_r} b^\kappa_r(s_p) = b^\kappa_r(s_r) \]

\[\forall \kappa, r \in R(\kappa), s_r \quad b^\kappa_r(s_r) = b_r(s_r). \]

Since we aim at leveraging the structure of the constraint set, i.e., since we want to send messages along corresponding graphs, we optimize the dual problem. To derive the dual problem we maximize the Lagrangian by making use of the Fenchel conjugate of the entropy. This results in the dual cost function given within the following claim:

**Claim 2** Set \( \nu_{\kappa \rightarrow r}(s_r) = 0 \) for every \( \kappa, r \notin G_P, s_r \). Then

\[
\sum_{\kappa, r} \frac{\hat{c}_r}{\beta} \ln \sum_{s_r} \exp \frac{\hat{\theta}_r(s_r) - \sum_{p \in P(r)} \lambda_{\kappa \rightarrow p}(s_r) + \sum_{c \in C(r) \cap \kappa} \lambda_{c \rightarrow r}(s_c) + \nu_{\kappa \rightarrow r}(s_r)}{\hat{c}_r / \beta} \tag{4.11}
\]

is the objective of the dual program with constraints \( \sum_{\kappa \in M(r)} \nu_{\kappa \rightarrow r}(s_r) = 0 \ \forall r, s_r \).

**Proof:** Using the Lagrangian of the program given in Eq. (4.10) and the fact that the soft-max function is the conjugate dual of the entropy yields the claim. ■

Next we derive a block-coordinate descent algorithm to optimize the dual program. To this end we choose a region \( r \) and optimize for \( \lambda_{\kappa \rightarrow p}(s_r) \ \forall p \in P(r), s_r \) while keeping all remaining variables fixed. In a second step we optimize for \( \nu_{\kappa \rightarrow r}(s_r) \ \forall \kappa \in M(r) \).
Claim 3 For every $\kappa$, set $\mu_{p\rightarrow r}(s_r)$ as defined by Eq. (4.7) in Fig. 4.5, then the block-coordinate descent update on $\lambda_{r\rightarrow p}(s_r)$ takes the form given in Eq. (4.8). The block-coordinate descent on $\nu_{\kappa\rightarrow r}(s_r)$ takes the form given in Eq. (4.9). Those block coordinate descent steps are guaranteed to converge for $\beta, c_r \geq 0$ and they are guaranteed to reach the optimum of both the primal and the dual for $c_r > 0, \beta \neq \infty$.

Proof: The proof follows (Schwing et al., 2011a).

This concludes the derivation for the distributed inference task. The entire algorithm is given in Fig. 4.5. We emphasize that convergence guarantees do not require a single iteration within the first step. As such, we can perform a few block-coordinate descent steps on the local problem before exchanging information between computers as given in the second step. We refer the reader to the results presented in the experimental Sec. 4.2.2 to observe the monotonic convergence of the derived algorithm.

4.2.4. Summary

To recap, we discussed a method that allows to partition the inference task of a given structured model onto multiple computers. Importantly, we manage to distribute both computational and memory requirements and hence obtain speed-ups comparable to the number of compute units.

4.3. Globally Convergent LP Relaxation Solver

While the aforementioned block-coordinate descent approach is concerned with computational complexity, we now turn our attention to a globally convergent algorithm for $\beta = \infty$. Setting $\beta = \infty$ is interesting because it corresponds to the maximum a-posteriori (MAP) problem. It is however also challenging since the primal problem is no longer strictly convex which results in a non-smooth dual (Rockafellar, 1970). Within this setting, block-coordinate descent approaches are guaranteed to converge since they are lower bounded by the primal problem, but they might get stuck in “corners.” For a sketch of this issue, consider Fig. 4.11 where we visualize the level curves of a 2D function. Assuming the current iterate to be the corner highlighted by a circle, neither proceeding in $\lambda_1$ direction nor walking along $\lambda_2$ direction is capable of finding a better iterate.

To overcome this sub-optimality problem, different solutions have been proposed, e.g., smoothing (Johnson, 2008; Jojic et al., 2010; Hazan & Shashua, 2010; Savchynskyy et al., 2012), proximal updates (Ravikumar et al., 2010) and augmented Lagrangian methods (Martins et al., 2011; Meshi & Globerson, 2011). Due to the modifications of the cost function, convergence speed is however reduced.

Globally convergent methods that directly consider the original non-smooth dual cost function are based on subgradients (Komodakis et al., 2010) and bundle methods (Lemaréchal, 1974; Kappes et al., 2012). However, applying subgradients either directly (Komodakis et al., 2010) or indirectly, to improve a polyhedral approximation (Kappes et al., 2012), is known to converge slowly since arbitrary gradient directions are employed.
We consider the subgradients in Sec. 4.3.1, and provide an efficient algorithm for detecting corners, as well as for decoding a primal optimal solution from a dual optimal solution. This is an intermediate step which facilitates the margin analysis of the Fenchel-Young duality theorem in Sec. 4.3.2. It provides a way to get out of corners, and to reach the optimal dual value.

To provide some intuition, let us consider a convex piecewise linear function $q(\lambda)$ and its corresponding non-smooth dual $q^*(d)$ as visualized in Fig. 4.12. The set of subdifferentials $\partial q(\lambda)$ of the function $q(\lambda)$ is depicted in Fig. 4.13(a). It consists of a single element corresponding to the gradient of $q(\lambda)$ unless the left and right limit of the gradient do not agree at $\lambda$. For the latter case the subdifferential consists of all hyperplane gradients supporting the function $q(\lambda)$ from below.

To improve convergence, the steepest descent direction among the elements in the subdifferential may be considered. Depending on the step size rule, methods that employ the steepest subdifferential descent direction may however get stuck far from the optimum (Bertsekas, 1999). To obtain guaranteed convergence we enlarge the set of subdifferentials such that it does not only contain the sub-gradients at the current iterate but also sub-gradients of neighboring points. Continuing our illustrative example we visualize the lower and upper bound of those enlarged sets $\partial_\epsilon q(\lambda)$ with green and blue colored lines in Fig. 4.13(b) and Fig. 4.13(c) for two different neighborhood regions. We emphasize that $\partial q(\lambda) = \partial_0 q(\lambda)$. We denote those enlarged sets as $\epsilon$-subdifferentials.

As detailed in Sec. 4.3.2, finding the steepest descent direction within those $\epsilon$-subdifferentials amounts to solving a quadratic program subject to linear constraints. In Sec. 4.3.3 we subsequently design an efficient algorithm based on a conditional gradient scheme to obtain a decoupled set of linear programs. We evaluate its performance in Sec. 4.3.4.
4. Inference in Structured Models

Figure 4.12.: The 1-dimensional function \( q(\lambda) \) in (a) and its dual \( q^*(d) \) in (b) constructed via y-intercepts of lines of a given slope \( d \) supporting \( q(\lambda) \) from below as visualized via red lines in (a).

Figure 4.13.: Subdifferentials

4.3.1. Subdifferentials

Subgradients are generalizations of gradients for non-smooth convex functions. Obtained from Eq. (4.5) by letting \( \beta \to \infty \), we state the linear programming (LP) relaxation as the primal problem of interest, i.e.,

\[
\max_{b} \sum_{r \in R, s_r} b_r(s_r) \theta_r(s_r) \tag{4.12}
\]

\[
s.t. \quad \forall r, s_r, p \in P(r) \quad b_r \in \Delta \sum_{s_p \not\in s_r} b_p(s_p) = b_r(s_r).
\]

To leverage the graphical model structure encoded within the constraints we consider its non-smooth re-parameterization dual, either derived directly from Eq. (4.12) or
obtained as the result of the limit β → ∞ from Eq. (4.6), which explicitly reads as

$$q(\lambda) = \sum_r \max_{s_r} \left\{ \theta_r(s_r) - \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_r) + \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c) \right\}. \quad (4.13)$$

A vector $d$ is called a subgradient of $q(\lambda)$ if it supports the epigraph of $q(\lambda)$ at $\lambda$, i.e.,

$$\forall \lambda \quad q(\lambda) - d^T \lambda \geq q(\lambda) - d^T \lambda. \quad (4.14)$$

The supporting hyperplane at $(\lambda, q(\lambda))$ with slope $d$ takes the form $d^T \lambda - q^*(d)$, when defining the conjugate dual as $q^*(d) = \max_{\lambda} \{d^T \lambda - q(\lambda)\}$. From the definition of $q^*(d)$ one can derive the Fenchel-Young duality theorem: $q(\lambda) + q^*(d) \geq d^T \lambda$, where equality holds if and only if $d$ is a supporting hyperplane at $(\lambda, q(\lambda))$. The set of all subgradients is called the subdifferential, denoted by $\partial q(\lambda)$, which can be characterized using the Fenchel-Young theorem as $\partial q(\lambda) = \{d : q(\lambda) + q^*(d) = \lambda^T d\}$. The subdifferential provides a way to reason about the optimal solutions of $q(\lambda)$.

Using Eq. (4.14) we can verify that $\partial q(\lambda)$ is dual optimal if and only if $0 \in \partial \bar{q}(\lambda)$. In the following claim we characterize the subdifferential of the dual function $\bar{q}(\lambda)$ using the Fenchel-Young duality theorem:

**Claim 4** Consider the dual function $q(\lambda)$ in Eq. (4.13). Let the set of maximizing states be denoted by $S^*_r = \text{argmax}_{s_r} \{\theta_r(s_r) - \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_r) + \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c)\}$. Then $d \in \partial q(\lambda)$, if and only if $d_{r \rightarrow p}(s_r) = \sum_{s_p \backslash s_r} b_p(s_p) - b_r(s_r)$ for probability distributions $b_r(s_r)$ whose nonzero entries belong to $S^*_r$ respectively.

**Proof:** Using the Fenchel-Young characterization of Eq. (4.14) for the max-function we obtain the set of maximizing elements $S^*_r$. Summing Eq. (4.14) over all regions $r$ while noticing the change of sign for elements in $\lambda$, we obtain the marginalization disagreements $d_{r \rightarrow p}(s_r)$.

The convex max-product algorithm performs block coordinate descent updates. Thus it iterates over regions $r$ and computes optimal solutions $\lambda_{r \rightarrow p}(s_r)$ for every $s_r, p \in P(r)$ analytically, while keeping all other variables fixed. The claim above implies that the convex max-product iterates over $r$ and generates beliefs $b_r(s_r)$ for every $s_r, p \in P(r)$ that agree on their marginal probabilities. This interpretation provides an insight into the non-optimal stationary points of the convex max-product, i.e., points for which it is not able to generate consistent beliefs $b_r(s_r)$ when it iterates over regions $r$. The representation of the subdifferential as the amount of disagreement between the marginalization constraints provides a simple procedure to verify dual optimality, as well as to construct primal optimal solutions. This is summarized in the corollary below.

**Corollary 1** Given a point $\lambda$, and sets $S^*_r$ as defined in Claim 4, let $s^*_r$ be elements in $S^*_r$ respectively. Consider the quadratic program

$$\min \sum_{r, s^*_r, p \in P(r)} \left( \sum_{s^*_p \backslash s^*_r} b_p(s^*_p) - b_r(s^*_r) \right)^2 \quad \text{s.t.} \quad \begin{cases} \forall r, s^*_r : b_r(s^*_r) \geq 0 \\ \forall r : \sum_{s^*_r} b_r(s^*_r) = 1. \end{cases}$$
4. Inference in Structured Models

λ is a dual optimal solution if and only if the value of the above program equals zero. Moreover, if λ is a dual optimal solution, then the optimal beliefs \( b_r^*(s_r) \) are also the optimal solution of the primal program in Eq. (4.12). However, if λ is not dual optimal, then the vector \( d_{r \rightarrow p}^* (s_r) = \sum_{s_p \backslash s_r} b_p^*(s_p^*) - b_r^*(s_r^*) \) points towards the steepest descent direction of the dual function, i.e.,

\[
d^* = \arg \min_{\|d\| \leq 1} \lim_{\eta \downarrow 0} \eta q(\lambda + \eta d) - q(\lambda).
\]

**Proof:** The steepest descent direction \( d \) of \( q \) is given by minimizing the directional derivative \( q'(d) \),

\[
\min_{\|d\| \leq 1} q'(d)(\lambda) = \max_{y \in \partial q} \min_{\|y\| \leq 1} d^T y = \max_{y \in \partial q} -\|y\|_2^2,
\]

which yields the above program (cf. Bertsekas et al. (2003), Chapter 4). If the zero vector is part of the subdifferential, we are dual optimal. Primal optimality follows from Claim 4.

One can monotonically decrease the dual objective by minimizing it along the steepest descent direction. Unfortunately, following the steepest descent direction does not guarantee convergence to the global minimum of the dual function (Wolfe, 1975). Performing steepest descent might keep minimizing the dual objective with smaller and smaller increments, thus converging to a suboptimal solution. The main drawback of steepest descent as well as block coordinate descent when applied to the dual objective in Eq. (4.13) is that both procedures only consider the support of \( S_r^* \) defined in Claim 4. In the following we show that by considering the \( \epsilon \)-margin of these supports we can guarantee that at every iteration we decrease the dual value by at least \( \epsilon \). This procedure results in an efficient algorithm that reaches both dual and primal optimal solutions.

### 4.3.2. \( \epsilon \)-Subgradients and Steepest \( \epsilon \)-Descent

To monotonically decrease the dual value while converging to the optimum, we suggest to explore the \( \epsilon \)-neighborhood of the dual objective in Eq. (4.13) around the current iterate \( \lambda \). For this purpose, we explore its family of \( \epsilon \)-subgradients. Given our convex dual function \( q(\lambda) \) and a positive scalar \( \epsilon \), we say that a vector \( d \) is an \( \epsilon \)-subgradient at \( \lambda \) if it supports the epigraph of \( q(\lambda) \) with an \( \epsilon \)-margin:

\[
\forall \hat{\lambda} \quad q(\hat{\lambda}) - d^T \hat{\lambda} \geq q(\lambda) - d^T \lambda - \epsilon.
\]  

The subgradients of a convex function are also \( \epsilon \)-subgradients. The family of \( \epsilon \)-subgradients is called the \( \epsilon \)-subdifferential and is denoted by \( \partial_\epsilon q(\lambda) \). Using the conjugate dual \( q^*(d) \), we characterize the \( \epsilon \)-subdifferential by employing the \( \epsilon \)-margin Fenchel-Young duality theorem:

\[
(\epsilon \text{-margin Fenchel-Young duality}) \quad \partial_\epsilon q(\lambda) = \left\{ d : 0 \leq q(\lambda) + q^*(d) - d^T \lambda \leq \epsilon \right\}.
\]  

48
The $\epsilon$-subdifferential augments the subdifferential of $q(\lambda)$ with additional directions $d$ which are controlled by the $\epsilon$-neighborhood of the function. Whenever one finds a steepest descent direction within $\partial_\epsilon q(\lambda)$, it is guaranteed to improve the dual objective by at least $\epsilon$. Moreover, if one cannot find such a direction within the $\epsilon$-subdifferential, then $q(\lambda)$ is guaranteed to be $\epsilon$-close to the dual optimum. This is summarized in the following claim.

**Claim 5** Let $q(\lambda)$ be a convex function and let $\epsilon$ be a given positive scalar. The $\epsilon$-subdifferential $\partial_\epsilon q(\lambda)$ is a convex and compact set. If $0 \not\in \partial_\epsilon q(\lambda)$ then the direction $d^* = \text{argmin} \|d\|$ subject to $d \in \partial_\epsilon q(\lambda)$ is a descent direction and $\inf_{\eta > 0} q(\lambda + \eta d) < q(\lambda) - \epsilon$. On the other hand, if $0 \in \partial_\epsilon q(\lambda)$ then $q(\lambda) \leq \inf_\lambda q(\lambda) + \epsilon$.

**Proof:** It follows Bertsekas et al. (2003), Proposition 4.3.1. Although $\partial_\epsilon q(\lambda)$ is a convex and compact set, finding its direction of descent is computationally challenging. Fortunately, it can be approximated whenever the convex function is a sum of simple convex functions, i.e., $q(\lambda) = \sum_{r \in R} q_r(\lambda)$. The approximation $\tilde{\partial}_\epsilon q(\lambda) = \sum_r \hat{\partial}_\epsilon q_r(\lambda)$ satisfies $\tilde{\partial}_\epsilon q(\lambda) \subset \tilde{\partial}_\epsilon q(\lambda) \subset \partial_{|R|\epsilon} q(\lambda)$, (see, e.g., Bertsekas et al. (2003)). On the one hand, if $0 \not\in \hat{\partial}_\epsilon q(\lambda)$ then the direction of steepest descent taken from $\hat{\partial}_\epsilon q(\lambda)$ reduces the dual objective by at least $\epsilon$. If $0 \in \hat{\partial}_\epsilon q(\lambda)$ then $q(\lambda)$ is $|R|\epsilon$-close to the dual optimum. In the following claim we use the $\epsilon$-margin Fenchel-Young duality in Eq. (4.16) to characterize the approximated $\epsilon$-subdifferential of the dual function.

**Claim 6** Consider the dual function $q(\lambda)$ in Eq. (4.13). Then the approximated $\epsilon$-subdifferential consists of vectors $d$ whose entries correspond to marginalization disagreements, i.e., $d \in \partial_\epsilon q(\lambda)$ if and only if $d_{r \rightarrow p}(s_r) = \sum_{s_p \mid s_r} b_p(s_p) - b_r(s_r)$ for probability distributions $b_r(s_r)$ that satisfy

$$\forall r \max_{s_r} \left\{ \theta_r(s_r) - \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_r) + \sum_{c \in C(r)} \lambda_{c \rightarrow p}(s_c) \right\} - \epsilon \leq$$

$$\leq \sum_{s_r} b_r(s_r) \left( \theta_r(s_r) - \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_r) + \sum_{c \in C(r)} \lambda_{c \rightarrow p}(s_c) \right).$$

**Proof:** Eq. (4.16) implies $b \in \partial_\epsilon q_r(\hat{\theta})$ if and only if $q_r(\hat{\theta}) + q_r^*(b) - b^T \hat{\theta} \leq \epsilon$ with $q_r^*(b)$ denoting the conjugate dual of $q_r(\hat{\theta})$. Plugging in $q_r, q_r^*$ we obtain not only the maximizing beliefs but all beliefs with an $\epsilon$-margin. Summing over $r$ while noticing that $\lambda_{r \rightarrow p}(s_r)$ change signs between $q_p$ and $q_r$ we obtain the marginalization disagreements $d_{r \rightarrow p}(s_r) = \sum_{s_p \mid s_r} b_p(s_p) - b_r(s_r)$. $\tilde{\partial}_\epsilon q(\lambda)$ is described using beliefs $b_r(s_r)$ that satisfy linear constraints, therefore finding a direction of $\epsilon$-descent can be done efficiently. Claim 5 ensures that minimizing the dual objective along a direction of descent decreases its value by at least $\epsilon$. Moreover, we are guaranteed to be $|R|\epsilon$-close to a dual optimal solution if no direction of
4. Inference in Structured Models

descent is found in \( \partial_\epsilon \tilde q(\lambda) \). Therefore, we are able to get out of corners and efficiently reach an approximated dual optimal solution. The interpretation of the Fenchel-Young margin as the amount of disagreement between the marginalization constraints also provides a simple way to reconstruct an approximately optimal primal solution. This is summarized in the following corollary.

**Corollary 2** Given a point \( \lambda \), set the re-parameterization potential \( \hat \theta_r(s_r) = \theta_r(s_r) - \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_p) + \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c) \). Consider the quadratic program

\[
\min_{b_r} \sum_{r, s_r, p \in P(r)} \left( \sum_{s_p \setminus s_r} b_p(s_p) - b_r(s_r) \right)^2 \\
\text{s.t.} \quad C = \begin{cases} \\
\forall r, s_r \\
\forall r \\
\forall r \\
\forall \end{cases} \\
\sum_{s_r} b_r(s_r) \geq 0 \\
\sum_{s_r} b_r(s_r) = 1 \\
\sum_{s_r} b_r(s_r) \hat \theta_r(s_r) \geq \max_{s_r} \{ \hat \theta_r(s_r) \} - \epsilon. 
\]

\( q(\lambda) \) is \(|\mathcal{R}|\epsilon\)-close to the dual optimal value if and only if the value of the above program equals zero. Moreover, the optimal beliefs \( b^*_r(s_r) \) primal value is \(|\mathcal{R}|\epsilon\)-close to the optimal primal value in Eq. (4.12). However, if \( q(\lambda) \) is not \(|\mathcal{R}|\epsilon\)-close to the dual optimal value then the vector \( d^*_r(s_r) = \sum_{s_p \setminus s_r} b^*_p(s_p) - b^*_r(s_r) \) points towards the steepest \( \epsilon \)-descent direction of the function, namely

\[
d^* = \arg \min_{\|d\| \leq 1} \lim_{\eta \downarrow 0} \frac{q(\lambda + \eta d) - q(\lambda) + \epsilon}{\eta}.
\]

**Proof:** The steepest \( \epsilon \)-descent direction is given by the minimum norm element of the \( \epsilon \)-subdifferential, described in Claim 6. \(|\mathcal{R}|\epsilon\)-closeness to the dual optimum is given by (Bertsekas et al. 2003, Proposition 4.3.1) once we find the value of the quadratic program to be zero. Note that the superset \( \tilde \partial_\epsilon \) is composed of \(|\mathcal{R}|\) subdifferentials. If the value of the above program equals zero, the beliefs fulfill marginalization constraints and they denote a probability distribution. Summing both \( \epsilon \)-margin inequalities w.r.t. \( r \), we obtain

\[
\sum_{r, s_r} b_r(s_r) \hat \theta_r(s_r) \geq \sum_{r} \max_{s_r} \hat \theta_r(s_r) - |\mathcal{R}|\epsilon.
\]

where the primal on the left hand side of the resulting inequality is larger then the dual subtracted by \( |\mathcal{R}|\epsilon \). With the dual itself upper bounding the primal, the corollary follows. 

Thus, we can construct an algorithm that performs \( \epsilon \) improvements over the dual function in each iteration. We can either perform block-coordinate dual descent (i.e., convex max-product updates) or steepest \( \epsilon \)-descent steps. Since both methods monotonically improve the same dual function, our approach is guaranteed to reach the optimal dual solution and to recover the primal optimal solution.
4.3. Globally Convergent LP Relaxation Solver

4.3.3. Conditional Gradient for \( \epsilon \)-descent Directions

In what follows, we propose to employ the Frank-Wolfe schema, also known as conditional gradients, to decouple the problem and solve it more efficiently. The intuition is based on the fact that programs with nonlinear cost functions and independent linear constraints can typically be solved via an efficient sequence of much simpler linear programs (Bertsekas, 1999).

To this end we adopt a Frank-Wolfe schema (Frank & Wolfe, 1956), which employs the gradient of the cost function. Assuming beliefs \( b \) to lie within the constraint set \( C \), the gradient of the cost function \( f(b) = \sum_{r,s,p} \left( \sum_{s_p \setminus s_r} b_{sp} - b_r(s_r) \right)^2 \) w.r.t. \( b_r(s_r) \) is given by

\[
\nabla_{b_r(s_r)} f = \sum_{p \in P(r)} -2 \left( \sum_{s_p \setminus s_r} b_{sp} - b_r(s_r) \right) + \sum_{c \in C(r)} 2 \left( \sum_{s_r \setminus s_c} b_r(s_r) - b_c(s_c) \right)
\]

\[
= 2 \left( \sum_{c \in C(r)} d_{c \rightarrow r}(s_c) - \sum_{p \in P(r)} d_{r \rightarrow p}(s_r) \right),
\]

with \( d_{r \rightarrow p}(s_r) = \sum_{s_p \setminus s_r} b_{sp} - b_r(s_r) \) denoting the marginalization disagreements.

The conditional gradient method to compute the steepest \( \epsilon \)-descent direction, i.e., to solve the quadratic program given in Eq. (4.17), is summarized in Fig. 4.14. Similar to the standard Frank-Wolfe schema it proceeds by iterating three steps. Firstly, a descent direction of the linearized quadratic cost function at the current iterate \( b_r \) is found by solving the following program:

\[
u^* = \arg \min_u \sum_{r,s_r} u_r(s_r) \nabla_{b_r(s_r)} f \quad \text{s.t.} \quad u \in C.
\] (4.18)

Secondly, we compute the optimal step length \( \gamma^* \) before updating the beliefs while making sure not to leave the constraint set, i.e., \( 0 \leq \gamma^* \leq 1 \). The latter two operations can be efficiently computed analytically in closed form and involve only simple arithmetic operations.

Due to the structure within the constraint set \( C \), the LP given in Eq. (4.18) decomposes such that the optimum is found by considering each region \( r \) independently. We therefore solve for all regions \( r \) in parallel and denote the corresponding local constraint set by \( C_r \). The program given in Eq. (4.18) is hence replaced by small programs, one for every region. These have the following form \( \forall r \):

\[
u^*_r = \arg \min_{u_r} \sum_{s_r} u_r(s_r) \nabla_{b_r(s_r)} f \quad \text{s.t.} \quad u_r \in C_r = \left\{ u_r : \begin{array}{l} \forall s_r \quad u_r(s_r) \geq 0 \\ \sum_{s_r} u_r(s_r) = 1 \\ \sum_{s_r} u_r(s_r) \hat{\theta}_r(s_r) \geq \max_{s_r} \hat{\theta}_r(s_r) - \epsilon. \end{array} \right\}
\] (4.19)

51
4. Inference in Structured Models

**Algorithm: Frank-Wolfe**

Iterate until convergence:

1. Solve the LP
   \[ u^* = \arg \min_u \sum_{r,s_r} u_r(s_r) \nabla b_r(s_r) f \quad \text{s.t.} \quad u \in C. \]

2. Compute the optimal step size \( \gamma^* \) in closed form by solving
   \[
   \gamma^* = \arg \min_{0 \leq \gamma \leq 1} \sum_{r,s_r,p \in P(r)} \left( \sum_{s_p \setminus s_r} (b_p(s_p) + \gamma (u^*_p(s_p) - b_p(s_p))) - (b_r(s_r) + \gamma (u^*_r(s_r) - b_r(s_r)))) \right)^2.
   \]

3. Update the beliefs via
   \[ b_r(s_r) \leftarrow b_r(s_r) + \gamma^* (u^*_r(s_r) - b_r(s_r)) \quad \forall r, s_r. \]

Figure 4.14.: Frank-Wolfe algorithm for finding the solution of the program given in Eq. (4.17).

While solving many small decoupled linear programs is arguably faster than a single large program, we are interested in a solution that does not require a general LP solver. The procedure is hence as efficient as possible and allows to exactly specify the required computational complexity.

**Construction of a Primal Optimal Solution**

To this end we show existence and availability of a construction for the primal optimal solution of the program given in Eq. (4.19). The method first identifies the non-zero domain for a feasible solution. A primal optimal solution is then analytically computed on this domain by either fixing a single entry to equal one or by solving a \( 2 \times 2 \) sized system of linear equations.

**Theorem 1** There exists a non-zero domain \( S^*_r \) with \( 1 \leq |S^*_r| \leq 2 \), and a distribution \( u_r(s_r) \) which is primal feasible and primal optimal for the program given in Eq. (4.19), such that \( u_r(s_r) \geq 0 \) if \( s_r \in S^*_r \).

**Proof:** To prove the theorem we consider the Karush-Kuhn-Tucker (KKT) conditions of the program given in Eq. (4.19). We introduce Lagrange multipliers \( \mu \) for the constraint that the weighted sum exceeds the constant \( \max_{s_r} \hat{\theta}_r(s_r) - \epsilon \), multipliers \( \sigma_r(s_r) \)
for the positivity constraint and multiplier $\gamma$ for the summation over states equaling one. Hence the stationarity, primal feasibility, dual feasibility and complementary slackness requirements read as follows:

\[
\nabla_{b_r(s_r)} f - \mu \hat{\theta}_r(s_r) - \sigma_r(s_r) + \gamma = 0 \quad \forall s_r \quad \text{(Stationarity)}
\]

\[
u_r(s_r) \geq 0 \quad \forall s_r \quad \text{(Primal Feasibility)}
\]

\[
\sum_{s_r} u_r(s_r) = 1 \quad \text{(Primal Feasibility)}
\]

\[
\max_{s_r} \hat{\theta}_r(s_r) - \epsilon - \sum_{s_r} u_r(s_r) \hat{\theta}_r(s_r) \leq 0 \quad \text{(Primal Feasibility)}
\]

\[
\mu \geq 0 \quad \text{(Dual Feasibility)}
\]

\[
\sigma_r(s_r) \geq 0 \quad \forall s_r \quad \text{(Dual Feasibility)}
\]

\[
\mu \left( \max_{s_r} \hat{\theta}_r(s_r) - \epsilon - \sum_{s_r} u_r(s_r) \hat{\theta}_r(s_r) \right) = 0 \quad \text{(Complementary Slackness)}
\]

\[
\sigma_r(s_r) u_r(s_r) = 0 \quad \forall s_r \quad \text{(Complementary Slackness)}
\]

Plugging the stationarity requirement given in Eq. (4.20) into the complementary slackness constraint provided in Eq. (4.27) yields $u_r(s_r) \left( \nabla_{b_r(s_r)} f - \mu \hat{\theta}_r(s_r) + \gamma \right) = 0 \forall s_r$ with $\sigma_r(s_r) = \nabla_{b_r(s_r)} f - \mu \hat{\theta}_r(s_r) + \gamma \geq 0 \forall s_r$ following from Eq. (4.25).

Assuming $\mu = 0$ requires $\sigma_r(s_r) = \nabla_{b_r(s_r)} f + \gamma \geq 0$. To ensure non-negativity and a minimum cost function value for the program given in Eq. (4.19), we set $\gamma$ to be the negative value of the smallest gradient, i.e., $\gamma = -\min_{s_r} \nabla_{b_r(s_r)} f$, and choose one state $S^*_r \in \arg \max_{s_r: \nabla_{b_r(s_r)} f = -\gamma} \hat{\theta}_r(s_r)$ to obtain $|S^*_r| = 1$. Setting $u_r(s_r) = 1$ if $s_r \in S^*_r$ and to zero otherwise fulfills dual feasibility and complementary slackness constraints. All primal feasibility constraints are fulfilled if $\max_{s_r} \hat{\theta}_r(s_r) - \epsilon - \sum_{s_r} u_r(s_r) \hat{\theta}_r(s_r) \leq 0$ for $s_r \in S^*_r$.

Phrasing the aforementioned mathematical statements in words: we take that single state $S^*_r$ among the minimizing elements of the cost function (Eq. (4.19)) which has the largest $\hat{\theta}_r(s_r)$. If the distribution $u_r(s_r)$ placing all its mass on that state fulfills the primal feasibility constraint given in Eq. (4.23) we have found the optimal feasible solution with $|S^*_r| = 1$.

Let us for now assume existence of a solution for the distribution $u_r(s_r)$ that has at most two non-zero entries. Using the condition obtained earlier by combination of stationarity with complementary slackness, i.e., $u_r(s_r) \left( \nabla_{b_r(s_r)} f - \mu \hat{\theta}_r(s_r) + \gamma \right) = 0$, for the two non-zero states enables computation of $\mu$ and $\gamma$ by inverting a $2 \times 2$ sized linear system of equations analytically. Assuming for now that dual feasibility holds, we construct a primal feasible solution by inverting the $2 \times 2$ sized linear system arising

53
from Eq. (4.22) and by enforcing Eq. (4.23) to hold with equality. We observe primal feasibility, and in particular also Eq. (4.21) to hold, if the set \( S^*_r = \{ s_1, s_2 \} \) contains one state \( s_1 \) with \( \hat{\theta}_r(s_1) > \max \hat{s}_r \hat{\theta}_r(\hat{s}_r) - \epsilon \) and another one with \( \hat{\theta}_r(s_2) \leq \max \hat{s}_r \hat{\theta}_r(\hat{s}_r) - \epsilon \).

It remains to be shown that we can find two such states \( s_1, s_2 \) which also fulfill dual feasibility. For dual feasibility to hold we require \( \mu \geq 0 \) and \( \sigma_r(s_r) = \nabla b_r(s_r) f - \mu \hat{\theta}_r(s_r) + \gamma \geq 0 \) \( \forall s_r \). To interpret this program we refer the reader to Fig. 4.15. Every state \( s_r \) defines the linear function \( \nabla b_r(s_r) f - \mu \hat{\theta}_r(s_r) \) which depends on the Lagrange multiplier \( \mu \). We emphasize that there exists at least one linear function with slope strictly smaller than \( -\max \hat{s}_r \hat{\theta}_r(\hat{s}_r) + \epsilon \) due to \( \epsilon > 0 \). Otherwise the problem would be unbounded. Note the change of sign, i.e., the slope is \(-\hat{\theta}_r(s_r)\).

Intersecting every line \( s_1 \) with slope strictly smaller than \(-\max \hat{s}_r \hat{\theta}_r(\hat{s}_r) + \epsilon \) with every line \( s_2 \) having a slope larger or equal to this constant gives the point \((\mu', -\gamma')\). Importantly dual feasibility only holds if \( \sigma_r(s_r) = \nabla b_r(s_r) f - \mu' \hat{\theta}_r(s_r) + \gamma' \geq 0 \) \( \forall s_r \), i.e., if \( \nabla b_r(s_r) f - \mu' \hat{\theta}_r(s_r) \geq -\gamma' \) \( \forall s_r \) while \( \mu' \geq 0 \). In words: every line has to pass above the point of intersection being \((\mu', -\gamma')\). Existence of at least one combination \( S^*_r = \{ s_1, s_2 \} \) is guaranteed if we found the case \( \mu = 0 \) not to yield a valid solution.

This concludes the proof of the theorem where we showed that we can construct a primal feasible solution \( u_r(s_r) \) with the non-zero domain being at most two.

In the following lemma we provide a statement regarding the computational complexity of constructing a primal optimal and primal feasible solution when following the procedure designed during the proof of Theorem 1.

**Lemma 1** A feasible primal optimal solution \( u_r(s_r) \) for the program in Eq. (4.19) is found with computational complexity at most \( O(|S_r|^2) \).

**Proof:** To prove the lemma we note that intersecting the line defined by \( \nabla b_r(s_r) f - \mu \hat{\theta}_r(s_r) \) for every \( s_r \) with every other state \( s_r \) as described in Theorem 1 is of quadratic complexity which proves the claim.

---

4. Inference in Structured Models

![Image of a graph showing the minimum given a set of lines illustrated in black color.](image-url)
We obtain the dual function $g$ solving the dual problem Algorithm 1 which proves the claim.

Proof: To prove this claim we note that a possible Lagrangian for the program in maximizing $y \overline{\phi}$ over $s_r^1 \in S$ follows as

$$ L() = \sum_{s_r} u_r(s_r) \nabla_{b_r(s_r)} f + \mu \left( \max_{\hat{s}_r} \hat{\theta}_r(\hat{s}_r) - \epsilon - \sum_{s_r} u_r(s_r) \hat{\theta}_r(s_r) \right). $$

We obtain the dual function $g(\mu)$ by minimizing $L()$ over $u_r \in \Delta$ where $\Delta$ is the probability simplex of corresponding dimension. The dual function is given by $g(\mu) = \min_{s_r} \left\{ \nabla_{b_r(s_r)} f - \mu \hat{\theta}_r(s_r) \right\} + \mu \left( \max_{\hat{s}_r} \hat{\theta}_r(\hat{s}_r) - \epsilon \right)$ resulting in the dual program maximizing $\mu \geq 0 g(\mu)$. This task equivalently reads as

$$ \max_{\mu \geq 0, y} \forall s_r \quad y \leq \nabla_{b_r(s_r)} f - \mu \left( \hat{\theta}_r(s_r) - \max_{\hat{s}_r} \hat{\theta}_r(\hat{s}_r) + \epsilon \right), $$

which proves the claim.

Relationship with the Dual Program

Subsequently we describe how to find the dual feasible solution and the non-zero domain of the distribution, i.e., $S_r^*$ more efficiently than by intersecting all combinations of lines. The worst case complexity is nonetheless quadratic in the number of states, i.e., Lemma 1 remains valid. The main idea is based on the fact that the task of finding the non-zero domain is equivalent to solving the dual program of the optimization given in Eq. (4.19).

Claim 7 The dual program for the task given in Eq. (4.19) reads as

$$ \max_{\mu \geq 0, y} \forall s_r \quad y \leq \nabla_{b_r(s_r)} f - \mu \left( \hat{\theta}_r(s_r) - \max_{\hat{s}_r} \hat{\theta}_r(\hat{s}_r) + \epsilon \right). \quad (4.28) $$

Proof: To prove this claim we note that a possible Lagrangian for the program in Eq. (4.19) is given by

$$ L() = \sum_{s_r} u_r(s_r) \nabla_{b_r(s_r)} f + \mu \left( \max_{\hat{s}_r} \hat{\theta}_r(\hat{s}_r) - \epsilon - \sum_{s_r} u_r(s_r) \hat{\theta}_r(s_r) \right). \quad (4.29) $$

We obtain the dual function $g(\mu)$ by minimizing $L()$ over $u_r \in \Delta$ where $\Delta$ is the probability simplex of corresponding dimension. The dual function is given by $g(\mu) = \min_{s_r} \left\{ \nabla_{b_r(s_r)} f - \mu \hat{\theta}_r(s_r) \right\} + \mu \left( \max_{\hat{s}_r} \hat{\theta}_r(\hat{s}_r) - \epsilon \right)$ resulting in the dual program maximizing $\mu \geq 0 g(\mu)$. This task equivalently reads as

$$ \max_{\mu \geq 0, y} \forall s_r \quad y \leq \nabla_{b_r(s_r)} f - \mu \left( \hat{\theta}_r(s_r) - \max_{\hat{s}_r} \hat{\theta}_r(\hat{s}_r) + \epsilon \right), \quad (4.30) $$

which proves the claim.
4. Inference in Structured Models

Algorithm: Efficient Globally Convergent Parallel MAP LP Relaxation Solver

Let \( \hat{\theta}_r(s_r) = \theta_r(s_r) + \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_r) - \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c) \)

Iterate until convergence:

1. Iterate
   a) \( \forall r \) in parallel: construct a primal feasible solution \( u_r(s_r) \)
   b) compute the optimal step size \( \gamma^* \) and update the region beliefs \( b_r(s_r) \) as detailed in Fig. 4.14

2. Compute the disagreement:
   \[ d_{r \rightarrow p}(s_r) = \sum_{s_p \setminus s_r} b_p(s_p) - b_r(s_r) \]

3. Update messages with stepsize \( \eta \) obtained through line search to improve \( q(\lambda) \):
   \[ \lambda_{r \rightarrow p}(s_r) \leftarrow \lambda_{r \rightarrow p}(s_r) + \eta d_{r \rightarrow p}(s_r) \]

4. Update potentials
   \[ \hat{\theta}_r(s_r) \leftarrow \theta_r(s_r) + \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c) - \sum_{p \in P(r)} \lambda_{r \rightarrow p}(s_r) \]

Figure 4.16.: An efficient, parallel and provably convergent MAP LP Relaxation solver.

To provide some intuition for this task we refer the reader to Fig. 4.15. We are interested in finding the maximum of the pointwise minimum of a set of lines. This is exactly the program required to be solved in Theorem 1 when considering \( \mu \geq 0 \): at least two linear functions have to intersect while all the other ones are above and the gradients of the lines have opposite sign, \( i.e. \), one is larger than \( z_r = \max_{s_r} \hat{\theta}_r(s_r) - \epsilon \) and the other one is smaller or equal. The active constraints \( S^*_r \) are the minimizing elements of \( \nabla_{b_r(s_r)} f - \mu \hat{\theta}_r(s_r) \) which is the non-zero domain of \( u_r(s_r) \). We take those two states \( s_1, s_2 \) closest to \( z_r \) while fulfilling the required constraints \( \hat{\theta}_r(s_1) > z_r, \hat{\theta}_r(s_2) \leq z_r \).

Subsequently we design an algorithm to find \( \mu \geq 0, y = g(\mu) \) and all active constraints \( s_r \in S^*_r \). To this end we denote the gradients of the lines via \( m(s_r) = -\hat{\theta}_r(s_r) + \max_{\tilde{s}_r} \tilde{\theta}_r(\tilde{s}_r) - \epsilon \). The details of the algorithm are given in Alg. 1. We start with \( \mu = 0 \) and find the set of lines with minimum y-axis intercept (cf. Alg. 1 line 1). If the gradients of all lines are larger or equal zero we intersect the line with smallest gradient with the set of all other lines having an even smaller gradient and we find
4.3. Globally Convergent LP Relaxation Solver

Figure 4.17.: (a): Difference between the minimal dual value attained by convex max-product \( q(\lambda_{\text{CMP}}) \) and our approach \( q(\lambda_\epsilon) \). Convex max-product gets stuck in about 20% of all cases. (b): The dual value w.r.t. time for different algorithms.

all the lines \( S_r^2 \) that have the smallest point of intersection \( \mu \) \((\text{cf. Alg. 1 line 7f})\). If there is one line with negative gradient among \( S_r^2 \) we terminate while having found \( \mu, y \) and \( S_r \) as given in Alg. 1. In the worst case we have to intersect every line with every other line, i.e., the algorithm complexity is quadratic in the number of states of the region.

We hence designed a method that retrieves an optimal solution for the program given in Eq. (4.19) with a computational complexity as stated in Lemma 1. Summarizing the constructive proof, we first investigate whether a primal optimal and feasible solution can be found for \( \mu = 0 \) which is possible in linear time. If no such solution is found we proceed by successively walking along the lower envelope highlighted with black color in Fig. 4.15.

It is important to note that it is generally beneficial to interleave the optimization for finding the steepest \( \epsilon \)-descent direction with the update of the dual variables, i.e., we blend the conditional gradient procedure outlined in Fig. 4.14 with an update of the dual variables \( \lambda \). After some conditional gradient iterations we can potentially verify whether a sufficiently large (\( \geq \epsilon \)) improvement of the dual \( q(\lambda) \) is possible. A single Frank-Wolfe step might however require frequent verifications of the dual improvement. Therefore there is a trade-off between fewer conditional gradient iterations and more verifications of the resulting \( \epsilon \)-descent directions. A possible procedure for an efficient globally optimal maximum a-posteriori (MAP) LP relaxation solver is outlined in Fig. 4.16. We compute the \( \epsilon \)-steepest descent direction by iterating step 1(a) and step 1(b) for an arbitrary number of times, verifying a possible \( \epsilon \) improvement with step 2 and step 3 and continue to iterate after reparameterizing in step 4.
4. Inference in Structured Models

4.3.4. Experiments

We compare our approach to a wide variety of state-of-the-art baselines using $10 \times 10$ sized spin-glass models and energy functions arising from a protein design task. For all experiments we successively decrease $\epsilon$ starting from $\epsilon = 0.01$ if the model is sufficiently close to $|\mathcal{R}|\epsilon$ optimality.

Spin Glasses

We consider spin glass models that consist of local factors, each having 3 states with values randomly chosen according to a zero mean, unit variance normal distribution $\mathcal{N}(0, 1)$. We use three states as convex max-product is optimal for pairwise spin glass models with only two states per random variable. The pairwise factors of the regular grid are weighted potentials with +1 on the diagonal, and off-diagonal entries being −1. The weights are independently drawn from $\mathcal{N}(0, 1)$.

In a first experiment we are interested in estimating how often convex max-product gets stuck in corners. We generate a set of 1000 spin glass models and estimate the distribution of the dual value difference comparing the $\epsilon$-descent approach with the convex max-product result after 10,000 iterations. We observe in Fig. 4.17(a) that about 20% of the spin glass models have a dual value difference larger than zero.

Having observed that convex max-product does not achieve optimality for 20% of the models, we now turn our attention to evaluating the run-time of different algorithms. We can therefore start with efficient block-coordinate descent steps before switching to computing the $\epsilon$-steepest descent directions via either the Frank-Wolfe procedure or a quadratic programming (QP) solver. We denote these approaches as either “FW,” parallelized to employ all cores or “QP” employing a parallelized commercial QP solver. Another baseline is the alternating direction method for dual MAP LP relaxations (ADLP) (Meshi & Globerson, 2011). In addition, we illustrate the performance of convex max-product (CMP) and convex sum-product (CSP) (Hazan & Shashua, 2010).
4.3. Globally Convergent LP Relaxation Solver

and compare with the dual-decomposition work of Komodakis et al. (2010) provided in a generic (DDG), a re-weighted (DDR) and a smoothed (DDS) version in the STAIR library by Gould et al. (2011). Note that ADLP is also implemented in this library. All algorithms are restricted to at most 5,000 iterations.

We draw the readers’ attention to Fig. 4.17(b), where we evaluate a single spin glass model and illustrate the dual value obtained after a certain amount of time. As given by the derivations, CMP is a monotonically decreasing algorithm that can get stuck in corners. Convex sum product (CSP) is guaranteed to monotonically converge to the optimum of a augmented problem which could be far from the MAP solution. It is important to note that our \(\epsilon\)-descent approaches denoted by FW and QP are monotonically decreasing as well, which contrasts the remaining investigated algorithms (ADLP, DDG, DDR, DDS).

We then evaluate how many conditional gradient iterations are required before checking whether we can find an \(\epsilon\)-descent direction. To this end, we compare our implementation using 800, 1200 and 1600 Frank-Wolfe iterations with the QP formulation of Schwing et al. (2012c). The results for two spin glass models are given in Fig. 4.18. We observe in Fig. 4.18(a) that 800 conditional gradient iterations are sufficient for this particular spin glass model. As apparent in Fig. 4.18(b) choosing the number of Frank-Wolfe iterations too low might however result in slow convergence due to frequent \(\epsilon\)-descent checks that are not successful. We subsequently set the number of conditional gradient iterations to 1200. But we note that an adaptive strategy, which
successively increases the number of Frank-Wolfe iterations to take advantage of the fact that \( \epsilon \)-descent directions are easily found during the first few rounds might be preferred.

Subsequently we evaluate the time it requires the different algorithms to achieve a given accuracy. We focus on “hard” problems, where we defined “hard” as those spin glass models whose difference between convex max-product and the \( \epsilon \)-descent method is larger than 0.2. To obtain statistically meaningful results we average over 30 hard problems. The results are visualized in Fig. 4.19. We illustrate the percentage of samples that achieved the indicated deviation from the optimum within a certain amount of time. We observe that significantly more samples achieve a lower deviation from the optimum in a smaller amount of time when using the proposed CESD approach. The maximum of 5,000 iterations limits convergence for all samples to low error deviation. Naturally, some algorithms did not get within the indicated range of optimality.

In a next experiment we evaluate the fraction of samples that achieved a deviation from the optimum that is smaller than a certain error given a specific amount of time. The results are visualized in Fig. 4.20. Again the proposed combination of convex max-product with a steepest \( \epsilon \)-descent method achieves very good performance.
Protein Design

Next we consider a protein design task and compare our algorithms (FW and QP) to the performance of the most competitive methods, i.e., ADLP and CMP. To this end, we make use of the eight models publicly available from the probabilistic inference challenge\(^1\). For all algorithms we use the same setting as before, except that we increase the number of the maximally possible iterations to 50,000.

Fig. 4.21 illustrates the dual energy over time. We observe fast convergence when the convex max-product (CMP) algorithm is optimal (see Fig. 4.21(a)). Similar to spin glass models, CMP gets stuck in corners as shown in Fig. 4.21(b)-(d). Our proposed \(\epsilon\)-steepest descent approach successfully finds the global optimum while monotonically decreasing the dual energy.

Similar to the spin glass models we evaluate for all eight protein design tasks the time required to achieve a specific optimum and the cumulative error distribution after

\(^1\)http://www.cs.huji.ac.il/project/PASCAL/index.php
4. Inference in Structured Models

Figure 4.22.: Time it requires to get a fraction of the samples to within the indicated optimality.

Figure 4.23.: Percentage of samples that achieved a smaller error after the indicated time.

a given amount of time in Fig. 4.22 and Fig. 4.23 respectively. We observe that the proposed algorithm achieves very good performance. Note that CMP gets stuck in non-optimal points while ADLP does not monotonically decrease the dual function.

4.3.5. Summary

To summarize, we discussed the challenges of the maximum a-posteriori (MAP) task arising due to a non-smooth dual function. We proposed to address the issue of convergence to the global optimum via an $\epsilon$-steepest descent method and decomposed the resulting optimization problem via a conditional gradient approach to obtain an efficient parallel algorithm that is guaranteed to monotonically converge to the global optimum.

4.4. Conclusion

Within this chapter we discussed inference algorithms for structured models. We presented efficient algorithms and discussed their challenges of computational complexity and convergence. For both issues we presented a possible solution.
5. Learning in Structured Models

While inference as discussed in Chapter 4 assumes availability of a scoring function \( \theta(s) \) to define a probability distribution \( p(s \mid \theta) \propto \exp(\beta \theta(s)) \), we are often facing the problem of designing appropriate functions. Given a set of feature or measurement functions \( \phi(x, s) \), this chapter considers how to obtain a discriminative linear combination. To this end we first review existing techniques in Sec. 5.1 before describing distributed structured prediction algorithms suitable for big data sets in Sec. 5.2.

5.1. Review

In the previous chapter we assumed the potential function \( \theta(s, x) \) to be computable for a specific output space configuration \( s \) once given input data \( x \). How do we design this function?

For learning with structured models we consider a log-linear parametrization throughout this chapter. As before, we operate with a constructed feature vector \( \phi(x, s) \) which maps from the input and output product space to an \( F \)-dimensional vector, \( \phi(x, s) : \mathcal{X} \times \mathcal{S} \mapsto \mathbb{R}^F \). Linearly combining the elements of this feature vector yields our potential function

\[
\theta(s, x) = w^\top \phi(x, s).
\]

This linear combination with \( w \in \mathbb{R}^W \) requires the number of parameters to correspond to the dimensionality of the feature vector, \( W = F \).

As mentioned before it is among others construction of the feature vector \( \phi(x, s) \) which makes the learning task imponderable without domain knowledge. Particularly since every element \( \phi_k(x, s), k \in \{1, \ldots, F\} \) of the feature vector has to be considered and designed carefully.

Important for algorithmic considerations and similar to the decomposition assumption described for the potential function in the previous chapter, we refer to a subset of the output space variables via the index set \( r \). We assume the \( k \)-th feature vector element to decompose according to

\[
\phi_k(x, s) = \sum_{r \in \mathcal{R}_k} \phi_{k,r}(x, s_r) \tag{5.1}
\]

where \( \mathcal{R}_k \) subsumes the index sets \( r \) required for computation of the \( k \)-th feature vector entry.

To illustrate that such a decomposition occurs in practice, we provide a brief but illustrative example in the following. Consider a gray-scale image denoising task where we receive an intensity value \( x_i \) for every pixel \( i \in \{1, \ldots, N\} \). We assume the variable
s_i to take a state close to its observed value referred to via x_i. Hence let the first feature vector entry compute the sum of the absolute differences between the observation and the chosen state for all pixel, i.e., in equations: \( \phi_1(x, s) = \sum_{r \in \mathcal{R}_1} \phi_{1,r}(x, s_r) = \sum_{r \in \mathcal{R}_1} |s_r - x_r| \). The set of regions required for computation of the first feature vector entry is given by the individual variables, i.e., \( \mathcal{R}_1 = \{\{i\}_{i=1}^N\} \). We say that the individual potentials \( \phi_{1,r}(x, s_r) = |s_r - x_r| \) are unaries since they depend on a single variable as \(|r| = 1 \forall r \in \mathcal{R}_1\).

In addition we might assume neighboring variables i and j to take the same states, i.e., \( \phi_2(x, s) = \sum_{r \in \mathcal{R}_2} \phi_{2,r}(x, s_r) = \sum_{(i,j) \in \mathcal{R}_2} |s_i - s_j| \) with \( \mathcal{R}_2 \) containing sets each being a pair \((i,j)\) of neighboring pixel. Since two variables are involved in computing an element \( \phi_{2,r} = |s_i - s_j| \) we refer to those potentials as pairwise or as being of order two.

All in all we obtain an \( F = 2 \) dimensional feature vector \( \phi(x, s) = [\phi_1(x, s) \ \phi_2(x, s)]^\top \). Without learning, the domain expert chooses two numbers \( w = [w_1 \ w_2]^\top \) to obtain the potential function \( \theta(x, s) = w^\top \phi(x, s) \). Rather than tuning the weights manually we obtain a data driven approach when leveraging a data set \( \mathcal{D} \) containing pairs \((x, s)\) of input space and annotated output space objects, i.e., \( \mathcal{D} = \{(x^{(i)}, s^{(i)})\}_{i=1}^{|\mathcal{D}|} \).

A domain expert might have a good idea about the sign of a specific weight but the order of magnitude is certainly hard to estimate manually. Continuing the aforementioned denoising example we expect a learning algorithm to yield \( w_1, w_2 < 0 \). The obtained signs of the parameter vector are hence useful for sanity checks.

Let us proceed step by step to derive the cost function for a data driven learning algorithm. The data set \( \mathcal{D} \) contains the ground truth output space configuration \( s \), i.e., the labeling we expect the inference algorithm to retrieve when given the corresponding input space object \( x \). Therefore we want an algorithm to choose a weight vector such that any other configuration \( \hat{s} \) scores smaller than the ground truth configuration.

More formally we obtain

\[
\forall \hat{s} \in S \quad w^\top \phi(x, \hat{s}) \leq w^\top \phi(x, s).
\]

Since all configurations \( \hat{s} \in S \) should score less or equal to the ground truth configuration \( s \), we equivalently require the maximizing configuration to have a score less or equal to the ground truth, i.e., \( \max_{\hat{s} \in S} w^\top \phi(x, \hat{s}) \leq w^\top \phi(x, s) \). Let’s ask for some additional margin \( \ell_{(x,s)}(\hat{s}) \geq 0 \) when considering data sample \((x, s)\) and configuration \( \hat{s} \) by requiring a weight vector \( w \) to achieve \( \max_{\hat{s} \in S} w^\top \phi(x, \hat{s}) + \ell_{(x,s)}(\hat{s}) \leq w^\top \phi(x, s) \). Subsequently we also refer to the margin \( \ell \) as a loss function.

Given this insight we linearly penalize our cost function if we find a configuration \( \hat{s} \) that scores larger than the ground truth output space object, i.e., we linearly penalize whenever \( \max_{\hat{s} \in S} w^\top \phi(x, \hat{s}) + \ell_{(x,s)}(\hat{s}) \geq w^\top \phi(x, s) \) which is achieved with the following function: \( \max\{0, \max_{\hat{s} \in S} w^\top \phi(x, \hat{s}) + \ell_{(x,s)}(\hat{s}) - w^\top \phi(x, s)\} \). Due to \( \ell_{(x,s)}(\hat{s}) \geq 0 \) we note that \( \max_{\hat{s} \in S} w^\top \phi(x, \hat{s}) + \ell_{(x,s)}(\hat{s}) - w^\top \phi(x, s) \geq 0 \). By additionally introducing a regularizer on our parameter vector \( w \), we obtain the following minimization, optimizing a convex regularized surrogate loss function that approximates the discrete
and possibly non-convex loss $\ell$: 

$$
\min_w \frac{C}{p} \|w\|_p^p + \sum_{(x,s) \in \mathcal{D}} \max_{\hat{s} \in \mathcal{S}} \left( w^\top \phi(x, \hat{s}) + \ell(x, s)(\hat{s}) \right) - w^\top d. 
$$

(5.2)

The sum of measurements is denoted by $d = \sum_{(x,s) \in \mathcal{D}} \phi(x, s)$. Note the similarity to the structured support vector machine formulation by Tsochantaridis et al. (2005) and the max-margin Markov networks as detailed by Taskar et al. (2003).

For a more probabilistic derivation we define the loss-augmented likelihood of an estimate $\hat{s}$ given the data sample $(x, s)$ to read as 

$$
p(x, s)(\hat{s} | w) \propto \exp \left( w^\top \phi(x, \hat{s}) + \ell(x, s)(\hat{s}) \right).
$$

The negative log-posterior of a data set $\mathcal{D}$ with independently and identically distributed elements is given by 

$$
- \ln \left( p(w) \prod_{(x,s) \in \mathcal{D}} p(x, s)(s | w) \right).
$$

Importantly we don’t measure the loss-augmented likelihood of an arbitrary estimate $\hat{s}$, but rather the ground truth configuration provided within the data set $\mathcal{D}$. We therefore refer to $p(x, s)(s | w)$ as the loss-augmented data likelihood. Assuming a prior function $p(w) = \exp \left( -\frac{C\beta}{p} \|w\|_p^p \right)$ and plugging in the loss-augmented data likelihood yields the following program:

$$
\min_w \frac{C}{p} \|w\|_p^p + \sum_{(x,s) \in \mathcal{D}} \frac{1}{\beta} \ln \sum_{\hat{s} \in \mathcal{S}} \exp \left( w^\top \phi(x, \hat{s}) + \ell(x, s)(\hat{s}) \right) - w^\top d. 
$$

(5.3)

With $\beta = \infty$ we obtain the max-margin formulation given in Eq. (5.2) while $\beta = 1$ retrieves the maximum likelihood approach of conditional random fields originally proposed by Lafferty et al. (2001).

Note that the program in Eq. (5.3) is equivalently derived by minimizing a Kullback-Leibler (KL) divergence, i.e., by solving $\min_w D_{\text{KL}}(q \mid p(w) \prod_{(x,s) \in \mathcal{D}} p(x, s)(s | w))$. Hence we observe structured prediction to be an M-projection task. More general Bregman divergences could be employed.

On the positive side, the two programs given in Eq. (5.2) and Eq. (5.3) are convex and unconstrained. It is the summation or maximization over the output space $\mathcal{S}$ which imposes potential challenges and renders the tasks intractable in general. For computational tractability we again reside to approximations similar to the ones detailed in Sec. 4.1 and derived by Hazan & Urtasun (2010). In short, we obtain the dual of the program given in Eq. (5.3) to result in a maximization of an entropy over joint distributions (Pietra et al., 2001; Dudík et al., 2004; Dudík & Schapire, 2006).
5. Learning in Structured Models

\[
\begin{align*}
\max_{b(x,s),r \in \Delta} & \quad \sum_{(x,s) \in D,r} \left( \frac{c_r}{\beta} H(b_{(x,s),r}) + \sum_{\hat{s}_r} b_{(x,s),r}(\hat{s}_r) \ell_{(x,s),r}(\hat{s}_r) \right) \\
& \quad - \frac{C^{1-q}}{q} \sum_k \left| \sum_{(x,s),r,\hat{s}_r} b_{(x,s),r}(\hat{s}_r) \phi_{(x,s),k,r}(x,\hat{s}_r) - d_k \right|^q \\
\text{s.t.} & \quad \forall (x, s), r, s_r, p \in P(r) \sum_{s \neq p \setminus s_r} b_p(s_p) = b_r(s_r)
\end{align*}
\]

Figure 5.1.: The approximated dual program of the learning task.

\[
\begin{align*}
\min_w & \quad \sum_{(x,s),r} \frac{c_r}{\beta} \ln \sum_{\hat{s}_r} \exp \frac{-C^{1-q}}{q} \sum_{k} \left| \sum_{(x,s),r,\hat{s}_r} b_{(x,s),r}(\hat{s}_r) \phi_{(x,s),k,r}(x,\hat{s}_r) - d_k \right|^q \\
& \quad - w^T d + \frac{C}{p} \|w\|^p_p
\end{align*}
\]

Figure 5.2.: The approximated learning task.

Towards this goal, we consider the dual problem of Eq. (5.3) which reads as follows

\[
\max_{p(x,s) \in \Delta} \sum_{(x,s) \in D} \left( \frac{1}{\beta} H(p(x,s)) + \sum_{\hat{s}} p(x,s)(\hat{s}) \ell(x,s)(\hat{s}) \right) - \frac{C^{1-q}}{q} \left| \sum_{(x,s),\hat{s}} p(x,s)(\hat{s}) \phi(x,\hat{s}) - d \right|^q
\]

with \(1/q + 1/p = 1\). This dual provides some intuition for why M-projection is also referred to as moment matching. We are interested in a distribution \(p(x,s) \in \Delta\) with the expected feature statistics \(\sum_{(x,s) \in D} E_{p(x,s)}[\phi(x,s)(x,\hat{s})]\) matching our empirical observations \(d\).

Similar to the inference task, we optimize over the output space of a joint distribution for every data sample which is intractable in general. Since we assumed every element of the feature vector to decompose as detailed in Eq. (5.1), we follow the procedure in Chapter 4 and introduce regions \(r\), counting numbers \(c_r\) and beliefs \(b_{(x,s),r}(s_r)\) which are required to be locally consistent. We enforce those marginalization constraints by following a Hasse diagram structure defined via a set of parents \(P(r)\) for every region \(r\). The resulting program is illustrated in Fig. 5.1. For clarity of notation we neglect dependence of the set of parents \(P(r)\) on each sample \((x, s)\).

To leverage the structure of the marginalization constraints we employ Lagrange multipliers \(\lambda_{(x,s),r \rightarrow p}(s_r)\) and transform the dual problem back to an approximated primal,
Algorithm: Convex Structured Prediction
Repeat until convergence

1. Iterate over all samples \((x, s)\) and regions \(r\):
   \[
   \forall p \in P(r), s_r \mu_{(x,s), p \rightarrow r}(s_r) = \frac{c_p}{\beta} \ln \sum_{s_p \backslash s_r} \exp \left( \frac{\tilde{\phi}_{(x,s), p}(s_p) - \sum_{p' \in P(p)} \lambda_{(x,s), p' \rightarrow p}(s_{p'}) + \sum_{r' \in C \setminus r} \lambda_{(x,s), r' \rightarrow r}(s_{r'})}{c_p / \beta} \right)
   \]

2. Weight vector update with stepsize \(\eta\) determined via Armijo iterations for \(p = 2\)
   \[
   w \leftarrow w - \eta \left( \sum_{(x,s), r} b_{(x,s), r}(\hat{s}_r) \phi_{(x,s), r}(\hat{s}_r) - \phi_{(x,s), r}(s_r) \right) + Cw
   \]

Figure 5.3.: A block-coordinate descent algorithm for the learning task.

which is illustrated in Fig. 5.2 where we let \(\tilde{\phi}_{(x,s), r}(\hat{s}_r) = \sum_{k, r \in \mathcal{R}_k} w_k \phi_{(x,s), k, r}(x, \hat{s}_r) + \ell_{(x,s), r}(\hat{s}_r)\) while assuming – just like for inference – that the \(k\)-th feature vector element for every sample decomposes into a sum of regions \(\mathcal{R}_k\), i.e., \(\phi_{(x,s), k} = \sum_{r \in \mathcal{R}_k} \phi_{(x,s), k, r}(x, \hat{s}_r)\) as described in Eq. (5.1).

Counting numbers \(c_r\) approximate the joint soft-max function via a sum of local soft-max functions ranging over small subsets of output space variables \(s_r\). To enforce consistency between the individual subsets of a sample \((x, s)\), messages \(\lambda_{(x,s), r \rightarrow p}(s_r)\) exchange information between a region \(r\) and its parents \(P(r)\).

If \(\beta, c_r \geq 0\) the cost function is convex and convergence is guaranteed when employing a block-coordinate descent approach which iterates between updating the weights \(w\) and the Lagrange multipliers \(\lambda\). Even more importantly and to increase efficiency, convexity permits to interleave updates for the Lagrange multipliers \(\lambda\) and gradient steps w.r.t. the parameters \(w\) (Hazan & Urtasun, 2010).

Following Hazan & Urtasun (2010), the resulting convex structured prediction algorithm is summarized in Fig. 5.3. The first step gives the block-coordinate descent steps for the Lagrange multipliers \(\lambda\), the second part illustrates the gradient update
5. Learning in Structured Models

\[
\max_{b(x,s), r \in \Delta} \sum_{(x,s) \in D, \kappa, r} \left( \hat{c}_r H(b(x,s), r) + \sum_{\hat{s}_r} b_{(x,s), r}(\hat{s}_r) \beta \ln \sum_{\hat{s}_c} \exp \left( \tilde{\phi}(x,s,r|x, \hat{s}_c) - \sum_{p \in P(r)} \lambda_{(x,s), r \rightarrow p}(\hat{s}_r) + \sum_{c \in C(r)} \lambda_{(x,s), c \rightarrow r}(\hat{s}_c) \right) / (c_r / \beta) \right) - \frac{C_1 - q}{q} \sum_k \left| \sum_{(x,s), \kappa, r, \hat{s}_r} b_{(x,s), r}(\hat{s}_r) \phi(x,s,k,r x, \hat{s}_r) - d_k \right|_q
\]

s.t.
\[
\begin{align*}
\forall (x,s), \kappa, r, s_r, p \in P(r) \quad \sum_{s_p \neq s_r} b^{c_p}_p(s_p) &= b^{c_p}_p(s_r) \\
\forall (x,s), \kappa, r, s_r \quad b^{c_r}_r(s_r) &= b_r(s_r)
\end{align*}
\]

Figure 5.4.: The approximated dual program of the distributed learning task.

\[
\min_w \sum_{\kappa, (x,s), r} \hat{c}_r \ln \sum_{\hat{s}_r} \exp \left( \tilde{\phi}(x,s,r|x, \hat{s}_r) - \sum_{p \in P(r)} \lambda_{(x,s), r \rightarrow p}(\hat{s}_r) + \sum_{c \in C(r)} \lambda_{(x,s), c \rightarrow r}(\hat{s}_c) \right) / (c_r / \beta) - w^T d + \frac{C}{p} \|w\|_p
\]

Figure 5.5.: The distributed and approximated learning task.

using beliefs

\[
b_{(x,s), r}(\hat{s}_r) \propto \exp \left( \tilde{\phi}(x,s,r|x, \hat{s}_r) - \sum_{p \in P(r)} \lambda_{(x,s), r \rightarrow p}(\hat{s}_r) + \sum_{c \in C(r)} \lambda_{(x,s), c \rightarrow r}(\hat{s}_c) \right) / (c_r / \beta)
\]

if \(c_r / \beta > 0\). Following the inference schema for \(c_r / \beta = 0\), we choose a uniform distribution over the beliefs nonzero domain which is given by the set of maximizing states.

5.2. Distributed Message Passing Algorithm

5.2.1. Intuition

Before deriving a distributed learning algorithm we note that it is trivially possible to parallelize the common max-margin or maximum likelihood learning task w.r.t. the number of samples. However, this approach does not scale to large graphical models requiring large amounts of memory.

We begin by investigating the approximated dual of the learning task which is given by the program illustrated in Fig. 5.4. Following the procedure detailed in Chapter 4 we
partition the output space $S$ of every sample onto different computers $\kappa$. This imposes a partitioning of the regions $r$ and hence local beliefs $b^\kappa(x, s, r)$ if a region is assigned to machine $\kappa$. Analogously to the distributed inference approach we are required to enforce consistency between regions upon convergence. This consistency constraint is the only coupling between the different computers, hence we decompose the dual into almost separable parts. The resulting program is given in Fig. 5.4. We again distribute the counting numbers, the loss and the features equally between the computers, i.e.,

$$\hat{c}_r = c_r / |M(r)|, \hat{\ell}(x, s, r) = \ell(x, s, r) / |M(r)| \text{ and } \hat{\phi}(x, s, k, r) = \phi(x, s, k, r) / |M(r)|,$$

where the set $M(r)$ denotes the set of machines node $r$ is assigned to.

Following previous ideas we aim at leveraging the structure given within the constraint set of the decomposed dual. We therefore transform the decomposed approximated dual back to the primal domain. The resulting cost function to be optimized is illustrated in Fig. 5.5 where we let

$$\hat{\phi}(x, s, r) = \sum_{k: r \in R_k} w_k \hat{\phi}(x, s, k, r) + \hat{\ell}(x, s, r).$$

The important differences to standard approximate convex learning originally derived by Hazan & Urtasun (2010) and illustrated in Fig. 5.2 are the summation over the computers $\kappa$ and an additional message $\nu(x, s, \kappa \to r(\hat{s}_r))$ to be transferred from machine $\kappa$ to region $r$. As for inference, the additional messages $\nu$ correspond to Lagrange multipliers for the consistency constraints enforcing distributed region beliefs to agree upon convergence.

In a subsequent step we derive a block-coordinate descent algorithm for the distributed approximated primal problem. Neglecting minor modifications, the updates w.r.t. $\lambda$ and $w$ are identical to the non-distributed algorithm. To update the Lagrange multipliers $\nu$ an additional step is introduced and only required to be performed occasionally. Given some messages we compute the beliefs

$$b(x, s, r)(\hat{s}_r) \propto \exp \left( \hat{\phi}(x, s, r) - \sum_{p \in P(r)} \lambda_{(x, s), r \to p}(\hat{s}_r) + \sum_{c \in C(r)} \lambda_{(x, s), c \to r}(\hat{s}_c) + \nu_{(x, s), \kappa \to r}(\hat{s}_r) \right) / (c_r / \beta)$$

if $c_r / \beta > 0$. For $c_r / \beta = 0$ the beliefs nonzero domain is again given by the maximizing states. The complete algorithm is summarized in Fig. 5.6 and we refer the reader to Sec. 4.2.3 and Sec. 5.2.3 for the derivations and statements regarding convergence guarantees.

Convex approximations of the learning task benefit from interleaving the gradient computation and the line-search while maintaining the convergence guarantees. This is also true when dividing the model onto separate resources and it is required to only transfer information between computers occasionally. Note that merging the gradient onto one machine requires usually a lot less information to be transmitted than exchange of factors during message passing since the dimension of the feature
Algorithm: Distributed Convex Structured Prediction

Repeat until convergence

1. For every $\kappa, (x_s)$ in parallel: iterate over $r : \forall p \in P(r), s_r$

$$\mu_{(x,s),p\rightarrow r}(s_r) = \frac{\hat{c}_p}{\beta} \ln \sum_{s_p \setminus s_r} \exp \left( \left( \hat{\phi}_{(x,s),p}(s_p) - \sum_{p' \in P(p)} \lambda_{(x,s),p\rightarrow p'}(s_p') + \sum_{r' \in C(p) \cap \kappa \setminus r} \lambda_{(x,s),r'\rightarrow p}(s_{r'}) + \nu_{(x,s),\kappa\rightarrow p}(s_p) \right) / (\hat{c}_p / \beta) \right)$$

$$\lambda_{(x,s),r\rightarrow p}(s_r) \propto \frac{\hat{c}_p}{\hat{c}_r} + \sum_{p \in P(r)} \frac{\hat{c}_p}{\hat{c}_r} \left( \hat{\phi}_r(s_r) + \sum_{c \in C(r) \setminus \kappa} \lambda_{(x,s),c\rightarrow r}(s_c) + \nu_{(x,s),\kappa\rightarrow r}(s_r) + \sum_{p \in P(r)} \mu_{(x,s),p\rightarrow r}(s_r) \right) - \mu_{(x,s),p\rightarrow r}(s_r)$$

2. Iterate over $r \in \mathcal{G}P : \forall \kappa \in M(r)$

$$\nu_{(x,s),\kappa\rightarrow r}(s_r) = \frac{1}{|M(r)|} \sum_{c \in C(r)} \lambda_{(x,s),c\rightarrow r}(s_c) - \sum_{c \in C(r) \setminus \kappa} \lambda_{(x,s),c\rightarrow r}(s_c) + \sum_{p \in P(r)} \lambda_{(x,s),r\rightarrow p}(s_r) - \frac{1}{|M(r)|} \sum_{\kappa \in M(r), p \in P(r)} \lambda_{(x,s),r\rightarrow p}(s_r)$$

3. Find a stepsize $\eta$ to decrease the cost function (Armijo rule) for $p = 2$

$$w \leftarrow w - \eta \left( \sum_{(x,s),r} b_{(x,s),r}(\hat{s}_r) \phi_{(x,s),r}(\hat{s}_r) - \phi_{(x,s),r}(s_r) + Cw \right) \quad (5.4)$$

Figure 5.6.: A block-coordinate descent algorithm for the distributed learning task.
5.2. Distributed Message Passing Algorithm

vector is usually significantly smaller than the state-space of all the regions divided onto multiple machines.

5.2.2. Experimental Results

To investigate the performance we distribute the learning task onto 16 computers and obtain the parameter vector \( w \) as detailed in Fig. 5.6. For illustration we chose a denoising problem. We are given a 4 bit ground truth image, \( i.e. \), 16 gray scale levels, and ten noisy observations as visualized for one example in Fig. 5.7. Our two dimensional feature vector measures on the one hand the absolute difference of the noisy observation to the 16 gray scale levels. On the other hand we measure the smoothness by employing a truncated linear pairwise potential. We aim at learning a linear combination of those two features, \( i.e. \), \( w \in \mathbb{R}^2 \).

**Frequency of transmission:** Most importantly we investigate the occurring effects when exchanging information between machines at different frequencies for different values of the temperature \( \beta \). The results are illustrated in Fig. 5.8. In our case, depending on \( \beta \), we observe the best convergence speed w.r.t. time when exchanging information between two to ten iterations. Naturally this depends on the considered task, \( i.e. \), the connectivity of the problem and the latency of the network connection. We highlight that rarely exchanging information leads to undesired staircase like convergence. The lower \( \beta \), the more sever, particularly if the algorithm is almost converged. But following the distributed inference, we also emphasize that the proposed distributed learning approach maintains the strictly monotonic decreasing behavior, irrespective of staircase effects.

**Time per iteration:** In a next experiment we measure the time required for a certain number of iterations and illustrate the results in Fig. 5.9. We compare the distributed algorithms exchanging information at different rates to an approach which parallelizes inference on a single eight-core machine. The latter is illustrated with a magenta line.
5. Learning in Structured Models

<table>
<thead>
<tr>
<th></th>
<th>25</th>
<th>10</th>
<th>5</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>∞</td>
<td>15.74</td>
<td>14.84</td>
<td>13.41</td>
<td>10.62</td>
<td>7.92</td>
</tr>
<tr>
<td>100</td>
<td>12.81</td>
<td>14.61</td>
<td>13.44</td>
<td>10.72</td>
<td>8.13</td>
</tr>
<tr>
<td>10</td>
<td>15.17</td>
<td>14.38</td>
<td>13.14</td>
<td>10.59</td>
<td>8.02</td>
</tr>
<tr>
<td>1</td>
<td>15.09</td>
<td>14.02</td>
<td>12.69</td>
<td>9.74</td>
<td>7.14</td>
</tr>
</tbody>
</table>

Table 5.1.: Learning speedup after 200 iterations for different $\beta$ (rows) and information exchange rates (columns).

Analogously to inference we observe significant improvement when distributing the problem onto multiple computers. The factors for the improvement in speed for 100 iterations are provided in Tab. 5.1.

To conclude the experimental section we showed the benefits for the speed of convergence when distributing a given problem onto multiple machines. A notable additional advantage are the distributed resource requirements. Since only part of the task is stored on a machine, memory requirements per computer are lower.

### 5.2.3. Detailed Derivation

The reader may skip this section without missing information required subsequently.

To derive the block coordinate descent algorithm for the learning task we first notice the similarity between the dual of the inference problem given in Eq. (4.11) and the primal of the distributed learning problem given in Fig. 5.5. Replacing the re-parameterized potentials $\hat{\theta}_r(s_r)$ of the distributed inference algorithm illustrated in Fig. 4.5 with $\tilde{\phi}(x,s)_r(x,s_r)$ directly gives the block coordinates descent steps for the Lagrange multipliers $\lambda$ and $\nu$. It therefore only remains to derive the gradient of the cost function given in Fig. 5.5 w.r.t. $w$. Learning hence augments the inference algorithm by iterations for each sample and a gradient descent step to update the weight vector. For convergence guarantees we require the latter line search iterations to make sure that the Armijo rule is satisfied.

**Claim 8** The gradient of the cost function given in Fig. 5.5 w.r.t. the weight vector $w$ reads as stated in Eq. (5.4) (see Fig. 5.6) where

\[
\begin{align*}
  b(x,s)_r(s_r) &\propto \exp \left( \tilde{\phi}(x,s)_r(x,s_r) - \sum_{p \in P(r)} \lambda_{(x,s),r \rightarrow p}(s_r) + \
  \sum_{c \in C(r)} \lambda_{(x,s),c \rightarrow r}(s_c) / (c_r/\beta) \right) \\
\end{align*}
\]

if $c_r/\beta > 0$. For $c_r/\beta = 0$ we let $b(x,s)_r(s_r)$ be a uniform distribution over the
5.2. Distributed Message Passing Algorithm

Figure 5.8.: Primal of the distributed learning algorithm over time with information exchange at different frequencies.

maximizing states, i.e.,

\[ S^*_r = \arg \max_{s_r} \left( \tilde{\phi}(x, s), r(x, s_r) - \sum_{p \in P(r)} \lambda(x, s, r \rightarrow p(s_r)) + \sum_{c \in C(r)} \lambda(x, s, c \rightarrow r(s_c) + \nu(x, s, \kappa \rightarrow r(s_r)) \right). \]

**Proof:** Straightforward computation of the gradient for \( c_r/\beta > 0 \) and Danskin’s Theorem (cf. Bertsekas et al. (2003)) for \( c_r/\beta = 0 \) yields the claim.

For completeness we provide the full algorithm in Fig. 5.6 and state its convergence properties in the following claim.

**Claim 9** The algorithm given in Fig. 5.6 is guaranteed to converge for \( \beta, c_r \geq 0 \) and guaranteed to find the optimum of the cost function given in Fig. 5.5 for strict convexity, i.e., for \( c_r > 0, \beta \neq \infty \).
5. Learning in Structured Models

Figure 5.9.: Comparing the time of non-distributed learning (dotted magenta) with the distributed learning algorithm exchanging information at different frequencies when considering a given number of iterations.

Proof: The proof follows from the requirement of strict convexity for block coordinate descent algorithms (Tseng, 1993).

This concludes the derivation and convergence statements for the block coordinate descent algorithm of the learning task. The entire algorithm is summarized in Fig. 5.6. We refer the reader to the results presented in the experimental Sec. 5.2.2 to observe the monotonic convergence of the derived algorithm.

5.3. Notes on a Globally Convergent Algorithm

Following the previous chapter about inference we highlight that convergence to the optimum of a linear programming relaxation is not guaranteed for $\beta = \infty$ since the block-coordinate descent optimization w.r.t. the Lagrange multipliers can get stuck in local optima due to non-smoothness of the dual function. We can however design an efficient approach based on $\epsilon$-subdifferentials and the conditional gradient algorithm to obtain global convergence. The extension is a straightforward application of Sec. 4.3 to multiple samples within a data set $D$ and a standard gradient step for the parameter vector $w$. For brevity we omit further details.
5.4. Conclusion

Within this chapter we discussed how to distribute learning algorithms onto multiple compute units each having its own memory, \emph{i.e.}, an algorithm for distributed memory architectures. Since frequent access to arbitrary variables is time consuming, we minimize communication overhead using a dual decomposition strategy while maintaining convergence guarantees.
5. Learning in Structured Models
6. 3D Indoor Scene Understanding

In the preceding chapters we discussed methods for inference and learning that are application independent given a suitable and computationally tractable choice for the features $\phi(x, s)$ and the parameterization of the output space $S$. However we also discussed that approximations, e.g., for the entropy, or relaxations are employed to obtain tractability. Despite those computationally necessary adjustments, good practical results are often obtained when employing those methods.

Nonetheless we argue in the following that, occasionally, a specific choice for the features and the output space leads to algorithms that permit construction of globally optimal yet efficient inference procedures. We observe this to be true when restricting ourselves to sub-modular scores $\theta(x, s) = w^\top \phi(x, s)$ $\forall w, x, s$ while being interested in max-margin methods. Efficient dynamic programming algorithms are also available when restricting ourselves to tree-structured decompositions of $w^\top \phi(x, s)$. Contrasting those two directions, we argue for an efficient branch-and-bound approach within this chapter.

More concretely, we investigate a 3D scene understanding application: given a single input image of an indoor scene we are interested in finding a box that best describes the observed room layout. In addition we also extend this application to jointly reason about the 3D room layout and an object within the room being aligned to the layout walls as illustrated in Fig. 6.1. Such an application is useful, e.g., to understand the observed scene in the first place, to more realistically render the scene from a different perspective or to construct floor plans in a fully automatic manner.

Just like other existing approaches for this task, we rely on the Manhattan world assumption, which states that there exist three dominant vanishing points (vp) that are orthogonal. We take advantage of the Manhattan world assumption, and let the object and the room be aligned with the three main dominant orientations. We thus compute vanishing points, and perform joint inference over the remaining degrees of freedom afterwards.

After having reviewed related work in Sec. 6.1 we discuss different parameterizations and their suitability in Sec. 6.2, first, for the layout estimation only, and afterwards for extending the task towards object prediction. This section is intended to highlighted the importance of defining the output space $S$ appropriately.

In Sec. 6.3 we subsequently describe the features that have been commonly employed to the task of 3D indoor scene estimation. This section is intended to emphasize the crucial importance of using or designing informative feature cues.

Afterwards we detail in Sec. 6.4 our globally optimal layout inference procedure which follows a branch-and-bound approach. Then we describe in Sec. 6.5 how to efficiently compute the required energies using an approach we refer to as “Integral Geometry,” such that we obtain a globally optimal yet fast inference procedure. We
6. 3D Indoor Scene Understanding

Figure 6.1.: Image with overlayed ground truth (blue, cyan) and our detection result (red, magenta) as well as the ground truth (GT) floor plan (top right) and our prediction (Ours) (bottom right).

provide some details regarding the learning algorithm in Sec. 6.6 and evaluate the proposed approach in Sec. 6.7.

6.1. Review

Most 3D scene understanding approaches for outdoor scenarios produce mainly qualitative results (Gupta et al., 2010; Hoiem et al., 2008; Saxena et al., 2008). Some notable exceptions are (Geiger et al., 2011; Bao & Savarese, 2011), which rely on short video sequences or uncalibrated image pairs. For self-calibration and metric reconstruction from uncalibrated image sequences we refer the reader to seminal work by Pollefeys (1999). While outdoor scenarios remain fairly unexplored, estimating the 3D layout of indoor scenes has experienced increasing popularity. This is mainly attributed to the fact that indoor scenes behave mostly as ‘Manhattan worlds,’ simplifying the estimation problem. Most monocular approaches approximate the layout of rooms by 3D cuboids (Hedau et al., 2009; Lee et al., 2010; Wang et al., 2010; Hedau et al., 2010; Schwing et al., 2012a; Hedau et al., 2012; Schwing & Urtasun, 2012). An exception is (Lee et al., 2009), which estimates the 3D layout of corridors by sweeping lines and the work by Flint et al. (2011), which advertises the use of dynamic programming.

Early approaches to 3D layout estimation (Hedau et al., 2009; Lee et al., 2010) reduce the complexity of the problem by utilizing a small set of hypothesis which however limits performance. Generative models were explored by del Pero et al. (2012), and inference is performed using Markov Chain Monte Carlo sampling. Wang et al. (2010) parameterized the problem using a Markov random field with only four degrees of freedom. While effective, the employed regions are of higher order, involving up to four random variables. As a consequence they used a very crude discretization which limits performance. In (Schwing et al., 2012a), the potentials typically employed in the literature were shown to be decomposable into pairwise potentials. As a consequence denser parameterizations were used, resulting in significant gains in performance. More recently, Schwing & Urtasun (2012) showed that the global optimum of typical layout scoring functions is obtained by pursuing a branch and bound approach. This results
in provably optimal solutions that are computed in real time on a single core.

A wide variety of 3D object detection approaches make use of 2D appearance models from multiple viewpoints (Schneiderman & Kanade, 2000; Torralba et al., 2007) to obtain a weak form of 3D information (Kushal et al., 2007; Thomas et al., 2006; Hoiem et al., 2007b; Sun et al., 2009). Alternatively, object centered methods utilize parametric models (Glasner et al., 2011; Brooks, 1983; Dickinson et al., 1992; Sun et al., 2010) and deformable part-based approaches (Felzenszwalb et al., 2010) have been adapted to predict 3D cuboids (Gu & Ren, 2010; Pepik et al., 2012; Fidler et al., 2012; Xiang & Savarese, 2012; Hedau et al., 2010). In this chapter we make use of 2D and 3D deformable part-based models in order to jointly estimate the layout as well as the objects present in the scene.

Objects and layout were also combined in (Wang et al., 2010; Lee et al., 2010; Hedau et al., 2010), and used in (Gupta et al., 2011) to predict affordances as well as to investigate the interaction between humans and objects (Delaitre et al., 2012). While Wang et al. (2010) is more concerned about predicting ‘clutter’ rather than actual objects, Lee et al. (2010) proposes to augment the space of layout candidates by a set of possible objects that are chosen to be either present or absent. Since the dimension of the state-space being object-layout candidates increases tremendously, a heuristic optimization with beam search is performed. In (Hedau et al., 2010) the layout prediction is used to guide a 3D object detector.

Unfortunately, most approaches trade the complexity of one of the tasks (i.e., object and layout prediction) by proposing a small set of candidates. As a consequence, the space of hypotheses is not well explored, resulting in suboptimal solutions. In contrast, this thesis proposes a provably exact solution to the joint problem, which reasons about the exponentially many layouts as well as the exponentially many object locations and sizes. Since the complexity is at least five orders of magnitude larger than a standard layout task the problem is much more difficult to solve. The challenges are two-fold: finding an efficient parametrization that permits reasonable inference time while being able to deal with occlusions which arise from object-layout interactions. Towards this goal we make use of object detectors as well as geometric features and show significant improvements over state-of-the-art.

6.2. Parameterization and Scoring Functions

Following Chapter 4 we are interested in inference to predict the most likely output space configuration \( s^* \in \mathcal{S} \) by maximizing a score. More formally considering Eq. (4.1) with \( \beta \to \infty \) yields the following inference task to be considered within this chapter:

\[
\arg \max_{s \in \mathcal{S}} \theta(x, s).
\]

The probably most crucial step for designing algorithms to solve a specific application is the definition of the output space \( \mathcal{S} \). An unfavorable representation of the output space objects \( s \in \mathcal{S} \) may result in intractable or computationally expensive inference. It is therefore advisable to spend a significant amount of time on this challenge, particularly since there is no right or wrong.
6. 3D Indoor Scene Understanding

![Diagram of 3D indoor scene understanding]

Figure 6.2.: Parameterization and score for the layout task

Subsequently, we first discuss how to parameterize the indoor 3D scene understanding task concerned with only predicting a 3D parametric box that describes the room observed in a given image. In a second step we extend the parameterization to also cover the task of reasoning about objects.

6.2.1. Parameterization and Score for Layout Prediction

Two different parameterizations have been proposed for the problem of 3D indoor scene understanding. Both assume that the three dominant vanishing points can be reliably detected a-priori.

In (Hedau et al., 2009; Lee et al., 2010), candidate 3D boxes are generated, and inference is formulated in terms of a single high dimensional discrete random variable. Hence, one state of such a variable denotes one candidate 3D layout and the output space $S$ is not assumed to decompose. This limits significantly the amount of candidate boxes, e.g., only $\approx 1000$ candidates are employed by Hedau et al. (2009).

Wang et al. (2010) showed that given the three dominant vanishing points, only 4 degrees of freedom are necessary to represent the layout, consisting of four rays originating from two distinct vanishing points. We adopt the latter parameterization, and model the problem in terms of four random variables $(s_1, \ldots, s_4) \in S_L$ that correspond to the angles encoding rays that originate from two distinct vanishing points. As illustrated in Fig. 6.2, these rays fully describe the 3D cuboid, defining the layout. We refer to the output space of the layout task via $S_L$ and to a specific layout via $s_L = (s_1, \ldots, s_4) \in S_L$.

Note that $s_1$ refers to rays that are strictly above the line connecting the two horizontal vanishing points $v_{p0}$ and $v_{p2}$, while $s_2$ defines rays strictly below. Similarly, $s_3$ and $s_4$ refer to rays that pass the vanishing point closer to the image center, i.e., $v_{p2}$, only to the left and only to the right respectively. This definition ensures parameterization of only valid layouts. A suitable parameterization is obtained if rays originate from any two distinct vanishing points.
6.2. Parameterization and Scoring Functions

Considering Fig. 6.2, a reasonably dense grid of possible intersections of rays \( r_i, r_j \), \( i \in \{1, 2\}, j \in \{3, 4\} \) requires about \( |S_i| = 25 \) angles, i.e., states for each discrete random variable. For (Hedau et al., 2009; Lee et al., 2010), unary potentials of size \( \prod_{i=1}^{4} |S_i| \) should be considered to define the same hypothesis space. More generally, our parameterization is in fact image dependent, i.e., the dimensionality of the state space adapts to the location of the vanishing points. In essence we ensure a sufficient density of the resulting grid by restricting the size of the image area circumscribed by rays situated next to each other.

Decomposing the state space seems favorable and is our method of choice for this application. We note however that decomposition is not necessarily desirable. If the features \( \phi(x, s) \) to be discussed later on don’t at least partly follow the employed decomposition of the output space, we would not gain anything. More concretely considering the layout estimation, if inference employs features that depend on the four output space variables of our task, i.e., if some features \( \phi \) are a function of all four variables \( \phi(x, s_1, s_2, s_3, s_4) \), the decomposition is likely useless. We detail the exact decomposition suitable for the considered task in Sec. 6.5.

In addition to the above two parameterizations we could consider an estimate for every pixel, i.e., a parameterization that predicts the type of wall for every point in the image. While certainly more flexible, it is also harder to restrict prediction to only physically plausible layout configurations.

To differentiate the parameterized layouts \( s_L \in S_L \), we design a scoring function \( \theta \) being a linear combination of some features \( \phi_{\text{lay}}(x, s) \). Besides parameterizing the output space properly, it’s the features and their decomposition that ensures success of an application. We measure the fitness of a layout estimate \( \hat{s}_L \) via a scoring function \( \theta \), summing weighted counts \( \phi_{\text{lay}, \alpha}(x, \hat{s}_L) \) over each possible face \( \alpha \in F = \{1, \ldots, 5\} \) denoting, floor, front-wall, right-wall, left-wall and ceiling. We hence sum the content for each face of the hypothesized layout configuration. The content we integrate is described more precisely in Sec. 6.3 and we refer to the counts via \( \phi_{\text{lay}, \alpha} \) for now. More formally,

\[
\theta_{\text{layout}}(x, \hat{s}_L) = w^\top_{\text{lay}} \phi_{\text{lay}}(x, \hat{s}_L) = \sum_{\alpha \in F} w^\top_{\text{lay}, \alpha} \phi_{\text{lay}, \alpha}(x, \hat{s}_L).
\]

6.2.2. Parameterization and Score for Layout and Object Prediction

In a next step we extend the four variable parameterization of the layout to also include an object within the scene.

We are interested in predicting the layout of the room as well as the objects present in the scene given monocular imagery. We mainly focus on predicting a single object, and search over all possible 3D object locations and sizes. Following existing approaches (Hedau et al., 2012, 2010; Lee et al., 2010; Schwing et al., 2012a), we constrain the object to be aligned with the main dominant orientations. We advocate for a joint approach, as we would like to exploit the relationships that exist between the layout and object prediction tasks. The main challenges to solve are dealing with the complexity of the search space as well as handling occlusions properly. Towards this goal, we
6. 3D Indoor Scene Understanding

Figure 6.3.: Jointly inferring room layout and 3D object with occlusion reasoning. The parameterization is indicated in (a) while the object and the pure layout evidence are illustrated in (b) and (c) respectively.

extend a branch and bound method, which is guaranteed to find the global optimum of the score representing the joint problem. We also develop a greedy approach, which produces accurate estimates very fast.

To predict an object we augmenting the four variable parametrization of the layout proposed originally by Wang et al. (2010). In the case of an object, given the vanishing points, 5 degrees of freedom are necessary, consisting of three rays originating from one vanishing point and two rays from another. We refer the reader to Fig. 6.3(a) for an illustration of the parameterization. We thus augment the layout variables \( s_L \in S_L \) by the object space \( s_O = (s_5, \ldots, s_9) \in S_O \) to be a product space with four and five factors respectively. Again, other parameterizations are possible but we found the aforementioned minimal parameterization to be sufficient.

Given an image \( x \), we are interested in predicting the layout \( s_L \in S_L \) as well as the object \( s_O \in S_O \) present in the scene. As image evidence, we exploit both top-down (class-specific) features in the form of 2D and 3D object detectors, as well as bottom-up (class independent) geometric features as detailed in Sec. 6.3.

To differentiate the output space configurations \( s \in S_L \times S_O \), we define the score of a joint configuration \( s \) as the sum of layout and object scores. These scores encode how well the layout and object estimates represent the image evidence. An additional term \( \theta_{pen}(x, s_L, s_O) \) makes sure that objects cannot penetrate walls, and an occam razor term \( \theta_{occam}(x, s_O) \) encodes the fact that we prefer simple explanations. This is necessary in order to handle rooms that do not contain objects. We thus have

\[
\theta_{\text{total}}(x, s_L, s_O) = \theta_{\text{lay-occ}}(x, s_L, s_O) + \theta_{\text{object}}(x, s_O) + \theta_{\text{pen}}(x, s_L, s_O) + \theta_{\text{occam}}(s_O).
\]

Note that the score of the layout depends on the 3D location and size of the object. This is due to the fact that the layout should only explain the image evidence that has not yet been explained by the object, as the object occludes the layout which is illustrated in Fig. 6.3. These occlusions make the problem computationally challenging, as the score \( \theta_{\text{lay-occ}} \) depends a priori on a large set of random variables.
6.2. Parameterization and Scoring Functions

Figure 6.4.: (a) Front face of an object is occluding the floor (blue color in (b)). Decomposition of the occluding area into a larger triangle in (c) and two triangles to be subtracted (d). Decomposition of the triangle in (c) into two positive parts (e) and (f) and a negative part (g) all depending on only two angles illustrating the generalization of integral geometry to triangular shapes.

Object Score: We define an additive score which decomposes over the faces of the object, summing the evidence inside each object face $\gamma \in \{1, \ldots, 4\}$ as illustrated in Fig. 6.3(b).

$$\theta_{object}(x, s_O) = \sum_{\gamma=1}^{4} \theta_{object, \gamma}(x, s_O) = \sum_{\gamma=1}^{4} w_{obj, \gamma}^T \phi_{obj, \gamma}(x, s_O)$$

Assuming that the object is situated on the floor, there are only 4 possible faces $\gamma$ that can be visible (i.e., top, front, left, right). Furthermore, for a given image only a maximum of three faces are actually visible. We define the features for each face to be weighted counts of image cues inside that face, as this will allow us to compute bounds in constant time as discussed in Sec. 6.5.

Layout Occlusion Score: The layout occlusion score $\theta_{lay-occ}$ is defined as

$$\theta_{lay-occ}(x, s_L, s_O) = \theta_{layout}(x, s_L) - \theta_{occ}(x, s_L, s_O)$$

where the last term discounts the image evidence which is already explained by the object, i.e., the pixels for each layout face $\alpha \in \mathcal{F}$ that are occluded by the object (see
Fig. 6.3(c)). We define features for each face $\alpha$ of the layout and object occlusion as weighted counts, i.e.,

$$
\theta_{\text{layout}}(x, s_L) = \sum_{\alpha \in F} w_{\text{lay}, \alpha}^\top \phi_{\text{lay}, \alpha}(x, s_L),
$$

$$
\theta_{\text{occ}}(x, s_L, s_O) = \sum_{\alpha \in F} w_{\text{lay}, \alpha}^\top \left( \sum_{\gamma=1}^{4} \phi_{\text{occ}, \alpha, \gamma}(x, s_L, s_O) \right).
$$

Note that we have shared the weights $w_{\text{lay}, \alpha}$ between the layout score $\theta_{\text{layout}}(x, s_L)$ and the occlusion term $\theta_{\text{occ}}(x, s_L, s_O)$ to properly represent occlusions. Fig. 6.4 (a), (b) provides the details in case $\alpha$ represents the floor and $\gamma$ denotes the front face of the object. The area covered by blue color in Fig. 6.4(b) highlights the floor pixels occluded by the object’s front face.

**Penetration Score:** This score makes sure that the object cannot penetrate the walls defined by the layout, i.e., it equals 0 whenever the object is inside the layout and is $-\infty$ in the case of penetration. Since inference aims at maximizing the score, we prevent the algorithm from retrieving configurations where objects are cut by walls defined by the layout. Note that due to discretization artifacts we occasionally observe objects to slightly interact with walls.

**Occam razor:** Given an image, we do not know a priori if there is an object in the scene. To prevent the model to always put an object, we introduce a fixed penalty for solutions that contain an object. In practice we set the penalty to be 10% of the layout score for the best configuration.

### 6.3. Features

As image evidence, we exploit both top-down (class-specific) features in the form of 2D and 3D object detectors, as well as bottom-up (class independent) geometric features. The geometric cues we employ are orientation maps (OM) (Lee et al., 2009) and geometric context (GC) (Hedau et al., 2009), as they were shown to produce impressive results on the layout task considered, e.g., by Lee et al. (2010); Schwing et al. (2012a); Schwing & Urtasun (2012). In addition we employ object detectors based on the deformable part-based model (Felzenszwalb et al., 2010).

#### 6.3.1. Orientation Maps

Given edges detected in the image, orientation maps (OMs) (Lee et al., 2009) estimate a normal orientation for each pixel by sweeping pairs of lines. Using the vanishing point configuration we can convert these normals into wall estimates, resulting in a five-dimensional feature for each pixel as illustrated for two images in Fig. 6.5(a). The colors refer to different walls that orientation maps assigns to a particular pixel. Given
those image cues, we already have a pretty good idea about what the room might look like. The black color indicates that no estimate is available for this pixel.

### 6.3.2. Geometric Context

Geometric context (GC) features (Hoiem et al., 2007a; Hedau et al., 2009) are six-dimensional features that utilize classifiers to predict the probability of each wall as well as clutter. More precisely, superpixels are computed and a boosting classifier reasons about the six possible labels being the five wall types and an object class in addition. The wall estimates for every pixel are visualized in Fig. 6.5(b) using the same color code as the orientation maps while the black color refers to the object class.

### 6.3.3. Object Features

Since the object class of geometric context features does not disambiguate the different object faces, we additionally consider the 3D object detector of Fidler et al. (2012). It provides us with four values per pixel representing the likelihood of belonging to one of the four possible object faces. We also extended the deformable part-based model (Felzenszwalb et al., 2010) to be supervised in terms of viewpoint, which makes up for one additional feature that represents the probability of a pixel belonging to an object. These are computed via soft-masks estimated from training data for each component.
6. 3D Indoor Scene Understanding

**Algorithm 2** branch and bound (BB) inference

put pair \((\bar{\theta}(S), S)\) into queue and set \(\hat{S} = S\)

repeat

split \(\hat{S} = \hat{S}_1 \times \hat{S}_2\) with \(\hat{S}_1 \cap \hat{S}_2 = \emptyset\)

put pair \((\bar{\theta}(\hat{S}_1), \hat{S}_1)\) into priority queue

put pair \((\bar{\theta}(\hat{S}_2), \hat{S}_2)\) into priority queue

retrieve \(\hat{S}\) having highest score

until \(|\hat{S}| = 1\)

6.4. Globally Optimal Inference

Having considered the parameterization of the output space \(S_L, S_O\) and the features \(\phi\), we subsequently tackle the problem of predicting the room layout \(s_L \in S_L\) of indoor scenes from a single image \(x \in \mathcal{X}\).

6.4.1. Layout Inference

The layout is commonly represented in terms of the spatial configuration of the faces of a rectangular 3D cuboid, \(i.e.,\) left, front and right wall as well as floor and ceiling. This problem is complex, as typical scenes contain objects that partly occlude the walls. In order to simplify the inference process, following existing approaches (Hedau et al., 2009; Lee et al., 2010; Wang et al., 2010; Schwing et al., 2012a) we rely on vanishing point detection and the Manhattan world properties of man-made indoor scenes.

To obtain an accurate prediction \(s_L^*\) given an image \(x\) we need to solve the following inference task:

\[
\begin{align*}
\begin{array}{rl}
s_L^* & = \arg\max_{s_L \in S_L} w_{lay}^T \phi_{lay}(x, s_L).
\end{array}
\end{align*}
\]

(6.1)

Given the inference procedure discussed subsequently, we obtain the model parameters \(w\), commonly referred to as weights with structured prediction learning algorithms such as max-margin Markov networks (Taskar et al., 2003), structured support vector machines (Tsochantaridis et al., 2005), conditional random fields (Lafferty et al., 2001) or approximate structured prediction (Hazan & Urtasun, 2010).

As discussed before and following Lee et al. (2010), we employ geometric context (GC) (Hedau et al., 2009) and orientation maps (OMs) (Lee et al., 2009) as image information to construct the feature vector \(\phi_{lay}(x, s_L)\). For a subset of the pixels, orientation maps provide a label corresponding to one of the five faces \(\alpha\) of the 3D cuboid that are potentially visible in the image. Geometric context (GC) (Hedau et al., 2009), on the other hand, provides for every pixel the probability that this pixel belongs to each surface label including objects in addition to the five labels in \(\mathcal{F}\). Given these image features, potentials are constructed by counting for each face each feature.
6.4. Globally Optimal Inference

(a) Front Wall. (b) Minimal left wall. (c) Maximal left wall.

Figure 6.6.: (a) illustrates the max and min of the front wall in red and black respectively. (b) magenta colors the minimally possible left wall within the set of layouts bounded from below and above by the black and red rays. (c) provides the maximal left wall.

type. We thus define $\phi(x, s_L)$ as a sum of potentials

$$w_{lay}^T \phi_{lay}(x, s_L) = \sum_{\alpha \in F} w_{lay,o,\alpha}^T \phi_{lay,o,\alpha}(x, s_L, \alpha) + \sum_{\alpha \in F} w_{lay,g,\alpha}^T \phi_{lay,g,\alpha}(x, s_L, \alpha),$$

(6.2)

where the subscripts $o$ and $g$ denote OM and GC image cues respectively. Note that the vectors $w_{lay,o,\alpha}$ and $w_{lay,g,\alpha}$ consist of 5 elements $w_{lay,o,\alpha,k} \in \mathbb{R}$ with $k \in \{1, \ldots, 5\}$ for each face $\alpha$ in the case of orientation maps and 6 elements $w_{lay,g,\alpha,k} \in \mathbb{R}$ with $k \in \{1, \ldots, 6\}$ per face $\alpha$ for geometric context.

We now describe how branch and bound is employed for our problem. We start with a trivial set (i.e., all possible layouts $S_L$), and at any given branch and bound iteration we have a priority queue where the considered sets are ordered in terms of a quality bound function. This function upper bounds the maximum score that any layout member of that set can possibly achieve. The best candidate $\hat{S}_L$ of the layout sets within the queue is considered. If it is a single layout, i.e., if $|\hat{S}_L| = 1$ and consequently $s^*_L = \hat{S}_L$, we have obtained the optimum. If it is a set of layouts, we split the set into two disjoint candidate sets $\hat{S}_{L,1}$ and $\hat{S}_{L,2}$. New bounds for the scores of those two sets are computed and denoted by $\bar{\theta}(\hat{S}_{L,i})$ with $i \in \{1, 2\}$, and both candidate sets are included into the priority queue. As the bound is more tight (smaller sets), it may be that none of these candidates will be on top of the priority queue. The algorithm terminates when a single hypothesis is returned, and such a hypothesis is guaranteed to be optimal. The beauty of branch and bound is that it does not explore regions which are not promising, allowing for efficient exact inference. We refer the reader to Alg. 2 for a schematic and general illustration referring to the output space via $\mathcal{S}$.

In order to apply branch and bound to the layout estimation task, we have to define a parameterization of the sets $\hat{S}_L$ as well as a bound $\bar{\theta}(\cdot)$ for the inference function of interest being $w_{lay}^T \phi_{lay}(x, s_L)$. We parameterize the set of hypothesis in terms of
intervals of candidate 2D ray intersections. Let \( \hat{S}_L = \{ S_{L,1} \cdot S_{L,2} \cdot S_{L,3} \cdot S_{L,4} \} \) denote a set of candidate layouts defined by the product space of intervals \( S_{L,i} = [s_{i,min}, s_{i,max}] \). Note that unlike defined by Lampert et al. (2009), these intervals do not represent axis aligned image regions, but regions that are in accordance with the vanishing points. This is illustrated for an arbitrary \( \hat{S}_L \) in Fig. 6.6(a) where the black rays indicate the smallest \( s_i \) in the interval, i.e., \( s_{i,min} \), and the red rays are drawn according to the biggest \( s_i \) in the intervals, i.e., \( s_{i,max} \).

Given the parameterization of the sets, we still require bounds \( \bar{\theta} \) for the original layout scoring function \( \theta_{layout} = w_{lay}^\top \phi_{lay}(x, s_L) \). In order for branch and bound to recover the exact solution, we need our bounds to satisfy the following two properties:

1. The bound of the interval \( \hat{S}_L \) has to upper-bound the true cost of each hypothesis \( s_L \in \hat{S}_L \), i.e., \( \forall s_L \in \hat{S}_L, \bar{\theta}(\hat{S}_L) \geq w_{lay}^\top \phi_{lay}(x, s_L) \).

2. The bound has to be exact for every single hypothesis, i.e., \( \forall s_L \in SL, \bar{\theta}(s_L) = w_{lay}^\top \phi_{lay}(x, s_L) \).

Going back to our task: as our features \( \phi_{lay,i,\alpha}^k(x, s_L, \alpha) \) for \( i \in \{ o, g \} \) are always positive (i.e., they represent counts), we can split the potentials in Eq. (6.2) into those with strictly positive weights and those with weights less or equal to zero as follows:

\[
w_{lay}^\top \phi_{lay}(x, s_L) = \sum_{\{(i, \alpha, k) : i \in \{ o, g \}, \alpha \in \mathcal{F}, w_{lay,i,\alpha}^k > 0 \}} w_{lay,i,\alpha}^k \phi_{lay,i,\alpha}^k(x, s_L, \alpha) + \sum_{\{(i, \alpha, k) : i \in \{ o, g \}, \alpha \in \mathcal{F}, w_{lay,i,\alpha}^k \leq 0 \}} w_{lay,i,\alpha}^k \phi_{lay,i,\alpha}^k(x, s_L, \alpha).
\]

Recall that \( \mathcal{F} = \{ 1, \ldots, 5 \} \) refers to the five layout faces. We can thus collapse these potentials into two functions, one that is strictly positive and one that is zero or negative by defining

\[
\theta_{lay}^+(x, s_L) = \sum_{\{(i, \alpha, k) : i \in \{ o, g \}, \alpha \in \mathcal{F}, w_{lay,i,\alpha}^k > 0 \}} w_{lay,i,\alpha}^k \phi_{lay,i,\alpha}^k(x, s_L, \alpha),
\]

\[
\theta_{lay}^-(x, s_L) = \sum_{\{(i, \alpha, k) : i \in \{ o, g \}, \alpha \in \mathcal{F}, w_{lay,i,\alpha}^k \leq 0 \}} w_{lay,i,\alpha}^k \phi_{lay,i,\alpha}^k(x, s_L, \alpha).
\]

Inference is hence equivalently stated via the problem

\[
s_L^* = \arg\max_{s_L \in S_L} \theta_{lay}^+(x, s_L) + \theta_{lay}^-(x, s_L),
\]

where we have used the fact that \( w_{lay}^\top \phi_{lay}(x, s_L) = \theta_{lay}^+(x, s_L) + \theta_{lay}^-(x, s_L) \).

Note that in the efficient subwindow search (ESS) of Lampert et al. (2009), the bounding box scoring function was also decomposed into the sum of negative and positive terms. The bounds were constructed by summing all positive terms within
6.4. Globally Optimal Inference

the maximally possible rectangle of the considered interval product space, \(i.e.,\) the union of all possible members, while adding the negative terms within the minimally required rectangle, \(i.e.,\) the intersection of all possible set members. This approach is suitable if the quality function depends only on contributions from within the rectangle. With our cost function being defined on the entire image we have to find another bounding strategy. Moreover, our faces are not axis-aligned but more general convex quadrilaterals.

As our functions \(\theta^+\) and \(\theta^-\) naturally decompose into a weighted sum over the different faces of the layout, we construct bounds by answering the question of what is the maximum positive contribution and minimum negative contribution of the score function within the set of layout candidates \(\hat{S}_L\) for each face \(\alpha \in \mathcal{F} = \{1, \ldots, 5\}\). The answer is simple, we need to bound each face separately by considering the minimum and maximum area that each face can take in the set. This is illustrated in Fig. 6.6(a) for the front wall trivially having a minimal contribution when taking \(s_{i,\text{min}} \forall i \in \{1, \ldots, 4\}\) and a maximal contribution when taking \(s_{i,\text{max}} \forall i \in \{1, \ldots, 4\}\). Hence we bound the contribution of the front face using

\[
\bar{\theta}_\text{lay,front-wall}(\hat{S}_L) = \theta^+_{\text{lay,front-wall}}(x, s_{\text{max}}) + \theta^-_{\text{lay,front-wall}}(x, s_{\text{min}}),
\]

where we used the subscript to restrict summation over the faces \(\alpha\) within \(\theta^\pm\) to the indicated set being in this case the front-wall. Let \(s_{\text{max}}\) and \(s_{\text{min}}\) be the 4-tuple \((s_{i,\text{max}})_{i=1}^4\) and \((s_{i,\text{min}})_{i=1}^4\). For the remaining four faces \(\alpha \in \mathcal{F} \setminus \text{front-wall}\) we need combinations of upper and lower bounding angles \(s_{i,\text{min}}\) and \(s_{i,\text{max}}\) to construct a valid function \(\bar{\theta}_{\text{lay,}\alpha}(\hat{S}_L)\) that fulfills previously mentioned properties. We illustrate such a combination of angles for a minimal and maximal contribution of the left wall in Fig. 6.6(b) and Fig. 6.6(c). Altogether the function for scoring sets of layouts \(\hat{S}_L\) is given by

\[
\bar{\theta}_\text{lay}(\hat{S}_L) = \sum_{\alpha \in \mathcal{F}} \bar{\theta}_{\text{lay,}\alpha}(\hat{S}_L).
\]

In order for our branch and bound inference to be practical, we need to be able to compute the bounds very efficiently. The bounds employed in Lampert et al. (2009) were efficiently computed using integral images. However, integral images are not applicable to our case, as we reason about 3D faces. Instead, we can use integral geometry (Schwing et al., 2012a) in order to compute these bounds in constant time as described in Sec. 6.5.

6.4.2. Layout and Object Inference

The scoring function is somewhat more complex when also considering objects. During inference we are again interested in computing the maximum a-posteriori (MAP) estimate of the joint problem defined as

\[
\arg \max_{s_L, s_O} \theta_{\text{total}}(x, s_L, s_O).
\]
Finding a global maximizer of the layout task, \( \theta_{\text{layout}}(x, s_L) \), is possible using branch and bound (Schwing & Urtasun, 2012). Subsequently, we generalize this approach to solve the joint layout and object problem with an explicit occlusion reasoning.

The branch and bound inference algorithm we employ follows our earlier description and is formally detailed in Alg. 2. We now operate on hypothesis sets \( \hat{S} \subseteq S_L \times S_O \) containing a multiplicity of different object-layout configurations, and start with a single interval being the full hypothesis set. As before we proceed iteratively, taking the most promising set on a priority queue at each iteration. If this set contains multiple hypotheses, the set is divided into two disjoint subsets \( \hat{S}_1 \) and \( \hat{S}_2 \) (i.e., \( \hat{S}_1 \cap \hat{S}_1 = \emptyset \) and \( \hat{S} = \hat{S}_1 \cup \hat{S}_2 \)). For each one we compute a lower-bound and insert the pair of score and set into the priority queue, ordered by the quality of the bound. The algorithm terminates when the element on top of the priority queue consists of a single hypothesis. In the worst case this algorithm investigates an exponential number of hypotheses, but if the bounds are tight, typically only a small fraction needs to be considered. As before, in order to return a global optimum, the bounds have to be true upper bounds for all the elements in the sets, and the bounds have to be exact in case a single hypothesis is evaluated. The bounds developed here satisfy these two properties, and thus we retrieve the global optimum of the joint problem.

In order to utilize branch and bound, we need to parametrize sets of hypotheses, and derive bounds which are both efficient to compute and tight. We parameterize sets of hypotheses by intervals of the form \( [s_{1,\text{min}}, s_{1,\text{max}}] \times \cdots \times [s_{9,\text{min}}, s_{9,\text{max}}] \), as such a parameterization simplifies our bounding functions. To keep the complexity level reasonable, we discretize the possible angles, having on average 18.4 states per layout variable and 28.1 states per object parameter. The variability in the number of states is due to the vanishing point locations.

Next we define valid bounds. As the score is a sum of terms, we bound each element separately and compute the final bound by summing the individual ones. It is easy to see that this is a valid bound. While bounding the objects is a straightforward
extension of Schwing & Urtasun (2012), bounding the occlusion term is much more cumbersome. We do not require to bound the penetration score as we can simply carve the space to consider only objects which are contained within the layout. As far as the occam razor potential is concerned, we add to the priority queue the best layout configuration found in the absence of an object with bound equal to its score plus the penalty.

**Layout bounds:** For the layout, we utilize the lower bounds described by Schwing & Urtasun (2012), which are obtained by dividing the layout scoring function into two parts, one containing positive weights and one containing negative weights as before:

$$\theta_{\text{layout}}(x, s_L) = \theta_{\text{lay}}^+(x, s_L) + \theta_{\text{lay}}^-(x, s_L)$$

Upper bounds are easily estimated by summing the biggest possible face for the positive features and subtract the smallest possible face for the negative ones. The bound for the right layout face is illustrated in Fig. 6.7(a), where the leftmost ray is depicted in green and the rightmost one in red, and the maximally possible right face area is colored in blue while the minimally possible right wall is highlighted in green. We emphasize again that computing the content of maximal and minimal faces depends on the four intervals for the front face and on three intervals for all other walls, floor and ceiling. Using integral geometry (Schwing et al., 2012a) as detailed in Sec. 6.5, we are able to decompose those functions into sums of accumulators that depend on at most two random variables. This allows computation of bounds in constant time while being memory efficient as well.

**Object bounds:** Object faces are amenable to a similar strategy. We split the score into negative and positive components, and bound the counts using the minimally and maximally possible faces. This is illustrated for the object’s top face in Fig. 6.7(b) with green and blue rays denoting the leftmost and rightmost rays. All object faces naively depend on four intervals, but, similar to the layout bounds, we can utilize integral geometry (Schwing et al., 2012a). By decomposing the faces into sums of pairwise accumulators, we compute the bounds in constant time and are again memory efficient.

**Occlusion bounds:** Also the occlusion bounds can be computed by dividing the score into negative and positive components while counting the content of minimally and maximally possible regions. To this end it is necessary to employ triangular shapes. Recall the blue highlighted area already explained by the object estimate in Fig. 6.4(b). To computing the content subsumed by the floor face that has not been explained by an object, we subtract the blue region referred to as $\theta_{\text{occ}}$. To compute this region we decompose it into the triangular shapes illustrated in Fig. 6.4(c) and (d).

**Improving speed**

**Carving:** Dividing an interval imposes new constraints that can be used to improve efficiency, e.g., the ray describing the top edge of the front face is required to be above
6.3D Indoor Scene Understanding

(a) Parameterization of the 3D layout estimation task.
(b) Integral geometry for the left-wall.
(c) Using accumulators for faces.

Figure 6.8.: Our problem formulation in terms of four random variables $y_i$ is illustrated in (a). Decomposing the third order potential for $\alpha = \text{left-wall}$ into two shaded second order potentials is shown in (b). A schematic on how “integral geometry” uses accumulators $A$ is given in (c).

the ray describing the bottom edge of that face. To avoid those intervals we carve out spaces that are physically impossible. Importantly, global optimality is not violated since the regions not considered are guaranteed to be scored with $-\infty$.

Greedy approach: We also illustrate a greedy strategy that speeds up computation by reducing the search space. To this end we first optimize $\theta_{\text{total}}$ w.r.t. $s_L \in S_L$ while fixing the object $s_O$ to remain outside the image. Intuitively we explain the scene without considering objects. In a second step we fix the previously obtained layout prediction $s_L^*$ and optimize $\theta_{\text{total}}$ w.r.t. $s_O \in S_O$. While this is not guaranteed to yield a global optimum, our experiments show that it results in performance, similar to the joint model when employing object detectors. This is expected as those make the two tasks more independent. In the case of only employing GCs and OMIs, our features contain only 5 labels and are hence ambiguous (both problems are tightly coupled). Thus, the greedy approach is significantly worst in this setting.

6.5. Efficient Score Computation via Integral Geometry

It remains to answer how to construct the potentials $\phi$ for accurate and efficient prediction. The complexity of the structured prediction problem depends on the order of the potentials involved, and its size, i.e., the number of states. Without objects, the methods presented by Hedau et al. (2009); Lee et al. (2010) operate with only a single variable and hence inevitably many states. When not considering the clutter denoted by hidden variables, Wang et al. (2010) employ functions of order up to four, i.e., functions depending on four variables.

Considering Fig. 6.8(a), a reasonably dense grid of possible intersections of rays $r_i; r_j, i \in \{1, 2\}, j \in \{3, 4\}$ requires about $|S_i| \approx 25$ angles per ray, i.e., states for each discrete random variable. With the cardinality of the variables being 25, the size of
6.5. Efficient Score Computation via Integral Geometry

![Graphs](a) BB 0.01s, cBP 0.45s (b) BB 0.01s, cBP 0.15s

Figure 6.9.: We illustrate the run time required to obtain a fraction of completed test images on the layout data set for the proposed inference approach (BB) and a standard message passing algorithm (cBP). The results are averaged over a large set of parameters $C$ for the two different models SSVM in (a) and approx in (b). The average times are indicated below the figures.

...those forth order potentials amounts to $|S| \approx 25^4$. Similarly, for (Hedau et al., 2009; Lee et al., 2010), functions of domain size $25^4$ should be considered to define the same hypothesis space. Both methods are computationally demanding and do not scale well in the number of states. The problem is even more severe when considering clutter or when reasoning about objects (Lee et al., 2010; Wang et al., 2010). In (Lee et al., 2010) all objects are ideally connected to each other, i.e., the former unary potential of size $25^4$ is now augmented by as many binary variables as object hypotheses are present in a scene. Hence the potential consists of $25^4 \cdot 2 \# \text{Objects}$ values. To tractably deal with this space, existing approaches use either fewer states (Hedau et al., 2009), and hence suffer from discretization artifacts, or introduce ad-hoc approximations for learning and inference (Lee et al., 2010; Wang et al., 2010).

In the remainder of the section, we show how the functions employed in literature can be decomposed into pairwise potentials by extending the concept of integral images to integral geometry.

Recall that our features are counts representing the content of a hypothesized wall or object face. Parameterization of a specific wall depends on four variables for the front wall and three variables for all the other ones. Since our features are linear we can however decompose computation of the content into a difference of two terms, each depending on only two variables. This concept is illustrated in Fig. 6.8(b) for the example of a left wall. The difference of the blue and green shaded area retrieves the content of the left wall with both areas depending on the state of two rays only.

Integral images perform partial computations in accumulators such that the generation of image features at different locations and scales can be performed efficiently by a few accesses to these accumulators (Viola & Jones, 2001). In the spirit of integral...
images, as shown in Fig. 6.8(c), we construct 2D accumulators, each counting features (probabilities) in regions of the space defined by two rays originating from two different vanishing points. Suppose we want to compute a potential defined as counts in the gray shaded area. We can easily obtain this potential, by treating each cell as a pixel (by counting the contribution in the cell) and applying integral images to those cells. This is the concept of integral geometry introduced by Schwing et al. (2012a). Thus a function defined in the shaded area in Fig. 6.8(c) is computed by adding the integral geometry cells to the top left of the red dotted corners while subtracting the ones to the top left of the blue dotted intersections.

The occlusion terms can be decomposed in a similar manner. To effectively compute $\theta_{\text{occ}}(x,s_L,s_O)$ we decompose the score into sums over triangular faces. This is illustrated in Fig. 6.4 for the case of the front face of an object occluding the floor. The occlusion is highlight with blue color in Fig. 6.4(b). We decompose the occlusion region, into the sum over three triangular shapes, i.e., from the red triangle in Fig. 6.4(c) we subtract the black and purple triangles in Fig. 6.4(d). More generally, a fourth positively counted triangle with its cathetus intersecting at an existing upper left corner is additionally required. While we have illustrated this decomposition with an example, all overlaps between object faces and layout walls are decomposed and computed in a similar manner. Furthermore, for each triangle, we compute the counts inside by again decomposing the computation into the sum of three accumulators. This is demonstrated in Fig. 6.4(e) – (g), where the pink and cyan areas are counted positively while the yellow area is subtracted. These accumulators are pairwise potentials, as each of the highlighted areas depends on only two angles. To compute the

Figure 6.10.: We illustrate the run time required to obtain a fraction of completed test images on the bedroom data set for the proposed inference approach (BB) and a standard message passing algorithm (cBP). The results are averaged over a large set of parameters $C$ for the two different models SSVM in (a) and approx in (b). The average times are indicated below the figures.
Table 6.1.: We compare the inference run time of standard message passing (cBP) and our approach (BB) averaged over a large set of parameters $C$ in all test instances of the (a) layout and (b) bedroom dataset for models learned with SSVM and approx.

required bounds we split the potentials into negative and positive terms and bound each with either its minimal or maximal face depending on the sign. This procedure again provides bounds computable in constant time.

6.6. Learning

The score of the joint problem depends linearly on the parameters $w$. Since we constructed an algorithm to provably retrieve the global optimizer of our scoring function, we utilize max-margin Markov networks or structured support vector machines (Taskar et al., 2003; Tsochantaridis et al., 2005) to learn the weights. Towards this goal, we designed a parallel algorithm that exploits multiple machines. The details of our cutting plane implementation are listed more precisely in Appendix A.4.

6.7. Experimental Evaluation

6.7.1. Layout

We first investigate the efficiency of our method and then illustrate the state-of-the-art performance obtained by our approach via exact inference. Importantly, we are able not only to do exact inference, but also to do it faster than any other approximate inference technique. All evaluations are carried out on two different data sets commonly used in the literature. The layout data set (Hedau et al., 2009) contains 314 images with ground truth annotation of faces, i.e., left, right and front wall as well as ceiling and floor. We employed the vanishing point detector of (Hedau et al., 2009), which failed in 9 training images and was successful for all test images, 105 in total.
addition, we evaluate our method on the bedroom data set (Hedau et al., 2010) which contains 309 labeled images. The data is split into training and test sets of size 181 and 128 respectively. In accordance with previous work on those data sets, we use a pixel based error measure, counting the percentage of pixel that disagree with the provided ground truth labeling. Note that the ground truth labeling is not necessarily aligned with detected vanishing points.

**Efficiency:** We trained our models to learn the parameters $w$ using a cutting plane algorithm to solve the structured support vector machine objective, denoted “SSVM,” and an approximation, denoted “approx,” which makes use of the same implementation but computes the cutting plane updates only approximately via convex belief propagation. Throughout this submission the stopping criteria for the message passing inference (i.e., our baseline) was set to a relative duality gap of 1e-5 or a maximum of 1000 iterations. We compare the running-time of our proposed branch and bound inference technique (BB) to a message passing baseline in the form of convex belief propagation (cBP) (Hazan & Shashua, 2010). This baseline makes use of integral geometry, and thus is equivalent to the efficient approach of Schwing et al. (2012a). Fig. 6.9(a) and Fig. 6.9(b) illustrate the percentage of completed layout test set images given a certain amount of time for models trained with “SSVM” as well as “approx” in the data set of Hedau et al. (2009). For the bedroom data set, we provide the results for “SSVM” training in Fig. 6.10(a) and for “approx” training in Fig. 6.10(b). To obtain meaningful statistics the given curves are averaged over a wide range of 14 logarithmically spaced regularization tradeoff parameters $C \in [0.001, 1]$. The average run time is provided in the caption.
6.7. Experimental Evaluation

<table>
<thead>
<tr>
<th></th>
<th>OM</th>
<th>GC</th>
<th>OM + GC</th>
<th>Others</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hoiem et al. (2007a)</td>
<td>-</td>
<td>28.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Hedau et al. (2009)</td>
<td>-</td>
<td>21.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Wang et al. (2010)</td>
<td>22.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Lee et al. (2010)</td>
<td>24.7</td>
<td>22.7</td>
<td>18.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>del Pero et al. (2012)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>16.3</td>
<td>12min</td>
</tr>
<tr>
<td>Schwing et al. (2012a)</td>
<td>18.64</td>
<td>15.35</td>
<td>13.59</td>
<td>-</td>
<td>0.15s</td>
</tr>
<tr>
<td>Schwing &amp; Urtasun (2012)</td>
<td>18.63</td>
<td>15.35</td>
<td>13.59</td>
<td>-</td>
<td>0.007s</td>
</tr>
</tbody>
</table>

Table 6.2.: Comparison to state-of-the-art that uses the same image information on the layout data set of Hedau et al. (2009). Pixel classification error is given in %.

<table>
<thead>
<tr>
<th></th>
<th>Error [%]</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>del Pero et al. (2011)</td>
<td>29.59</td>
<td>-</td>
</tr>
<tr>
<td>Hoiem et al. (2007a)</td>
<td>23.04</td>
<td>-</td>
</tr>
<tr>
<td>Hedau et al. (2009)</td>
<td>22.94</td>
<td>-</td>
</tr>
<tr>
<td>Schwing et al. (2012a)</td>
<td>16.46</td>
<td>0.17</td>
</tr>
<tr>
<td>Schwing &amp; Urtasun (2012)</td>
<td>16.46</td>
<td>0.007</td>
</tr>
</tbody>
</table>

Table 6.3.: Comparison to state-of-the-art on the bedroom data set provided by Hedau et al. (2010). Pixel classification error is given in %.

We note that cBP inference has a plateau-like behavior, i.e., there are easy instances, but also harder ones requiring more iterations, some of them requiring the maximum number of iterations we allow. Importantly, the proposed BB approach has a roughly constant incline, and converges to the optimum faster. In contrast, the baseline is not guaranteed to get the optimum. Note that we display the results using logarithmic time scale, and thus cBP takes significantly more time for a large set of images.

To obtain quantitative results we provide the average time for inference for the layout data set and the bedroom data set in Tab. 6.1(a) and Tab. 6.1(b) respectively. The proposed BB approach outperforms the message passing technique on both data sets. Also note that the approach proposed by del Pero et al. (2012) requires about 12 minutes per image when employed to the layout data set.

**Number of splits:** The number of iterations in branch and bound is a direct measure for the number of splits required to achieve the optimal solution. It also provides an indication for the tightness of the bounds constructed in the previous section. We emphasize that the proposed branch and bound scheme utilizes integral geometry, and thus only accesses a few values on the accumulators to efficiently upper bound a set of layouts. Therefore one iteration is very fast and computable in constant time. Fig. 6.11(a) and Fig. 6.11(b) depict the cumulative distribution of the number of iterations required to obtain the optimal solution for the layout and bedroom datasets.

97
Figure 6.12.: The best classification results on the layout data set. The first and fourth column illustrate the image overlaid by the best possible layout obtained from ground truth labels in blue and our prediction result (red) given orientation map and geometric context features illustrated in columns 2, 5 and columns 3, 6 respectively.

respectively. As before, we average over a large range of parameters $C$ employed to learn our models, and compare “SSVM” and “approx.” We observe that the behavior on both data sets is roughly identical, with the bedroom data set requiring on average somewhat more iterations. Independent of the way the parameters $w$ are learned, we find that roughly similar number of iterations are required during inference.

**Accuracy:** Our branch and bound approach is faster than any previously published 3D layout estimation method. We now show its predictive performance on the layout and bedroom data sets. To this end we take the best learning settings reported in (Schwing et al., 2012a) and use them for the proposed BB inference. Tab. 6.2 and Tab. 6.3 show the results. Interestingly, we obtain identical results to the ones presented by Schwing et al. (2012a). This means that in practice cBP converges (although not guaranteed) to the optimum. Therefore we conclude that identical accuracies are obtained with our faster but most importantly, provably exact, BB inference mechanism.
6.7. Experimental Evaluation

Figure 6.13.: The best classification results on the bedroom data set. The first and fourth column illustrate the image overlaid by the best possible layout obtained from ground truth labels in blue and our prediction result (red) given orientation map and geometric context features illustrated in columns 2, 5 and columns 3, 6 respectively.

Visual Results: Finally, we provide visualizations of our results as well as orientation map and geometric context features for both data sets in Fig. 6.12 and Fig. 6.13. Here we use the model learned with OM and GC features. The blue lines overlaying the image provide the best possible result given the employed discretization, while the red lines illustrate our prediction. The error with respect to the pixelwise face labelings is indicated below each image. Orientation maps are given in column 2 and 5 while geometric context features are depicted in column 3 and 6.

We also provide failure cases for the layout data set (top row) and the bedroom data set (bottom row) in Fig. 6.14. We observe two main reasons causing prediction errors. First, a failing vanishing point detection can typically not be recovered from as the rays and their configuration can be far from any true layout. The second reason for failure modes is non-informative image features due to a failing prediction in case of geometric context or wrong line detections causing misleading orientation maps.
6. 3D Indoor Scene Understanding

![Image](image_url)

Figure 6.14.: The worst classification results on the layout data set (top row) and the bedroom data set (bottom row). The first and fourth column illustrate the image overlaid by the best possible layout obtained from ground truth labels in blue and our prediction result (red) given orientation map and geometric context features illustrated in columns 2, 5 and columns 3, 6 respectively.

### 6.7.2. Layout and Object

For joint layout and object estimation we perform our experiments on the bedroom data set (Hedau et al., 2010) which contains 309 images labeled with objects. The data is split into a training and test set of size 181 and 128 respectively. We again employ the vanishing point detector of Hedau et al. (2009). We measure the performance via a pixel-wise error metric that counts the percentage of pixels that disagree with ground truth and evaluate on 9-label and 5-label metrics. Whereas the latter captures the performance on estimating orientation, irrespective of being part of the layout or the object, the 9-label metric takes into account this distinction, making it significantly harder. To evaluate the object detection performance we use the F-measure computed for detections with intersection over union (IOU) higher than 0.5 as utilized by Pascal. We report this measure to detect the top face (Top), all the side faces jointly (Side), the convex hull (Hull) of the object as well as a 2D bounding box (BB). In addition we provide the pixelwise labeling error. Moreover, we follow Hedau et al. (2012) and also evaluate the free-space being a true 3D error metric.

We perform two tasks. First we look into the problem of localization where we know about the existence of an object in the scene, and the goal is to find it in 3D. For the second task, we performed 3D detection, where we have no knowledge about the presence of an object. We refer to these tasks via “loc” and “det” respectively.
6.7. Experimental Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Top</th>
<th>Side</th>
<th>Hull</th>
<th>BB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo</td>
<td>25.51</td>
<td>19.39</td>
<td>48.98</td>
<td>64.29</td>
</tr>
<tr>
<td>Geo+2D</td>
<td>33.67</td>
<td>27.55</td>
<td>60.20</td>
<td>65.31</td>
</tr>
<tr>
<td>Geo+3D</td>
<td><strong>37.76</strong></td>
<td>38.78</td>
<td>60.20</td>
<td>71.43</td>
</tr>
<tr>
<td>Geo+2D+3D</td>
<td>35.05</td>
<td><strong>39.18</strong></td>
<td><strong>68.04</strong></td>
<td><strong>74.23</strong></td>
</tr>
<tr>
<td>Geo</td>
<td>36.30</td>
<td>32.59</td>
<td>51.11</td>
<td>54.07</td>
</tr>
<tr>
<td>Geo+2D</td>
<td>42.22</td>
<td>38.52</td>
<td>62.22</td>
<td>66.67</td>
</tr>
<tr>
<td>Geo+3D</td>
<td><strong>44.44</strong></td>
<td>43.70</td>
<td>58.52</td>
<td>60.74</td>
</tr>
<tr>
<td>Geo+2D+3D</td>
<td>42.96</td>
<td><strong>47.41</strong></td>
<td><strong>66.67</strong></td>
<td><strong>69.63</strong></td>
</tr>
</tbody>
</table>

Table 6.4.: Comparison to state-of-the-art.

<table>
<thead>
<tr>
<th></th>
<th>Top</th>
<th>Side</th>
<th>Hull</th>
<th>BB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo</td>
<td>26.53</td>
<td>24.49</td>
<td>50.00</td>
<td>63.27</td>
</tr>
<tr>
<td>Geo+2D</td>
<td>33.67</td>
<td>27.55</td>
<td>60.20</td>
<td>65.31</td>
</tr>
<tr>
<td>Geo+3D</td>
<td><strong>35.71</strong></td>
<td>37.76</td>
<td>60.20</td>
<td>69.39</td>
</tr>
<tr>
<td>Geo+2D+3D</td>
<td>34.69</td>
<td><strong>38.78</strong></td>
<td><strong>65.31</strong></td>
<td><strong>74.49</strong></td>
</tr>
</tbody>
</table>

Table 6.5.: Importance of the features: note that every feature we add generally improves detection. We refer to OM+GC features via Geo, the 2D detector via 2D, and the 3D detector via 3D.

**Comparison to state-of-the-art:** We begin our experimentation by comparing our approach to the 3D detector of Fidler et al. (2012) and the deformable part-based model (Felzenszwalb et al., 2010). We utilize the occam-razor score only in the “det” case, since we do not know if the image contains an object. We provide the results in Tab. 6.4 and observe that our approach significantly improves over both baselines.

**Feature Importance:** Tab. 6.5 and Tab. 6.6 show our results when employing different types of features. We observe that each source of information, i.e., geometric features (OM and GC), 2D and 3D detectors increases performance.

**Oracle:** To illustrate the best achievable performance of our approach, we investigate its performance when providing ground truth features. We do so in two scenarios, by using 5-label and 9-label features referred to as “Oracle 5L” and “Oracle 9L” respectively. The former mimics the case where only geometric features are provided, while the latter represents the case where one has access to perfect detections and geometry estimates. Tab. 6.7 and Tab. 6.8 show that while the performance of the greedy approach is more or less identical when providing 9-label information, joint inference outperforms the greedy approach in the 5-label case. This is again the same phenomena as with real features.
6. 3D Indoor Scene Understanding

Table 6.6.: Importance of the features: note that every feature we add generally improves detection. We refer to OM+GC features via Geo, the 2D detector via 2D, and the 3D detector via 3D.

<table>
<thead>
<tr>
<th></th>
<th>Labeling measures</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>joint 9L</td>
<td>joint 5L</td>
<td>greedy 9L</td>
<td>greedy 5L</td>
<td></td>
</tr>
<tr>
<td>loc Geo</td>
<td>26.16</td>
<td>22.00</td>
<td></td>
<td>26.62</td>
<td>22.70</td>
<td></td>
</tr>
<tr>
<td>Geo+2D</td>
<td>23.20</td>
<td>20.43</td>
<td></td>
<td>23.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geo+3D</td>
<td></td>
<td></td>
<td>21.03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geo+2D+3D</td>
<td>22.65</td>
<td>20.30</td>
<td></td>
<td>23.81</td>
<td>21.22</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Labeling measures</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>det 9L</td>
<td>det 5L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>loc Geo</td>
<td>27.84</td>
<td>23.81</td>
<td></td>
<td>26.95</td>
<td>23.05</td>
<td></td>
</tr>
<tr>
<td>Geo+2D</td>
<td>25.77</td>
<td>22.94</td>
<td></td>
<td>24.50</td>
<td>21.64</td>
<td></td>
</tr>
<tr>
<td>Geo+3D</td>
<td></td>
<td></td>
<td>24.57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geo+2D+3D</td>
<td>24.66</td>
<td>21.67</td>
<td></td>
<td>24.57</td>
<td>21.73</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.7.: Comparison of F1 scores and labeling error for the sparse and dense parameterization using oracle features.

**Density of the Parameterization:** The major failures of our approach are wrong vanishing points as well as discretization artifacts. To illustrate the performance gain when increasing the discretization, we almost double the average number of states per layout variable from 18.4 to 34.6 and similarly increase the state space for object parameters from 28.1 for the sparse approach to 52.9. As illustrated in Tab. 6.7 and Tab. 6.8 for oracle features, the performance increases significantly for some of the measures. On real features, however, we observe almost no gain which is mainly due to the captured noise in the features.

**Computational Complexity:** Tab. 6.9 shows the average inference time for both the greedy and joint approach when employing different types of features. As expected, an increasing amount of features results in slower inference. The greedy approach is about two orders of magnitude faster for oracle features and three orders of magnitude faster for real features. We further emphasize that high quality features yield faster inference since the energy landscape is smoother.
6.7. Experimental Evaluation

<table>
<thead>
<tr>
<th>Sparse</th>
<th>det</th>
<th>loc</th>
<th>joint 9L</th>
<th>5L</th>
<th>greedy 9L</th>
<th>5L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle 9L</td>
<td>7.82</td>
<td>6.27</td>
<td>8.22</td>
<td>6.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle 5L</td>
<td>7.68</td>
<td>6.10</td>
<td>10.89</td>
<td>7.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle 9L</td>
<td>10.28</td>
<td>7.51</td>
<td>8.65</td>
<td>6.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle 5L</td>
<td>11.77</td>
<td>6.96</td>
<td>11.94</td>
<td>7.48</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dense</th>
<th>det</th>
<th>loc</th>
<th>joint 9L</th>
<th>5L</th>
<th>greedy 9L</th>
<th>5L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle 9L</td>
<td>5.79</td>
<td>5.54</td>
<td>4.52</td>
<td>4.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle 5L</td>
<td>5.78</td>
<td>4.60</td>
<td>9.80</td>
<td>6.88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle 9L</td>
<td>7.06</td>
<td>6.83</td>
<td>5.04</td>
<td>4.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle 5L</td>
<td>8.24</td>
<td>5.43</td>
<td>10.09</td>
<td>6.63</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.8.: Comparison of F1 scores and labeling error for the sparse and dense parameterization using oracle features.

<table>
<thead>
<tr>
<th></th>
<th>joint</th>
<th>greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle 9L</td>
<td>12.88s</td>
<td>0.07s</td>
</tr>
<tr>
<td>Oracle 5L</td>
<td>6.95s</td>
<td>0.07s</td>
</tr>
<tr>
<td>Geo</td>
<td>331.43s</td>
<td>0.37s</td>
</tr>
<tr>
<td>Geo+2D</td>
<td>230.68s</td>
<td>0.30s</td>
</tr>
<tr>
<td>Geo+3D</td>
<td>583.18s</td>
<td>0.43s</td>
</tr>
<tr>
<td>Geo+2D+3D</td>
<td>3333.09s</td>
<td>1.58s</td>
</tr>
</tbody>
</table>

Table 6.9.: Average inference time in seconds for the joint and greedy approach with different features provided

**Estimating free space:** Following Gupta et al. (2011) and Hedau et al. (2012) we compute the average F1 score using $\text{IOU} \geq 0.5$ and standard average of our estimation for the floor, the ground face of the object and the free space in Tab. 6.10. We observe average free-space estimation accuracies of up to 40% which improves over Hedau et al. (2012).

**Qualitative results:** Qualitative results are illustrated in Fig. 6.15. In general, our approach does a great job at estimating both the layout and object. We refer the reader to (Schwing et al., 2013) for results on each of the test images. The main failures illustrated in the bottom right corner of Fig. 6.15 are due to vanishing points not being detected properly and noisy features which are errors we cannot recover from.

**Estimating multiple objects:** We extend our approach to detect multiple objects in a greedy fashion, by fixing the layout and the previously detected object, subtracting the explanation from the image evidence and solving for the next object. Fig. 6.16 shows examples, with the layout, the first and second objects depicted in magenta, red and green. The heuristic approach works well if the image features provide clear evidence. We are however not guaranteed to find a globally optimal configuration.
6. 3D Indoor Scene Understanding

Figure 6.15.: Illustration of prediction results (red, magenta) and best found ground truth state (blue, cyan) given vanishing points for joint object and layout inference overlaying the image. Below each image we provide visible annotation floor plan (gray) and object on the left while corresponding prediction result on the right. A failure case due to wrong vanishing points is illustrated in bottom right figure.

6.8. Conclusion

We addressed the problem of recovering the scene layout in the form of a 3D parametric box given a single image. To this end we presented a novel branch and bound approach which splits the label space in terms of candidate sets of 3D layouts, and bounds the score for entire sets by constructing upper-bounding contributions of each individual face. We employed integral geometry in order to evaluate these bounds in constant time, and showed that we obtain not only the exact solution, but also in less time than approximate inference tools such as message-passing. We demonstrated the effectiveness of our approach in two benchmarks and showed that our bounds are tight, and only a few evaluations are necessary.

In addition we presented an approach to joint 3D room layout and object reasoning that predicts the optimal box within a box. To this end we carefully modeled the occlusions and phrased the problem as a structured prediction task that permits exact inference via a branch and bound algorithm. The main technical difficulty resides in the development of occlusion bounds which require the generalization of integral geometry to triangular shapes.
Table 6.10.: Computation of average F1 score for intersection over union of floor, object footprint and free-space for joint inference with indicated features. While the Pascal approach counts scores larger than 0.5 as correct detections, we also provide the mean.

<table>
<thead>
<tr>
<th></th>
<th>Pascal</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Floor</td>
<td>Object</td>
</tr>
<tr>
<td>Oracle 9L</td>
<td>89.76</td>
<td>62.22</td>
</tr>
<tr>
<td>Oracle 5L</td>
<td>90.55</td>
<td>60.00</td>
</tr>
<tr>
<td>Geo</td>
<td>63.78</td>
<td>29.63</td>
</tr>
<tr>
<td>Geo+2D</td>
<td>71.65</td>
<td>29.63</td>
</tr>
<tr>
<td>Geo+3D</td>
<td>68.50</td>
<td>37.78</td>
</tr>
<tr>
<td>Geo+2D+3D</td>
<td>70.63</td>
<td>37.04</td>
</tr>
</tbody>
</table>

Figure 6.16.: After jointly inferring layout (magenta) and object (red), we re-apply the object part to obtain a second object (green).
6. 3D Indoor Scene Understanding
7. Learning with Latent Variables

The learning algorithms for structured models discussed in Chapter 5 assumed a given data set $\mathcal{D}$ consisting of pairs of input and output space objects. Importantly, the output space objects are required to be completely labeled, i.e., a ground truth value for every single random variable has to be provided and missing data was not considered at all. In the following we waive this restriction.

7.1. Loss Minimization with Latent Variables

To this end we discuss a general framework for loss minimization with latent variables. For readability we repeat some earlier definitions. Consider the setting where $\mathcal{X}$ is the input space (e.g., an image or a sentence) and $\mathcal{S}$ is a structured output space (e.g., an image segmentation or a parse tree).

Note that $\mathcal{S}$ can depend on the example $x \in \mathcal{X}$. For clarity of notation we drop this dependency. Let $\phi : \mathcal{X} \times \mathcal{S} \rightarrow \mathbb{R}^F$ denote a mapping from input and label space to an $F$-dimensional feature space. We are interested in finding the parameters $w \in \mathbb{R}^F$ of a log-linear model, which best describe the possible configurations $s \in \mathcal{S}$ of $x \in \mathcal{X}$, i.e.,

$$p(s \mid x, w) \propto \exp \beta w^\top \phi(x, s). \quad (7.1)$$

We address the weakly supervised setting, where we are given a training set $\mathcal{D} = \{(x^{(i)}, y^{(i)})_{i=1}^{|\mathcal{D}|}\}$ containing $|\mathcal{D}|$ pairs, each composed by an input $x \in \mathcal{X}$ and some partially labeled data $y \in \mathcal{Y} \subseteq \mathcal{S}$. For every training pair, we divide the label space $\mathcal{S} = \mathcal{Y} \times \mathcal{H}$ into two non-intersecting subspaces $\mathcal{Y}$ and $\mathcal{H}$ and refer to the missing information $h \in \mathcal{H}$ as hidden or latent.

Following the derivation of the fully supervised setting, we want to linearly penalize configurations if the best prediction over the complete output space $\mathcal{S}$ exceeds the score obtained with the best prediction achieved when clamping the annotation. More formally, the cost function shall linearly penalize configurations for which

$$\max_{\hat{s} \in \mathcal{S}} \left\{w^\top \phi(x, \hat{s}) + \ell_c(x, y)(\hat{s})\right\} - \max_{\hat{h} \in \mathcal{H}} \left\{w^\top \phi(x, (y, \hat{h})) + \ell_c(x, y)(y, \hat{h})\right\}.$$

Naturally, the left hand side is always at least as large as the right hand side. When including a margin $\ell$, we obtain the following max-margin formulation in case latent variables are present within the data set $\mathcal{D}$:

$$\min_w \frac{C}{p} \|w\|_p^p + \sum_{(x, s)} \left(\max_{\hat{s}} \left\{w^\top \phi(x, \hat{s}) + \ell(x, y)(\hat{s})\right\} - \max_{\hat{h}} \left\{w^\top \phi(x, (y, \hat{h})) + \ell_c(x, y)(y, \hat{h})\right\}\right). \quad (7.2)$$

107
7. Learning with Latent Variables

The margin $\ell$ is also known as a loss function $\ell_{(x,y)}(\hat{s})$ which compares an estimated configuration $\hat{s}$ with the labeled data $(x,y) \in \mathcal{D}$, providing a measure for the fitness of the estimate. For an alternative derivation using a probabilistic interpretation we incorporate this loss function into a learning approach by considering the loss-augmented likelihood

$$p_{(x,y)}(\hat{s}|w) \propto \exp \left( \beta \phi^T(x, \hat{s}) + \ell_{(x,y)}(\hat{s}) \right).$$

(7.3)

Intuitively it places more probability mass on those parts of the space $S$ that have a high loss.

A maximum likelihood approach aims at finding model parameters $w$ which assign highest probability to the data $\mathcal{D}$. As we have no information available for the unobserved space $H$ we marginalize it out, i.e., we average over all possible hidden states. Therefore, we define the loss-augmented likelihood of a prediction $\hat{y} \in \mathcal{Y}$ when observing the pair $(x,y)$ as

$$p_{(x,y)}(\hat{y}|w) \propto \sum_{\hat{h} \in H} p_{(x,y)}(\hat{y}, \hat{h}|w) = \sum_{\hat{h} \in H} p_{(x,y)}(\hat{s}|w).$$

(7.4)

Assuming the data to be independent and identically distributed (i.i.d.), our goal is to minimize the negative log-posterior $-\ln[p(w) \prod_{(x,y) \in \mathcal{D}} p_{(x,y)}(y|w)]$ with $p(w) = \exp \left( \frac{-C_p}{\beta} \|w\|_p \right)$ being a prior function on the model parameters. As a result, the negative log-posterior is a difference of convex terms

$$\frac{C_p}{\beta} \|w\|_p + \sum_{(x,y) \in \mathcal{D}} \left( \frac{1}{\beta} \ln \sum_{\hat{s} \in S} \exp \beta \left( \phi^T(x, \hat{s}) + \ell_{(x,y)}(\hat{s}) \right) - \frac{1}{\beta} \ln \sum_{\hat{h} \in H} \exp \beta \left( \phi^T(x,y, \hat{h}) + \ell_{(x,y)}^c((y, \hat{h})) \right) \right),$$

(7.5)

with the first two parts being the sum of the negative log-prior function and the logarithm of the partition function. We take the loss of a ground truth configuration $\ell_{(x,y)}^c((y, \hat{h})) = \ell_{(x,y)}((y, \hat{h})) \equiv 0$, independent of any estimate $\hat{h}$.

Importantly, the temperature parameter $\beta$ defines an entire family of structured prediction tasks with latent variables. For $\beta = 1$ we obtain the maximum likelihood framework also known as hidden conditional random field (Quattoni et al., 2007), while $\beta = \infty$ results in the max-margin formulation known as structured support vector machines for latent variables (Yu & Joachims, 2009), i.e., exactly the program given in Eq. (7.2). The soft-max functions introduced via the probabilistic formulation smoothly approximate the max function for $\beta \to \infty$. 

108
7.2. Efficient Message Passing Algorithm

7.2.1. Approximate Latent Structured Loss Minimization

The unconstrained minimization problem in Eq. (7.5) w.r.t. \( w \) is challenging as it involves a sum of convex and concave terms containing exponentially sized sums. To make the minimization more tractable, we follow Yuille & Rangarajan (2003) and upper-bound the concave part via a minimization over a set of dual variables subsequently referred to as \( q(x, y) \). This results in a convex dual and a non-convex bi-linear term as described in the following claim.

Claim 10 The function

\[
\frac{C}{2} \frac{||w||^2}{p} + \sum_{(x, y) \in D} \left( \sum_{r \in R} \frac{c_r}{\beta} \ln \sum_{s_r} \exp \left( \tilde{\phi}(x, y, r)(s_r) \right) + \right.
\]

\[
\left. \sum_{c \in C(r)} \lambda(x, y, c \rightarrow r)(s_c) - \sum_{p \in P(r)} \lambda(x, y, r \rightarrow p)(s_r) \right) \left( \frac{c_r}{\beta} \right)
\]

\[
-\sum_k w_k \left( \sum_{(x, y)} \left( \sum_{r \subseteq Y} \phi_{k, r}(x, y, r) + \sum_{r \not\subseteq Y, h_r} \phi_{k, r}(x, y, h_r) d(x, y, r(h_r)) \right) \right)
\]

\[
-\sum_{(x, y)} \left( \sum_{r \not\subseteq Y, h_r} \ell^c_{r}(x, y, r(h_r)) d(x, y, r(h_r)) \right) - \sum_{r \not\subseteq Y} \hat{c}_r \frac{1}{\beta} H(d(x, y, r))
\]

s.t. \( \sum_{p \setminus h_r} d(x, y, p(h_p) = d(x, y, r(h_r)) \forall (x, y), r, p \in P(r), h_r \}

\[
\bigg\{ d(x, y) \in C(x, y) \bigg\} := \bigg\{ (x, y) \in D \bigg\}
\]

convex in \( w \) and \( q(x, y) \) separately, is an upper bound on Eq. (7.5), \( \forall q(x, y)(h) \in \Delta \), with \( \Delta \) the probability simplex, \( H \) the entropy and \( \mathbb{E} \) the expectation w.r.t. the stated distribution. The bound holds with equality for that \( q^*_*(x, y)(h) \) minimizing this cost function (Eq. (7.6)).
Proof: The partition function is the conjugate dual of the entropy as derived in Chapter 5. When replacing the partition function over the latent space $H$ via

$$
\frac{1}{\beta} \ln \sum_{\hat{h}} \exp \beta \left( w^\top \phi(x, y, \hat{h}) + \ell_c^{\hat{h}}(y, \hat{h}) \right) =
$$

we obtain the problem stated in Eq. (7.6). It is easy to see that the cost function given in Eq. (7.6) is convex in $w$ and $q(x, y)$, and not jointly convex in $w$ and $q(x, y)$. Neglecting minimization w.r.t. $q(x, y)$ results in an upper bound to the original problem. The original problem is attained for optimal $q^*(x, y)$ which concludes the proof.

The bound illustrated in Eq. (7.6) provides the intuition for the divergence we optimize when working with latent variables. Consider minimization of the Kullback-Leibler (KL) divergence w.r.t. both, parameters $w$ and distributions $q(x, y)(\hat{h})$, i.e.,

$$
\min_{w,q(x, y)\in\Delta} \sum_{(x, y)} D_{KL} \left( q(x, y) \mid \frac{p(w)}{Z(x, y)} \exp \beta \left( w^\top \phi(x, y, \hat{h}) + \ell(x, y)(y, \hat{h}) \right) \right).
$$

The cost function of this program is equivalent to Eq. (7.6) if we set our prior function to be $p(w) = \exp \left( -\frac{C\beta}{|D|p} \|w\|_p^p \right)$. We therefore obtain an I-projection when optimizing w.r.t. the distributions $q(x, y)(\hat{h})$ while an M-projection is performed when optimizing w.r.t. the parameters $w$. We will see later that the resulting optimization scheme alternates between those two tasks.

For many real-world applications, the program in Claim 10 involves sums over exponentially sized sets $S$ and $H$. They are exponentially sized as the observed and unobserved labels $y = (s_i)_{i \in \mathcal{Y}} \in \mathcal{Y}$ and $h = (s_i)_{i \in \mathcal{H}} \in \mathcal{H}$ are often tuples with elements $s_i \in S_i$ taking $|S_i|$ discrete states. Note that the set $S$ of all $N$ variables within the model is obtained by $S = \{1, \ldots, N\} = \mathcal{Y} \cup \mathcal{H}$, with product spaces $\mathcal{Y} = \prod_{i \in \mathcal{Y}} S_i$ and $\mathcal{H} = \prod_{i \in \mathcal{H}} S_i$. For many applications the features usually describe interactions only between smaller local subsets of random variables, i.e.,

$$
\phi_k(x, s) = \sum_{r \in \mathcal{R}_k} \phi_{k,r}(x, s_r), \quad (7.7)
$$

where $\mathcal{R}_k$ denotes the set of regions involved in the $k$-th entry of the feature vector. The set of all regions is referred to via $\mathcal{R} = \bigcup_k \mathcal{R}_k$. Note that each feature is described by a Hasse diagram as before. An edge connects two regions $r$ and $p$ if $p$ is a parent of $r$, i.e., if $p \in P(r)$. Alternatively, two regions are connected if $r$ is a child of $p$, i.e., if $r \in C(p)$. In many applications the loss functions $\ell$ and $\ell_c$ factorize in a similar fashion and are easily introduced into the Hasse diagram, i.e., $\ell(x, y)(s)$ decomposes into local
terms \( \ell_{(x,y),r}(s_r) \), whereas \( \ell_{(x,y),r}^c(h_r) \) is structured according to the locally defined regions \( \ell_{(x,y),r}^c(s_r), \forall \{ r \in R : r \subseteq \mathcal{H} \} \). For notational convenience we subsequently abbreviate this set of regions via \( r \subseteq \mathcal{H} \) and employ similar shorthands for \( r \subseteq \mathcal{Y} \) and \( r \not\subseteq \mathcal{Y} \).

We make use of the local structure of features and loss, and approximate the intractable function in Claim 10. In particular we follow Chapter 4 and let the probability distribution \( q_{(x,y)}(h) \) be described by local beliefs \( d_{(x,y),r}(h_r) \in \Delta \). We approximate the marginal polytope by a local one using the marginalization constraints

\[
\sum_{p \in P(r)} d_{(x,y),r}(h_r) = d_{(x,y),r}(h_r) \quad \forall (x,y) \in \mathcal{D}, r \not\subseteq \mathcal{Y}, p \in P(r), h_r \in S_r.
\]

Moreover, we introduce counting numbers \( \hat{c}_r \) to allow for more flexibility in the entropy approximation.

To further obtain a tractable approximation for the partition function over \( S \) in Eq. (7.6) we approximate its Legendre transform, an entropy ranging over \( s \in S \), via local terms. As those local terms are required to fulfill marginalization constraints for global consistency in the dual domain, we obtain Lagrange multipliers \( \lambda_{(x,y),r}(p(s_r)) \forall (x,y) \in \mathcal{D}, r \in R, p \in P(r) \) and \( s_r \in S_r \) for the primal formulation. Note that those Lagrange multipliers are often interpreted as messages. For generality and accuracy of the entropy approximations we again allow for counting numbers \( c_r \). The approximation is formally stated as follows:

**Theorem 2** The approximation of the program in Eq. (7.6) takes the form given in Program 1 where

\[
\tilde{\phi}_{(x,y),r}(s_r) = \ell_{(x,y),r}(x, s_r) + \sum_{k : r \in \mathcal{R}_k} w_k \phi_{k,r}(x, s_r).
\]

**Proof:** In Sec. 7.2.4.

### 7.2.2. Message Passing Algorithm

Before deriving an algorithm for solving Program 1, we begin by discussing the properties of the approximation. For counting numbers and inverse-temperature parameter \( \beta \) larger than zero, it is jointly convex in the messages \( \lambda_{(x,y),r}(p(s_r)) \forall (x,y) \in \mathcal{D}, r \in S, p \in P(r) \) and \( s_r \subseteq S_r \) and the model parameters \( w \). It is also jointly convex in the messages and the beliefs \( d_{(x,y),r} \forall r \subseteq \mathcal{H} \), but not jointly convex when optimizing for both the weights and the beliefs. Cycling through blocks of variables and updating them in a block-coordinate descent manner is not guaranteed to converge as we cannot fulfill pseudoconvexity in every pair of coordinate blocks. Similar to other latent variable frameworks we obtain convergence guarantees when employing instances of variational methods discussed, e.g., by Jordan et al. (1999); Neal & Hinton (1999). An example is the concave-convex procedure (CCCP) (Yuille & Rangarajan, 2003; Sriperumbudur & Lanckriet, 2009) which separates the cost function into two functions \( f_1(w, \lambda) \) and \( f_3(d) \), convex in their parameters, and a bilinear term \( f_2(w, d) \), connecting the two. \( \lambda \) is the vector of all messages, \( d \) the vector of all beliefs, and \( C_{(x,y)} \forall (x,y) \in \mathcal{D} \) the set of all marginalization constraints. We refer the reader to Program 1 for the definition of these functions and the constraint set.

Without loss of generality we assume Program 1 to be bounded from below. Considering the biconvex cost function, it is intuitive to alternate between solving for the
7. Learning with Latent Variables

beliefs and then performing a gradient step in the direction of the weights and the messages. Due to the fact that the program is unconstrained in the messages and model parameters, one gradient step of the latter is sufficient. We refer the reader to Sec. 7.2.4 for a detailed derivation of the algorithm. In short, updating the beliefs, i.e., the ‘latent variable prediction problem’ requires solving

$$\min_{d(x,y)} f_2(w, d) + f_3(d) \quad \text{s.t.} \quad d(x,y) \in C(x,y)$$

(7.8)

for every \((x, y) \in \mathcal{D}\) independently, hence possibly in parallel. Not astonishingly this problem reduces to a standard (convex) belief propagation task (Weiss et al., 2007; Meltzer et al., 2009; Hazan & Shashua, 2010), i.e., an I-projection which is guaranteed to find the global optimum for strictly positive counting numbers \(c_r\) and temperature parameter \(\beta \neq \infty\). To update the weights and messages we are required to decrease the cost function of the following unconstrained program, convex in \(w\) and \(\lambda\):

$$\min_{w,\lambda} f_1(w, \lambda) + f_2(w, d).$$

(7.9)

Similar to the program given in Eq. (7.8), convergence is guaranteed for counting numbers and annealing factor being strictly positive. More importantly, for weights \(w\), one gradient step of length \(\eta\) obtained via line search ensuring, e.g., the Armijo rule, is sufficient for convergence guarantees. A solution for a block-coordinate descent step \(\nabla_{\lambda(x,y),r} (f_1 + f_2) = 0 \) w.r.t. \(\lambda_{(x,y),r \rightarrow p}(s_r)\) for \((x, y) \in \mathcal{D}, r \in \mathcal{R}\) can be analytically computed jointly \(\forall p \in P(r), s_r\). We briefly state the proposed algorithm for latent structured prediction in Alg. 3 while pointing the interested reader to Sec. 7.2.4 for details. The convergence properties of the proposed algorithm are summarized in the following claim.

**Claim 11** Alg. 3 is guaranteed to decrease the cost function of Program 1 at every iteration and guaranteed to converge to a minimum or a saddle point for \(c_r, \hat{c}_r > 0\) and \(\beta \neq \infty\).

**Proof:** Recalling Theorem 5 in Yuille & Rangarajan (2003) we notice that alternating optimization of the approximated program in Theorem 2 w.r.t. \(d\) and \(\lambda, w\) is equivalent to CCCP, which is guaranteed to converge to a stationary point if the respective functions are convex. Convexity is ensured for \(c_r, \hat{c}_r > 0\) and \(\beta \neq \infty\). \(\blacksquare\)

7.2.3. Experiments

In this section we demonstrate the effectiveness of our approach using the tasks of image segmentation as well as 3D scene understanding, and show that our method significantly outperforms a latent structured support vector machine (LSSVM) in terms of performance and hidden conditional random fields (HCRFs) in terms of speed.

**Segmentation:** Our first task addresses segmentation of weakly labeled images. This is an interesting example, as the graphical model contains many loops. As ground truth we use the \(14 \times 40\) sized “ICML” tag given in Fig. 7.1. We created a dataset
Algorithm 3 latent structured prediction

repeat
  repeat
    // to solve latent variable prediction problem
    \[ \min_d f_2 + f_3 \text{ s.t. } \forall (x, y) \in \mathcal{D}, d(x, y) \in C(x, y) \]
  until convergence

// message passing update
\[ \forall (x, y), r \in \mathcal{R} \quad \lambda(x, y), r \leftarrow \nabla \lambda(x, y), r (f_1 + f_2) = 0 \]

// gradient step with step size \( \eta \)
\[ w \leftarrow w - \eta \nabla_w (f_1 + f_2) \]
until convergence

Figure 7.1.: Segmenting the ICML tag with 90% latent variables: (left) ground truth and our result, (right) wrong model learned by LSSVM.

composed of 10 training and 10 test instances, where each observation \( x \) is obtained by adding zero mean, uniform noise on the ground truth labels \( s_i \in S_i = \{1, \ldots, 5\} \).

We employ \( F = 2 \) features, a local potential based on the observations and a pairwise linear smoothness potential. In our experiments, we gradually increase the amount of missing labels from 0% to 100%, and determine at random which variables are hidden/latent.

We compare our approach to a convex HCRF and to the LSSVM of Tsochantaridis et al. (2005) which uses belief propagation to solve the respective sub-problems. We use at most 200 outer iterations, 1000 inner iterations for our approach, and 200 outer iterations, 1000 message passing iterations, 20 cutting plane iterations for LSSVM. For computational reasons the HCRF method is restricted to only 10 outer iterations, 1000 message passing iterations and 5 CRF iterations resulting in a maximum of 50 updates of the model parameters. For a fair comparison, we also use 50 outer iterations and 1000 message passing iterations resulting in a maximum of 50 updates for our approach with \( \beta = 1 \). All algorithms employed the same initialization. For our framework we additionally vary \( \beta \) from \( \infty \) for the max-margin formulation to 1 for the maximum-likelihood formulation. Mean results averaged over 5 runs are depicted in Fig. 7.2. Our method results in good prediction for all values of \( \beta \), while the LSSVM
7. Learning with Latent Variables

Figure 7.2.: Performance as a function of the amount of latent variables, averaged over 5 runs.

fails in the presence of large amounts of latent variables. This is due to the fact that the cutting planes are not exactly computable for loopy models \(((14 \cdot 40)^5\) possibilities), and thus no decrease in the cost function is guaranteed. An example of the prediction of our approach and LSSVM in the presence of 90% latent variables is illustrated in Fig. 7.1, where LSSVM learns a wrong model that favors neighboring pixels to be different. In contrast, the HCRF performs similarly to our approach, but it takes on average 213.2min to compute a single HCRF experiment while only 6.2min are required for our approach with \(\beta = 1\). Since HCRF is not practical, we take LSSVM as a baseline for the reminder of the experimental evaluation.

3D Scene Understanding: Recovering the spatial layout of indoor scenes from a single image is an important problem in applications such as surveillance and robotics. Existing approaches formulate the problem as a structured prediction task focusing on estimating the 3D box which best describes the scene layout. Taking advantage of the Manhattan world assumption, i.e., there exist three dominant vanishing points which are orthogonal, the problem can be formulated as inference in a pairwise graphical model composed of four random variables. As shown in Fig. 7.3(a), these variables represent the angles encoding the rays that originate from the respective vanishing points. Following existing approaches (Hedau et al., 2009; Lee et al., 2010), we employ \(F = 55\) features based on geometric context (GC) and orientation maps (OM) and refer the interested reader to Hoiem et al. (2007a) and Lee et al. (2009) for respective details. Our features count for each face in the cuboid (given a particular configuration of the layout) the number of pixels with a certain label for OM and the probability that such label exists for GC. Performance is measured as the percentage of pixels that have been correctly labeled, with labels, i.e., left-wall, right-wall, front-wall, ceiling, floor.

We first investigate how the layout estimation can benefit from the use of weakly labeled data. To this end we use a set of fully annotated images, denoted ‘fixed,’ and
7.2. Efficient Message Passing Algorithm

![Parameterization of 3D scene understanding task](image)

Figure 7.3.: The parameterization of the 3D scene understanding task is illustrated in (a). Comparing different amount \( \{25,50,100\} \) of weakly labeled additional information with 50\% and 75\% missing data in (b) and (c).

<table>
<thead>
<tr>
<th></th>
<th>fully</th>
<th>weakly</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hoiem et al. (2007a)</td>
<td>209</td>
<td>0</td>
<td>28.9%</td>
</tr>
<tr>
<td>Hedau et al. (2009)</td>
<td>209</td>
<td>0</td>
<td>21.2%</td>
</tr>
<tr>
<td>Wang et al. (2010)</td>
<td>209</td>
<td>0</td>
<td>22.2%</td>
</tr>
<tr>
<td>Lee et al. (2010)</td>
<td>209</td>
<td>0</td>
<td>18.6%</td>
</tr>
<tr>
<td>Schwing et al. (2012b)</td>
<td>10</td>
<td>25 (75%)</td>
<td>15.5%</td>
</tr>
<tr>
<td>Schwing et al. (2012b)</td>
<td>10</td>
<td>50 (75%)</td>
<td>15.0%</td>
</tr>
<tr>
<td>Schwing et al. (2012b)</td>
<td>10</td>
<td>100 (75%)</td>
<td>14.7%</td>
</tr>
</tbody>
</table>

Table 7.1.: Comparison to state-of-the-art on the layout data set of Hedau et al. (2009). 75\% of the information is missing for each weakly annotated image.

add a varying number \( \{25,50,100\} \) of images with only 1 or 2 angles labeled, \textit{i.e.}, 75\% or 50\% missing information. The randomly chosen unlabeled angles are treated as latent variables. All results are averaged over 12 runs, each being trained on a varying portion of the training set. Learning is performed with parameters \( C = 1, \beta = 100 \) and all counting numbers equal to one.

The results for 50\% and 75\% of missing information are detailed in Fig. 7.3(b) and Fig. 7.3(c) respectively. As expected, the prediction performance improves as a function of the number of fully labeled images, but more importantly, the performance significantly improves as a function of the amount of weakly labeled data. Our performance also increases as a function of how much supervision the weakly annotated images have, \textit{i.e.}, 2 hidden variables outperforms having 3 latent variables per image.

A comparison of our approach to the state-of-the-art is shown in Tab. 7.1. Great performance is achieved with a small amount of supervision. Our fully supervised approach with 200 completely labeled examples results in a prediction error of 13.6\% Schwing et al. (2012a).

Fig. 7.4 depicts improvements achieved by our approach compared to only using the fixed set of 10 fully labeled training images, as well as LSSVM. For LSSVM and our
7. Learning with Latent Variables

![Images showing prediction results of different approaches](image)

**Figure 7.4.** Column 1 and 2 show prediction results of our approach and LSSVM after training with 10 fully labeled images and an additional 100 images labeled with 50% missing information. Column 3 illustrates prediction results after training with only 10 fully labeled images. The pixel errors are as indicated below each figure. Column 4 and 5 provide the orientation map and geometric context image features.

Approach, we used an additional 100 images with 50% missing annotations. Prediction errors are indicated below the figures. We also provide illustrations for the image features we employed in the last two columns, i.e., orientation maps and geometric context. Interestingly, when trained with only 10 images, the model tends to miss walls and the ceiling.

### 7.2.4. Detailed Derivation

**Proof of Theorem 2**

As usual we assume that each element of the feature vector $\phi$ decomposes into a graphical model structure, i.e., the $k$-th feature vector element takes the following form:

$$
\phi_k(x, s) = \sum_{r \in \mathcal{R}_k} \phi_{k,r}(x, s_r),
$$

(7.10)

with $\mathcal{R}_k$ denoting the set of regions involved in the $k$-th feature vector element. Following the common notation, an edge connects regions $r, p \in \mathcal{R}_k$ if and only if region $p$ is within the set of parents of region $r$, i.e., $p \in P(r)$. The Hasse diagram for the inference problem contains the union of all regions, i.e., $\mathcal{R} = \bigcup_k \mathcal{R}_k$. The set of parents $P(r)$ and children $C(r)$ is defined as usual and inherited from the sub-graphs.
In many applications the loss functions $\ell$ and $\ell^c$ factorize in a way, similar to the features. Hence, we assume them to be easily included into the graphical model via

\[
\ell(x,y)(s) = \sum_{r \in \mathcal{R}} \ell(x,y),r(s_r), \quad \ell^c(x,y)(y,\hat{h}) = \sum_{r \in \mathcal{R}} \ell^c(x,y),r(y_r,\hat{h}_r). \quad (7.11)
\]

Let $d_{(x,y),r}$ be the marginals of $q_{(x,y)}(h)$ corresponding to the regions within the Hasse diagram introduced by the features and the loss functions. The expected loss given the latent variable distribution equivalently reads as

\[
\sum_{\hat{h}} \ell^c(x,y)(y,\hat{h})q_{(x,y)}(\hat{h}) = \sum_{r \subseteq \mathcal{Y}} \ell^c(x,y),r(y_r) + \sum_{r \not\subseteq \mathcal{Y},h_r} \ell^c(x,y),r(y_r,h_r)d_{(x,y),r}(h_r). \quad (7.13)
\]

Since $\ell^c(x,y),r(y_r)$ is a constant it has no effect on the considered optimization problem and can be neglected. The expected feature statistics over the latent variable distribution are denoted by $v$. Similar to the expected loss, we compute and abbreviate them via

\[
\sum_{\hat{h}} \phi_k(x,y,\hat{h})q_{(x,y)}(\hat{h}) = v_{(x,y),k} = \sum_{r \subseteq \mathcal{Y}} \phi_k,r(x,y_r) + \sum_{r \not\subseteq \mathcal{Y},h_r} \phi_k,r(x,y_r,h_r)d_{(x,y),r}(h_r), \quad (7.14)
\]

In addition to rewriting above expectations exactly via marginals, we approximate the joint entropy by local terms, each ranging over a marginal of a corresponding region. For better flexibility of the approximation we introduce counting numbers $\hat{c}_r$. Moreover, we approximate the marginal polytope via the local polytope given by the marginalization and simplex constraints. All in all we obtain the following
approximated program:

\[
\begin{align*}
\min_{w,d,\sigma} \quad & \frac{C_p}{p} \|w\|_p^p + \sum_{(x,y) \in D} \left( \frac{1}{\beta} \ln \sum_s \exp \beta \left( \sigma(x, s) + \ell(x,y)(s) \right) - \sum_r w_r v(x,y),r \right) \left( f_2 - \ell_c(x,y) + \frac{1}{\beta} \ln \sum_s \exp \beta \left( \sigma(x, s) + \ell(x,y)(s) \right) - \sum_r w_r v(x,y),r \right) \right) \\
\text{s.t.} \quad & \forall (x, y), r \not\in \mathcal{Y}, p \in P(r), h_r \sum_{h_p \not\in h_r} d_{(x,y),p}(h_p) = d_{(x,y),r}(h_r) \\
& d_{(x,y),r} \in \Delta \\
& \forall (x, y), s \quad \sigma(x, s) = w^\top \phi(x, s).
\end{align*}
\]

For clarity of the presentation and since \( f_3 \) is independent of \( w \) and \( \sigma \), let us address minimization w.r.t. \( w \) and \( \sigma \) without regards to \( f_3 \) and the marginalization constraints on \( d \) required for global consistency. We therefore consider the following optimization problem:

\[
\begin{align*}
\min_{w,\sigma} \quad & \sum_{(x,y)} \frac{1}{\beta} \ln \sum_s \exp \beta \left( \sigma(x, s) + \ell(x,y)(s) \right) - \sum_r w_r v_r + \frac{C_p}{p} \|w\|_p^p \\
\text{s.t.} \quad & \forall (x, y), s \quad \sigma(x, s) = w^\top \phi(x, s).
\end{align*}
\]

Though convex in \( w \), this task is challenging due to the exponentially sized sum over the output space \( \mathcal{S} \). However, it is identical to a standard structured prediction task and the same approximation techniques apply. We therefore aim at approximating the program by considering the dual. In a subsequent step we therefore minimize the Lagrangian w.r.t. the primal variables \( w \) and \( \sigma \), i.e.,

\[
\begin{align*}
\sum_{(x,y)} \min_{\sigma} \left( \frac{1}{\beta} \ln \sum_s \exp \beta \left( \sigma(x, s) + \ell(x,y)(s) \right) - \sum_s p_{(x,y)}(s) \sigma(x, s) \right) + \\
\min_{w} \left( \frac{C_p}{p} \|w\|_p^p + \sum_{(x,y),s} p_{(x,y)}(s) \phi(x, s) - v \right).
\end{align*}
\]
Note that we introduced Lagrangian multipliers $p_{(x,y)}(s) \forall (x,y), s$. Analytically carrying out the minimization, we obtain the following dual problem

$$\max_{p_{(x,y)} \in \Delta} \sum_{(x,y)} \frac{1}{\beta} H(p_{(x,y)}) + \sum_s p_{(x,y)}(s)(w^\top \phi(x, s) + \ell(x, y)(s))$$

$$- \frac{C^{1-q}}{q} \sum_k \left| \sum_{(x,y), s} p_{(x,y)}(s)\phi_k(x, s) - v(x, y), r \right|^q.$$ 

Similar to our previous arguments we assume that the probability distribution $p_{(x,y)}(s)$ is defined via marginals $b_{(x,y), r}$. Again we approximate the marginal polytope via a local one given by the marginalization and simplex constraints. Hence we obtain,

$$\sum_s \ell(x, y)(s)p_{(x,y)}(s) = \ell(x, y) = \sum_{r \in R, s_r} \ell(x, y), r (s_r)b_{(x,y), r}(s_r),$$

$$\sum_s \phi_k(x, s)p_{(x,y)}(s) = u(x, y), k = \sum_{r \in R, s_r} \phi_k, r(x, s_r)b_{(x,y), r}(s_r),$$

$$u_k = \sum_{(x,y)} u(x, y), k.$$ 

The approximated dual program with counting numbers $c_r$ reads as

$$\max_{b, u} \sum_{(x,y), r} \frac{c_r}{\beta} H(b_{(x,y), r}) + \sum_{(x,y), r, s_r} b_{(x,y), r}(s_r)\tilde{\phi}(x, y), r (s_r) - \frac{C^{1-q}}{q} \sum_k |u_r - v_r|^q$$

$$\text{s.t. } \forall (x, y), r, p \in P(r), s_r \sum_{s_p \setminus s_r} b_{(x,y), p}(s_p) = b_{(x,y), r}(s_r) \tag{7.20}$$

$$b_{(x,y), r} \in \Delta. \tag{7.21}$$

By maximizing the Lagrangian with corresponding multipliers $\lambda(x,y), r \rightarrow p(s_r)$ for the constraint given in Eq. (7.20) and after combining with $f_3$ and corresponding constraints on $d$ we obtain the claim.

**Algorithmic Details**

Similar to a CCCP approach (Yuille & Rangarajan, 2003) we address optimization of Program 1 by alternating between two tasks. One optimization considers only the beliefs $d$, while the other one operates on $\lambda$ and $w$. When updating the beliefs $d$, we obtain the ‘latent variable prediction problem’ which requires solving

$$\sum_{(x,y) \in \mathcal{D}} \min_{d(x,y)} f_2(w, d) + f_3(d) \tag{7.22}$$

$$\text{s.t. } \forall (x, y) \in \mathcal{D} \quad d(x,y) \in \mathcal{C}(x,y).$$
7. Learning with Latent Variables

Explicitly, this task reads as

$$\sum_{(x,y)} \max_d \frac{\hat{c}_r}{\beta} H(d(x,y), r) + \sum_{r \subseteq \mathbb{Y}, h_r} d(x,y), r(h_r) \left( \sum_{k: r \subseteq \mathbb{Y}} w_k \phi_k, r(x, y_r, h_r) + \beta \sum_{r \subseteq \mathbb{Y}} \phi_k, r(x, y_r, h_r) \right)$$

s.t. $$\forall (x, y), r \not\subseteq \mathbb{Y}, p \in P(r), h_r \sum_{h_p \not\subset h_r} d(x,y), p(h_p) = d(x,y), r(h_r)$$

$$\forall (x, y) \quad d(x,y), r \in \Delta.$$ 

We observe that this task corresponds to a standard (convex) belief propagation task with region potentials $$\tilde{\phi}_{(x,y), r}(h_r) = \sum_{k: r \subseteq \mathbb{Y}} w_k \phi_k, r(x, y_r, h_r) + \beta \sum_{r \subseteq \mathbb{Y}} \phi_k, r(x, y_r, h_r).$$

We solve it by minimizing its unconstrained dual program using a message passing algorithm on the graph defined by all the regions containing latent variables, i.e., $$r \not\subseteq \mathbb{Y}$$. The algorithm is guaranteed to find the optimal solution for counting numbers $$\hat{c}_r > 0$$ and $$\beta \neq \infty$$.

To optimize for model parameters $$w$$ and messages $$\lambda$$ we gradually solve the following unconstrained problem:

$$\min_{w, \lambda} f_1(w) + f_2(w, d). \quad (7.23)$$

‘Gradually’ means that it is sufficient to perform a single update step before re-computing the marginals of the latent variable distribution. Given $$\tilde{\phi}_{(x,y), r}(s_r) = \ell_{(x,y), r}(x, s_r) + \sum_{k: r \subseteq \mathbb{Y}} w_k \phi_k, r(x, s_r),$$ this program explicitly reads as

$$\min_{\lambda, w} \frac{C}{2} \|w\|^2 + \sum_{(x,y) \in \mathcal{D}} \left( \sum_r \frac{c_r}{\beta} \ln \sum_{s_r} \exp \left( \tilde{\phi}_{(x,y), r}(s_r) \right) - \sum_{p \in P(r)} \lambda_{(x,y), r \rightarrow p}(s_r) + \sum_{c \in C(r)} \lambda_{(x,y), c \rightarrow r}(s_c) \right) / (c_r / \beta) - \sum_k w_k \left( \sum_{(x,y)} \left( \sum_{r \subseteq \mathbb{Y}} \phi_k, r(x, y_r) + \sum_{r \not\subseteq \mathbb{Y}, h_r} \phi_k, r(x, h_r) \right) \right).$$

It is an approximated structured prediction task with empirical observations $$v_k$$. Hence the same solution applies. We briefly sketch the message passing update rules for the Lagrange multipliers $$\lambda$$ in Lemma 2. The update rules for the weight vector $$w$$ are derived in Lemma 3.
7.2. Efficient Message Passing Algorithm

**Lemma 2** Given a region \( r \in \mathcal{R} \) of the Hasse diagram, the optimal \( \lambda_{(x,y), r \to p}(s_r) \) \( \forall p \in P(r), s_r \in S_r, (x, y) \in \mathcal{D} \) of Theorem 2 (resp. Eq. (7.23)) satisfies

\[
\lambda_{(x,y), r \to p}(s_r) \propto \frac{c_p}{c_r + \sum_{p \in P(r)} c_p} \left( \tilde{\phi}_{(x,y), r}(s_r) + \sum_{c \in C(r)} \lambda_{c \to r}(s_c) \right) + \sum_{p \in P(r)} \mu_{(x,y), p \to r}(s_r) \tag{7.24}
\]

with

\[
\mu_{(x,y), p \to r}(s_r) = \frac{c_p}{\beta} \ln \sum_{s_p \setminus s_r} \exp \left( \frac{\tilde{\phi}_{(x,y), p}(s_p) - \sum_{p' \in P(p)} \lambda_{(x,y), p' \to p}(s_{p'}) + \sum_{r' \in C(p)} \lambda_{(x,y), r' \to p}(s_{r'})}{c_p / \beta} \right).
\tag{7.25}
\]

**Proof:**

To update the messages we take every \((x, y), r \in \mathcal{R}\) and obtain an analytic solution of the first order optimality condition w.r.t. \( \lambda_{(x,y), r \to p}(s_r) \) \( \forall p \in P(r), s_r \). We obtain the following simplified optimization problem:

\[
\min_{\lambda_{(x,y), r \to p}(s_r)} \frac{c_r}{\beta} \ln \sum_{s_r} \exp \left( \frac{\tilde{\phi}_{(x,y), r}(s_r) - \sum_{p \in P(r)} \lambda_{(x,y), r \to p}(s_r) + \sum_{c \in C(r)} \lambda_{(x,y), c \to r}(s_c)}{c_r / \beta} \right) + \sum_{p \in P(r)} \frac{c_p}{\beta} \ln \sum_{s_r} \exp \left( \frac{\mu_{(x,y), p \to r}(s_r) + \lambda_{(x,y), r \to p}(s_r)}{c_p \beta} \right).
\tag{7.26}
\]

We find the optimal \( \lambda_{(x,y), r \to p}(s_r) \) \( \forall p \in P(r), s_r \) whenever the gradient vanishes which is achieved for

\[
\lambda_{(x,y), r \to p}(s_r) \propto \frac{c_p}{c_r + \sum_{p \in P(r)} c_p} \left( \tilde{\phi}_{(x,y), r}(s_r) + \sum_{c \in C(r)} \lambda_{c \to r}(s_c) \right) + \sum_{p \in P(r)} \mu_{(x,y), p \to r}(s_r) - \mu_{(x,y), p \to r}(s_r),
\]

which completes the proof. \( \blacksquare \)
Program 2 Message passing algorithm for approximated structured prediction with latent variables

Repeat

1. Solve ‘latent variable prediction problem’ (Program (7.23)) until convergence

2. For each \((x, y) \in \mathcal{D}, r \in \mathcal{R}\) (by Lemma 2):

\[
\lambda_{(x, y), r \rightarrow p}(s_r) \propto \frac{c_p}{c_r + \sum_{p \in P(r)} c_p} \left( \tilde{\phi}_{(x, y), r}(s_r) + \sum_{c \in C(r)} \lambda_{c \rightarrow r}(s_c) \right)
+ \sum_{p \in P(r)} \mu_{(x, y), p \rightarrow r}(s_r) - \mu_{(x, y), p \rightarrow r}(s_r).
\]

3. For each \(k\) (by Lemma 3): find a stepsize \(\eta\) that reduces \(f_1 + f_2\) as required by the Armijo rule and update

\[
w_k \leftarrow w_k - \eta \left( \sum_{(x, y) \in \mathcal{S}_k, s_r} b_{(x, y), r}(s_r) \phi_{k, r}(x, s_r) - v_r + C|w_r|^{p-1} \text{sign}(w_r) \right)
\]

Lemma 3 The gradient of the approximated program given in Theorem 2 (resp. Eq. (7.23)) w.r.t. \(w_k\) equals

\[
\sum_{(x,y)} \left( \sum_{r \in \mathcal{S}_k, s_r} b_{(x, y), r}(s_r) \phi_{k, r}(x, s_r) \right) - v_r + C|w_r|^{p-1} \text{sign}(w_r), \tag{7.27}
\]

with

\[
b_{(x,y), r}(s_r) \propto \exp \left( \frac{\tilde{\phi}_{(x,y), r}(s_r) - \sum_{p \in P(r)} \lambda_{(x,y), r \rightarrow p}(s_r) + \sum_{c \in C(r)} \lambda_{(x,y), c \rightarrow r}(s_c)}{c_r / \beta} \right).
\]

Proof: This is a direct computation of the gradient w.r.t. \(w_r\) and application of Danskin’s theorem in case \(c_r / \beta = 0\).

Putting all the update rules together we obtain our proposed algorithm for structured prediction with latent variables which is summarized by the three steps given within Program 2, i.e., (1) solving the ‘latent variable prediction problem,’ (2) a single update...
7.3. Relationship with Mixture Models and the k-Means Algorithm

7.3.1. Details

Let a scaled multivariate Gaussian distribution with mean vector $\mu \in \mathbb{R}^M$, covariance matrix $\frac{1}{\beta} \Sigma \in \mathbb{R}^{M \times M}$ and the scalar responsibility $\hat{r} \in \mathbb{R}$ be denoted by

$$p(y \mid \mu, \Sigma, \hat{r}) = \frac{\hat{r}}{\sqrt{(2\pi)^M \det \frac{1}{\beta}\Sigma}} \exp\left(-\frac{1}{2} (y - \mu)^\top \left(\frac{1}{\beta} \Sigma^{-1}\right) (y - \mu)\right). \quad (7.28)$$

As a member of the exponential family we equivalently write

$$p(y \mid w) = h(y) \exp\left(w^\top \phi(y) - Z(w)\right) \quad (7.29)$$

with natural parameters $w$ depending on mean $\mu$, covariance $\frac{1}{\beta} \Sigma$ and responsibility $\hat{r}$, sufficient statistics $\phi(y)$, underlying measure $h(y)$ and the log-partition function $Z(w)$. We note that due to summation, a mixture of Gaussian $\sum_i p(y \mid w_i)$ is not a member of the exponential family since it cannot be phrased to follow the form given in Eq. (7.29).

To cast learning such that it fits into the previously proposed framework, suppose our observations $y$ arise from a set of discrete states. Let us further assume that we are not interested in the full output space $\mathcal{S}$ but rather the space of observations $\mathcal{Y}$ identical for all samples. By marginalizing the latent space $\mathcal{H}$ we equivalently rewrite the two-dimensional mixture of Gaussian $\sum_i p(y \mid w_i)$ via

$$p(y \mid w) = \sum_h p(s \mid w) \propto \sum_h \exp\left(\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}^\top \begin{bmatrix} \phi(y)\delta(h = 1) \\ \phi(y)\delta(h = 2) \end{bmatrix}\right), \quad (7.30)$$
7. Learning with Latent Variables

(a) normalized data

(b) $p(s|x)$ for $\beta = 1$

(c) $p(s|x)$ for $\beta = 100$

Figure 7.5.: Given the normalized ‘Old Faithful’ data set illustrated in (a) we infer the means and variances of the mixture model with $\beta = 1$ and $\beta = 100$ illustrated in (b) and (c) respectively.

where $\delta(\cdot)$ refers to the Kronecker delta function also known as Iverson bracket, which takes a value of one if the containing argument is true and evaluates to zero otherwise. Note that marginalizing over arbitrary latent spaces is computationally intractable in general, and marginal maximum a-posteriori (MAP) approximations are required to compute $p(y \mid w)$, e.g., by Liu & Ihler (2011).

Due to the discrete nature of the latent variable $h$ and the assumption of discrete states for the observable $y$, we are nonetheless able to learn the parameters $w$ of a mixture of Gaussian distributions and, in general, a mixture of any member of the exponential family using the framework discussed in Sec. 7.2. The computational complexity depends on the graphical model structure induced by the sufficient statistics. In our case we obtain potentials of order 3.

7.3.2. Experimental Evaluation

To illustrate the results of the proposed framework in case there exist variables which are never observed, we aim at learning the mean vectors and covariance matrices of a 2-dimensional Gaussian distribution. To this end we employ the publicly available ‘Old Faithful’ data set\(^1\) illustrated in Fig. 7.5(a) after normalization and discretization.

The resulting mixture distributions are visualized in Fig. 7.5(b) and Fig. 7.5(c) for $\beta = 1$ and $\beta = 100$ respectively. Note the effects known from the relationship between Gaussian mixture models and the k-means algorithm, i.e., the larger the inverse-temperature parameter $\beta$ the more peaked the distribution.

7.3.3. Summary

This leads us to summarize that our proposed framework which relates latent structured support vector machines (LSSVMs) and hidden conditional random fields (HCRFs),

\(^1\)available from http://research.microsoft.com/en-us/um/people/cmbishop/prml/webdatasets/datasets.htm
7.4. Conclusion

Figure 7.6.: Just like GMM and k-Means being related via a scaling of the covariance with inverse-temperature parameter $\beta$, we design a more general framework to contain HCRF and LSSVM as special instances and illustrate them to generalize GMM and k-Means via the feature vector $\phi$ also known as sufficient statistics.

or max-margin and max-likelihood via $\beta = \infty$ and $\beta = 1$ also generalizes discrete variants of the k-means (Lloyd, 1982) algorithm and Gaussian mixture models (GMM). We provide a visualization of this relationship in Fig. 7.6. Similar to k-Means being a zero temperature variant of a Gaussian mixture model, we argue that the max-margin approach employed within a structured support vector machine with latent variables is a zero temperature variant of the maximum likelihood method referred to as hidden conditional random field. Modifying the vector of sufficient statistics allows to obtain the more general structured prediction framework from well known mixture models.

7.4. Conclusion

Within this chapter we discussed methods for learning with weakly labeled or hidden data. We presented a framework that unifies max-margin and maximum likelihood methods, proposed approximations to be solved exactly and verified that the approach is capable of extracting information from weakly labeled data. In addition we investigated the relationship to mixture models and the k-means algorithm and visualized that the proposed method is also capable of extracting information from data, even if some variables are never observed.
7. Learning with Latent Variables
8. Conclusions and Future Work

8.1. Main Message

In the preceding chapters we discussed multiple contributions to frequently employed inference and learning algorithms. We detailed how to improve efficiency of inference for Random Forest classification in Chapter 3. For structured models considered in Chapter 4 we distributed inference onto multiple computers to decrease the required wall-clock time and derived an efficient solver which is guaranteed to converge to the global optimum of the standard LP relaxation formulation. Similarly, in Chapter 5 we discussed a method to adapt the learning task to high performance computing environments if the models are large. In Chapter 6 we show that the task of predicting a box which best describes the observed room layout given a single image admits a globally optimal inference algorithm. We used this application in Chapter 7 to illustrate that a more efficient learning algorithm permits extraction of information from weakly annotated data. Given the set of algorithms and their features summarized in Appendix A we envisage a couple of generalizations to be discussed in the following.

8.2. Future Work

Numerous extensions for future work, certainly not limited to the directions summarized subsequently, are possible. Note that some of the questions given below might be straightforward to answer and others might have been answered already.

8.2.1. Applications

A first natural approach is usage of the presented and implemented algorithms for applications other than 3D indoor scene understanding. Having presented a few fields and corresponding tasks in the introduction and having discussed in detail the 3D scene understanding application of predicting a 3D parametric box, we imagine that the presented algorithms are useful for many more challenges. In fact the implementations have already been used for 3D reconstruction (Salzmann & Urtasun, 2012), object detection, scene classification and semantic segmentation (Yao et al., 2012), stereo estimation (Yamaguchi et al., 2012), holistic scene understanding (Fidler et al., 2013) and flow estimation (Yamaguchi et al., 2013). Algorithms have also been extended to continuous variables, e.g., by Peng et al. (2012).
8. Conclusions and Future Work

8.2.2. Relationship to DP-mixture and DP-k-means Algorithm

In Sec. 7.3 we showed that the presented structured prediction algorithm generalizes a discrete version of mixture models for exponential family distributions. The temperature parameter relates hidden conditional random fields to latent structured support vector machines just the same way that the Gaussian mixture model relates to the k-means algorithm, i.e., via scaling of the covariance.

Similar to standard mixture models we hence fix the number of clusters a-priori. Bayesian non-parametric models, e.g., the infinite Dirichlet process mixture (Hjort et al., 2010), are more flexible in the sense that the number of clusters is not specified in advance. Following the relationship between ‘soft’ and ‘hard’ assignments of clustering methods, an algorithm named DP-k-means was derived recently by Kulis & Jordan (2012) as a version of Dirichlet process mixtures operating with hard assignments.

Viewing hidden conditional random fields and latent structured support vector machines as extensions of clustering algorithms we ask about generalizations of DP-mixtures and DP-k-means. Existence of those will most likely follow the variational inference scheme presented by Blei & Jordan (2006).

8.2.3. Tightening LP Relaxations

In Sec. 4.3 we designed a globally optimal LP relaxation solver using the formalism of ϵ-subdifferentials. While the presented approach is guaranteed to find the global optimum for a given relaxation, it depends on the tightness of the relaxation whether an integral maximum a-posteriori (MAP) solution to the corresponding integer linear program is equivalent.

To be more specific, we are given a set of potentials \( \theta_r(s_r) \), e.g., of order two, which are part of the model. While we can enforce marginalization constraints between the corresponding beliefs \( b_r(s_r) \) only, it is also possible to include all zeros higher order potentials and corresponding beliefs to enforce marginalization constraints more globally by specifying adequate parent sets \( P(r) \).

Tightening has been suggested as a method to successively approach an integral MAP solution. The proposed methods are governed by heuristic choices on which marginalization constraints to add to the optimization. Importantly we observe that our notation and implementation using Hasse diagrams encompasses factor graphs just like junction trees. While we might obtain an integral MAP solution for some models without including further constraints, a junction tree algorithm and hence tightening up to the tree-width of the model is required to retrieve the MAP solution in other cases.

We are therefore working on how tightening is realized within the ϵ-subdifferential formulation, seeking an understanding on when tightening is required, whether we can determine in advance to which level we need to tighten such that an integral MAP solution is obtained and on relationships with linear programming hierarchies such as Lovász-Schrijver, Sherali-Adams or Laserre.
8.2. Future Work

8.2.4. Latent Variable Models

The optimization approach for latent variable models presented in Chapter 7 leaves a few questions unanswered. Most importantly, we are solving a non-convex approximation irrespective of the choice of counting numbers. We are therefore wondering whether specific choices of counting numbers yield optimization problems that are solvable more efficiently.

In addition we are investigating the consequences of “stacking” latent variables, i.e., constructing models that follow the deep Boltzmann machine architecture (Hinton & Salakhutdinov, 2006; Salakhutdinov & Hinton, 2012). While theoretically solvable by the implemented algorithms we are not sure about resulting performance and use-cases.

8.2.5. Nonlinear Structured Prediction Algorithms

All the learning algorithms presented within this thesis model log-linear distributions. What are the advantages and disadvantages of non-linear extensions? What do we gain when trading linearity for more expressive functions? What are the connections between layers of latent variables and non-linearity? Answering these questions likely yields novel models and algorithms with advantages and disadvantages.

To provide a pointer, suppose our scoring function $\theta(s)$ depends non-linearly on a set of parameters $w$. We can reside to the learning framework for structured models presented within this thesis if we use a linear Taylor expansion of $\theta(s)$ around some arbitrarily chosen but fixed parameter $w_0$. It remains open how to effectively update $w_0$ but a variational approach seems applicable.

8.2.6. Learning Representations

All of the methods presented within this thesis and all of the future directions outlined above assume the model to be given. Hence we assume a domain expert to design a representation for data and answers, i.e., for features $\phi(x, s)$ and output space $S$, such that a tractable learning and inference algorithm that generalizes well to unseen data is obtained. True intelligence is therefore required when imagining possible descriptions for data and answers.

But wouldn’t it be amazing if a computer could take over that task in a fully automatic manner? This question aims at finding a model given only data, i.e., finding causalities in given data. Numerous approaches have been proposed in the past, e.g., summarized by Spirtes et al. (1993) in general and more recently by Jaakkola et al. (2010) using LP relaxations as well as by Janzing et al. (2012) who aim at inferring causal directions in Bayesian networks. Learning representations is also discussed within the deep belief network community, e.g., by Bengio et al. (2012).

In order to obtain true artificial intelligence we therefore have to construct algorithms such that computers imagine representations given data, rather than computers extracting knowledge in terms of parameters, particularly if it holds that:

“The true sign of intelligence is not knowledge but imagination.”

(Albert Einstein)

129
8. Conclusions and Future Work
A. Features of Algorithm Implementations

All the algorithms presented within this thesis are implemented as libraries in C++. The source code packages have at least the generality presented in the previous chapters unless otherwise noted and are publicly available from http://alexander-schwing.de, licensed under GPLv3. We briefly summarize the main features of the implementations subsequently.

A.1. Adaptive Random Forest (ARF)

The random forest library publicly available from http://alexander-schwing.de is just a standard implementation. The modified test procedure discussed in Chapter 3 is available upon request only.

A.2. Distributed convex Belief Propagation (dcBP)

The $dcBP$ package provides distributed inference algorithms for factor graphs only. Hence no support for the more general region graphs is implemented. If required we refer the reader to Sec. A.3 for a more general implementation. The main features of the $dcBP$ package are as follows:

- **Generality:** Support for both computation of marginals as well as computation of the most likely labeling, \textit{i.e.}, the maximum a-posteriori task with arbitrary counting numbers for variables and factors within the factor graph.

- **Support for high performance computing:** The package parallelizes inference w.r.t. the partitioning provided by the user and w.r.t. graph coloring of the factor graph to make best use of distributed and shared memory architectures. To suit large clusters we utilize standard message passing libraries like OpenMPI or MPICH.

- **Input formats:** The user can either employ the UAI file format\footnote{see http://www.cs.huji.ac.il/project/PASCAL/fileFormat.php for details} of the probabilistic inference challenge or a custom file format to specify counting numbers and the partitioning of the inference task onto multiple machines.
A.3. Distributed Structured Prediction (dSP)

The dSP package provides distributed learning algorithms and additionally, extends distributed inference of the dcBP package to Hasse diagrams. We provide a summary of the main features of the dSP package in the following:

- **Sparse potentials:** The dSP implementation differentiates between sparse and dense potentials to handle memory more efficiently.

- **Support for higher order regions:** Arbitrary Hasse diagrams are taken as input for both learning and inference. Therefore dSP supports junction tree optimization just like ordinary message passing on a factor graph. In addition and at the expense of exponential complexity, models can be manually tightened up to the tree-width.

- **Support for arbitrary counting numbers:** The user is free to specify the counting numbers. Therefore, non-convex Bethe approximations are supported just like convex belief propagation. Parameters determine whether updates for the Lagrange multipliers are interleaved with gradient steps for the model parameters.

- **Suitable for hidden conditional random fields or structured support vector machines:** Changing the parameter $\beta$ allows optimization of the structured support vector machine (SSVM) or max-margin Markov network (M3N) objective (max-margin) for $\beta = \infty$ just like the conditional random field (CRF) cost function (max-likelihood) for $\beta = 1$.

- **Support for high performance computing:** The package contains modules that parallelize learning w.r.t. samples on a single machine or w.r.t. the graph coloring of the Hasse diagram. In addition, learning for distributed memory environments supports parallelization either by partitioning the model or by distributing the samples. Similarly, parameters determine whether inference processes multiple samples concurrently or whether a single model is solved in parallel using graph coloring. To suit large clusters we utilize standard message passing libraries like OpenMPI or MPICH. To save time for the transmission, we also merge messages of different regions into a single package.

- **Different input formats:** We designed both a textual as well as a binary input file format. While the first is suitable for debugging purposes, the latter targets large scale data sets. Besides file formats a Matlab mex function provides direct access to the modules.

- **Latent variable models:** We provide a module to learn from weakly labeled data, i.e., we support latent structured support vector machines (Yu & Joachims, 2009) just like hidden conditional random fields (Quattoni et al., 2007) by following Schwing et al. (2012b). Parallelization w.r.t. samples and by graph coloring is also supported for latent variable models.
A.4. Cutting Plane Structured Prediction Algorithm

The provided implementation for learning structured predictors using a cutting plane algorithm was, among others, employed to learn the parameters of the presented 3D layout estimation task. The main features of this package are:

- **Support for high performance computing**: The package distributes the training set onto multiple computers within a compute cluster such that the task of finding the most violated constraint is performed in parallel.

- **Warm starting**: The algorithm can store its entire state on the hard disk such that we can resume learning at any iteration if required.

A.5. Globally convergent LP Relaxation Solver (gcLPR)

The gcLPR package to be made available reads factor graphs specified via UAI files but also operates with the more general Hasse diagrams. The latest implementation is free of any library dependencies. Since the internal data structures are equivalent, it should be straightforward to include gcLPR as a solver into the dSP framework.
Bibliography


Bibliography


Bibliography


Liu, Q. and Ihler, A. Variational algorithms for marginal MAP. In *Proc. UAI*, 2011.


Bibliography


Bibliography


Bibliography


