Universal Exact Algorithm for Globally Augmented MAP Inference in Structured Prediction

vorgelegt von
Dipl.-Inform.
Alexander Bauer
geb. in Kopejsk

Von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Thomas Wiegand
1. Gutachter:	Prof. Dr. Manfred Opper
2. Gutachter:	Prof. Dr. Benjamin Blankertz
3. Gutachter:	Prof. Dr. Marius Kloft

Tag der wissenschaftlichen Aussprache: 27 November 2017

Berlin 2017

"Nothing is more practical than a good theory."

V. Vapnik

Technische Universität Berlin

Abstract

Universal Exact Algorithm for Globally Augmented MAP Inference in Structured Prediction

by ALEXANDER BAUER

The ultimate goal of discriminative learning is to train a prediction system by optimizing a desired measure of performance. Unlike in the standard learning scenario with univariate real-valued outputs, in structured prediction we aim at predicting a structured label corresponding to complex objects such as sequences, alignments, sets, or graphs. Here, structural support vector machine (SSVM) enables us to build complex and accurate models and directly integrate the desired performance measure into the optimization process. However, it relies on the availability of efficient inference algorithms — the state-of-the-art training algorithms repeatedly perform inference either to compute a subgradient or to find the most violating configuration. In the literature, the corresponding computational task is generally referred to as loss augmented (or adjusted) inference and is the main computational bottleneck during the training procedure. When the loss function is decomposable we often can perform inference efficiently by using the same algorithm we use for evaluating the prediction function. However, the most popular loss functions are non decomposable (or high order) and require new inference algorithms to benefit from learning with task-dependent loss functions.

The main goal of the thesis is to address the computational difficulties inherent in a family of discrete optimization problems which we here refer to as augmented MAP inference. In particular, our main focus is on the exact inference, which is known to be NP-hard in general. As the main result, I define a large class of tractable problem instances within the framework of graphical models and derive an exact message passing algorithm which always finds an optimal solution in polynomial time. The latter is universal in a sense that its applicability does not explicitly depend on the graph structure of a corresponding model but rather on the intrinsic properties like treewidth and number of states of the auxiliary variables. Moreover, it leaves the global interactions between the energy of the underlying model and the sufficient statistics of the global terms largely unspecified.

Due to its generic form, the presented algorithm can also be used in other application scenarios than as a subroutine for loss augmented inference in a training algorithm including evaluation of PAC-Bayesian generalization bounds for structured prediction and globally constrained MAP inference. I demonstrate its practical usefulness by training an SSVM for various non decomposable loss functions on the example of a few different applications of increasing complexity. To cope with more complex dependencies I analyze an accurate approximation framework of Lagrangian relaxation and dual decomposition in the context of globally augmented MAP inference.

Zusammenfassung

Der ultimative Zweck des diskriminativen Lernens is das Trainieren eines Vorhersagesystems durch das Optimieren einer gewünschten Zielfunktion. Anders als in dem herkömmlichen Ansatz des überwachten Lernens mit eindimensionalen reellwertigen Funktionen, die Labels in einer strukturellen Prädiktion (structured prediction) repräsentieren komplexe Objekte wie Sequenzen, Ausrichtungen, Mengen oder allgemeinere Graphen. Die strukturellen Support Vector Maschinen (SSVM) ermöglichen das Konstruieren von akkuraten und robusten Modellen für das Lernen mit strukturierten Ausgaben und integrieren das gewünschte Optimierungskriterium direkt in den Lernprozess. Jedoch, eine wichtige Voraussetzung dafür ist das Vorhandensein von effizienten Inferenzalgorithmen. Nämlich, die state-of-the-art Trainingsalgorithmen nutzen wiederholt einen Inferenzalgorithmus zum Berechnen eines Subgradienten oder Bestimmen einer maximal verletzenden Konfiguration. In der Literatur, das entsprechende Problem is bekannt als Loss Augmented Inference (mit Verlust erweiterte Inferenz), die den Flaschenhals in der Berechnung darstellt. Im Falle einer zerlegbaren Verlustfunktion können wir den gleichen Algorithmus verwenden zur Inferenz wie für das Auswerten von der Vorhersagefunktion. Jedoch sind die meisten der populären Verlustfunktionen nicht zerlegbar, so dass neue Inferenzalgorithmen notwendig sind um von dem Lernen mit diesen Funktionen zu profitieren.

In dieser Dissertation präsentiere ich eine generische Sicht auf das erweiterte Inferenzproblem aus der Perspektive einer exakten Berechnung, die im allgemeinen Fall NP-hart ist. Als ein Hauptergebnis definiere ich eine große Klasse von berechenbaren Probleminstanzen mithilfe der graphischen Modelle und präsentiere einen exakten Algorithmus mit einer polynomiellen Laufzeit. Wegen seiner generischen Form kann dieser Algorithmus auch zum Lösen weiterer Probleme eingesetzt werden wie zum Beispiel das Auswerten von Generalisierungsschranken für strukturelle Prädiktion und MAP Inferenz mit globalen Bedingungen. Ich demonstriere den praktischen Nutzen von diesem Algorithmus anhand des Trainierens von SSVMs für mehrere Beispielapplikationen mit unterschiedlichen nicht zerlegbaren Verlustfunktionen. Zusätzlich untersuche ich den Ansatz der Lagrangian Relaxiereung and der dualen Zerlegung zur Handhabung von komplexeren Beziehungen zwischen den Variablen eines Models.

Acknowledgements

At this point I want to use the opportunity to thank all the people who has supported me in various ways during my PhD time. First of all, I would like to thank my PhD advisors Prof. Dr. Klaus-Robert Müller for providing the necessary guidance and working environment in his Machine Learning Laboratory from the very beginning of my PhD studies. Moreover, I thank all members of his lab for creating such a pleasurable working atmosphere in the office each day. In particular, I thank two of my colleagues Nico Görnitz and Shinichi Nakajima for having long and fruitful discussions.

Contents

List of Figures

List of Tables

List of Abbreviations

CFG	Context free grammar
CKY	Cocke-Kasami-Younger
CNF	Chomsky normal form
CPA	Cutting plane algorithm
CRF	Conditional random field
DD	Dual decomposition
HMM	Hidden Markov model
HOP	high order potential
LP	Linear programming
MAP	Maximum a posteriori
MRF	Markov random field
MLN	Markov logic network
PCFG	Probabilistic context free grammar
PRM	Probabilistic relational model
SSVM	Structural support vector machine

List of Symbols

$\boldsymbol{x}$	vector containing input variables
$\boldsymbol{y}$	vector containing output variables
$\boldsymbol{l}_m$	vector-valued auxiliary variable with index m
C_k	a clique of variables from $\boldsymbol{y}$ with index k
$\boldsymbol{w}$	weight vector
$\boldsymbol{\Psi}$	joint feature map
Δ	loss function
$\mathbb{1}_\alpha[\cdot]$	indicator function yielding α if the statement in the brackets $[\cdot]$ is true and 0 otherwise
$\delta(\cdot,\cdot)$	Kronecker delta function yielding 1 if the arguments are equal and 0 otherwise
$\odot$	binary algebraic operation $\odot \in \{+,\cdot\}$
$\otimes$	direct tensor product, $(a \otimes b)_{i+(j-1)\cdot dim(a)} = a_i \cdot b_j$
M	the number of variables (or graph nodes) in the output
N	the maximal number of states for an output variable in $\boldsymbol{y}$
R	the maximal number of states for each auxiliary variable
τ	treewidth of a graph
ν	the maximal number of neighbors of a node in a given clique tree
$\boldsymbol{\theta}$	vector of weights in the standard overcomplete representation
$\boldsymbol{\mu}$	joint variable assignment in the standard overcomplete representation
$\mathcal{X}_G$	the set of valid assignments to the variables in an MRF G in the standard overcomplete representation
$\mathcal{M}_G$	marginal polytope for a graph G
L_G	local consistency polytope for a graph G
$\boldsymbol{u}$	dual variables

Chapter 1

Introduction

This chapter introduces the basic setting of structured output prediction and highlights the main contributions of the thesis. Here, we only provide a high-level discussion and defer to Chapter 2 for a formal introduction of this topic.

1.1 Structured Output Prediction

There has been a lot of progress in machine learning during the last two decades. However, learning general functional dependencies between arbitrary input and output spaces is still one of the key challenges in computational intelligence. Here, the framework of *structured output prediction* [TGK03; Tso+05] provides the necessary tools by generalizing the standard paradigms of supervised learning for training prediction systems with *multivariate* and *interdependent* outputs. Namely, unlike in the standard regression or classification tasks where predictions are real numbers, in structured prediction, we aim at predicting a structured label corresponding to complex objects such as sequences, alignments, sets, or graphs. Consider for example the task of predicting a sequence of part-of-speech tags for every word in a sentence (see Figure 1.1). In a simplified version we assume that possible tags can be only a noun, verb, determiner (dt), preposition (prep) or date. Clearly, the tag identity for

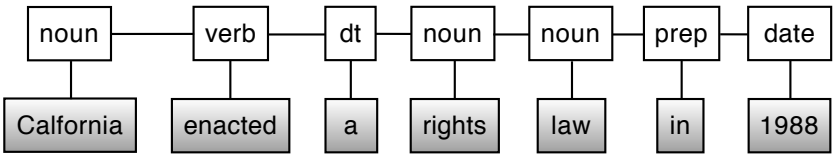


Figure 1.1: Simplified part-of-speech tagging example. The shaded rectangles represent the individual words in a sentence and the white rectangles the corresponding tags. The edges show the explicit dependencies in a corresponding model. Each tag is affected by the identity of the underlying word and by the identity of the neighbor tags.

a word at any position in the sentence is affected by the neighboring tags according to the grammar of a corresponding language. Some tags are more likely to follow each other than the others. For example, it is very likely for a determiner to be followed by a noun (e.g., "the house") while transitions between two determiners (e.g., "the the") never occur. While introducing such dependencies between the individual output variables generally increases the accuracy of the model, in the prediction and training phase we then have to reason over some variables jointly resulting in a more involved optimization. In this thesis we build on the framework of graphical models such as *Markov random fields* (MRFs) [KF09] which provide a formalism for describing joint probability distributions over a set of random variables by using a graphical representation.

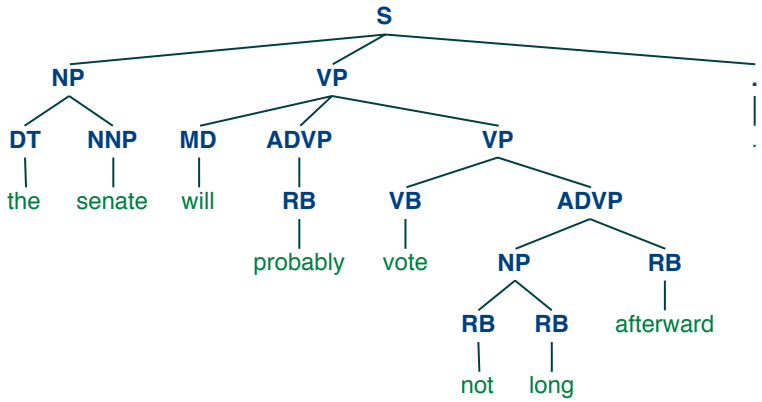
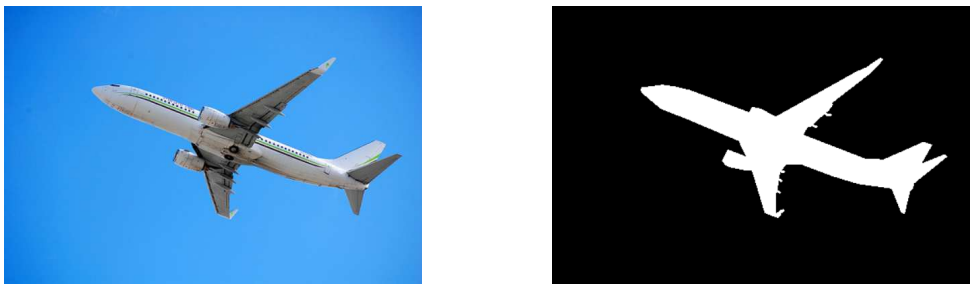


Figure 1.2: Illustration of a parsing example for the input sentence: "The senate will probably vote not long afterward .". The intermediate symbols above the words represent the part-of-speech tags and the remaining nodes describe how the individual words are organized into phrases. For example the first two words build a noun phrase (NP) and the remaining words belong to a verb phrase (VP).

Another example for learning with structured outputs is the task of natural language parsing where the goal is to predict a parse tree describing the phrase structure for a given input sentence as illustrated in Figure 1.2. Here, the individual output variables form a tree structure where each node together with its children corresponds to an application of a production rule according to some grammar formalism. In linguistics parsing helps to understand the exact meaning of a sentence by identifying the grammatical components, e.g., such as subject and predicate in English. More generally, it aims to assign an explicit structure to seemingly unstructured text and is an important intermediate step in the language processing pipeline. Namely, the corresponding syntactic structure allows to relate the semantic content of the individual word groups providing a powerful representation for the following analysis of the meaning of a given sentence. Note that the type of the corresponding dependencies cannot be expressed by an unstructured model as not every configuration of node labels results in a valid parse tree with respect to the given grammar.

A further example with a more dense dependency structure of the output variables presents the task of foreground-background image segmentation (see Figure



(a) Input image **(b)** Segmentation image

Figure 1.3: The input image on the left is segmented into foreground and background pixels to the right. The white pixels correspond to the foreground and the black pixels to the background. The images are taken from the dataset provided by the Pascal VOC challenge 2012.

1.3). The goal here is to segment the pixels in a given image into foreground and background. Every pixel (or a group of pixels) in the image can be represented by a single variable which is connected to their intermediate neighbors forming a grid-like structure. Here, the weights of individual pixels encode the preference of that pixel to be in a specific class and the edge weights can be used to enforce additional smoothness constraints on the output variables. Therefore, to find an optimal solution we need to perform a joint inference over all the pixels in the image.

Given a suitable problem representation the next question is how to adjust the corresponding model parameters according to a chosen measure of performance. Among popular learning algorithms is the structural support vector machine (SSVM) [Tso+05; Joa+09] which enables us to build complex and accurate models for structured prediction. However, it relies on the ability to perform exact inference during the training procedure. For this reason, since the original work on SSVMs the lack of the corresponding inference algorithms has been the main obstacle in using this framework for a range of practical applications.

1.2 Motivation and Use Cases

The research conducted within the scope of this thesis has been motivated by the following use cases:

- **Learning with High Order Loss Functions**

  The ultimate goal of discriminative learning is to train a prediction system by optimizing a desired measure of performance. Unlike in the standard learning scenario with univariate real-valued outputs, in structured prediction we aim at predicting a structured label corresponding to complex objects such as sequences, alignments, sets, or graphs. As a result, many structured prediction tasks have their own measure of performance, such as Hamming distance in sequence tagging, F_1-score in constituency parsing [MS99], intersection over union in image segmentation [Eve+15], or more task-related measures like BLEU in machine translation [Pap+02] or ROUGE in text summarization [Lin04], just to mention a few.

  On the other hand, structural support vector machine (SSVM) enables us to build complex and accurate models for structured prediction, and directly integrate the desired performance measure into the optimization process. However, its applicability relies on the availability of efficient inference algorithms — the state-of-the-art training algorithms (cutting planes [Kel60; JFY09], bundle methods [SVL07; Teo+10], subgradient methods [RBZ07a; RBZ07b], and Frank-Wolfe optimization [Lac+13]) repeatedly perform inference either to compute a subgradient or to find the most violating configuration. In the literature, the corresponding computational task is generally referred to as the *loss augmented (or adjusted) inference* and is the main computational bottleneck during the training procedure.

  In the case where the loss function is decomposable, that is, factorizes similarly to the prediction function, the two can be folded together and we can use for the task of loss augmented inference the same algorithm we use for evaluating the prediction function. However, the most popular loss functions are non decomposable (or high order) and require new inference algorithms to benefit from learning with task-dependent loss functions. Note that the state of the research preceding the work in this thesis allowed for training with non

decomposable loss functions only via approximations: by either performing approximate loss augmented inference or by approximating the loss with simple functions. Both approaches result in a suboptimal performance. Most importantly, however, the theoretical guarantees[1] of the corresponding training algorithms strongly rely on the assumption that the separation oracle for the loss augmented inference is exact and do not necessarily hold with approximations.

- **Enabling training of slack scaling formulation for structural SVMs**

  The maximum-margin framework of SSVMs comes with two loss-sensitive formulations known as the *margin scaling* and *slack scaling*. Since the original paper on SSVMs [Tso+05] there has been lots of speculation about the differences in training by either of the two formulations. In particular, training via slack scaling has been conjectured to be more accurate and beneficial than margin scaling. Nevertheless, it has been rarely used in practice due to the lack of known efficient inference algorithms. Even if the loss function is decomposable, due to the multiplication with the loss term the resulting objective shows no decomposable structure rendering the use of the established message passing algorithms intractable. Consequently, one had to sacrifice exact inference with approximate methods to benefit from slack scaling formulation — no general exact algorithm has been proposed previous to the work in the present thesis.

- **Evaluation of generalization bounds for structured prediction**

  Generalization bounds can give useful theoretical insights in behavior and stability of a learning algorithm by upper bounding the expected loss or the risk of a prediction function. Evaluating such a bound could provide certain guarantees how a system trained on some finite data will perform in the future on the unseen examples. Unlike in the standard regression or classification tasks with univariate real-valued outputs, in structured prediction, the existing bounds often involve complex terms imposing dense structural dependencies on the prediction variables. This results in a complex combinatorial optimization problem preventing a simple evaluation. Furthermore, the availability of an efficient inference algorithm could potentially motivate the development of new training algorithms aiming to directly minimize a corresponding bound.

- **Globally Constrained MAP Inference**

  In the general task of learning with structured outputs, the prediction function is defined as a maximization of a compatibility function between an input and a discrete output over a finite set of all possible outputs. In many cases, evaluating this function on a given input, technically corresponds to performing MAP inference on a discrete graphical model including Markov random fields (MRFs) [KF09], probabilistic context free grammars (PCFGs) [MS99; Joh98; Hee93; Cha97], hidden Markov models (HMMs) [Rab89], conditional random fields (CRFs) [Laf01], probabilistic relational models (PRMs) [Kol99; TAK13], and Markov logic networks (MLNs) [RD06]. This inference task is ubiquitous in structured learning and is involved in both phases, training and prediction.

  While simple models benefit from the existence of efficient inference algorithms, at the same time they lack of expressiveness and cannot describe more

[1] This includes the following guarantees: 1) polynomial time termination 2) correctness and 3) empirical risk bound. See Chapter 2 for more details.

complex variable interactions necessary for achieving a higher performance on a given task. In practice, however, we might want to modify the prediction function by imposing additional (global) constraints on its output. For example, we could perform a corresponding MAP inference subject to the constraints on the label counts specifying the size of an output or the distribution of the resulting labels, which is a common approach in applications like sequence tagging and image segmentation. Alternatively, we might want to get the best output with a score from a specific range which can provide deeper insights into the energy function of a corresponding model. Finally, we might want to restrict the set of possible outputs directly by excluding specific label configurations. The latter is closely related to the computational task known as the *(diverse) k-best MAP inference* [Bat+12; GKB13].

Note that instead of enforcing the global constraints directly we could augment the objective with a corresponding penalty term in form of a high order potential modifying the mode of the resulting distribution towards outputs with small constraint violation. Dealing with models involving global factors is also a common use case in many application areas including computer vision, information retrieval, computational biology, and natural language processing.

1.3 Thesis Outline and Own Contributions

The main goal of the thesis is to address the computational difficulties inherent in the general recurrent task of (augmented) MAP inference which plays a central role in structured prediction. We set up a corresponding problem within the framework of graphical models which allows for a concise theoretical analysis and provides an access to a range of powerful modeling techniques. It is well known that inference on models with bounded treewidth can be performed efficiently. Given such a model, we here consider two types of augmentation: 1) extending the model by additional global factors and 2) imposing global constraints on the output variables. In particular, we consider global cardinality-based potentials defined as an arbitrary function of multivariate label statistics. Global factors of this form are very useful and can model complex nonlinear variable interactions. For example, in the important task of loss augmented MAP inference, the most of the existing (non decomposable) loss functions can be represented by such a global factor. On the other hand such factors also could encode hard constraints on the model by means of indicator functions excluding invalid configurations. However, the above augmentation expressed as an MRF results in a fully connected graph preventing the use of established exact inference algorithms[2]. More specifically, the main contributions of this thesis can be summarized as follows. A more detailed summary of contributions can be found at the end of each chapter.

1. For the general task of (globally) augmented MAP inference in structured prediction we define a large class of tractable problem instances which can be solved efficiently. More precisely, we consider models with **bounded** treewidth, which additionally have been augmented by either a **global** potential or a **global** constraint. In particular, this covers the task of loss augmented inference for many **non decomposable** loss functions for both margin **and** slack scaling formulation of SSVM.

[2]This includes max-product (or max-sum algorithm) [Bis06] and junction tree algorithm [KF09; MC10] for MAP inference in discrete graphical models.

2. We derive an **exact** message passing algorithm for the tractable case, which is guaranteed to find an optimal solution in **polynomial time** with respect to the problem size. We provide a rigorous theoretical analysis of the computational complexity and prove the correctness of the presented algorithm. We emphasize that the presented algorithm **for the first time** enables training SSVMs for various **non decomposable** loss functions for a range of practical applications.

3. We show how the derived algorithmic framework can benefit **multiple** application scenarios including the loss augmented inference for SSVMs, evaluation of generalization bounds for structured prediction, and the general task of globally constrained MAP inference, which covers the **diverse** k-best MAP inference as a special case.

4. We also (theoretically) analyze an accurate and efficient approximation framework based on the technique of Lagrangian relaxation and dual decomposition, which can handle (augmented) inference problems with **unbounded** treewidth. As a result, we provide novel insights connecting the two important optimization techniques of LP relaxation and dual decomposition for discrete MAP inference. In particular, we show that in the case of binary **pairwise** MRFs both methods share the **partial optimality** property of their solutions. Combined with the results of previous works we can conclude that, in this case, the two methods are indeed **equivalent**. However, dual decomposition has a key advantage over LP relaxation because of its **distributed** computational scheme.

5. **For the first time**, we empirically analyze the effect of training an SSVM with various **non decomposable** loss functions (for margin **and** slack scaling) by using **exact** inference during the training procedure. More specifically, we consider three different examples of increasing complexity (with respect to the output structure) including learning tasks such as sequence tagging, sequence segmentation and natural language parsing with context free grammars.

The thesis is organized as follows.

In Chapter 1, I start with an informal introduction of structured output prediction, explain the motivation for the conducted research, and provide a road map through the thesis.

Chapter 2 provides some basic terminology and thematic background for the subsequent discussions in the thesis. In particular, it contains a customized introduction in the field of discrete graphical models and the related topics on graph transformations required for the main analysis in the following chapter.

In Chapter 3, I present an algorithmic framework for performing discrete MAP inference on models with bounded treewidth, which additionally have been augmented by either a global potential or a global constraint. Moreover, I provide the theoretical analysis of correctness and time complexity of the corresponding computations. The usability of the presented algorithm is demonstrated on an example of a few different use cases.

In Chapter 4, for the sake of completeness, I discuss known approximate methods to address models with unbounded treewidth. More specifically, I analyze an efficient framework of Lagrangian relaxation and a related technique of dual decomposition applied to the augmented MAP problem. Additionally, I investigate the relation between the popular optimization techniques of LP relaxation and dual decomposition and show that for the special case of binary pairwise models both techniques share the partial optimality property of their solutions.

In Chapter 5, I analyze the effect of the proposed exact algorithms on learning procedure of SSVMs for three different applications including tasks for sequence tagging, sequence segmentation and natural language parsing. For each application, I give a compact description of the learning problem along with task-related details for the corresponding inference algorithm. The experimental results are presented at the end of each application section.

In Chapter 6, we summarize the results from the individual chapters and discuss the theoretical and practical implications of the corresponding contributions.

1.4 Previously Published Work

The following publications are included in parts or in an extended version in this thesis:

- A. Bauer, N. Görnitz, F. Biegler, K.-R. Müller, and M. Kloft. **Efficient Algorithms for Exact Inference in Sequence Labeling SVMs**. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, May 2014. [Bau+14]
- A. Bauer, M. Braun, and K.-R. Müller. **Accurate Maximum-Margin Training for Parsing with Context-Free Grammars**. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, Dec 2015. [BBM17]
- A. Bauer, S. Nakajima, and K.-R. Müller. **Efficient Exact Inference With Loss Augmented Objective in Structured Learning**. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, Aug 2016. [BNM16]
- A. Bauer, S. Nakajima, N. Görnitz and K.-R. Müller. **Partial Optimality of Dual Decomposition for MAP Inference in Pairwise MRFs**. *arXiv preprint, 2017.* [Bau+17b]
- A. Bauer, S. Nakajima, N. Görnitz and K.-R. Müller. **Optimizing for Measure of Performance in Max-Margin Parsing**. *arXiv preprint, 2017.* [Bau+17a]

Additional research work not covered by the material in this thesis can be found in:

- J. Höner, S. Nakajima, A. Bauer, K.-R. Müller, and N. Görnitz. **Minimizing Trust Leaks for Robust Sybil Detection**. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Aug 2017. [Hön+17]

Chapter 2

Background

In this chapter we introduce some basic terminology and the thematic background of the thesis. In particular, we provide here a customized introduction in the framework of discrete graphical models such as Markov random fields and the related topics on graph transformations required for the main analysis in the following chapters. Moreover, we provide a noticeable new result regarding the properties of margin scaling as summarized below.

The main **contributions** in this chapter are the following:

- We show that margin scaling formulation of an SSVM is scaling-invariant with respect to the loss function in the sense of Proposition 3. Basically, it implies that we can deal with different scalings of the loss function implicitly by appropriately adjusting the value of the hyperparameter C.

Parts of this chapter including Section 2.2 are based on:

A. Bauer, N. Görnitz, F. Biegler, K.-R. Müller, and M. Kloft. Efficient Algorithms for Exact Inference in Sequence Labeling SVMs. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* © 2014 IEEE. [Bau+14]

2.1 Undirected Graphical Models

A *Markov network* or *Markov random field* (MRF) [KF09] is an undirected graph describing a family of joint probability distributions over a set of random variables by specifying a common set of conditional independence properties which must hold for every distribution in the family. The nodes in the graph of an MRF represent the variables, and the edges indicate direct probabilistic interactions between the neighboring variables. More precisely, the explicit form of a graph encodes the conditional independence properties of a corresponding distribution which can be verified by using a technique known as the graph separation. Namely, consider an MRF in Figure 2.1 and three different sets of nodes A, B, and C. To verify whether a conditional independence property $P(A, B|C) = P(A|C)P(B|C)$ holds for probability distributions defined by a corresponding MRF we have to consider all possible paths that connect the nodes in the set A to nodes in the set B. If every such path is blocked by at least one of the nodes from the set C, the independence property holds.

2.1.1 Gibbs Distribution and Markov Networks

Definition 1 (Gibbs Distribution)**.** *A distribution P_Φ is a Gibbs distribution parameterized by a set of factors $\Phi = \{\phi_1(C_1), ..., \phi_K(C_K)\}$, $C_k \subseteq \{y_1, ..., y_n\}$ if it is defined as*

$$P_\Phi(y_1, ..., y_n) = \frac{1}{Z}\tilde{P}_\Phi(y_1, ..., y_n)$$

where $\tilde{P}_\Phi(y_1, ..., y_n) = \phi_{C_1}(C_1) \times \phi_{C_2}(C_2) \times ... \times \phi_{C_K}(C_K)$ is an unnormalized measure and $Z = \sum_{y_1,...,y_n} \tilde{P}_\Phi(y_1, ..., y_n)$ is a normalizing constant called the partition function.

We say that a distribution P_ϕ with factors $\Phi = \{\phi_1(C_1), ..., \phi_K(C_K)\}$ *factorizes* over an MRF $\mathcal{H}$ if each C_k is a fully connected subgraph of $\mathcal{H}$. The factors that parametrize an MRF are also called *clique* potentials. Without loss of generality we can assume that factors are defined over maximal cliques.

2.1.2 Factor Graphs

An MRF does not generally reveal all the structure in a Gibbs distribution. In particular, we cannot see from a graph if the individual factors in the parametrization are defined over maximal cliques or subsets thereof. An alternative with a more finer-grained parametrization of an MRF provide the factor graphs. A factor graph makes explicit the factor structure in a network. It contains two types of nodes for the variables (represented by circles) and additional nodes for factors (represented by squares). Each factor node is connected to all the variables in its scope. An example is illustrated in Figure 2.2.

The simplest way to parametrize a distribution is by listing all possible values of the corresponding factors in a table. An alternative parametrization provide the so-called *log-linear* models which can result in a much more compact representation for many distributions. We say that a distribution P is a *log-linear* model over a Markov network $\mathcal{H}$ if it is associated with:

- a set of features $\{f_1(C_1), ..., f_k(C_k)\}$ where each C_i is a complete subgraph of $\mathcal{H}$,
- a set of weights $w_1, ..., w_k$,

such that

$$P(y_1, ..., y_n) = \frac{1}{Z}\exp\left[-\sum_{i=1}^{k} w_i f_i(C_i)\right]. \tag{2.1}$$

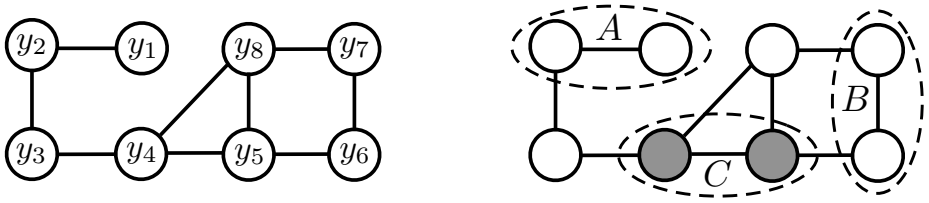


Figure 2.1: Illustration of an MRF example. The graph on the left presents an MRF over the variables $y_1, ..., y_8$. The graph on the right illustrates a conditional independence property $P(A, B|C) = P(A|C)P(B|C)$ for the sets $A = \{y_1, y_2\}$, $B = \{y_6, y_7\}$, and $C = \{y_4, y_5\}$. Using the graph separation technique we can verify that sets A and B are conditionally independent given the values of the variables in C.

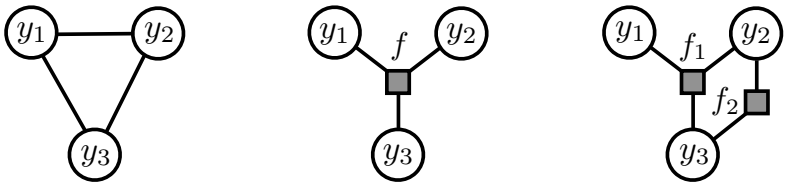


Figure 2.2: Finer-grained parametrization via a factor graph. The graph on the left presents an MRF over the the variables y_1, y_2, y_3, which contains only one (maximal) clique. The graph in the middle and on the right illustrates two possible finer parametrization of the MRF on the left. All three graphs can define the same distribution.

A probability distribution described by a log-linear model is also a Gibbs distributions since we can define the corresponding factors via $\phi(C) := \exp(-w \cdot f(C))$. The term $\varepsilon(C) = -\ln \phi(C)$ is often called the *energy function*.

2.1.3 Cluster Graphs

Here, we introduce the notion of a *cluster graph* — a data structure that provides a graphical flowchart of factor-manipulation processes in a graphical model. Each node in a cluster graph is associated with a subset of variables of a corresponding MRF. The graph also contains edges connecting cluster nodes whose scopes have non-empty intersection.

Definition 2 (Cluster Graph). *A cluster graph for a set of factors Φ over the variables $\mathcal{Y} = \{y_1, ..., y_n\}$ is an undirected graph, each of whose nodes i is associated with a subset $C_i \subseteq \mathcal{Y}$. A cluster graph must be family-preserving — each factor $\phi \in \Phi$ must be associated with a cluster C_i such that $scope[\phi] \subseteq C_i$. Each edge between two clusters C_i and C_j is associated with a sepset $S_{i,j} \subseteq C_i \cap C_j$.*

In the following we define an important property for a cluster graph which provides a fundamental basis for exact inference algorithms.

Definition 3 (Running Intersection Property). *Let $G = (\mathcal{V}, \mathcal{E})$ be a cluster graph over a set of factors Φ where $\mathcal{V}$ denotes the set of cluster nodes and $\mathcal{E}$ the set of the corresponding edges in G. We say that G has the running intersection property if, whenever there is a variable y such that $y \in C_i$ and $y \in C_j$, then y is also in every cluster in the (unique) path in G between C_i and C_j.*

If the cluster graph is a tree, then the running intersection property implies that all the nodes containing a variable y form a tree. In particular, every path in the tree is unique. Note that this property also implies that $S_{i,j} = C_i \cap C_j$. Finally, we introduce the following data structure.

Definition 4 (Clique Tree). *Let Φ be a set of factors over $\mathcal{Y}$. A cluster tree over Φ that satisfies the running intersection property is called a clique tree (also called a junction tree). In that case the clusters are also called cliques.*

We now shortly explain how a clique tree can be constructed from an MRF with a set of factors Φ. For this we need the concept of a chordal graph: an undirected graph is *chordal* if it contains no cycle of length greater than three without shortcuts, that is, every minimal loop has the length three. Consider for example an MRF in Figure 2.1 on the left. It contains a chordless cycle through the variables y_5, y_6, y_7, y_8 and is, therefore, not chordal. There are two basic approaches how to construct a clique tree for a given undirected graph. The first is based on the variable elimination algorithm

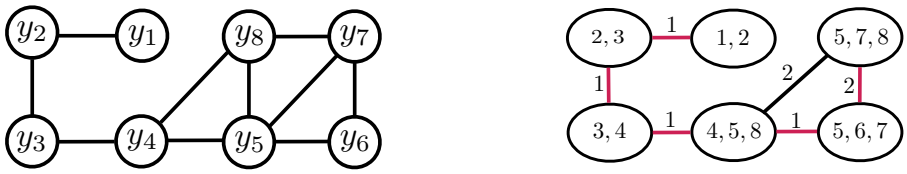


Figure 2.3: Clique tree construction for the MRF in Figure 2.1. To get a triangulated graph we added an additional edge between the nodes y_5 and y_7 to fix the only chordless cycle y_5-y_6-y_7-y_8. This results in a chordal graph on the left. Then we identify the maximal cliques which become the nodes in a cluster graph (on the right). The edge weights are given by the cardinality of the corresponding sepsets. The red edges mark a possible clique tree which we get by running the maximum spanning tree algorithm.

[KF09] and the second on direct graph manipulation. For any MRF $\mathcal{H}_\Phi$ with a set of factors Φ we can build a clique tree as follows. In the first step called *triangulation*, we construct a chordal graph from $\mathcal{H}_\Phi$. Given a chordal graph $\mathcal{H}^*$ the next step is to find the maximal cliques in the graph. One of the simplest methods is to run the maximum-cardinality search on it. Finally, to determine the edges in the clique tree we can use the maximum spanning tree algorithm where the weights of the edges are given by the cardinality of the corresponding sepsets $S_{i,j}$. To summarize we can construct a clique tree as follows:

1. Given a set of factors construct an undirected graph $\mathcal{H}_\Phi$
2. Triangulate $\mathcal{H}_\Phi$ to construct a chordal graph $\mathcal{H}^*$.
3. Find maximal cliques in $\mathcal{H}^*$ and make each one a node in a cluster graph.
4. Run the maximum spanning tree algorithm on the cluster graph to construct a tree.

Figure 2.3 illustrates the above procedure on an example. We note that in the graph theory the concept of a clique tree is often referred to as *tree decomposition* of a graph. It lies at the heart of exact inference algorithms for computing the marginal probabilities or finding a *maximum a posteriori* (MAP) assignment[1] of a corresponding joint distribution over a set of random (discrete) variables. For example, running the max-sum algorithm [Bis06] on a clique tree results in a computational time which is exponential in the maximal size of a clique in a corresponding clique tree. The maximal size of a clique in a clique tree is tightly connected to the notion of the *treewidth* of a graph in the following sense. Consider an arbitrary (loopy) MRF $\mathcal{H}$ and a corresponding clique tree $\mathcal{T}$. The maximal size of a clique in the clique tree $\mathcal{T}$ minus 1 is referred to as the *width* of $\mathcal{T}$. The treewidth of $\mathcal{H}$ is defined as the minimal width over all possible clique trees for $\mathcal{H}$ and thus describes the tree-likeness of a corresponding graph. We note that determining the treewidth of an arbitrary graph is an NP-hard problem [KF09]. However, many NP-hard problems on graphs of bounded[2] treewidth can be solved in polynomial time.

[1] A MAP assignment has the maximal probability under the model and is the mode of a corresponding distribution.

[2] Informally, we say that a treewidth is bounded if it does not explicitly depend on the graph size of a corresponding MRF.

2.2 Structured Output Prediction

Structured output prediction is a generalization of the standard paradigms of supervised learning tasks such as classification and regression to the case where the outputs are complex structured objects such as sequences, alignments, trees, lattices or more general graphs [TGK03; Tso+05; Joa+09]. In the following we introduce the existing framework of max-margin learning with structured outputs.

2.2.1 Joint Feature Maps

We consider a general problem of learning a functional dependency $h\colon \mathcal{X} \to \mathcal{Y}$ between an input space $\mathcal{X}$ and an arbitrary discrete output space $\mathcal{Y}$. A common approach is to learn a *joint discriminant function* that assigns each pair $(\boldsymbol{x}, \boldsymbol{y})$ with a real number encoding a degree of compatibility between an input $\boldsymbol{x}$ and an output $\boldsymbol{y}$. Hence, by maximizing this function over all $\boldsymbol{y} \in \mathcal{Y}$, we can determine, for a given input $\boldsymbol{x}$, the most compatible output

$$h(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \; \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}). \tag{2.2}$$

Here, $\boldsymbol{\Psi}\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ denotes a vector-valued mapping called *joint feature map*, whose concrete form depends on the nature of the underlying learning task. $\boldsymbol{w} \in \mathbb{R}^d$ is a weight vector to be learned. Apart from the necessity to capture important relationships between the individual variables the actual form of $\boldsymbol{\Psi}$ also has a strong influence on the efficiency of learning and prediction processes. Namely, evaluation of the prediction function in (2.2) results in a combinatorial problem, for which the computational complexity is directly affected by the type of variable interactions present in the joint feature map.

2.2.2 Loss Functions and Risk Minimization

The standard zero-one loss typically used for evaluating the classification performance is rather inappropriate for the most structured prediction tasks since it does not distinguish between partially incorrect predictions. In general, we need a performance measure which is able to assess a partial correctness of an output. For example, in the task of sequence learning, we would prefer predictions having a smaller Hamming distance with respect to the ground truth. Formally, we consider learning with arbitrary loss functions $\Delta\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. Here, $\Delta(\boldsymbol{y}^*, \boldsymbol{y})$ denotes the discrepancy between the true structured label $\boldsymbol{y}^*$ and a prediction label $\boldsymbol{y}$.

We consider the following supervised learning scenario. We assume that input-output pairs $(\boldsymbol{x}, \boldsymbol{y})$ are generated according to some fixed but unknown probability distribution $P(\boldsymbol{x}, \boldsymbol{y})$ and our goal is to minimize the risk

$$\mathcal{R}^{\Delta}_{P}(h) = \int_{\mathcal{X} \times \mathcal{Y}} \Delta(\boldsymbol{y}, h(\boldsymbol{x})) \; dP(\boldsymbol{x}, \boldsymbol{y}) \tag{2.3}$$

over the space of considered prediction functions. However, since the underlying distribution P is not known, we instead aim at minimizing the empirical risk

$$\mathcal{R}^{\Delta}_{S}(h) = \frac{1}{n} \sum_{i=1}^{n} \Delta(\boldsymbol{y}_i, h(\boldsymbol{x}_i)) \tag{2.4}$$

given a finite training set $S = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ of i.i.d. generated data points according to the distribution P.

2.2.3 Structural Support Vector Machine

Following a prevalent line of research [Tso+05; Joa+09; JFY09] we can learn the prediction function h by training a *structural support vector machine* (SSVM) using one of its loss-sensitive extensions either *margin* or *slack* scaling. In slack scaling, we scale the slack variables ξ_i according to the loss incurred in each of the linear constraints, which gives rise to the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{w},\, \xi_i \geqslant 0}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y}) \geqslant 1 - \frac{\xi_i}{\Delta(\boldsymbol{y}_i, \boldsymbol{y})}, \;\; \forall i, \forall \boldsymbol{y} \in \mathcal{Y}\backslash\{\boldsymbol{y}_i\} \end{aligned} \tag{2.5}$$

In margin scaling, we scale the margin according to the incurred loss as follows:

$$\begin{aligned} \underset{\boldsymbol{w},\, \xi_i \geqslant 0}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y}) \geqslant \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \xi_i, \;\; \forall i, \forall \boldsymbol{y} \in \mathcal{Y}\backslash\{\boldsymbol{y}_i\} \end{aligned} \tag{2.6}$$

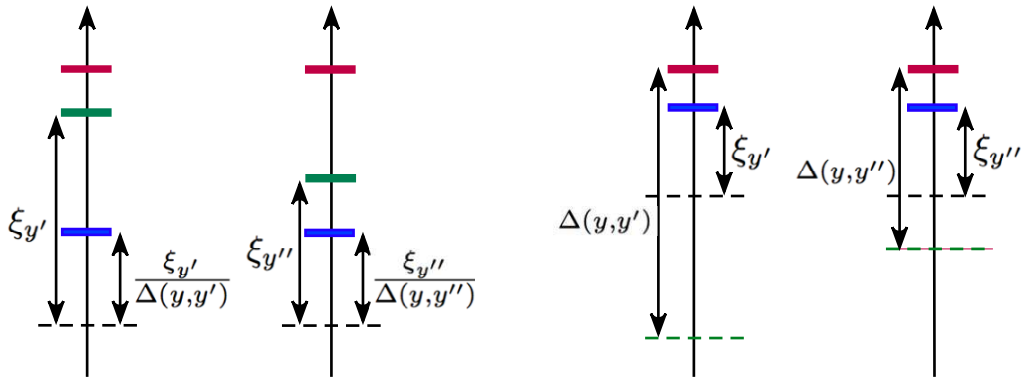
Fig. 2.4 illustrates the effect of rescaling on slack variables and margin. In the case of slack scaling in (a) we observe that the outputs $\boldsymbol{y}'$ and $\boldsymbol{y}''$ violate the margin to the same extent $\xi_{\boldsymbol{y}'}/\Delta(\boldsymbol{y}, \boldsymbol{y}') = \xi_{\boldsymbol{y}''}/\Delta(\boldsymbol{y}, \boldsymbol{y}'')$, but, since their corresponding loss values are different, the output $\boldsymbol{y}'$ contributes a greater value $\xi_{\boldsymbol{y}'}$ to the objective function than the output $\boldsymbol{y}''$. In the case of margin scaling we observe a similar effect. Although the outputs $\boldsymbol{y}'$ and $\boldsymbol{y}''$ violate the margin by the same value $\xi_{\boldsymbol{y}'} = \xi_{\boldsymbol{y}''}$ (before rescaling), because of $\Delta(\boldsymbol{y}, \boldsymbol{y}') > \Delta(\boldsymbol{y}, \boldsymbol{y}'')$, the output $\boldsymbol{y}'$ contributes a greater value $\Delta(\boldsymbol{y}, \boldsymbol{y}') - 1 + \xi_{\boldsymbol{y}'}$ to the objective function than the value $\Delta(\boldsymbol{y}, \boldsymbol{y}'') - 1 + \xi_{\boldsymbol{y}''}$ of the output $\boldsymbol{y}''$. However, a potential disadvantage of the margin rescaling (when contrasted to slack rescaling) is that it seems to give more importance to labellings having large errors — even if they are well separated from the true output.

Both scaling formulations have the property that they minimize an upper bound on the (regularized) empirical risk as follows. The corresponding proofs can be found in [Tso+05].

Proposition 1. *Denote by $\xi_i^*(\boldsymbol{w})$ and optimal solution of a slack variable in the problem (2.5) (or (2.6)) for a given weight vector $\boldsymbol{w}$. Then $\frac{1}{n}\sum_{i=1}^{n}\xi_i^*(\boldsymbol{w})$ is an upper bound on the empirical risk $\mathcal{R}_S^{\Delta}(h)$ in (2.4).*

Another appealing property of the slack scaling formulation is its scaling invariance with respect to the loss function as shown in [Tso+05].

Proposition 2. *Suppose $\Delta' = \alpha \cdot \Delta$ with $\alpha > 0$, i.e., Δ' is a scaled version of the original loss Δ. By rescaling $C' = \frac{C}{\alpha}$ both optimization problems in (2.5) with respect to C', Δ' and C, Δ are equivalent in the sense that the optimal weight vector $\boldsymbol{w}^*$ is the same in both cases.*



(a) slack scaling **(b)** margin scaling

Figure 2.4: © 2014 IEEE. Illustration of the behavior of slack variables ξ_i due to the slack and margin scaling. Each vertical axis corresponds to a subset of constraints in a corresponding optimization problem with respect to the same data point $(\boldsymbol{x}, \boldsymbol{y})$. The score value of the true output $\boldsymbol{y}$ is marked by a red bar, while the blue bars mark the score values of two (support) outputs $\boldsymbol{y}'$ and $\boldsymbol{y}''$ with $\Delta(\boldsymbol{y}, \boldsymbol{y}') > \Delta(\boldsymbol{y}, \boldsymbol{y}'')$. In (a) the green bars illustrate the effect of slack scaling on the corresponding slack variables, where the distance between a red bar and the dotted line corresponds to the margin. In (b) the distance between a dotted black line and a red bar corresponds to the margin of the length 1, while the effect on slack variables due to the margin scaling is visualized by the green dotted line.

Proof. First note that each $\boldsymbol{w}$ is feasible for both problems in the sense that we can find slack variables such that all the constraints are satisfied. Furthermore, we consider the unconstrained formulation:

$$f(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \max \left\{0, \max_{\hat{\boldsymbol{y}} \neq \boldsymbol{y}_i} \{\Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) \cdot (1 - \boldsymbol{w}^\top \delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}})\}\right\}$$

and

$$f'(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C'}{n}\sum_{i=1}^{n} \max \left\{0, \max_{\hat{\boldsymbol{y}} \neq \boldsymbol{y}_i} \{\Delta'(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) \cdot (1 - \boldsymbol{w}^\top \delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}}))\}\right\}$$

where we used the following short notation $\delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}}) = \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \boldsymbol{\Psi}(\boldsymbol{x}_i, \hat{\boldsymbol{y}})$. It suffices to show that $f(\boldsymbol{w}) = f'(\boldsymbol{w})$ holds for each $\boldsymbol{w}$. Using the definition of C' and Δ' we get:

$$\begin{aligned} f'(\boldsymbol{w}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{\alpha \cdot n}\sum_{i=1}^{n} \max \left\{0, \max_{\hat{\boldsymbol{y}} \neq \boldsymbol{y}_i} \{\alpha \cdot \Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) \cdot (1 - \boldsymbol{w}^\top \delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}}))\}\right\} \\ &= \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{\alpha \cdot n}\sum_{i=1}^{n} \alpha \cdot \max \left\{0, \max_{\hat{\boldsymbol{y}} \neq \boldsymbol{y}_i} \{\Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) \cdot (1 - \boldsymbol{w}^\top \delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}}))\}\right\} \\ &= f(\boldsymbol{w}) \end{aligned}$$

where in the second step we pulled α out of the inner and outer $\max$, since on the one hand it is a positive constant and on the other hand $\alpha \cdot 0 = 0$. □

The last proposition shows that while using the slack scaling approach we do not need to care about the scale of the loss function explicitly, since it is implicitly represented in terms of the hyperparameter C making it convenient for the end-user to tune the scale of Δ by changing the values of C instead.

Similar to the slack scaling (Proposition 2), the margin scaling has certain invariance property with respect to the scaling of the loss function.

Proposition 3. *Suppose $\Delta' = \alpha \cdot \Delta$ with $\alpha > 0$, i.e., Δ' is a scaled version of the original loss Δ. By rescaling $C' = C \cdot \alpha$ both optimization problems in (2.6) with respect to C, Δ and C', Δ' are equivalent in the sense that $\boldsymbol{w}^*$ is an optimal solution of the original problem if and only if $\alpha \cdot \boldsymbol{w}^*$ is an optimal solution of the scaled problem.*

Proof. First note that each $\boldsymbol{w}$ is feasible for both problems in the sense that we can find slack variables such that all the constraints are satisfied. Furthermore, we consider the unconstrained formulation:

$$f(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \max\ \left\{0, \max_{\hat{\boldsymbol{y}} \neq \boldsymbol{y}_i}\ \{\Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) - \langle \boldsymbol{w}, \delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}})\rangle\}\right\}$$

and

$$f'(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C'}{n}\sum_{i=1}^{n} \max\ \left\{0, \max_{\hat{\boldsymbol{y}} \neq \boldsymbol{y}_i}\ \{\Delta'(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) - \boldsymbol{w}^\top\delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}})\}\right\}$$

where we used the following short notation $\delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}}) = \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \boldsymbol{\Psi}(\boldsymbol{x}_i, \hat{\boldsymbol{y}})$.
It suffices to show that for each $\boldsymbol{w}$ the equality $f'(\alpha \cdot \boldsymbol{w}) = \alpha^2 \cdot f(\boldsymbol{w})$ holds. Using the definition of C' and Δ' we get:

$$\begin{aligned} f'(\alpha \cdot \boldsymbol{w}) &= \frac{1}{2}\|\alpha \cdot \boldsymbol{w}\|^2 + \frac{C \cdot \alpha}{n}\sum_{i=1}^{n} \max\ \left\{0, \max_{\hat{\boldsymbol{y}} \neq \boldsymbol{y}_i}\ \{\alpha \cdot \Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) - \alpha \cdot \boldsymbol{w}^\top\delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}})\}\right\} \\ &= \frac{\alpha^2}{2}\|\boldsymbol{w}\|^2 + \frac{C \cdot \alpha^2}{n}\sum_{i=1}^{n} \max\ \left\{0, \max_{\hat{\boldsymbol{y}} \neq \boldsymbol{y}_i}\ \{\Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) - \boldsymbol{w}^\top\delta\boldsymbol{\Psi}_i(\hat{\boldsymbol{y}})\}\right\} \\ &= \alpha^2 \cdot f(\boldsymbol{w}) \end{aligned}$$

where in the second step we pulled α out of the inner and outer $\max$, since on the one hand it is a positive constant and on the other hand $\alpha \cdot 0 = 0$. □

Although we cannot tune the scale of Δ by changing the values of C in the same way as for slack scaling, this is not really a disadvantage, since the corresponding solution vectors point in the same direction and only differ in their scaling. The latter does not matter for the actual prediction function, since

$$\operatorname*{argmax}_{\hat{\boldsymbol{y}}}\ \boldsymbol{w}^\top\boldsymbol{\Psi}(\boldsymbol{x}, \hat{\boldsymbol{y}}) = \operatorname*{argmax}_{\hat{\boldsymbol{y}}}\ \alpha \cdot \boldsymbol{w}^\top\boldsymbol{\Psi}(\boldsymbol{x}, \hat{\boldsymbol{y}})$$

for any $\alpha > 0$. Therefore, as for slack scaling, we also do not need to care much about the scale of the loss function and can handle different scalings by means of the hyperparameter C.

2.2.4 Training Algorithms

At first glance both scaling formulation show a similar behavior. In particular, both aim at minimizing an upper bound on the empirical risk with respect to the loss function Δ. However, a potential disadvantage of margin scaling is that it seems to give more importance to labellings with high loss — regardless of how well they are separated from the true output. In slack scaling, the labellings which are already well separated are not affected by the scaling of slack variables.

Algorithm 1 Cutting Plane Algorithm

1: $\mathcal{W} \leftarrow \emptyset$, $\boldsymbol{w} \leftarrow \mathbf{0}$, $\boldsymbol{\xi} \leftarrow \mathbf{0}$ Cst($\cdot$): Constraint set
2: **repeat**
3: **for** $i = 1$ **to** n **do**
4: $\widehat{\boldsymbol{y}} \leftarrow \arg\max_{\boldsymbol{y} \in \mathcal{Y}} H_{i,\boldsymbol{w}}(\boldsymbol{y})$
5: **if** $H_{i,\boldsymbol{w}}(\widehat{\boldsymbol{y}}) > \xi_i$ **then**
6: $\mathcal{W} \leftarrow \mathcal{W} \cup \{(i, \widehat{\boldsymbol{y}})\}$
7: $(\boldsymbol{w}, \boldsymbol{\xi}) \leftarrow \min_{\boldsymbol{w},\, \boldsymbol{\xi} \geqslant 0} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$ subject to Cst($\mathcal{W}$)
8: **end if**
9: **end for**
10: **until** $\mathcal{W}$ has not changed during the iteration

Another issue with SSVMs is that $|\mathcal{Y}|$ may grow exponentially in the description size of the outputs $\boldsymbol{y} \in \mathcal{Y}$ resulting in exponentially many constraints for the optimization problem as, for example, in the case of natural language parsing. Broadly speaking, there are two categories of methods for solving such quadratic problems. The first category is based on the idea of cutting plane algorithm (CPA) [Tso+05; JFY09] to avoid generating exponentially many constraints. This also includes stochastic gradient [Bor+07] and online subgradient methods [RBZ07b]. The second category of methods [TGK03; Tas+05; TLJ06] which are applied to the special case of margin scaling exploits the decomposability of the corresponding loss functions resulting in a joint program combining the inference part and parameter learning. The major problem with the second category, however, is that it is limited to applications where the corresponding reformulation exists.

Here, we focus on the cutting plane approach (see Algorithm 1) that iteratively constructs a subset of constraints in the full-sized optimization problem, which ensures a sufficiently accurate solution up to a specified precision. The bottleneck of this method, however, is that it is based on the concept of a separation oracle, which is used to deliver for each data point $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ a constraint that is most violated by a current solution (line 4). Here, $H_{i,\boldsymbol{w}}(\boldsymbol{y}) = (1 - \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y})) \cdot \Delta(\boldsymbol{y}_i, \boldsymbol{y})$ for slack scaling and $H_{i,\boldsymbol{w}}(\boldsymbol{y}) = \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y})$ for margin scaling. The corresponding constraints for each data point i (denoted as $\mathcal{W}_i$) are indexed by the individual outputs $\boldsymbol{y}$. That is, for each pair $(i, \boldsymbol{y})$ the actual constraint is obtained by replacing i and $\boldsymbol{y}$ in a corresponding inequality (2.5) or (2.6). After computing the most violated constraint for the current training point it is first added to the set of constraints (line 6) and a new solution $(\boldsymbol{w}, \boldsymbol{\xi})$ is recomputed according to the new constraint set (line 7). The algorithm terminates if $\mathcal{W}$ did not change during the last iteration. It has been shown [Tso+05] that there always exists a polynomial size subset of constraints such that the solution under these constraints is guaranteed to satisfy the full set of constraints up to an arbitrary precision [Tso+05; JFY09]. Consequently, the CPA, which sequentially adds the most violated constraint, converges in polynomial number if iterations.

Alternatively, we can use other algorithms for solving a corresponding quadratic problem including subgradient method [RBZ07a; RBZ07b], bundle methods [SVL07; Teo+10], and Frank-Wolfe algorithm [Lac+13]. Note that each of these algorithms repeatedly needs to solve the same loss augmented inference problem.

2.2.5 Loss Augmented Inference

The problem of finding the most violated constraint is generally referred to as the *loss augmented inference* and is, for many cases, the computational bottleneck during the training procedure. Ignoring the constant terms it can be generically written as

$$\widehat{\boldsymbol{y}} = \underset{\boldsymbol{y}\in\mathcal{Y}}{\operatorname{argmax}} \;\; \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \odot \Delta(\boldsymbol{y}^*, \boldsymbol{y}), \quad \odot \in \{+, \cdot\} \tag{2.7}$$

where $\boldsymbol{y}^*$ is the (fixed) true output for a given input $\boldsymbol{x}$.

There is a high diversity of existing applications which can be naturally embedded in the maximum-margin framework of structured output prediction. This diversity is also reflected in the high number of existing (structured) loss functions which basically can be divided into two groups. The first group is often refereed to as *decomposable* losses indicating that the factorization of the loss function agrees with the factorization of the compatibility function in (2.2) – so the two can be folded together and we can use the same algorithm for problem (2.7) we use for evaluating the prediction function. The other group corresponds to the more complex case where the loss function does not decompose in a desired way sometimes referred to as *high-order* loss – interpreted as a global factor in a graphical model which imposes high-order dependencies on the individual variables. While the simple case corresponds to training an SSVM via margin scaling given a decomposable loss, this excludes the slack scaling formulation and, more importantly, the use of popular non decomposable loss functions. Therefore, previous works had to rely on approximation algorithms for inference in order to benefit from non decomposable losses. Furthermore, usage of approximate solutions for the maximal violators can have negative effects on the resulting performance as summarized below.

Given an exact separation oracle, training an SSVM via cutting plane algorithm or bundle methods, for example, comes with three theoretical guarantees:

1. Polynomial Time Termination: The training algorithm terminates in a polynomial number of iterations and thus overall polynomial time, provided the separation oracle runs in polynomial time.

2. Correctness: The solution given by the training algorithm is optimal, since it terminates only if no constraint of the original full-size problem is violated (up to a specified precision).

3. Empirical Risk Bound: The solution given by the training algorithm provides an upper bound on the empirical risk.

The existing proofs [Tso+05; JFY09] for these properties strongly rely on the assumption that the separation oracle is exact and do not necessarily hold with approximations. A related question is what happens under approximation as there are many cases where the exact inference is intractable. A few previous works address this problem. Kulesza et al. [KP07] provide a theoretical analysis of MRF structural learning with loopy belief propagation (LBP) and linear programming (LP) relaxation [KF09]. They emphasize that when approximation is used in a learning algorithm, the learning can fail even with approximate inference method with rigorous approximation guarantees. Joachims et al. [FJ08] also provide an analysis of the learning behavior under approximation. Although they derived new learning guarantees depending on the quality of approximation, one basic result (from the theoretical perspective) is that the only property preserved when using approximation during training is the polynomial time termination. In particular, the correctness property does not hold anymore.

2.3 Summary and Discussion

In this chapter we provided a customized introduction in the topic of discrete graphical models such as Markov random fields. In particular, we introduced the concepts of cliques, clique trees and treewidth of a graph. This provides the necessary background for the subsequent discussion in the next chapter.

Moreover, we formally introduced the max-margin framework of SSVMs with a particular focus on the difference between its two loss-sensitive formulations, margin and slack scaling. Here, we provide a noticeable new result regarding the properties of the margin scaling. Namely, in the literature the slack scaling is believed to be in many ways better-behaved than margin scaling. In particular, a statement is widespread that one of the alleged disadvantages of margin scaling (compared to slack scaling) is not being scaling-invariant with respect to the loss function, which makes this approach less manageable [Tso+05; SG08]. We showed (Proposition 3) that margin scaling also ensures invariance of some sort. More precisely, we showed that different loss scales also can be dealt with implicitly by appropriately adjusting the value of the hyperparameter C. Namely, the corresponding optimal weight vectors $\boldsymbol{w}^*$ point in the same direction and differ only in their scaling resulting in identical predictions.

Chapter 3

Exact Inference

In this chapter we present a generic view on the augmented MAP problem from a perspective of exact inference, which is NP-hard in general [CSH08; KF09]. We set up a corresponding problem within the framework of graphical models which allows for a concise theoretical analysis and also provides an access to a range of powerful modeling techniques. As a result we define a large class of tractable problem instances and derive a generic message passing algorithm which always finds an optimal solution in polynomial time. We complete our discussion by demonstrating how the presented algorithm can be used in several application scenarios including loss augmented inference, evaluation of generalization bounds for structured learning, and globally constrained MAP inference.

The main **contributions** in this chapter are the following:

- For the general task of (globally) augmented MAP inference we define a large class of tractable problem instances which can be solved efficiently.
- We derive an **exact** message passing algorithm for the tractable case, which is guaranteed to find an optimal solution in **polynomial time** with respect to the problem size.
- We provide a rigorous theoretical analysis of the computational complexity and prove the correctness of the presented algorithm.
- We show how the presented framework can be used in **multiple** application scenarios including loss augmented inference, evaluating PAC-Bayesian generalization bounds for structured prediction, and globally constrained MAP inference.

Parts of this chapter including Sections 3.1-3.4 are based on:

> A. Bauer, S. Nakajima, and K.-R. Müller. Efficient Exact Inference With Loss Augmented Objective in Structured Learning. *IEEE Trans. on Neural Networks and Learning Systems (TNNLS)* © 2017 IEEE [BNM16]

3.1 Problem Setting

We cast the maximization problem over a set of discrete variables $\boldsymbol{y} = (y_1, ..., y_M)$ as MAP inference on MRFs [Bis06; KF09]. More precisely, we identify the objective function $F(\boldsymbol{y}) = \sum_{t=1}^{T} f_t(\boldsymbol{y}_{C_t})$ and the MRF referring to each element y_m of $\boldsymbol{y}$ as a *node*, and denoting by $C_t \subseteq \{1, ..., M\}$ a *clique* corresponding to the index set of variables on which the t-th factor $f_t : \mathbb{R}^{|C_t|} \to \mathbb{R}$ depends. A corresponding realization of

the variables with indices in C_t is given by $\boldsymbol{y}_{C_t}$. To quantify the complexity of a corresponding objective we use the notion of *treewidth* of a graph [Bod93], and define the following:

Definition 5 (τ-decomposability)**.** *We say that a function $F : \mathcal{D} \subseteq \mathbb{R}^M \to \mathbb{R}$ is* τ-decomposable*, if the (unnormalized) probability $\exp(F(\boldsymbol{y}))$ factorizes over an MRF with a **bounded** treewidth τ.*

In other words, for any *τ-decomposable* function, there exists a tree decomposition with clique nodes (and potentials) $\{\tilde{f}_t, \tilde{C}_t\}_{t=1}^{\tilde{T}}$ such that $\max_t |\tilde{C}_t| - 1 = \tau$, and

$$F(\boldsymbol{y}) = \textstyle\sum_{t=1}^{\tilde{T}} \tilde{f}_t(\boldsymbol{y}_{\tilde{C}_t}). \tag{3.1}$$

If there are no loops running over multiple cliques, the treewidth of a corresponding MRF is equal to the maximal clique size minus one. This idea extends to arbitrary graphs supported by the concept of *junction trees* (or *clique trees*) [KF09; LS90]: a clique tree is constructed from a triangulated graph by creating a new node for each maximal clique; the cliques which share common variables are connected by edges shaping a tree such that the total number of shared variables is maximized. The maximal number of variables in a node is called the $width$ of a corresponding clique tree, and the treewidth is thus defined as the minimum width among all possible clique trees resulting from the triangulated graph. Although the problem of constructing a clique tree with a minimal width is NP-hard in general, there are several efficient techniques [KF09] which provide a good tree decomposition with the resulting width being close to the treewidth. In the following, $M = |\bigcup_{t=1}^{T} C_t|$ is the total number of nodes in the MRF over discrete variables $\{y_m\}_{m=1}^{M}$ with N being the maximum number of the possible states. Provided the maximization part dominates the time for the clique tree construction, we get the following known result [LS90]:

Proposition 4. *The computational time complexity for maximizing τ-decomposable functions is upper bounded by $O(M \cdot N^{\tau+1})$.*

We extend the notion of τ-decomposability for real-valued functions to mappings with multivariate outputs and define joint decomposability:

Definition 6 (Joint τ-decomposability)**.** *We say two mappings $\boldsymbol{G} : \mathcal{D} \subseteq \mathbb{R}^M \to \mathbb{R}^P$ and $\boldsymbol{G}' : \mathcal{D} \subseteq \mathbb{R}^M \to \mathbb{R}^{P'}$ are* jointly τ-decomposable*, if there exists an MRF with a **bounded** treewidth τ that can express both $\boldsymbol{G}$ and $\boldsymbol{G}'$.*

Definition 6 requires that the mappings $\boldsymbol{G}$ and $\boldsymbol{G}'$ factorize over a common set of cliques, i.e., there exists a tree decomposition with clique nodes and the corresponding potentials $\{\boldsymbol{g}_t, \boldsymbol{g}'_t, C_t\}_{t=1}^{T}$ where $\max_t |C_t| - 1 = \tau$, and

$$\boldsymbol{G}(\boldsymbol{y}) = \textstyle\sum_{t=1}^{T} \boldsymbol{g}_t(\boldsymbol{y}_{C_t}), \quad \boldsymbol{G}'(\boldsymbol{y}) = \sum_{t=1}^{T} \boldsymbol{g}'_t(\boldsymbol{y}_{C_t}).$$

Note that the individual factor functions are allowed to have less variables in their scope than in a corresponding clique, i.e., $\text{scope}(\boldsymbol{g}_t), \text{scope}(\boldsymbol{g}'_t) \subseteq C_t$.

Our original (partial) motivation is to derive an efficient inference algorithm for the problem in (2.7), which is an instance[1] of a more general class of problems:

[1] Provided the compatibility function and the sufficient statistics of the loss function factorize over an MRF with a bounded treewidth.

Problem 1. *For $F\colon \mathcal{Y} \to \mathbb{R}$, $\boldsymbol{G}\colon \mathcal{Y} \to \mathbb{R}^P$, $H\colon \mathbb{R} \times \mathbb{R}^P \to \mathbb{R}$ with $\mathcal{Y} \subset \mathbb{R}^M$, $|\mathcal{Y}| \leqslant N^M$ and $P, M, N \in \mathbb{N}$, we consider the following discrete optimization problem:*

$$\underset{\boldsymbol{y}\in\mathcal{Y}}{\text{maximize}} \quad H(F(\boldsymbol{y}), \boldsymbol{G}(\boldsymbol{y})) \tag{3.2}$$

where we assume that: 1) F and $\boldsymbol{G}$ are jointly τ-decomposable, 2) H is non-decreasing in the first argument.

For slack scaling, for example, the mapping H can be defined as $H(F(\boldsymbol{y}), \boldsymbol{G}(\boldsymbol{y})) = F(\boldsymbol{y}) \cdot \eta(\boldsymbol{G}(\boldsymbol{y}))$ for some $\eta\colon \mathbb{R}^P \to \mathbb{R}_+$, where $F(\boldsymbol{y}) = 1 - \boldsymbol{w}^\top (\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}^*) - \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}))$ corresponds to the *relative compatibility*, and $\eta(\boldsymbol{G}(\boldsymbol{y})) = \Delta(\boldsymbol{y}^*, \boldsymbol{y})$ to a dissimilarity measure. In fact, a considerable number of popular performance measures in structured prediction can be generally represented in this form, that is, as a multivariate cardinality-based potential where the mapping $\boldsymbol{G}$ represents multivariate label statistics (e.g. the number of true or false positives) with respect to the ground truth. In Chapter 6, we give an overview of the existing loss functions along with this compact representation. It is important to note that the choice of $\boldsymbol{G}$ has a strong impact on the efficiency of the resulting computations. Namely, in order for an instance of Problem 1 to be solvable in polynomial time we need an additional assumption on $\boldsymbol{G}$ requiring that the maximal number of states of each auxiliary variable (to be introduced in the next section) must be polynomially bounded in M, that is, in the number of variables in a corresponding MRF.

3.2 Intuition behind the Algorithmic Idea

Here we give an intuition why efficient inference is possible for Problem 1 on the example of loss augmented inference for SSVMs. For the margin scaling formulation, if η is a linear function, the objective $F(\boldsymbol{y}) + \eta(\boldsymbol{G}(\boldsymbol{y}))$ inherits the τ-decomposability directly from F and $\boldsymbol{G}$. In that case efficient inference is guaranteed according to Proposition 4.

Additional source of difficulty in the slack scaling formulation originates from the multiplication between $F(\boldsymbol{y})$ and $\eta(\boldsymbol{G}(\boldsymbol{y}))$. Even for a linear η we cannot directly exploit the decomposability of F and $\boldsymbol{G}$, and the resulting graph is fully connected. Moreover, the non-linearity of η required by popular dissimilarity measures, in general, prevents efficient inference even for the margin scaling. Nevertheless, in many cases we can perform efficient inference as shown below. Namely, the global interactions between decomposable F and $\boldsymbol{G}$ can be controlled using auxiliary variables at a polynomial cost. We now illustrate this on a simple example.

Consider a *chain* of connected nodes with a 1-decomposable F and 0-decomposable $\boldsymbol{G}$. That is,

$$F(\boldsymbol{y}) = \sum_{m=1}^{M-1} f_m(y_m, y_{m+1}) \text{ and } \boldsymbol{G}(\boldsymbol{y}) = \sum_{m=1}^{M} \boldsymbol{g}_m(y_m). \tag{3.3}$$

Although 0-decomposability — MRF with no edges — seems to be very restrictive, in fact, it covers many practical cases (see Chapter 5). Our goal is to maximize an objective of the form $F(\boldsymbol{y}) \odot \eta(\boldsymbol{G}(\boldsymbol{y}))$ where $\odot\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ represents an algebraic operation like addition or multiplication.

A decomposable case for margin scaling (with $\eta(\boldsymbol{G}) = \boldsymbol{G}$) is illustrated by the leftmost factor graph in Figure 3.1. Here, the corresponding factors f_m and $\boldsymbol{g}_m$ can be folded together enabling an efficient inference according to Proposition 4. The

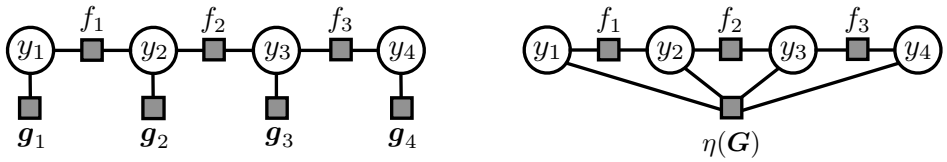


Figure 3.1: Factor graph representation for the margin scaling objective with $\eta(\boldsymbol{G}) = \boldsymbol{G}$ (on the left), and a high-order $\eta(\boldsymbol{G})$ (on the right).

nonlinearity of η, however, results (in the worst case!) in a global dependency between all the variable nodes giving rise to a high-order potential $\eta(\boldsymbol{G})$ as illustrated by the rightmost factor graph in Figure 3.1. In slack scaling, even for a linear η, after multiplying the individual factors we can see that the resulting model has an edge for every pair of variables resulting in a fully-connected graph. Thus, for the last two cases an efficient (exact) inference is infeasible in general. The key idea is to relax the dense connections in these graphs by introducing *auxiliary variables* $\boldsymbol{L} = (\boldsymbol{l}_1, \ldots, \boldsymbol{l}_M) \in \mathbb{R}^{P \times M}$ subject to the constraints

$$\boldsymbol{l}_m = \sum_{k=1}^{m} \boldsymbol{g}_k(y_k), \;\; m \in \{1, ..., M\}. \tag{3.4}$$

More precisely, for $H(F(\boldsymbol{y}), \boldsymbol{G}(\boldsymbol{y})) = F(\boldsymbol{y}) \odot \eta(\boldsymbol{G}(\boldsymbol{y}))$, Problem 1 is equivalent to the following constrained optimization problem in the sense that both have the same optimal value and the same set of optimal solutions with respect to $\boldsymbol{y}$:

$$\begin{aligned} \underset{\boldsymbol{y}, \boldsymbol{L}}{\text{maximize}} \quad & F(\boldsymbol{y}) \odot \eta(\boldsymbol{l}_M) \\ \text{subject to} \quad & \boldsymbol{l}_{m+1} = \boldsymbol{l}_m + \boldsymbol{g}_{m+1}(y_{m+1}) \quad \forall m \in \{1, ..., M-1\} \\ & \boldsymbol{l}_1 = \boldsymbol{g}_1(y_1) \end{aligned} \tag{3.5}$$

where the new objective involves no global dependencies , and is 1-decomposable if we regard $\eta(\boldsymbol{l}_M)$ as a constant. We can make the local dependency structure of the new formulation more explicit by taking the constraints directly into the objective as follows

$$Q(\boldsymbol{y}, \boldsymbol{L}) = F(\boldsymbol{y}) \odot \eta(\boldsymbol{l}_M) - \mathbb{1}_\infty[\boldsymbol{l}_1 \neq \boldsymbol{g}_1(y_1)] - \sum_{m=1}^{M-1} \mathbb{1}_\infty[\boldsymbol{l}_{m+1} \neq \boldsymbol{l}_m + \boldsymbol{g}_{m+1}(y_{m+1})]. \tag{3.6}$$

Here, $\mathbb{1}_\alpha[\cdot]$ denotes the indicator function such that $\mathbb{1}_\alpha[\cdot] = \alpha$ if the argument in $[\cdot]$ is true and $\mathbb{1}_\alpha[\cdot] = 0$ otherwise. The indicator functions rule out the configurations that do not satisfy Eq. (3.4) when maximization is performed. A corresponding factor graph for margin scaling is illustrated by the leftmost graph in Figure 3.2. We see that our new augmented objective (3.6) shows only local dependencies and is, in fact, 2-decomposable.

Applying the same scheme for slack scaling also yields a graph (see the rightmost graph in Figure 3.2) with much localized connections except $\boldsymbol{l}_4$ having a high degree, which we call a *hub* node. Actually, $Q(\boldsymbol{y}, \boldsymbol{L})$ is 2-decomposable if $\boldsymbol{l}_4$ is observed — assuming a fixed value of $\boldsymbol{l}_4$ renders the term $\eta(\boldsymbol{l}_4)$ constant, which can be multiplied into the corresponding factors of F. This way we can effectively reduce the overall treewidth at the expense of polynomial computation time supported by Proposition 4, provided the maximal number R of different states of each auxiliary variable is polynomially bounded in M, the number of nodes in the original graph.

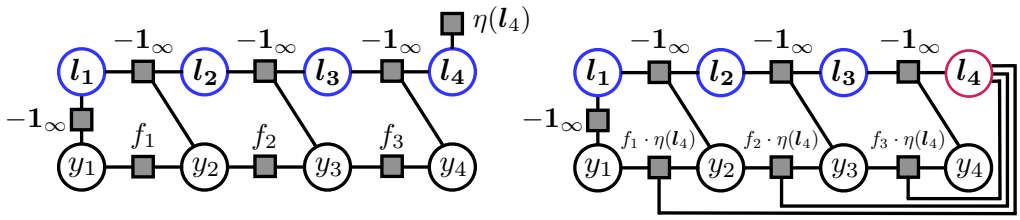


Figure 3.2: Factor graph representation for an augmented objective $Q(\boldsymbol{y}, \boldsymbol{L})$ for margin scaling (on the left) and for slack scaling (on the right). The auxiliary variables $\boldsymbol{L} = (\boldsymbol{l}_1, ..., \boldsymbol{l}_4)$ are marked blue (except $\boldsymbol{l}_4$ for slack scaling). $\boldsymbol{l}_4$ is the hub node.

Namely, after constructing a *clique tree* to the augmented graph, we can directly apply the max-sum [LS90; Bis06; KF09] algorithm at most R times – one for each possible observation of the hub node. However, performing inference independently for each observation of $\boldsymbol{l}_4$ would be an unnecessary waste of computational time and we present in the next subsection a more elaborated way. Namely, the corresponding constraints implicitly imposed by the auxiliary variables can be directly implemented within the message passing procedure as we show in the subsequent sections.

As already mentioned, an additional requirement for the resulting algorithm to run in polynomial time is that the maximal number of states R of an auxiliary variable (which is a property of the mapping $\boldsymbol{G}$ in Problem 1) must be polynomial in M. The good news is that all the popular dissimilarity measures satisfy this condition (see Chapter 5 for an overview). One practical example with R being polynomially bounded in M is the case where the individual factors of $\boldsymbol{G}$ are indicator functions with $\boldsymbol{g}_t(\boldsymbol{y}_{C_t}) \in \{0, 1\}^P$. In that case, R is of the order $O(M^P)$.

3.3 Message Passing Algorithm on Factor Graphs

The idea presented in the previous section is intuitive and allows for reusing of existing software. After the corresponding graph transformation due to the introduction of auxiliary variables we can use the standard max-sum algorithm for graphical models. Alternatively to the explicit graph transformation we can modify the message passing protocol instead, which is asymptotically at least one order of magnitude faster. In the following we derive an algorithm for solving an instance of Problem 1 via message passing in a factor graph. We now restrict ourselves to trees and later generalize the presented idea to loopy graphs via junction tree construction.

Assume now that the mapping F in Problem 1 is represented by a tree-structured factor graph with variable nodes $y_1, ..., y_M$. We denote by y_r the node chosen to be the root of the tree. Each variable y_m is associated with one auxiliary (multivariate) variable $\boldsymbol{l}_m$. The messages between neighboring nodes are sent according to the following message passing protocol. A factor node f_s can send a message $\mu_{f_s \to y_m}(Y, \boldsymbol{l})$ to a neighbor variable node y_m for fixed values Y_m and $\boldsymbol{l}$ (of the variable y_m and a corresponding auxiliary variable $\boldsymbol{l}_m$, respectively) if it received all messages from its other neighbors y_k for $k \in ne(f_s) \setminus \{m\}$, where $ne(f)$ denotes the set of indices of neighbor variable nodes of the factor f. Similarly, we can send a message from a variable node to a factor node. In that case we say that the factor or the variable node is *ready*.

The initial messages for the leaves which are factor nodes can be computed for all possible values Y and $\boldsymbol{l}$ according to

$$\mu_{f_s \to y_m}(Y, \boldsymbol{l}) = \begin{cases} f_s(Y), & \text{if } \boldsymbol{l} = \boldsymbol{g}_s(Y), \\ -\infty, & \text{else.} \end{cases} \tag{3.7}$$

If a variable node y_m is a leaf — that is there is no preceding factor node — we set

$$\mu_{y_m \to f_s}(Y, \boldsymbol{l}) = \begin{cases} 0, & \text{if } \boldsymbol{l} = \boldsymbol{g}_s(Y), \\ -\infty, & \text{else.} \end{cases} \tag{3.8}$$

The remaining messages from a factor node f_s to its neighboring variable node y_m on the way to the root can be computed according to the following equation

$$\mu_{f_s \to y_m}(Y_m, \boldsymbol{l}) = \max_{\{Y_k\}, \{\boldsymbol{l}_k\}} f_s(Y_m, \{Y_k\}) + \sum_{k \in ne(f_s) \setminus \{m\}} \mu_{y_k \to f_s}(Y_k, \boldsymbol{l}_k) \tag{3.9}$$

where we maximize over all configurations $\{Y_k\} = \{Y_k \colon k \in ne(f_s) \setminus \{m\}\}$ of the corresponding variables $\{y_k\}$ and over all parameters $\{\boldsymbol{l}_k\} = \{\boldsymbol{l}_k \colon k \in ne(f_s) \setminus \{m\}\}$ satisfying the equation $\sum_{k \in ne(f_s) \setminus \{m\}} \boldsymbol{l}_k = \boldsymbol{l} - \boldsymbol{g}_s(Y_m, \{Y_k\})$. Similarly, we send messages from a variable node y_m to a factor node f_s according to

$$\mu_{y_m \to f_s}(Y_m, \boldsymbol{l}) = \max_{\{\boldsymbol{l}_k\}} \sum_{k \in ne(y_m) \setminus \{s\}} \mu_{f_k \to y_m}(Y_m, \boldsymbol{l}_k) \tag{3.10}$$

maximizing over $\{\boldsymbol{l}_k\} = \{\boldsymbol{l}_k \colon k \in ne(y_m) \setminus \{s\}\}$ subject to $\sum_{k \in ne(y_m) \setminus \{s\}} \boldsymbol{l}_k = \boldsymbol{l}$.

The algorithm terminates if the root node y_r received all messages from its neighbors. We then compute the values

$$\mu(\boldsymbol{l}) = \max_{Y_r, \{\boldsymbol{l}_k\}} \sum_{k \in ne(y_r)} \mu_{f_k \to y_r}(Y_r, \boldsymbol{l}_k) \tag{3.11}$$

maximizing over all possible states Y_r of y_r and over $\{\boldsymbol{l}_k\} = \{\boldsymbol{l}_k \colon k \in ne(y_r)\}$ subject to the constraint $\sum_k \boldsymbol{l}_k = \boldsymbol{l}$. At this stage of the algorithm, each value $\mu(\boldsymbol{l})$ corresponds to a maximal value of $F(\cdot)$ over all $\boldsymbol{y} \in \mathcal{Y}$ subject to the constraint $\boldsymbol{G}(\boldsymbol{y}) = \boldsymbol{l}$. Therefore, we can use these values to get the optimal value p^* of Problem 1 according to

$$p^* = \max_{\boldsymbol{l}} H(\mu(\boldsymbol{l}), \boldsymbol{l}). \tag{3.12}$$

A corresponding optimal solution of Problem 1 can be obtained via backtracking using additional variables λ for saving optimal decisions in intermediate steps – as usually done in dynamic programming. The complete algorithm is summarized in Algorithm 2 supported by the following theorem. A proof is given in Appendix A.

Theorem 1. *Algorithm 2 always finds an optimal solution of Problem 1 if the factor graph of a corresponding MRF (associated with F) has a tree structure. The resulting time complexity is exponential in τ and in the maximal degree of the variable nodes minus 1.*

One issue with the factor graph representation is that the maximal degree of a variable node might be huge, even if F has a bounded treewidth. In fact, it can be linear in the graph size (e.g. for star-like shape). However, by using the concept of clique trees we can always find a suitable representation such that the inference can be performed in time polynomial in the number of variables where the degree of a

Algorithm 2 Augmented Max-Sum on a Tree-Structured Factor Graph

Compute the initial messages for leaf nodes according to Eq. (3.7) and (3.8)
while not finished **do**
  if there is a factor node f_s that is ready **then**
    for all Y and $\boldsymbol{l}$ **do**
      send a message $\mu_{f_s \to y_m}(Y, \boldsymbol{l})$ according to Eq. (3.9); save the maximizing arguments $\lambda_{f_s \to y_m}(Y, \boldsymbol{l}) := [\{Y_k\}^*; \{\boldsymbol{l}_k\}^*]$
    end for
  end if
  if there is a variable node y_m that is ready **then**
    for all Y and $\boldsymbol{l}$ **do**
      send a message $\mu_{y_m \to f_s}(Y, \boldsymbol{l})$ according to Eq. (3.10); save the maximizing arguments $\lambda_{y_m \to f_s}(Y, \boldsymbol{l}) := [\{\boldsymbol{l}_k\}^*]$
    end for
  end if
end while
$\boldsymbol{l}^* \leftarrow \operatorname{argmax}_{\boldsymbol{l}} H(\mu(\boldsymbol{l}), \boldsymbol{l})$, where $\mu(\boldsymbol{l})$ is defined by Eq. (3.11)
Let Y^* be a maximizing argument for $\mu(\boldsymbol{l}^*)$ in Eq. (3.11); starting with values $\boldsymbol{l}^*$ and Y^* recursively reconstruct an optimal configuration $\boldsymbol{y}^*$ using the information in λ according to the formulas in Eq. (3.9) and in Eq. (3.10).

corresponding polynomial is only a function of the treewidth τ — as we show in the next section.

3.4 Message Passing Algorithm on Clique Trees

The algorithmic idea presented in the previous section for tree-structured factor graphs can be applied to general MRFs with cycles by first transforming loopy graphs into a clique tree and then running an inference algorithm on the new representation as described below.

First, similar to conventional junction tree algorithm, we need to construct a clique tree for a given set of factors, which is family preserving and has the running intersection property [Bis06; KF09]. There are two equivalent approaches, the first based on variable elimination and the second on graph triangulation. A good description of both is given in [KF09]. In particular, the time complexity of the variable elimination algorithm is upper bounded by $O(M \cdot N^{\tau+1})$.

Assume now that a clique tree with cliques $C_1, ..., C_K$ is given, where C_i denotes a set of indices of variables contained in the i-th clique. That is, a corresponding set of variables is given by $\boldsymbol{y}_{C_i}$. We denote the clique potentials (or factors) related to the mappings F and $\boldsymbol{G}$ (see Problem 1) by $\{f_{C_i}\}_{i=1}^K$ and $\{\boldsymbol{g}_{C_i}\}_{i=1}^K$, respectively. Additionally, we denote by C_r a clique chosen to be the root of the clique tree. Finally, we use the notation $ne(C_i)$ for the indices of the neighbors of the clique C_i. We can now compute the optimal value of the objective in Problem 1 as follows. Starting at the leaves of the clique tree we iteratively send messages toward the root according to the following message passing protocol. A clique C_i can send a message to its parent clique C_j if it received all messages from all its other neighbors C_k for $k \in ne(C_i) \setminus \{j\}$. In that case we say that C_i is *ready*.

For each configuration of the variables $\boldsymbol{y}_{C_i \cap C_j}$ and parameters $\boldsymbol{l} \in \mathbb{R}^P$ (encoding the state of an auxiliary variable associated with the current clique C_i) a corresponding message from a clique C_i to a clique C_j can be computed according to the

Algorithm 3 Augmented Max-Sum on a Clique Tree

1: **while** root clique C_r did not receive all messages **do**
2: **if** a clique C_i is ready **then**
3: **for** all $\boldsymbol{y}_{C_i \cap C_j}$ and $\boldsymbol{l}$ **do**
4: send a message $\mu_{C_i \to C_j}(\boldsymbol{y}_{C_i \cap C_j}, \boldsymbol{l})$ to a parent clique C_j according to Eq. (3.13); save the maximizing arguments $\lambda_{C_i \to C_j}(\boldsymbol{y}_{C_i \cap C_j}, \boldsymbol{l}) := [\boldsymbol{y}^*_{C_i \setminus C_j}; \{\boldsymbol{l}_k\}^*]$
5: **end for**
6: **end if**
7: **end while**
8: $\boldsymbol{l}^* \leftarrow \operatorname{argmax}_{\boldsymbol{l}} H(\mu(\boldsymbol{l}), \boldsymbol{l})$, where $\mu(\boldsymbol{l})$ is defined by Eq. (3.14)
9: Let $\boldsymbol{y}^*_{C_r}$ be a maximizing argument for $\mu(\boldsymbol{l}^*)$ in Eq. (3.14); starting with values $\boldsymbol{l}^*$ and $\boldsymbol{y}^*_{C_r}$ recursively reconstruct an optimal configuration $\boldsymbol{y}^*$ using the information in λ according to the formula in Eq. (3.13).

following equation

$$\mu_{C_i \to C_j}(\boldsymbol{y}_{C_i \cap C_j}, \boldsymbol{l}) = \max_{\boldsymbol{y}_{C_i \setminus C_j}, \{\boldsymbol{l}_k\}} f_{C_i}(\boldsymbol{y}_{C_i}) + \sum_{k \in ne(C_i) \setminus \{j\}} \mu_{C_k \to C_i}(\boldsymbol{y}_{C_k \cap C_i}, \boldsymbol{l}_k) \tag{3.13}$$

where we maximize over all configurations of the variables $\boldsymbol{y}_{C_i \setminus C_j}$ and over all parameters $\{\boldsymbol{l}_k\} = \{\boldsymbol{l}_k \colon k \in ne(C_i) \setminus \{j\}\}$ subject to $\sum_{k \in ne(C_i) \setminus \{j\}} \boldsymbol{l}_k = \boldsymbol{l} - \boldsymbol{g}_{C_i}(\boldsymbol{y}_{C_i})$.

The algorithm terminates if the designated root clique C_r received all messages from its neighbors. We then compute the values

$$\mu(\boldsymbol{l}) = \max_{\boldsymbol{y}_{C_r}, \{\boldsymbol{l}_k\}} f_{C_r}(\boldsymbol{y}_{C_r}) + \sum_{k \in ne(C_r)} \mu_{C_k \to C_r}(\boldsymbol{y}_{C_k \cap C_r}, \boldsymbol{l}_k) \tag{3.14}$$

maximizing over all configurations of $\boldsymbol{y}_{C_r}$ and $\{\boldsymbol{l}_k\} = \{\boldsymbol{l}_k \colon k \in ne(C_r)\}$ subject to the constraint $\sum_k \boldsymbol{l}_k = \boldsymbol{l} - \boldsymbol{g}_{C_r}(\boldsymbol{y}_{C_r})$, which we use to get the optimal value p^* of Problem 1 according to

$$p^* = \max_{\boldsymbol{l}} H(\mu(\boldsymbol{l}), \boldsymbol{l}). \tag{3.15}$$

A corresponding optimal solution of Problem 1 can be obtained by backtracking the additional variables λ saving optimal decisions in intermediate steps. The complete algorithm is summarized in Algorithm 3 supported by the following theorem.

Theorem 2. *Algorithm 3 always finds an optimal solution of Problem 1. The computational complexity is of the order $O(M \cdot N^{\tau+1} \cdot R^{\nu-1})$ where ν is defined as the maximal number of neighbors of a node in a corresponding clique tree.*

Proof. We now show the correctness of the presented computations. For this purpose we first provide a semantic interpretation of messages as follows. Let $C_i - C_j$ be an edge in a clique tree. We denote by $\mathcal{F}_{\prec(i-j)}$ the set of clique factors f_{C_k} of the mapping F on the C_i-th side of the tree and by $\mathcal{G}_{\prec(i-j)}$ a corresponding set of clique factors of the mapping $\boldsymbol{G}$. Furthermore, we denote by $\mathcal{V}_{\prec(i-j)}$ the set of all variables appearing on the C_i-th side but not in the sepset $C_i \cap C_j$. Intuitively, a message $\mu_{C_i \to C_j}(\boldsymbol{y}_{C_i \cap C_j}, \boldsymbol{l})$ sent from a clique C_i to C_j corresponds to a sum of all factors contained in $\mathcal{F}_{\prec(i-j)}$ which is maximized (for fixed values of $\boldsymbol{y}_{C_i \cap C_j}$ and $\boldsymbol{l}$) over the variables in $\mathcal{V}_{\prec(i-j)}$ subject to the constraint $\boldsymbol{l} = \sum_{\boldsymbol{g}_{C_k} \in \mathcal{G}_{\prec(i-j)}} \boldsymbol{g}_{C_k}(\boldsymbol{y}_{C_k})$. That is, we

define the following induction hypothesis

$$\mu_{C_i \to C_j}(\boldsymbol{y}_{C_i \cap C_j}, \boldsymbol{l}) = \max_{\mathcal{V}_{\prec(i-j)} \colon \boldsymbol{l} = \sum_{\boldsymbol{g}_{C_k} \in \mathcal{G}_{\prec(i-j)}} \boldsymbol{g}_{C_k}(\boldsymbol{y}_{C_k})} \sum_{f_{C_k} \in \mathcal{F}_{\prec(i-j)}} f_{C_k}(\boldsymbol{y}_{C_k}). \tag{3.16}$$

Now consider an edge $(C_i - C_j)$ such that C_i is not a leaf. Let $i_1, ..., i_m$ be the neighboring cliques of C_i other than C_j. It follows from the running intersection property that $\mathcal{V}_{\prec(i-j)}$ is a disjoint union of $\mathcal{V}_{\prec(i_k - i)}$ for $k = 1, ..., m$ and the variables $\boldsymbol{y}_{C_i \setminus C_j}$ eliminated at C_i itself. Similarly, $\mathcal{F}_{\prec(i-j)}$ is the disjoint union of the $\mathcal{F}_{\prec(i_k - i)}$ and $\{f_{C_i}\}$. Finally, $\mathcal{G}_{\prec(i-j)}$ is the disjoint union of the $\mathcal{G}_{\prec(i_k - i)}$ and $\{\boldsymbol{g}_{C_i}\}$. In the following, we abbreviate the term $\mathcal{V}_{\prec(i_k - i)} \colon \boldsymbol{l}_{i_k} = \sum_{\boldsymbol{g} \in \mathcal{G}_{\prec(i_k - i)}} \boldsymbol{g}$ describing a range of variables in $\mathcal{V}_{\prec(i_k - i)}$ subject to a corresponding equality constraint with respect to $\boldsymbol{l}_{i_k}$ by $\mathcal{V}_{\prec(i_k - i)} \colon \boldsymbol{l}_{i_k}$. Thus, the right hand side of Eq. (3.16) is equal to

$$\max_{\boldsymbol{y}_{C_i \setminus C_j}} \max_{\{\boldsymbol{l}_{i_k}\}_{k=1}^m} \max_{\mathcal{V}_{\prec(i_1 - i)} \colon \boldsymbol{l}_{i_1}} \cdots \max_{\mathcal{V}_{\prec(i_m - i)} \colon \boldsymbol{l}_{i_m}} \left(\sum_{f \in \mathcal{F}_{\prec(i_1 - i)}} f \right) + \cdots + \left(\sum_{f \in \mathcal{F}_{\prec(i_m - i)}} f \right) + f_{C_i} \tag{3.17}$$

where in the second $\max$ we maximize over all configurations of $\{\boldsymbol{l}_{i_k}\}_{k=1}^m$ subject to the constraint $\sum_{k=1}^m \boldsymbol{l}_{i_k} = \boldsymbol{l} - \boldsymbol{g}_{C_i}(\boldsymbol{y}_{C_i})$. Since all the corresponding sets are disjoint the term (3.17) is equal to

$$\max_{\boldsymbol{y}_{C_i \setminus C_j}, \{\boldsymbol{l}_{i_k}\}_{k=1}^m} f_{C_i} + \underbrace{\max_{\mathcal{V}_{\prec(i_1 - i)} \colon \boldsymbol{l}_{i_1}} \left(\sum_{f \in \mathcal{F}_{\prec(i_1 - i)}} f \right)}_{\mu_{C_{i_1} \to C_i}(\boldsymbol{y}_{C_{i_1} \cap C_i}, \boldsymbol{l}_{i_1})} + \cdots + \underbrace{\max_{\mathcal{V}_{\prec(i_m - i)} \colon \boldsymbol{l}_{i_m}} \left(\sum_{f \in \mathcal{F}_{\prec(i_m - i)}} f \right)}_{\mu_{C_{i_m} \to C_i}(\boldsymbol{y}_{C_{i_m} \cap C_i}, \boldsymbol{l}_{i_m})} \tag{3.18}$$

where again the maximization over $\{\boldsymbol{l}_{i_k}\}_{k=1}^m$ is subject to the constraint $\sum_{k=1}^m \boldsymbol{l}_{i_k} = \boldsymbol{l} - \boldsymbol{g}_{C_i}(y_{C_i})$. Using the induction hypothesis in the last expression we get the right hand side of Eq. (3.13) proving the claim in Eq. (3.16).

Now look at Eq. (3.14). Using Eq. (3.16) and the fact that all involved sets of variables and factors for different messages are disjoint we conclude that the computed values $\mu(\boldsymbol{l})$ corresponds to the sum of all factors f for the mapping F over the variables in $\boldsymbol{y}$ which is maximized subject to the constraint $\boldsymbol{G}(\boldsymbol{y}) = \boldsymbol{l}$. Therefore, by performing maximization over all values $\boldsymbol{l}$ according to Eq. (3.15) we get the optimal value of Problem 1.

By inspecting the formula for the message passing in Eq. (3.13) we conclude that the corresponding operations can be done in $O(M \cdot N^{\tau+1} \cdot R^{\nu-1})$ time where ν denotes the maximal number of neighbors of any clique node C_i. First, the summation in Eq. (3.13) involves $|ne(C_i)|$ terms. Second, a maximization is performed first over $|C_i \setminus C_j|$ variables with a cost $N^{|C_i \setminus C_j|}$. This, however, is done for each configuration of $\boldsymbol{y}_{C_i \cap C_j}$ where $|C_i \setminus C_j| + |C_i \cap C_j| = |C_i| \leqslant \tau + 1$ resulting in $N^{\tau+1}$. Then a maximization over $\{\boldsymbol{l}_k\}$ costs additionally $R^{|ne(C_i)|-2}$. Together with possible values for $\boldsymbol{l}$ it yields $R^{\nu-1}$ where we upper bounded $|ne(C_i)|$ by ν. Therefore, sending a message for all possible configurations of $(\boldsymbol{y}_{C_i \cap C_j}; \boldsymbol{l})$ on the edge $C_i - C_j$ costs $O(N^{\tau+1} \cdot |ne(C_i)| \cdot R^{\nu-1})$ time. Finally, we need to do these operations for each edge $(i, j) \in \mathcal{E}$ in the clique tree. The resulting cost can be estimated as follows: $\sum_{(i,j) \in \mathcal{E}} N^{\tau+1} \cdot R^{\nu-1} \cdot |ne(C_i)| = N^{\tau+1} \cdot R^{\nu-1} \sum_{(i,j) \in \mathcal{E}} |ne(C_i)| \leqslant N^{\tau+1} \cdot R^{\nu-1} \cdot 2|\mathcal{E}| \leqslant N^{\tau+1} \cdot R^{\nu-1} \cdot 2|M|$. Therefore, the total complexity is upper bounded by $O(M \cdot N^{\tau+1} \cdot R^{\nu-1})$. □

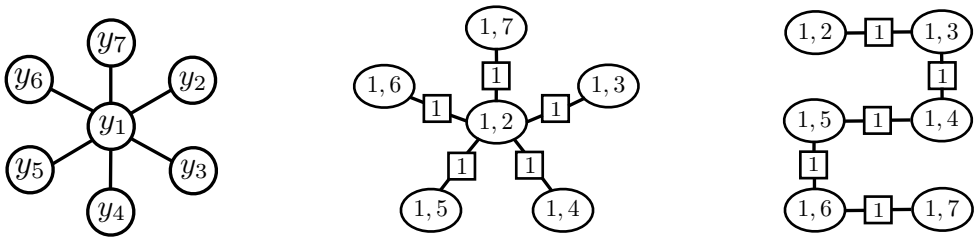


Figure 3.3: Illustration of an extreme example where ν can be linear in the graph size. The leftmost graph represents an MRF with $M = 7$ variables and treewidth $\tau = 1$. The graph in the middle shows a valid clique tree for the MRF on the left where the clique $\{y_1, y_2\}$ has $M - 1$ neighbors. That is, ν is linear in the graph size for that clique tree. The rightmost graph represents another clique tree which has a chain form where $\nu = \tau + 1 = 2$. The squared nodes denote here the corresponding sepsets.

Besides the treewidth τ, the value of the parameter ν is also crucial for the resulting running time of Algorithm 3 since the corresponding complexity is also exponential in ν. The following proposition suggests that ν tends to take on small values (provided τ is small) and effectively does not depend on the size of a corresponding MRF. We provide a corresponding proof in the Appendix A.

Proposition 5. *For any MRF with treewidth τ, there is a clique tree, for which the maximal number of neighbors ν is upper bounded according to $\nu \leqslant 2^{\tau+2} - 4$.*

To support the above proposition we consider the following extreme example illustrated in Figure 3.3. We are given an MRF with a star-like shape (on the left) having $M = 7$ variables and treewidth $\tau = 1$. One valid clique tree for this MRF is shown in the middle. In particular, the clique containing the variables y_1, y_2 has $\nu = M - 1$ neighbors. Therefore, running Algorithm 3 on that clique tree results in a computational time exponential in the graph size M. However, it is easy to modify that clique tree to one with a small number of neighbors for each node (shown on the right) upper bounded by $\nu = 2 = \tau + 1$. Note that the bound given in the Proposition 5 describes the worst case scenario and is much lower for many practical applications including the examples presented in Chapter 5. However, the important result here is that ν effectively depends only on τ and not on the graph size.

We complete our discussion here by giving an alternative view on the computational complexity of the presented idea which shows the connection to the conventional junction tree algorithm in the case where $\nu \leqslant \tau + 2$. Namely, the constraint message passing algorithm (Algorithm 3) can be seen as a conventional message passing on a clique tree (for the mapping F in Problem 1) without auxiliary variables, but where the size of the state space for each variable y_i (previously upper bounded by N) is now increased to $N \cdot R$. Provided $\nu \leqslant \tau + 2$, Proposition 4 then guarantees an exact inference in time of the order $O(M \cdot (N \cdot R)^{\tau+1})$. The summation constraints with respect to the auxiliary variables can be ensured by extending the corresponding potential functions f_{C_i} to take on $-\infty$ forbidding inconsistent state transitions between individual variables. The same observation holds also for message passing on factor graphs (Algorithm 2). To summarize, we can remove the global dependencies (imposed by the mapping H in Problem 1) reducing the overall treewidth by cleverly introducing auxiliary variables, but pay a price that the size of the label space becomes a function of the graph size (R is in general dependent on M). If R grows only polynomially with M – which is true in many cases – we can perform exact inference in polynomial time!

3.5 General Use Cases

In this section we demonstrate the usability of the presented algorithm in several application scenarios. Each use case in turn represents multiple practical applications.

3.5.1 Loss Augmented Inference with High Order Loss Functions

As already mentioned, Problem 1 covers the task of loss augmented inference (for margin and slack scaling) as a special case. Namely, for the generic representation given in (2.7) we can define $F(\boldsymbol{y}) = \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$, $\eta(\boldsymbol{G}(\boldsymbol{y})) = \Delta(\boldsymbol{y}^*, \boldsymbol{y})$ for a suitable $\eta\colon \mathbb{R}^P \to \mathbb{R}_+$, and

$$H(F(\boldsymbol{y}), \boldsymbol{G}(\boldsymbol{y})) = F(\boldsymbol{y}) \odot \eta(\boldsymbol{G}(\boldsymbol{y})). \tag{3.19}$$

It is worth noting that a considerable number of non decomposable (or high order) loss functions in structured prediction can be represented as a multivariate cardinality-based potential $\eta(\boldsymbol{G}(\cdot))$ where the mapping $\boldsymbol{G}$ encodes the label statistics, e.g., the number of true or false positives with respect to the ground truth. Furthermore, the maximal number of states for the corresponding auxiliary variables related to $\boldsymbol{G}$ is polynomially bounded in the the number of variables M. In Chapter 5, we give an overview of the existing loss functions along with this compact representation and the resulting computational cost affecting the running time of the corresponding inference algorithm.

3.5.2 Evaluating Generalization Bounds for Structured Prediction

Generalization bounds can give useful theoretical insights in behavior and stability of a learning algorithm by upper bounding the expected loss or the risk of a prediction function. Evaluating such a bound could provide certain guarantees how a system trained on some finite data will perform in the future on the unseen examples. Unlike in the standard regression or classification tasks with univariate real-valued outputs, in structured prediction, in order to evaluate a corresponding bound we need to solve a complex combinatorial optimization problem limiting its use in practice. In the following we demonstrate how the presented algorithmic idea can be used to evaluate PAC-Bayesian generalization bounds for max-margin structured prediction. As a working example we consider the following generalization theorem stated in [McA06]:

Theorem 3. *Assume that* $0 \leqslant \Delta(\boldsymbol{y}^*, \boldsymbol{y}) \leqslant 1$. *With probability at least* $1 - \delta$ *over the draw of the training set* $\mathcal{S} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_1, \boldsymbol{y}_n)\}$ *of size* $n \in \mathbb{N}$, *the following holds simultaneously for all weight vectors* $\boldsymbol{w}$*:*

$$\begin{aligned}\mathbf{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\rho}[\Delta(\boldsymbol{y}, h_{\boldsymbol{w}}(\boldsymbol{x}))] \leqslant \frac{\|\boldsymbol{w}\|^2}{n} + \sqrt{\frac{\|\boldsymbol{w}\|^2 \ln(\frac{2dn}{\|\boldsymbol{w}\|^2}) + \ln(\frac{n}{\delta})}{2(n-1)}} \\ + \frac{1}{n}\sum_{i=1}^{n} \max_{\hat{\boldsymbol{y}}} \mathbb{1}_1 \left[\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}_i, \hat{\boldsymbol{y}}) \leqslant \Delta_{HD}(\boldsymbol{y}_i, \hat{\boldsymbol{y}})\right] \cdot \Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}})\end{aligned} \tag{3.20}$$

where $h_{\boldsymbol{w}}(\boldsymbol{x})$ *denotes the prediction function given in (2.2).*

An interesting observation is that Hamming distance Δ_{HD} still appears in the bound term regardless of the choice of the actual loss function Δ we aim to optimize for during the training procedure.

The right hand side of the inequality (3.20) contains two types of terms. While the first term can be easily evaluated, the second term requires for each data point $(\boldsymbol{x}, \boldsymbol{y}^*)$ a maximization over $\boldsymbol{y} \in \mathcal{Y}$ according to

$$\max_{\boldsymbol{y} \in \mathcal{Y}} \mathbb{1}_1 \left[\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}^*) - \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \leqslant \Delta_{\text{HD}}(\boldsymbol{y}^*, \boldsymbol{y}) \right] \cdot \Delta(\boldsymbol{y}^*, \boldsymbol{y}).$$

We now show that evaluating this term (for models with bounded treewidth) is an instance of Problem 1. More precisely, we consider an example with F_1-loss (see Table 5.1 in Chapter 5) and define $F(\boldsymbol{y}) = \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$, and $\eta(\boldsymbol{G}(\boldsymbol{y})) = \Delta_{F_1}(\boldsymbol{y}^*, \boldsymbol{y})$, and

$$H(F(\boldsymbol{y}), \boldsymbol{G}(\boldsymbol{y})) = \mathbb{1}_1 \left[\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}^*) - F(\boldsymbol{y}) \leqslant |\boldsymbol{y}^*| - G_1(\boldsymbol{y}) + G_2(\boldsymbol{y}) \right] \cdot \eta(\boldsymbol{G}(\boldsymbol{y}))$$

where we use $\Delta_{\text{HD}}(\boldsymbol{y}^*, \boldsymbol{y}) = FP + FN$, $FN = |\boldsymbol{y}^*| - TP$, which removes the need of additional auxiliary variables for Hamming distance reducing the resulting computational cost. Here, TP, FP, and FN denote the numbers of true and false positives, as well as the number of false negatives, respectively. $|\boldsymbol{y}^*|$ denotes the size[2] of the output $\boldsymbol{y}^*$. Both $|\boldsymbol{y}^*|$ and $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}^*)$ are constant with respect to the maximization over $\boldsymbol{y}$. Note also that H is non decreasing in $F(\boldsymbol{y})$. Furthermore, the number of states of the auxiliary variables is upper bounded by $R = M^2$. Therefore, the computational complexity of Algorithm 3 (for each single data point) for the presented example is given by $O(M^{2\nu-1} \cdot N^{\tau+1})$. In Chapter 5 we evaluate the generalization bound in theorem 3.20 for a few trained models.

As a final remark we note that training an SSVM corresponds to solving a convex problem but is not consistent. It fails to converge to the optimal predictor even in the limit of infinite training data [Now+14]. However, minimizing the (non-convex) generalization bound is consistent. This provides a trade-off between convexity and consistency for max-margin structured prediction. The fact that we can evaluate this bound can potentially result in development of new training algorithms for structured prediction. For example, we could use Algorithm 3 to compute a subgradient of the right hand side of the generalization bound. Alternatively, we could use the bound value as an early stopping criterion while training an SSVM.

3.5.3 Globally Constrained MAP Inference

In some cases we might want to modify the prediction function by imposing additional constraints on the outputs. For example, we could perform a corresponding MAP inference subject to some constraints on the label statistics which specify the size of an output or the distribution of different label counts. Alternatively, we might want to get the best output with a score from a specific range which can be done by imposing additional constraints on the sufficient statistics of the prediction function.

Note that from a technical perspective, the problem of performing MAP inference subject to some global constraints on the statistics $\boldsymbol{G}(\boldsymbol{y})$ is equivalent to the MAP problem augmented with a global cardinality-based potential $\eta(\boldsymbol{G}(\boldsymbol{y}))$. Namely, we can always define η as an indicator function $\mathbb{1}_{-\infty}[\cdot]$, which returns $-\infty$ if the corresponding (global) constraint on $\boldsymbol{G}(\boldsymbol{y})$ is violated. Furthermore, the form of η does not affect the message passing of the presented algorithms. We can always check the validity of a corresponding constraint after all the necessary statistics have been computed.

[2]Note that in a binary case with fixed graph size, the number of false negatives has a different interpretation and $|\boldsymbol{y}^*|$ must be replaced by the number P of positive labels in $\boldsymbol{y}^*$, since $FN = P - TP$.

Constraints on Label Counts

As a simple example consider the binary sequence tagging experiment. That is, every output $\boldsymbol{y} \in \mathcal{Y}$ is a sequence and each site in the sequence can be either 0 or 1. Given some prior information on the number b of positive labels we could improve the quality of the resulting solution by imposing a corresponding constraint on the outputs as follows

$$\begin{aligned} \underset{\boldsymbol{y} \in \mathcal{Y}}{\text{maximize}} \quad & \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} \quad & \sum_{i=1}^{M} y_i = b \end{aligned} \tag{3.21}$$

We can write this as an instance of Problem 1 by setting $F(\boldsymbol{y}) = \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$, and $G(\boldsymbol{y}) = \sum_{i=1}^{M} y_i$, and

$$H(F(\boldsymbol{y}), G(\boldsymbol{y})) = F(\boldsymbol{y}) + \mathbb{1}_{-\infty}[G(\boldsymbol{y}) \neq b]. \tag{3.22}$$

Since all the variables in $\boldsymbol{y}$ are binary, the number of states R of the corresponding auxiliary variables $l_m = \sum_{i=1}^{m} y_i$ is upper bounded by M. Also, because the output graph is a sequence we have $\nu = 2$. Therefore, the computational complexity of Algorithm 2 and 3 for the presented example is of the order $O(M^2 \cdot N^{\tau+1})$.

As already mentioned, the evaluation of the corresponding constraints is done after the actual message passing procedure. Therefore, we can perform inference subject to arbitrary non linear (inequality) constraints on the label counts.

Constraints on the Value of Prediction Function

We continue with binary sequence tagging example (with pairwise interactions). To enforce constraints on the score to be in a specific range as in

$$\begin{aligned} \underset{\boldsymbol{y} \in \mathcal{Y}}{\text{maximize}} \quad & \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to} \quad & a \leqslant \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \leqslant b \end{aligned} \tag{3.23}$$

we first rewrite the prediction function in therm of its sufficient statistics as follows:

$$\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{o,s} w_{o,s} \underbrace{\sum_{t=1}^{M} \mathbb{1}_1[x_t = o \wedge y_t = s]}_{=:G_{o,s}(\boldsymbol{y})} + \sum_{s_1,s_2} w_{s_1,s_2} \underbrace{\sum_{t=2}^{M} \mathbb{1}_1[y_{t-1} = s_1 \wedge y_t = s_2]}_{=:G_{s_1,s_2}(\boldsymbol{y})} \tag{3.24}$$

where $\boldsymbol{w} = (..., w_{o,s}, ..., w_{s_1,s_2}, ...)$, $\boldsymbol{G} = (..., G_{o,s}, ..., G_{s_1,s_2}, ...)$, and

$$H(F(\boldsymbol{y}), \boldsymbol{G}(\boldsymbol{y})) = F(\boldsymbol{y}) + \mathbb{1}_{-\infty}[\boldsymbol{w}^\top \boldsymbol{G}(\boldsymbol{y}) \notin [a, b]]. \tag{3.25}$$

Note that $\boldsymbol{G}$ contains all the sufficient statistics of F such that $F(\boldsymbol{y}) = \boldsymbol{w}^\top \boldsymbol{G}(\boldsymbol{y})$. Here again we could replace $a \leqslant \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{y}) \leqslant b$ by any (non linear) constraint on the sufficient statistics given by the joint feature map $\boldsymbol{\Psi}(\boldsymbol{y})$.

The corresponding computational complexity can be derived by considering an urn problem, one with $D \cdot N$ and one with N^2 distinguishable urns and M indistinguishable balls. Here, D denotes the size of the dictionary for the observations x_t in the input sequence $\boldsymbol{x}$. Note that the dictionary of the input symbols can be large

with respect to other problem parameters. However, we can reduce D to the size of the vocabulary part only occurring in the current input $\boldsymbol{x}$. The first urn problem corresponds to the unary observation-state statistics $G_{o,s}(\boldsymbol{y})$ and the second to the pairwise statistics for the state transition $G_{s_1,s_2}(\boldsymbol{y})$. The resulting number of possible distributions of balls over the urns is given by

$$\underbrace{\binom{M + D \cdot N - 1}{M}}_{\leqslant M^{D \cdot N}} \cdot \underbrace{\binom{M + N^2 - 1}{M}}_{\leqslant M^{N^2}} \leqslant \underbrace{M^{D \cdot N + N^2}}_{=R}. \tag{3.26}$$

Although the resulting complexity (due to $\nu = 2$) being $O(M^{D \cdot N + N^2 + 1} \cdot N^{\tau+1})$ is still a polynomial in the number of variables M, the degree is quite high such that we can consider only short sequences. For practical use we recommend the efficient approximation framework of Lagrangian relaxation (see Chapter 4).

Constraints on the Search Space

The constraint on the search space can be different from the constraints we can impose on the label counts. For example, we might want to exclude a set of K complete outputs $\{\boldsymbol{y}^1, ..., \boldsymbol{y}^K\}$ from the feasible set $\mathcal{Y}$ by using an exclusion potential $\mathbb{1}_{-\infty}[\boldsymbol{y} \in \{\boldsymbol{y}^1, ..., \boldsymbol{y}^K\}]$. This is also related to the task of finding K-best assignment[3].

For simplicity, we again consider a sequence tagging example. Given a set of K patterns to exclude we can introduce auxiliary variables $\boldsymbol{l}_m \in \{0, 1\}^K$ where for each pattern $\boldsymbol{y}^k$ we have a constraint[4] $(\boldsymbol{l}_m)_k = \max\{\mathbb{1}_1[y^k_m \neq y_m], (\boldsymbol{l}_{m-1})_k\}$. Therefore, the maximal number of states for $\boldsymbol{l}_m$ is given by $R = 2^K$. The resulting computational complexity for finding an optimal solution over $\boldsymbol{y} \in \mathcal{Y} \setminus \{\boldsymbol{y}^1, ..., \boldsymbol{y}^K\}$ is (due to $\nu = 2$) of the order $O(2^K \cdot M \cdot N^{\tau+1})$.

A related problem is finding a *diverse* K-best solution. Here, the goal is to produce best solutions which are sufficiently different from each other according to some diversity function, e. g., a loss function like Hamming distance Δ_{HD}. More precisely, after computing the MAP solution $\boldsymbol{y}^1$ we compute the second best (diverse) output $\boldsymbol{y}^2$ with $\Delta_{\text{HD}}(\boldsymbol{y}^1, \boldsymbol{y}^2) \geqslant m_1$. For the third best solution we then require $\Delta_{\text{HD}}(\boldsymbol{y}^1, \boldsymbol{y}^3) \geqslant m_2$ and $\Delta_{\text{HD}}(\boldsymbol{y}^2, \boldsymbol{y}^3) \geqslant m_2$ and so on. That is, we search for an optimal output $\boldsymbol{y}^K$ such that $\Delta_{\text{HD}}(\boldsymbol{y}^k, \boldsymbol{y}^K) \geqslant m_{K-1}$, $m_k \in \mathbb{N}$ for all $k \in \{1, ..., K-1\}$.

For this purpose, we define auxiliary variables $\boldsymbol{l}_m \in \{0, ..., M\}^{K-1}$ where for each pattern $\boldsymbol{y}^k$ we have a constraint $(\boldsymbol{l}_m)_k = (\boldsymbol{l}_{m-1})_k + \mathbb{1}_1[y^k_m \neq y_m]$ computing the Hamming distance of a solution $\boldsymbol{y}$ with respect to the pattern $\boldsymbol{y}^k$. Therefore, we can define

$$H(F(\boldsymbol{y}), G(\boldsymbol{y})) = F(\boldsymbol{y}) - \mathbb{1}_\infty[G_k(\boldsymbol{y}) \geqslant m_k \text{ for } k \in \{1, ..., K-1\}] \tag{3.27}$$

where $\boldsymbol{G} = (G_1, ..., G_{K-1})$ and at the final stage (due to $\boldsymbol{G}(\boldsymbol{y}) = \boldsymbol{l}_M$) we have all the necessary information to evaluate the constraints with respect to the diversity function (here Hamming distance). The maximal number of states R for the auxiliary variables is upper bounded by M^{K-1}. Therefore, the resulting running time is (due to $\nu = 2$) of the order $O(M^K \cdot N^{\tau+1})$.

[3]Note that for the standard problem of finding K-best assignment for an MRF, instead of using an exclusion potential, we could use the existing approach which is more efficient. Finding K-best assignment, however, is not equivalent to the problem of excluding specific patterns.

[4]More precisely, we modify the message computation in (3.13) with respect to the auxiliary variables by replacing the corresponding constraints $(\boldsymbol{l}_m)_k = (\boldsymbol{l}_{m-1})_k + \mathbb{1}_1[y^k_m \neq y_m]$ in the maximization over $\{\boldsymbol{l}_m\}$ by the constraints $(\boldsymbol{l}_m)_k = \max\{\mathbb{1}_1[y^k_m \neq y_m], (\boldsymbol{l}_{m-1})_k\}$.

Finally, we note that the concept of diverse K-best solutions can also be used during the training of SSVMs to speed up the convergence of a corresponding algorithm by generating diverse cutting planes or subgradients as described in [GKB13]. An appealing property of Algorithm 2 and Algorithm 3 is that we get some part of the necessary information for free as a side-effect of the message passing.

Structured AUC Optimization Framework

As a further potential use case we use the presented algorithmic idea to motivate a novel learning approach which aims at optimizing for a new structured AUC measure of performance. See Appendix A.3 for details.

3.6 Summary and Discussion

The general task of (globally) augmented MAP inference is ubiquitous in structured prediction and occurs in various contexts during the training, prediction, and post processing phase. Although the family of the corresponding problems can significantly vary in its precise shape a considerable number of its members shares the same unifying property that the information on the global variable interactions imposed by either a global factor or a global constraint can be locally propagated trough the network by means of the dynamic programming (or message passing). Based on this insight we defined a generic representation of the augmented MAP problem (see Problem 1), which explicitly describes the essential properties enabling efficient exact inference: joint τ-decomposability of involved mappings and the limited number of states of their sufficient statistics. In particular, the precise shape of the wrapping function H is largely unrestricted, which allows for modeling complex nonlinear dependencies by combining the score function F and the label statistics $\boldsymbol{G}$ in (almost) any imaginable way. We demonstrated this flexibility on a few examples of potential use cases.

We presented an exact message passing algorithm, which is guaranteed to find an optimal solution in time polynomial in the description size of the problem instance – provided the number of states of the auxiliary variables is also polynomially bounded, which is true in the most practical cases. We implemented two different versions of the algorithm: one for message passing on factor graphs (Algorithm 2) and one on clique trees (Algorithm 3). Message passing on factor graphs is straightforward and does not require additional graph transformations but suffers from the fact that the resulting time complexity is exponential in the maximal degree of the variable nodes. As illustrated in Figure 3.3 this can render the corresponding computations intractable. However, using the concept of clique trees it is always possible (see Proposition 5) to find a suitable representation for efficient inference. We proved the correctness of the presented algorithms and provided a detailed analysis of their computational complexity (see Theorem 1, Theorem 2).

Furthermore, we demonstrated how the presented framework can be used in several application scenarios including the general task of loss augmented inference, evaluation of PAC-Bayesian generalization bounds for structured prediction and globally constrained MAP inference. As a further potential application we used the presented algorithmic idea to motivate a novel learning approach for SSVMs to optimize for a new structured AUC measure of performance.

Chapter 4

Approximate and Distributed Inference — Theory

Although the main focus in the thesis is on exact inference, for the sake of completeness, we here discuss known approximate methods, for which we provide a few novel findings. More precisely, we extend the range of handleable problems by dropping tractability assumptions made in Chapter 3, which enabled exact inference. In particular, we consider models with unbounded treewidth and present an approximate inference approach based on the powerful optimization techniques of Lagrangian relaxation and, more specifically, dual decomposition. An appealing property of these techniques is that they provide feasible solutions with a certificate of optimality, which allows for an efficient verification whether a given solution is optimal. Furthermore, the computations can be performed in a distributed way.

In the first part of the chapter we provide novel insights connecting the two established optimization techniques of linear programming (LP) relaxation and dual decomposition (DD) for the general task of discrete MAP inference. In the second part we discuss the intricacies of the mentioned relaxation techniques when applied to the task of (globally) augmented MAP inference and show how they can benefit from the exact algorithm proposed in the previous chapter.

The main **contributions** in this chapter are the following:

- We provide **novel** insights connecting the two established optimization techniques of LP relaxation and DD for the general task of MAP inference in discrete MRFs.
- For the case of binary pairwise MRFs we show the **partial optimality** property of DD and present a procedure for extracting the **strongly persistent** part from the corresponding solutions.
- We analyze the intricacies of an accurate and efficient approximation framework of Lagrangian relaxation and DD when applied to the task of (globally) augmented MAP inference.
- We propose a **simple** bisection method for the special case of MAP inference subject to a **global** linear constraint.

Parts of this chapter are based on:

> A. Bauer, S. Nakajima, N. Görnitz and K.-R. Müller. Partial Optimality of Dual Decomposition for MAP Inference in Pairwise MRFs. *arXiv preprint, 2017.*

4.1 MAP Inference on Pairwise MRFs

In the following we introduce the existing approaches of linear programming (LP) relaxation and dual decomposition (DD) for the general task of MAP inference on the example of pairwise MRFs and provide novel insights regarding the (partial) optimality of the corresponding solutions for this case. Some of the findings generalize also to models with arbitrary higher order dependencies.

4.1.1 MAP Inference as an Optimization Problem

For a set of n discrete variables $\boldsymbol{x} = \{x_1, ..., x_n\}$ taking values from a finite set S we define the energy of a pairwise MRF factorizing over a graph $G = (\mathcal{V}, \mathcal{E})$ according to:

$$E(\boldsymbol{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{i,j}(x_i, x_j), \tag{4.1}$$

where the functions $\theta_i(\cdot)\colon S \to \mathbb{R}$, $\theta_{i,j}(\cdot,\cdot)\colon S \times S \to \mathbb{R}$ denote the corresponding unary and pairwise potentials, respectively. The *maximum a posteriori* (MAP) problem, that is, computing an assignment with the highest probability is equivalent to the problem of finding an assignment which minimizes the energy.

Probably the most popular method for solving this problem is based on the linear programming (LP) relaxation technique. For this purpose, the MAP problem is first represented as an (equivalent) integer linear problem (ILP):

$$\underset{\boldsymbol{\mu} \in \mathcal{X}_G}{\text{minimize}} \quad \boldsymbol{\theta}^\top \boldsymbol{\mu} \tag{4.2}$$

where $\boldsymbol{\mu}$ corresponds to a joint variable assignment $\boldsymbol{x}$ in the standard overcomplete representation [WJ08]. That is, $\boldsymbol{\theta}$ is a vector with entries $\theta_i(x_i)$ for all $i \in \mathcal{V}, x_i \in S$ and $\theta_{i,j}(x_i, x_j)$ for all $(i,j) \in \mathcal{E}$, $x_i, x_j \in S$, and $\boldsymbol{\mu}$ is a binary vector of indicator functions for nodes and edges $\mu_i(x_i), \mu_{i,j}(x_i, x_j) \in \{0,1\}$, where $\mu_i(s) = 1 \Leftrightarrow x_i = s$ and $\mu_{i,j}(s_1, s_2) = 1 \Leftrightarrow x_i = s_1 \wedge x_j = s_2$. The set $\mathcal{X}_G$ corresponds to all valid assignments of a pairwise MRF over a graph G and has the following compact representation:

$$\mathcal{X}_G := \left\{ \boldsymbol{\mu} \in \mathbb{R}^d \;\middle|\; \begin{array}{ll} \sum_{x_i} \mu_i(x_i) = 1 & \forall i \in \mathcal{V} \\ \sum_{x_i} \mu_{i,j}(x_i, x_j) = \mu_j(x_j) & \forall (i,j) \in \mathcal{E}, \forall x_j \in S \\ \sum_{x_j} \mu_{i,j}(x_i, x_j) = \mu_i(x_i) & \forall (i,j) \in \mathcal{E}, \forall x_i \in S \\ \mu_i(x_i) \in \{0,1\} & \forall i \in \mathcal{V}, \forall x_i \in S \\ \mu_{i,j}(x_i, x_j) \in \{0,1\} & \forall (i,j) \in \mathcal{E}, \forall x_i, x_j \in S \end{array} \right\} \tag{4.3}$$

A convex hull of this set, which we denote by $\mathcal{M}_G := \text{conv}\, \mathcal{X}_G$ plays a special role in the optimization and is known as the *marginal polytope* of a corresponding MRF. Namely, the problem (4.2) is equivalent to the one where we replace the set $\mathcal{X}_G$ by its convex hull $\mathcal{M}_G$, that is

$$\min_{\boldsymbol{\mu} \in \mathcal{X}_G} \boldsymbol{\theta}^\top \boldsymbol{\mu} = \min_{\boldsymbol{\mu} \in \mathcal{M}_G} \boldsymbol{\theta}^\top \boldsymbol{\mu}. \tag{4.4}$$

Since finding an optimal solution of the above ILP or equivalently minimizing its linear objective over the marginal polytope is in general intractable, we usually consider the following relaxation:

$$\underset{\boldsymbol{\mu} \in L_G}{\text{minimize}} \quad \boldsymbol{\theta}^\top \boldsymbol{\mu} \tag{4.5}$$

where we optimize over a bigger set $L_G \supseteq \mathcal{M}_G \supseteq \mathcal{X}_G$ called the *local consistency polytope* of a MRF over a graph G, which results from relaxing the integrality constraints $\mu_i(x_i), \mu_{i,j}(x_i, x_j) \in \{0, 1\}$ in the definition of $\mathcal{X}_G$ by allowing the corresponding variables to take all real values in the interval $[0, 1]$. That is,

$$L_G := \left\{ \boldsymbol{\mu} \in \mathbb{R}^d \;\middle|\; \begin{array}{ll} \sum_{x_i} \mu_i(x_i) = 1 & \forall i \in \mathcal{V} \\ \sum_{x_i} \mu_{i,j}(x_i, x_j) = \mu_j(x_j) & \forall (i,j) \in \mathcal{E}, \forall x_j \\ \sum_{x_j} \mu_{i,j}(x_i, x_j) = \mu_i(x_i) & \forall (i,j) \in \mathcal{E}, \forall x_i \\ \mu_{i,j}(x_i, x_j) \geqslant 0 & \forall (i,j) \in \mathcal{E}, \forall x_i, x_j \in S \end{array} \right\} \tag{4.6}$$

Note that the non-negativity of the unary variables $\mu(x_i) \geqslant 0$ implicitly follows from the combination of the agreement constraints between node and edge variables and the non-negativity of the latter.

4.1.2 Optimization via Dual Decomposition

We now briefly review the DD framework for MAP inference in (pairwise) MRFs [KPT11]. The main idea is to decompose the original intractable optimization problem (OP) in (4.2) over a graph G into a set of tractable inference problems over subtrees $\{\mathcal{T}_j\}_{j=1}^m$, $\mathcal{T}_j \subseteq G$, which are coupled by a set of agreement constraints to ensure the consistency. That is, each of the individual subproblems $j \in \{1, ..., m\}$ corresponds to the MAP inference on a subtree $\mathcal{T}_j = (\mathcal{V}_j, \mathcal{E}_j)$ of the original MRF. More precisely, we define the following OP

$$\begin{array}{ll} \underset{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in \mathcal{X}_{\mathcal{T}_m}, \boldsymbol{\nu}}{\text{minimize}} & \displaystyle\sum_{j=1}^{m} {\boldsymbol{\theta}^j}^\top \boldsymbol{\mu}^j \\ \text{subject to} & \mu_i^j(x_i) = \nu_i(x_i) \quad \forall j \in \{1, ..., m\}, \forall i \in \mathcal{V}_j, \forall x_i \in S \end{array} \tag{4.7}$$

where each vector $\boldsymbol{\mu}^j$ denotes the variables of a local subproblem with respect to $\mathcal{T}_j$, and $\boldsymbol{\nu}$ is a set of global variables $\nu_i(x_i)$ on which the variables $\mu_i^j(x_i)$ of the (overlapping) subproblems must agree. We can choose any decomposition with the only condition that the corresponding trees together must cover all the nodes and edges of G, that is, $\mathcal{V} = \bigcup_{j=1}^m \mathcal{V}_j$ and $\mathcal{E} = \bigcup_{j=1}^m \mathcal{E}_j$, as well as $\boldsymbol{\theta}^\top \boldsymbol{\mu} = \sum_{j=1}^m \boldsymbol{\theta}_j^\top \boldsymbol{\mu}^j$. Note that OP in (4.7) is equivalent to the ILP in (4.2). A corresponding LP relaxation given by

$$\begin{array}{ll} \underset{\boldsymbol{\mu}^1 \in L_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in L_{\mathcal{T}_m}, \boldsymbol{\nu}}{\text{minimize}} & \displaystyle\sum_{j=1}^{m} {\boldsymbol{\theta}^j}^\top \boldsymbol{\mu}^j \\ \text{subject to} & \mu_i^j(x_i) = \nu_i(x_i) \quad \forall j \in \{1, ..., m\}, \forall i \in \mathcal{V}_j, \forall x_i \in S \end{array} \tag{4.8}$$

is equivalent to the OP in (4.5) in the sense that both have the same optimal value and the same optimal solution set.

In the corresponding dual problems the goal is to maximize the dual function of the OPs in (4.7) and (4.8) according to

$$\underset{\boldsymbol{u} \in \mathcal{U}}{\text{maximize}} \quad g_{4.7}(\boldsymbol{u}), \text{ where } g_{4.7}(\boldsymbol{u}) = \inf_{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in \mathcal{X}_{\mathcal{T}_m}} \left\{ \sum_{j=1}^{m} ({\boldsymbol{\theta}^j} + \boldsymbol{u}^j)^\top \boldsymbol{\mu}^j \right\} \tag{4.9}$$

and

$$\underset{\boldsymbol{u}\in\mathcal{U}}{\text{maximize}} \quad g_{4.8}(\boldsymbol{u}), \text{ where } g_{4.8}(\boldsymbol{u}) = \inf_{\boldsymbol{\mu}^1\in L_{\mathcal{T}_1},\dots,\boldsymbol{\mu}^m\in L_{\mathcal{T}_m}} \left\{ \sum_{j=1}^{m} (\boldsymbol{\theta}^j + \boldsymbol{u}^j)^\top \boldsymbol{\mu}^j \right\} \tag{4.10}$$

respectively, over a restricted set of dual values

$$\mathcal{U} := \left\{ \boldsymbol{u} \colon \sum_{j\colon i\in\mathcal{V}_j} \boldsymbol{u}_i^j(x_i) = 0, i \in \{1,\dots,n\}, x_i \in S \right\} \tag{4.11}$$

We here overload the notation in the following sense. The dual variables have the following form $\boldsymbol{u} = (\boldsymbol{u}^1,\dots,\boldsymbol{u}^m)$, $\boldsymbol{u}^j = (\dots, u_i^j(x_i), \dots)$. Therefore, since we ignore the edges, the number of dual variables $\boldsymbol{u}^j$ is smaller than the dimensionality of $\boldsymbol{\mu}^j$ (or $\boldsymbol{\theta}^j$). However, for the algebraic operations (e.g. inner product) to make sense we implicitly assume that the vector $\boldsymbol{u}^j$ (if required) is appropriately filled with zeros to get the same dimensionality as $\boldsymbol{\mu}^j$. A derivation of the above dual problems can be found in [KPT11]. There are different ways to solve a corresponding dual problem. The most popular is a subgradient method for convex non differentiable objectives. Alternatively, we could use a variant of block coordinate descent or cutting plane algorithm.

4.1.3 Connections between LP Relaxation and Dual Decomposition

The facts summarized in Case 1 below are mainly known. We provide them for the sake of completeness. In Case 2 we present new insights connecting the two optimization techniques.

Case 1: LP Relaxation Yields an Integral Solution

It is well known that if the LP relaxation is tight, both the LP relaxation and DD provide an integral optimal solution to the MAP problem. From a different perspective, this means that strong duality holds for the OP in (4.7). That is, there is zero duality gap between optimal values of OPs in (4.7) and (4.9). Equivalently, it implies the existence of consistent optimal assignments to subtrees according to a chosen decomposition. We summarize these insights in the following lemma.

Lemma 1. *The following claims are equivalent:*

(i) LP relaxation in (4.5) has an integral solution
(ii) strong duality holds for problem (4.7)
(iii) $\bar{\mu}_i^{j_1}(x_i) = \bar{\mu}_i^{j_2}(x_i) \quad \forall j_1, j_2 \in \{1,\dots,m\}, i \in \mathcal{V}_{j_1} \cap \mathcal{V}_{j_2}, x_i \in S$

where $\bar{\boldsymbol{\mu}} = (\bar{\boldsymbol{\mu}}^1,\dots,\bar{\boldsymbol{\mu}}^m)$ *is a (not necessarily unique) minimizer of the Lagrangian* $\mathcal{L}(\cdot,\dots,\cdot,\boldsymbol{u}^*)$ *for OP in (4.7) and* $\boldsymbol{u}^*$ *is a dual optimal.*

Case 2: LP Relaxation Yields a Fractional Solution

If the LP relaxation is not tight, a corresponding optimal solution $\boldsymbol{\mu}^*$ will have fractional components. Given such a fractional solution, we denote by $\mathcal{I} \subseteq \{1,\dots,n\}$ the indices of the variables $x_1,\dots,x_n$, which have been assigned an integral value in $\boldsymbol{\mu}^*$, and by $\mathcal{F} \subseteq \{1,\dots,n\}$ the remaining set of indices corresponding to the fractional part. Formally, $i \in \mathcal{I} \Leftrightarrow \forall x_i \in S\colon \mu_i^*(x_i) \in \{0,1\}$ and $\mathcal{F} = \{1,\dots,n\} \setminus \mathcal{I}$. In

contrast to LP relaxation, assignments produced via DD are fully integral. Given a tree decomposition of a corresponding MRF, the optimal assignments to different subtrees, however, will partially disagree on the overlapping parts. Here we denote by $\mathcal{A} \subseteq \{1, ..., n\}$ the indices of the variables $x_1, ..., x_n$ with unique assignment. Formally, $i \in \mathcal{A}$, if for every tree which contains x_i and every optimal assignment to that tree, x_i has the same value. Similarly we denote by $\mathcal{D} \subseteq \{1, ..., n\}$ the set of indices for which at least two different trees disagree on their optimal assignments, that is, $\mathcal{D} = \{1, ..., n\} \setminus \mathcal{A}$. We now provide a formal analysis of the relationship between the sets $\mathcal{I}$ and $\mathcal{A}$, or equivalently between $\mathcal{F}$ and $\mathcal{D}$.

Theorem 4. *Let $\boldsymbol{\mu}^*$ be an optimal fractional solution of the LP relaxation (4.5), $\bar{\boldsymbol{u}}$ a dual optimal for OP in (4.7), and $\mathcal{L}\colon \mathcal{X}_{\mathcal{T}_1} \times ... \times \mathcal{X}_{\mathcal{T}_m} \times \mathcal{U} \to \mathbb{R}$ a corresponding Lagrangian. There always exists a set of minimizers $\bar{\boldsymbol{\mu}}^1 \in \mathcal{X}_{\mathcal{T}_2}, ..., \bar{\boldsymbol{\mu}}^m \in \mathcal{X}_{\mathcal{T}_m}$ of the Lagrangian $\mathcal{L}(\cdot, ..., \cdot, \bar{\boldsymbol{u}})$ which agree with the integral part $\mathcal{I}$ of $\boldsymbol{\mu}^*$, that is,*

$$\forall j \in \{1, ..., m\}, i \in \mathcal{I} \cap \mathcal{V}_j, x_i \in S\colon \quad \bar{\mu}_i^j(x_i) = \mu_i^*(x_i),$$

for short $\bar{\boldsymbol{\mu}}_{\mathcal{I}}^j = \boldsymbol{\mu}_{\mathcal{I}}^*$.

The result in the above theorem has the most intuitive interpretation in the case of a decomposition into spanning trees, that is, when every tree $\mathcal{T}_j$ covers all the nodes ($\mathcal{V}_j = \mathcal{V}$) of the original graph. In that case Theorem 4 implies that for any optimal solution of the LP relaxation and a corresponding assignment $\boldsymbol{x}^*$ with an integral part $\mathcal{I}$, there exist optimal assignments $\boldsymbol{x}^1, ..., \boldsymbol{x}^m$ (from a dual solution $\bar{\boldsymbol{u}}$) for the different spanning trees which agree on a set of nodes $\mathcal{A}$ with $x_i^j = x_i^*$ for all $i \in \mathcal{I} \subseteq \mathcal{A}$. An immediate question arising is whether the two sets $\mathcal{I}$ and $\mathcal{A}$ are equal. The answer is no. In general, the two sets are not the same and $\mathcal{I}$ will usually be a proper subset of $\mathcal{A}$.

Theorem 4 motivates the following simple heuristic for getting an approximate integral solution, which is especially suitable for a decomposition over spanning trees when using subgradient optimization. Namely, we can consider optimal assignments for every spanning tree and choose the best according to the value of the primal objective. Increasing the number of trees in a decomposition also increases the chance of finding a good assignment. Furthermore, during the optimization we can repeat this for the intermediate results after each iteration of a corresponding optimization algorithm saving the currently best solution. Obviously, this only can improve the quality of the resulting assignment. Note that it is the usual practice with subgradient methods to save the intermediate results since the objective is not guaranteed to improve in every step but can even get worse. In that sense, the above heuristic does not impose additional computational cost.

Finally, it turns out that in a non degenerate case, where LP relaxation has a unique solution, the relationship $\mathcal{I} \subseteq \mathcal{A}$ (and therefore $\mathcal{D} \subseteq \mathcal{F}$) holds for all minimizers of a corresponding Lagrangian $\mathcal{L}(\cdot, ..., \cdot, \bar{\boldsymbol{u}})$ supported by the following theorem.

Theorem 5. *Let $\boldsymbol{\mu}^*$ be a unique optimal solution of the LP relaxation (4.5), $\bar{\boldsymbol{u}}$ a dual optimal for OP in (4.7), and $\mathcal{L}\colon \mathcal{X}_{\mathcal{T}_1} \times ... \times \mathcal{X}_{\mathcal{T}_m} \times \mathcal{U} \to \mathbb{R}$ a corresponding Lagrangian. Each set of minimizers $\bar{\boldsymbol{\mu}}^1 \in \mathcal{X}_{\mathcal{T}_2}, ..., \bar{\boldsymbol{\mu}}^m \in \mathcal{X}_{\mathcal{T}_m}$ of the Lagrangian $\mathcal{L}(\cdot, ..., \cdot, \bar{\boldsymbol{u}})$ agrees with the integral part $\mathcal{I}$ of $\boldsymbol{\mu}^*$, that is,*

$$\forall j \in \{1, ..., m\}, i \in \mathcal{I} \cap \mathcal{V}_j, x_i \in S\colon \quad \bar{\mu}_i^j(x_i) = \mu_i^*(x_i)$$

for short $\bar{\boldsymbol{\mu}}_{\mathcal{I}}^j = \boldsymbol{\mu}_{\mathcal{I}}^*$.

In particular, for binary pairwise MRFs this result implies that each assignment obtained via DD is partially optimal in a sense that it always contains the strongly persistent part of the (fractional) solution of the LP relaxation. Provided the fractional part is small, it suggests that the obtained assignments will often have a low energy close to the optimum even if the LP relaxation is not tight. This fact and the possibility of a parallel computation renders the dual decomposition a practical tool for the MAP inference upon the LP relaxation. We also note that the property $\mathcal{I} \subseteq \mathcal{A}$ in Theorem 4 still holds for non binary MRFs with arbitrary higher order cliques, but $\mathcal{I}$ is not guaranteed to be strongly persistent anymore. In the following we build on an additional lemma.

Lemma 2. *Assume the setting of Theorem 4. For every subproblem $j \in \{1, ..., m\}$ over a tree $\mathcal{T}_j$ there are minimizers $\bar{\boldsymbol{\mu}}^j, \hat{\boldsymbol{\mu}}^j \in \mathcal{X}_{\mathcal{T}_j}$, where*

$$\mu_i^*(x_i) = \frac{1}{2}(\bar{\mu}_i^j(x_i) + \hat{\mu}_i^j(x_i)) \tag{4.12}$$

holds for all $i \in \mathcal{V}_j, x_i \in S$.

The above lemma ensures that for each optimal solution $\boldsymbol{\mu}^*$ of the LP relaxation, for each tree in a given decomposition there always exist two different assignments which agree exactly on the nodes corresponding to the integral (strongly persistent) part $\mathcal{I}$ of $\boldsymbol{\mu}^*$. It is also worth noting that when LP relaxation is not tight the sets of minimizing assignments to the different trees are disjoint on the overlapping parts due to Lemma 1.

Lemma 2 and Theorem 5 together imply that the unambiguous part among all optimal assignments from all trees in a decomposition coincides with the integral part of the (fractional) optimal assignment via LP relaxation giving rise to the following theorem.

Theorem 6. *Let $\boldsymbol{\mu}^*$ be a unique optimal solution for the LP relaxation (4.5) and $\bar{\mathbf{u}}$ a dual optimal for OP in (4.7). Then the unambiguous part $\mathcal{A}$ of optimal assignments among all the overlapping subproblems in a decomposition coincides with the integral part $\mathcal{I}$ of $\boldsymbol{\mu}^*$. That is, $\mathcal{A} = \mathcal{I}$.*

That is, we can extract the strongly persistent part from DD by considering the intersection of optimal assignments for individual trees. This is in particular convenient for a decomposition over single edges, since computing the set of optimal assignments for an edge is straightforward.

Numerical Validation

Here we present a numerical experiment summarized in Figure 4.1 to validate the theoretical statements in the paper. For this purpose we considered a 5×5 Ising grid model corresponding to 25 pixels and defined its energy according to the following procedure. The unary potentials have been all set to zero. The values of the corresponding edge potentials have been selected uniformly at random from an interval $[-0.5, 0.5]$. This process has been repeated until the corresponding LP relaxation yielded a fractional solution $\boldsymbol{\mu}^*$ (see the fourth plot in the first row).

Given such an energy, we then considered a decomposition of the above Ising model into two spanning trees $\mathcal{T}_1$ (all the vertical edges) and $\mathcal{T}_2$ (all the horizontal edges). We used the subgradient method to solve a corresponding dual problem and got optimal assignments $\bar{\boldsymbol{\mu}}^1$ and $\bar{\boldsymbol{\mu}}^2$ corresponding to subproblems $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively (see the first two plots in the first row).

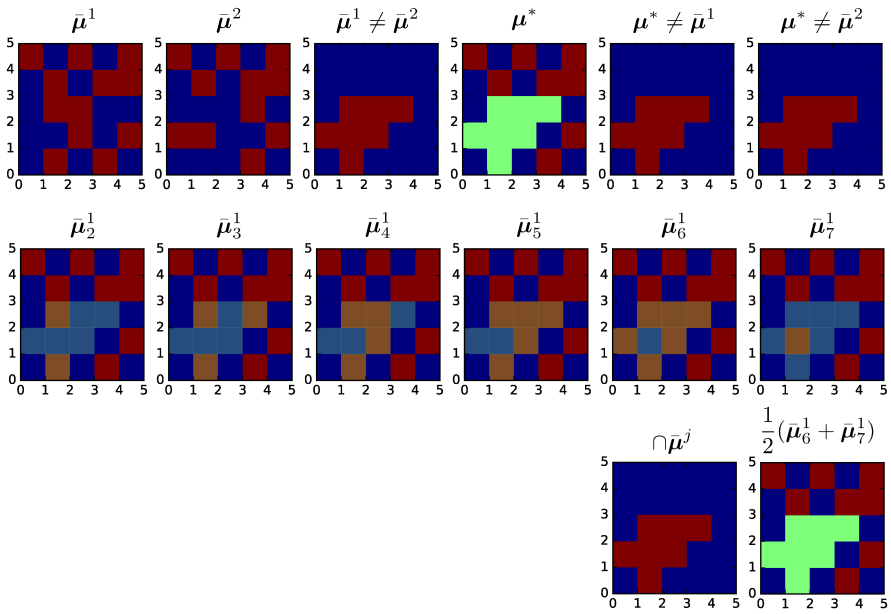


Figure 4.1: Illustration of numerical validation of the theoretical results including Theorem 4, 5, 6 and Lemma 2. We consider a decomposition of an 5×5 Ising grid model into two spanning trees (more precisely, disconnected forests): $\mathcal{T}_1$ consisting of all vertical edges and $\mathcal{T}_2$ consisting of all horizontal edges. The corresponding optimal assignments for both trees are denoted by $\bar{\boldsymbol{\mu}}^1$ and $\bar{\boldsymbol{\mu}}^2$, respectively. $\boldsymbol{\mu}^*$ is a corresponding optimal assignment from the LP relaxation. Here, the red area corresponds to label 1, blue area to label 0, and green area to label 0.5. The individual plots in the first row visualize a corresponding assignment or a disagreement between two assignments. The second row provides (additionally to $\bar{\boldsymbol{\mu}}^1$) further optimal assignments for the subproblem $\mathcal{T}_1$. The first plot in the third row illustrates the unambiguous part among all optimal and overlapping assignments in blue and the ambiguous part in red. The last plot illustrates an average of the two assignments above.

The last two plots in the upper row in Figure 4.1 can serve as a validation of Theorem 4. Namely, the red area in these two plots corresponds to the variables on which each assignment disagrees with the LP solution $\boldsymbol{\mu}^*$. As we can see it happens only for the fractional area, where each variable in $\boldsymbol{\mu}^*$ has the value 0.5. In other words, $\bar{\boldsymbol{\mu}}^1$ and $\bar{\boldsymbol{\mu}}^2$ agree with the integral part of $\boldsymbol{\mu}^*$.

To support Theorem 5 we computed further optimal assignments $\bar{\boldsymbol{\mu}}^1_2, ..., \bar{\boldsymbol{\mu}}^1_7$ for the subproblem $\mathcal{T}_1$ additionally to $\bar{\boldsymbol{\mu}}^1$. These are visualized in the second row in Figure 4.1. We overlay these assignments with the assignment $\boldsymbol{\mu}^*$ in a transparent way to emphasize the difference to the LP solution. We see that all these optimal assignments agree with the integral part of $\boldsymbol{\mu}^*$.

The last plot in the third row validates the statement in Lemma 2. It visualizes the average of the the two plots above corresponding to the optimal assignments $\bar{\boldsymbol{\mu}}^1_6$ and $\bar{\boldsymbol{\mu}}^1_7$ for the subproblem $\mathcal{T}_1$.

Finally, the first plot in the third row supports the claim in Theorem 6. Namely, the blue area corresponds to the unambiguous part $\mathcal{A}$ where each variable has a unique value in all the optimal assignments among the two subproblems. The red area marks the ambiguous part, where each variable has different values in different assignments. We see that the equality $\mathcal{A} = \mathcal{I}$ holds.

Extracting the Strongly Persistent Part

The presented theory suggests a simple way to extract the strongly persistent part of a solution of the LP relaxation (provided it is unique) from the assignments via DD in the case of binary pairwise MRFs as follows. If we consider a decomposition over

individual edges, then computing all optimal assignments for an edge in the MRF is straightforward by trying all possible node configurations. We do this for every edge in the MRF. Simultaneously, we keep track of labellings among optimal edge assignments for each node in the graph. At the end, the nodes with unique label belong to the strongly persistent part. We can proceed similarly in the case of arbitrary tree decompositions (e.g. spanning trees). Here, we can use the K-best Viterbi's algorithm [Wu+08] to extract all the optimal assignments for each tree in a decomposition. In the case of spanning trees we need to consider only one. Finally, we note that using K-best Viterbi's may find many (successive) optimal assignments which disagree with each other only in a few variables slowing the extraction progress. Here using the diverse K-best inference based on the idea of Algorithm 2 and 3 might help to *skip* some of the optimal assignments. Note, however, that by skipping optimal assignments, in general, we lose the guarantee that each node with a unique assignment belongs to the strongly persistent part.

4.2 Augmented MAP Inference on Low Order MRFs

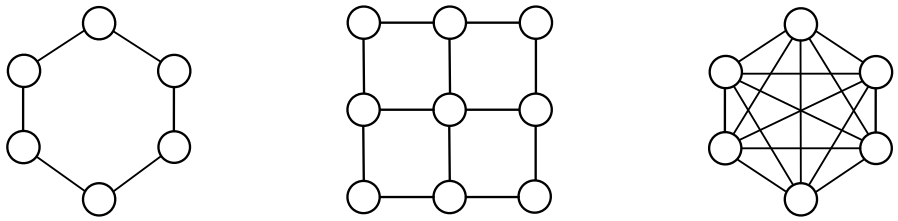
In the following we consider an extension of Problem 1 by dropping the assumption on the MRFs associated with the mappings F and $\boldsymbol{G}$ to have a bounded treewidth. More precisely, we assume that the maximal clique size in a given MRF is sufficiently small but allow the corresponding treewidth to be dependent on the graph size. Figure 4.2 illustrates the difference between the maximal clique size and the treewidth[1]. For simplicity we first consider a special case where a pairwise[2] MRF (with unbounded treewidth) is augmented by a further global potential as described below. We show how to deal with the more general case at the end of the section.

One important high order potential used for applications in natural language processing and computer vision is the cardinality potential [GDS07; TGZ10] that expresses constraints over the number of variables taking on a particular value. In the simplest case of binary variables it is given as a function of the number of variables which are on: $f(\sum_i x_i)$. This generalizes to functions of multivariate counts, e.g., in case of multiple possible labels. Assuming the standard overcomplete representation (introduced in the previous section), we can represent counts as a linear mapping. For example, for a variable assignment $\boldsymbol{\mu}$ in a binary pairwise MRF the sum $\sum_i x_i$ is given by $\mathbf{1}_+^\top \boldsymbol{\mu}$, where $\mathbf{1}_+$ is a binary vector with entries equal to one for all dimensions $\mu_i(x_i)$, $i \in \mathcal{V}$ where $x_i = 1$ and zero otherwise. Similarly, a global potential based on multivariate counts of different variable statistics can be represented by $f(A\boldsymbol{\mu})$, that is, as a composition of an arbitrary function $f\colon \mathbb{R}^r \to \mathbb{R}$ and a linear mapping $A \in \{0,1\}^{r \times d}$. The types of statistics which can be counted are encoded in the matrix A. To give an example different from simple label counts, we could count the number of edges which contain two disagreeing variable nodes. In the task of image segmentation, for example, this corresponds to the boundary length (in pixels) of a resulting segmentation with respect to the class label.

We now consider a special (but practical) problem where a pairwise MRF (in the standard overcomplete representation) is augmented with a high order potential

[1] To remind the reader, the treewidth is given as the maximal clique size (minus one) in a cordless graph after triangulation.

[2] We note that the results presented below generalize beyond pairwise MRFs allowing for higher order dependencies. The only applicability requirement is that the MAP inference on a corresponding subgraph can be performed efficiently. For graphs with sufficiently small cliques this can be done either by considering a decomposition over individual cliques or by considering subgraphs with no cycles over the individual cliques supported by the junction tree construction.



(a) $\tau = 2$ **(b)** $\tau = \sqrt{M}$ **(c)** $\tau = M - 1$

Figure 4.2: Illustration of the difference between the treewidth of a MRF and the maximal clique size. All three examples belong to pairwise models. That is, the maximal clique size is 2. However, the actual treewidth differs significantly between the graphs. In (a) the model is a circle with bounded treewidth $\tau = 2$. The treewidth of the other two graphs is unbounded being a function of the number of the graph nodes. More precisely, for an $n \times n$ grid model in (b) with $M = n^2$ nodes the treewidth is exactly $\tau = \sqrt{M}$. In (c) the graph is fully connected having only one maximal clique with M nodes. Therefore, its treewidth is given by $\tau = M - 1$.

according to

$$\underset{\boldsymbol{\mu} \in \mathcal{X}_G}{\text{minimize}} \quad \boldsymbol{\theta}^\top \boldsymbol{\mu} + f(A\boldsymbol{\mu}) \tag{4.13}$$

In the task of loss augmented inference, for example, the global factor $f(A\boldsymbol{\mu})$ plays the role of a loss function. E.g. for F_1-loss which is a function of the numbers of true (TP) and false positives (FP) given by $\Delta_{F_1}(\boldsymbol{y}^*, \boldsymbol{y}) = 1 - 2\text{TP}/(\text{TP} + \text{FP} + \text{P})$ we can define TP by $\mathbf{1}_{\text{TP}}^\top \boldsymbol{\mu}$, where $\mathbf{1}_{\text{TP}}$ is a binary vector with non zero entries only for dimensions $\mu_i(y_i)$, for which $y_i = y_i^* = 1$. Similarly, the number FP can be represented by $\mathbf{1}_{\text{FP}}^\top \boldsymbol{\mu}$, where $\mathbf{1}_{\text{FP}}$ is a binary vector with non zero entries only for dimension where $y_i = y_i^* = 0$. P denotes the number of positive labels in $\boldsymbol{y}^*$. Therefore, we can define $A = (\boldsymbol{a}_1, \boldsymbol{a}_2)^\top \in \mathbb{R}^{2 \times d}$ where $\boldsymbol{a}_1 := \mathbf{1}_{\text{TP}}$, and $\boldsymbol{a}_2 := \mathbf{1}_{\text{FP}}$, and $f(A\boldsymbol{\mu}) := 2\boldsymbol{a}_1^\top \boldsymbol{\mu}/((\boldsymbol{a}_1 + \boldsymbol{a}_2)^\top \boldsymbol{\mu} + P) - 1$.

Given the interpretation of $f(A\boldsymbol{\mu})$ as a loss function, we consider two different ways how to apply Lagrangian relaxation in order to solve the problem (4.13): 1) by enforcing the loss statistics via a global linear constraint and 2) by treating the loss function as a global potential and separating it from the rest of the graph via dual decomposition.

4.2.1 Enforcing Loss Statistics via Lagrangian Relaxation

Our first approach is intuitive given the explicit representation of the loss function as a composition of some function with a linear transformation. If the range of possible loss statistics given by $\boldsymbol{b} = A\boldsymbol{\mu}$ is rather small, instead of solving problem (4.13) we can consider solving multiple instances of the following constrained problem

$$\begin{aligned} &\underset{\boldsymbol{\mu} \in \mathcal{X}_G}{\text{minimize}} && \boldsymbol{\theta}^\top \boldsymbol{\mu} \\ &\text{subject to} && A\boldsymbol{\mu} = \boldsymbol{b} \end{aligned} \tag{4.14}$$

where the loss statistics have a fixed value $\boldsymbol{b}$, and then choose the best solution for the problem (4.13) according to the objective value. Each instance of the above constrained problem can be (approximately) solved via Lagrangian relaxation. More precisely, we consider a corresponding dual problem, where the goal is to maximize the dual function

$$g(\boldsymbol{u}) = \inf_{\boldsymbol{\mu} \in \mathcal{X}_G} \boldsymbol{\theta}^\top \boldsymbol{\mu} + \boldsymbol{u}^\top (A\boldsymbol{\mu} - \boldsymbol{b}) = \inf_{\boldsymbol{\mu} \in \mathcal{X}_G} (\boldsymbol{\theta} + A^\top \boldsymbol{u})^\top \boldsymbol{\mu} - \boldsymbol{u}^\top \boldsymbol{b}. \tag{4.15}$$

This can be done by using standard optimization techniques for convex non smooth objectives like subgradient methods or cutting plane approach. According to the form of the dual function a corresponding subgradient (or maximal violator) can be computed by using the same inference algorithm as for minimizing the objective in (4.14). Note that the presented method in general provides only an approximate solution in the following sense.

Lemma 3. *Let $\boldsymbol{u} \in \mathbb{R}^r$ be given. Then any optimal solution $\widehat{\boldsymbol{\mu}} = \widehat{\boldsymbol{\mu}}(\boldsymbol{u})$ of the unconstrained optimization problem*

$$\underset{\boldsymbol{\mu} \in \mathcal{X}_G}{\text{minimize}} \;\; (\boldsymbol{\theta} + A^\top \boldsymbol{u})^\top \boldsymbol{\mu} \tag{4.16}$$

is an optimal solution of the constrained optimization problem (4.14) for $\boldsymbol{b} = A\widehat{\boldsymbol{\mu}}$.

Proof. It holds:

$$\begin{aligned} g(\boldsymbol{u}) = \inf_{\boldsymbol{\mu} \in \mathcal{X}_G} \; (\boldsymbol{\theta} + A^\top \boldsymbol{u})^\top \boldsymbol{\mu} - \boldsymbol{u}^\top \boldsymbol{b} &= (\boldsymbol{\theta} + A^\top \boldsymbol{u})^\top \widehat{\boldsymbol{\mu}} - \boldsymbol{u}^\top \boldsymbol{b} \\ &= \boldsymbol{\theta}^\top \widehat{\boldsymbol{\mu}} + \boldsymbol{u}^\top (A\widehat{\boldsymbol{\mu}} - \boldsymbol{b}) \\ &= \boldsymbol{\theta}^\top \widehat{\boldsymbol{\mu}} \;\geqslant \inf_{\boldsymbol{\mu} \in \mathcal{X}_G} \boldsymbol{\theta}^\top \boldsymbol{\mu} \end{aligned} \tag{4.17}$$

Since the value of the dual function always provides a lower bound on the optimal primal value, we get an equality in the last step. That is, $\widehat{\boldsymbol{\mu}}$ has minimal energy and is also feasible with respect to the linear constraint with $\boldsymbol{b} = A\widehat{\boldsymbol{\mu}}$. □

That is, the approximation is not with respect to the energy of an assignment – it is always optimal, but the corresponding constraint $A\boldsymbol{\mu} = \boldsymbol{b}$ can be violated. Enforcing constraints on the model via Lagrangian relaxation is an old technique and has been successfully applied in computer vision for improving accuracy in tasks such as foreground-background segmentation [LJK14; KBR07]. The goal here is to segment the pixels of a given input image in two classes the foreground and background. If we are given some prior knowledge like the size of the foreground we might improve the resulting segmentation accuracy by imposing a corresponding global constraint $\boldsymbol{a}^\top \boldsymbol{\mu} = b$ on the model, where $\boldsymbol{a} = \mathbf{1}_{\mathrm{TP}}$ (as defined in the previous section) and b is the number of pixels with label 1 in the ground truth segmentation. Figure 4.3 illustrates the effectiveness of this approach on a simple example. Here we consider an Ising grid model [BM09] with submodular energies. That is, the nodes are arranged in a grid where we have only pairwise interactions between neighboring pixels. The weights of the individual node potentials have been estimated by maximum likelihood using the color intensities from the part of the image corresponding to the ground truth segmentation[3]. Since the global term corresponding to the size constraint affects only unary potentials and the original model is submodular we can use graph cuts algorithm [BJ01; BV06; KZ04] for inference in each step of a corresponding algorithm for maximizing the dual function in (4.15). As we can see a corresponding optimal segmentation violates the imposed size constraint by a few pixels, but still provides a better result than the segmentation without the global constraint.

If $\widehat{\boldsymbol{\mu}}$ (in Lemma 3) satisfies the corresponding constraint, it is guaranteed to be an optimal solution of the problem (4.14). Here, we denote by $\mathcal{B}$ the set of all loss parameters $\boldsymbol{b}$ for which we can find an optimal solution this way. To summarize, we can solve only a restricted set of instances of the problem (4.14) with respect to the

[3] It is a common approach in image segmentation where the user can marks some pixels in the foreground and background from which the the corresponding weights can be estimated [RKB04].

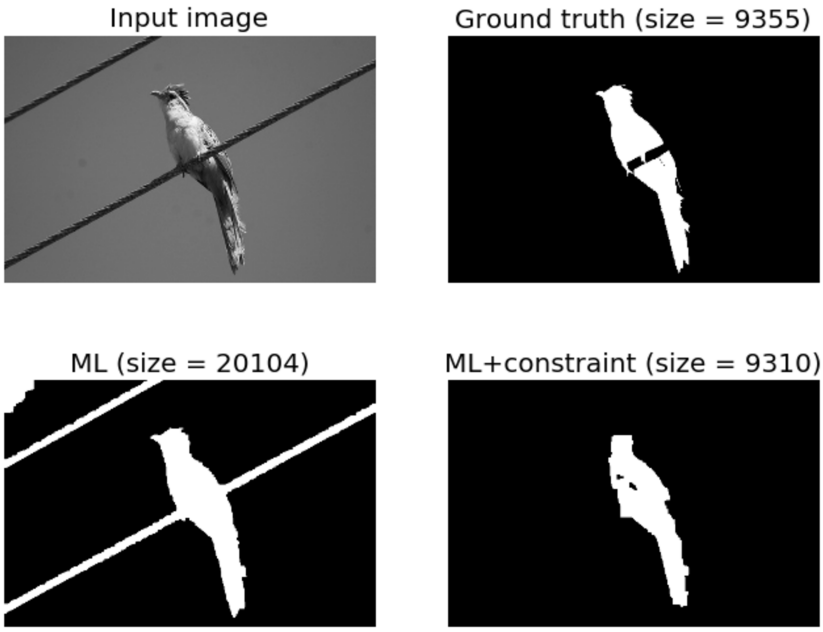


Figure 4.3: Illustration of image segmentation example. The two images in the first row represent an input image (on the left) and the ground truth segmentation in foreground and background (on the right). In the second row we see a segmentation given by maximum-likelihood solution (on the left) and maximum-likelihood solution subject to a global size constraint on the foreground (on the right).

parameter $\boldsymbol{b}$. However, each of the corresponding solutions is optimal. Therefore, we can solve the problem (4.13) exactly, provided the vector with the loss statistics for a corresponding optimal solution is in $\mathcal{B}$. Otherwise, the best candidate can be used as an approximate solution.

Finally, we note that the above approach is trivially parallelizable over different values of $\boldsymbol{b}$. Unfortunately, the range of the corresponding values can be huge. However, in some cases we can use different heuristics to decrease the number of possible loss values to be investigated. In the following we show for an important special case how the loss can be parametrized in a more compact way.

Special Case: Ratio of Counts

The main difficulty here is that the number of loss statistics can be huge. To deal with this problem, we can apply different heuristics to decrease the number of values to be investigated. In particular, in the special case where the loss is given as a ratio of counts (e.g. F_β-score or intersection over union) we can apply the following idea. Instead of iterating over different parameter values of the loss function we could solve multiple instances for different values $t = (\boldsymbol{a}_1^\top \boldsymbol{\mu} + d_1)/(\boldsymbol{a}_2^\top \boldsymbol{\mu} + d_2)$ of the loss function directly. This is equivalent to:

$$\begin{aligned} \underset{\boldsymbol{\mu} \in \mathcal{X}_G}{\text{minimize}} \quad & \boldsymbol{\theta}^\top \boldsymbol{\mu} \\ \text{subject to} \quad & (\boldsymbol{a}_1 - t \cdot \boldsymbol{a}_2)^\top \boldsymbol{\mu} = t \cdot d_2 - d_1 \end{aligned} \tag{4.18}$$

The main advantage of the formulation in (4.18) is that it is not not explicitly dependent on the counting statistics (which can be redundant) as in (4.14), but on the resulting loss value t, therefore reducing the number of points to be investigated. For a normalized loss function, for example, we could consider the equidistant values of $t \in [0, 1]$ according to a desired frequency. Furthermore, we could shrink the number of possible values due to the lemma below by computing the Viterbi assignment in the first step, which corresponds to inference without the high order term.

Lemma 4. *Let $\boldsymbol{\mu}_V^*$ be a solution of $\min_{\boldsymbol{\mu}} \boldsymbol{\theta}^\top \boldsymbol{\mu}$ and $\boldsymbol{\mu}_L^*$ a solution of $\min_{\boldsymbol{\mu}} \boldsymbol{\theta}^\top \boldsymbol{\mu} + f(A\boldsymbol{\mu})$. Then for the corresponding assignments $\boldsymbol{y}_V^*$, $\boldsymbol{y}_L^*$ and the loss $\Delta(\boldsymbol{y}) = -f(A\boldsymbol{\mu})$ the inequality $\Delta(\boldsymbol{y}_V^*) \leqslant \Delta(\boldsymbol{y}_L^*)$ always holds.*

As already mentioned, the approximation of the presented approach is with respect to the corresponding global constraint $A\boldsymbol{\mu} = \boldsymbol{b}$. That is, we will be able to solve (exactly) the instances of the problem (4.14) only for some set of values for $\boldsymbol{b}$. Unfortunately, while the problem (4.18) has a more compact parametrization of the loss function than problem (4.14), it might further shrink the range of possible instances which can be solved by the presented method. The following lemma suggests that the approach of the formulation (4.14) is potentially more accurate. More precisely, it shows that duality gap of the formulation in (4.18) is at least as big as for the formulation (4.14).

Lemma 5. *The equality $g_{4.18}(u) = g_{4.14}((u, -tu)^\top)$ holds, where $g_{4.14}$ and $g_{4.18}$ denote the dual functions of the problem (4.14) and (4.18), respectively. That is, $d_{4.18}^* \leqslant d_{4.14}^*$, where d^* denotes the optimal value of a a corresponding dual problem.*

Alternatively, we could consider the following formulation for different values of z:

$$\begin{aligned} \underset{\boldsymbol{\mu} \in \mathcal{X}_G}{\text{minimize}} \quad & \boldsymbol{\theta}^\top \boldsymbol{\mu} + \frac{1}{z}(\boldsymbol{a}_1^\top \boldsymbol{\mu} + d_1) \\ \text{subject to} \quad & \boldsymbol{a}_2^\top \boldsymbol{\mu} + d_2 = z \end{aligned} \tag{4.19}$$

The corresponding advantage here is that implicitly we only have to iterate over one parameter of a corresponding loss function with respect to $\boldsymbol{a}_2$ (via z), therefore, reducing the number of problem instances to look at. The main difference to the previous two formulations is that the objective is directly affected by the form of the function f in $f(A\boldsymbol{\mu})$ being ratio of counts. The problem is, however, that the constraint $\boldsymbol{a}_2^\top \boldsymbol{\mu} + d_2 = z$ might be not satisfiable and using z directly in the objective can negatively affect the results.

Bisection Method for a Global Linear Constraint

In case with a single global constraint $\boldsymbol{a}^\top \boldsymbol{\mu} = b$ we can solve a corresponding instance of problem (4.14) via a bisection method. Compared to the cutting plane approach the advantage here is that the corresponding algorithm is easy to implement and does not require an additional LP solver.

More precisely, we consider a mapping $B \colon u \mapsto \widehat{b}$ which maps each dual value u to the value of a corresponding global statistic $\widehat{b} = \boldsymbol{a}^\top \widehat{\boldsymbol{\mu}}$ where $\widehat{\boldsymbol{\mu}}$ is an optimal solution of problem (4.16) for the parameters u and $\boldsymbol{a}$. Due to Lemma 3 it is also an optimal solution of the problem (4.14) for $\boldsymbol{a}$ and $\widehat{b}$. An important observation here is that mapping B is monotonic as illustrated in Figure 4.4. This also holds for multiple constraints.

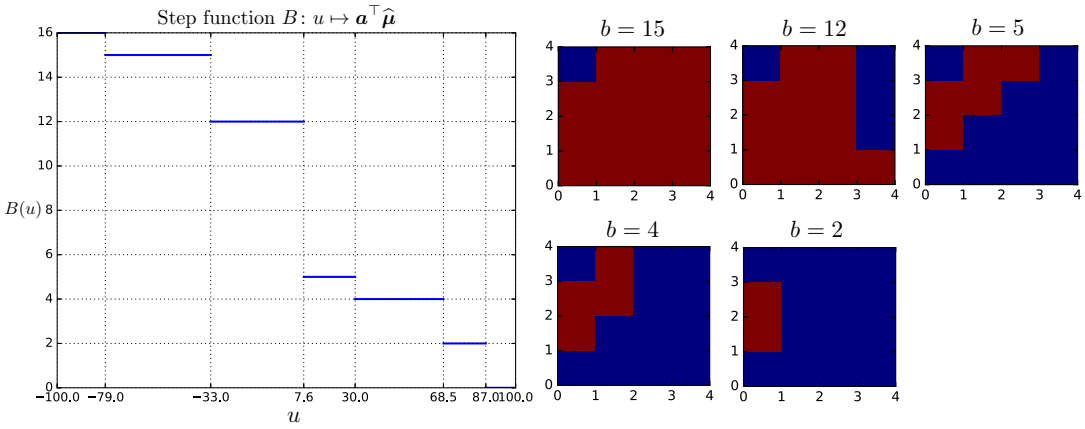


Figure 4.4: Illustration of the approximation effect of Lagrangian relaxation. We consider MAP inference on a 4×4 Ising grid model of binary variables (for some random weights $\boldsymbol{\theta}$) subject to the constraints that the number b of variables with positive values (visualized as red squares on the right) is fixed. The function B (visualized on the left) is a non increasing step function. The problem (4.14) can be solved (exactly) only for values $b \in \{0, 2, 4, 5, 12, 15, 16\}$ visualized on the right (except the trivial cases for $b = 0$ and $b = 16$).

Lemma 6. *Let the matrix A have only non negative entries $A_{i,j} \geqslant 0$. Then all the functions $u_k \mapsto (A\widehat{\boldsymbol{\mu}})_k$, where $\widehat{\boldsymbol{\mu}}$ is an optimal solution of the problem (4.16) for a given $\boldsymbol{u}$, are monotonic.*

The algorithmic idea (summarized in Algorithm 4) is to iteratively evaluate the value $B(u)$ by solving a combinatorial problem (4.16) and apply the bisection search until we find a dual value u which produces a $\widehat{b}$ closest to our desired parameter value b. Namely, we first evaluate B on two initial values u_l and u_r such that $B(u_l) > b > B(u_r)$ (provided B is non increasing – the non decreasing case is symmetric). Given a new dual value u_m (line 3) we first solve the problem (4.16) and get an optimal solution $\widehat{\boldsymbol{\mu}} = \widehat{\boldsymbol{\mu}}(u_m)$ (line 4) and a corresponding $\widehat{b} = \boldsymbol{a}^\top \widehat{\boldsymbol{\mu}}$ (line 8). If $b > \widehat{b}$ we continue the search in the left half $[u_l, u_m]$, and in the right half $[u_m, u_r]$ otherwise until there are no other breakpoints in the interval $[u_l, u_r]$.

An interesting observation is that the presented algorithm can be seen as a special version of the ES algorithm [Gus80; KBR07] adjusted to the binary search: instead of computing all the existing breakpoints of the problem in every step we bisect the search space. The bisection idea could also be applied to multiple constraints as a heuristic where we perform bisection on each loss statistic (that is, dimension of A) one at a time.

Finally, we note that the presented algorithm also provides a special case of the cutting plane algorithm for a single constraint according to the following lemma.

Lemma 7. *Consider the problem (4.14) with a single constraint. The cutting plane algorithm (CPA) (as given in [LJK14]) requires the same number of oracle calls as the bisection method (Algorithm 4). Furthermore, each violated constraint computed by CPA corresponds to the assignment produced by a corresponding oracle for u_m.*

Given the above lemma we now can relate the three algorithms: ES algorithm, CPA, and the presented bisection method. In particular, we are usually interested in solving multiple instances (e.g. loss augmented inference) of problem (4.14). Instead of solving each instance independently, we could consider them together by adjusting the boundaries of the search interval for each value of b simultaneously after each oracle call. Figure 4.5 presents a comparison of the three approaches with respect to

Algorithm 4 Bisection method for problem (4.14) with a single constraint

Compute the solutions $\boldsymbol{\mu}_l$ and $\boldsymbol{\mu}_r$ of the problem $\underset{\boldsymbol{\mu}\in\mathcal{X}_G}{\text{minimize}}\ (\boldsymbol{\theta}+u\cdot\boldsymbol{a})^\top\boldsymbol{\mu}$ for the initial values of the dual variable $u\in\{u_l,u_r\}$
while not finished **do**
  $u_m \leftarrow \boldsymbol{\theta}^\top(\boldsymbol{\mu}_r-\boldsymbol{\mu}_l)/(\boldsymbol{a}^\top(\boldsymbol{\mu}_l-\boldsymbol{\mu}_r))$
  $\boldsymbol{\mu}_m \leftarrow \underset{\boldsymbol{\mu}\in\mathcal{X}_G}{\text{minimize}}\ (\boldsymbol{\theta}+u_m\cdot\boldsymbol{a})^\top\boldsymbol{\mu}$
  if $u_m\in\{u_l,u_r\}$ **then**
    return $\underset{\boldsymbol{\mu}\in\{\boldsymbol{\mu}_l,\boldsymbol{\mu}_r\}}{\text{argmin}}\ |\boldsymbol{a}^\top\boldsymbol{\mu}-b|$
  end if
  $b_m \leftarrow \boldsymbol{a}^\top\boldsymbol{\mu}_m$
  if $b_m=b$ **then**
    return $\boldsymbol{\mu}_m$
  end if
  if $b_m>b$ **then**
    $u_l\leftarrow u_m$, $\boldsymbol{\mu}_l\leftarrow\boldsymbol{\mu}_m$
  end if
  if $b_m<b$ **then**
    $u_r\leftarrow u_m$, $\boldsymbol{\mu}_r\leftarrow\boldsymbol{\mu}_m$
  end if
end while

the number of oracle calls required until convergence. We also evaluated the number of oracle calls by using subgradient optimization. However, the corresponding number is of the charts resulting in thousands of iterations until convergence.

To summarize, the presented bisection method can be seen as a version of the ES algorithm adjusted to the bisection search. On the other hand, the bisection method performs the same computations as the CPA in case of a single constraint. The latter, however, is more general as it can be applied in the case with multiple constraints, but requires an additional LP solver.

4.2.2 Detaching High-Order Term via Dual Decomposition

For simplicity we consider a special but important case of the Ising grid model. In particular, we first assume that the energy $\boldsymbol{\theta}^\top\boldsymbol{\mu}$ of the corresponding MRF in the problem (4.13) is submodular. The key insight here is that a graph consisting only of a global cardinality potential and unary potentials can be dealt with efficiently. More precisely, we consider the following decomposition

$$\begin{aligned}&\underset{\boldsymbol{\mu},\boldsymbol{\nu}\in\mathcal{X}_G}{\text{minimize}} && \boldsymbol{\theta}^\top\boldsymbol{\mu}+f(A\boldsymbol{\nu})\\ &\text{subject to} && \mu_i(x_i)=\nu_i(x_i)\quad \forall i\in\mathcal{V},\forall x_i\in S\end{aligned}\tag{4.20}$$

That is, the global cardinality potential now belongs to a separate subproblem which can be solved according to the method proposed in [RVM12] in $O(r\cdot n\cdot\log n)$ time where n is the number of nodes and r is the number of rows in A. The corresponding dual problem can then be solved using subgradient methods. If f has nice properties like quasi-convexity, for example, we could consider another formulation to reduce the running time for dealing with the high order potential (HOP)

$$\begin{aligned}&\underset{\boldsymbol{\mu}\in\mathcal{X}_G,\boldsymbol{z}\in\text{dom}(f)}{\text{minimize}} && \boldsymbol{\theta}^\top\boldsymbol{\mu}+f(\boldsymbol{z})\\ &\text{subject to} && A\boldsymbol{\mu}=\boldsymbol{z}\end{aligned}\tag{4.21}$$

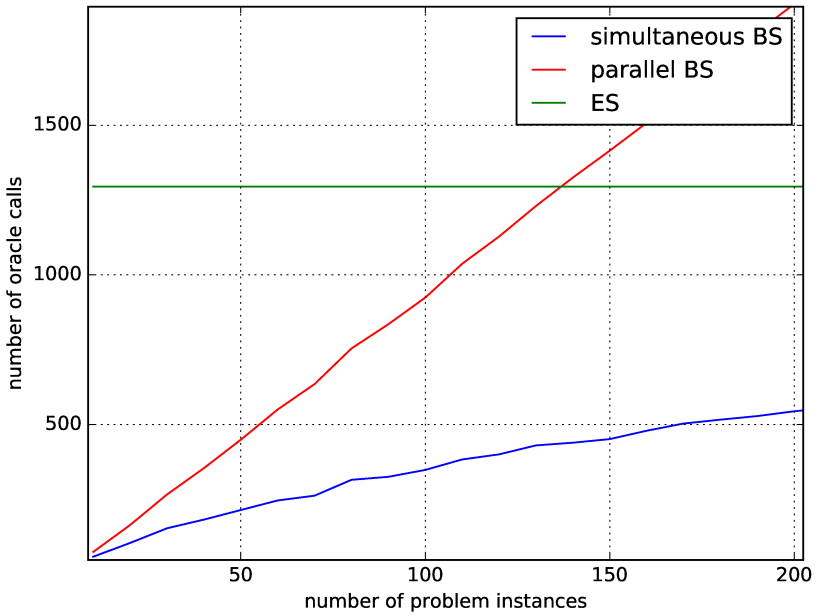


Figure 4.5: Performance comparison of the ES algorithm, simultaneous (equivalent to the CPA) and parallel bisection methods in terms of the total number of oracle calls as a function of the number of instances of problem (4.16) to be solved. We considered a 100×100 Ising grid model with random weights.

In that case we can solve a subproblem corresponding to the HOP in $O(n)$ time by using local search. Additionally, it allows for dealing with problems where the inference runs over a space $\mathcal{Y}$ including different output structures, that is, where the graph of a corresponding MRF is not fixed (see Chapter 5 for an example). Namely, when optimizing the subproblem involving $f(\boldsymbol{z})$ we do not care about the output structure but only about the values of the corresponding loss statistics for f. However, as the next lemma shows the above formulation results in a worse lower bound on the optimal value of the problem (4.13).

Lemma 8. *The equality $g_{4.21}(\boldsymbol{u}) = g_{4.20}(A^\top \boldsymbol{u})$ holds, where $g_{4.21}$ and $g_{4.20}$ denote the dual functions of the problem (4.21) and (4.20), respectively. That is, $d^*_{4.21} \leqslant d^*_{4.20}$, where d^* denotes the optimal value of a corresponding dual problem.*

If the energy is not submodular we again can consider tree decompositions of the Ising grid. To illustrate the following idea we consider a decomposition into two spanning trees (or forests) corresponding to all horizontal and vertical edges, respectively:

$$\begin{aligned} \underset{\boldsymbol{\mu}^1 \in \mathcal{X}_{T_1}, \boldsymbol{\mu}^2 \in \mathcal{X}_{T_2}, \boldsymbol{\mu}^3 \in \mathcal{X}_f, \boldsymbol{\nu}}{\text{minimize}} \quad & {\boldsymbol{\theta}^1}^\top \boldsymbol{\mu}^1 + {\boldsymbol{\theta}^2}^\top \boldsymbol{\mu}^2 + f(A\boldsymbol{\mu}^3) \\ \text{subject to} \quad & \mu^j_i(x_i) = \nu_i(x_i) \quad \forall j \in \{1,2,3\}, \forall i \in \mathcal{V}_j, \forall x_i \in S \end{aligned} \tag{4.22}$$

where we denote by $\mathcal{X}_f$ a set of valid assignments to the corresponding subproblem with the HOP. The above problem is equivalent to the LP relaxation presented in [KP09]. Note that we could also consider a subproblem where the HOP is combined

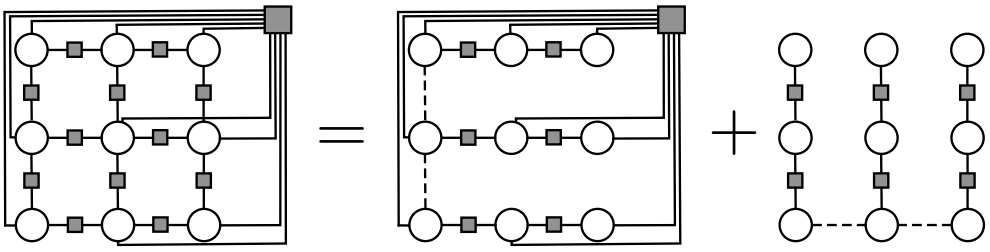


Figure 4.6: Possible decomposition in subproblems. We can deal with the first subproblem by using the message passing algorithm presented in Chapter 3. For the second subproblem we can use standard Viterbi algorithm for inference on trees. The dotted edges mark the missing connections to show how the information flows in a corresponding tree.

with one of the spanning trees (see Figure 4.6) as follows

$$\begin{aligned} \underset{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1}, \boldsymbol{\mu}^2 \in \mathcal{X}_f, \boldsymbol{\nu}}{\text{minimize}} \quad & \boldsymbol{\theta}^{1\top} \boldsymbol{\mu}^1 + (\boldsymbol{\theta}^{2\top} \boldsymbol{\mu}^2 + f(A\boldsymbol{\mu}^2)) \\ \text{subject to} \quad & \mu_i^j(x_i) = \nu_i(x_i) \quad \forall j \in \{1,2\}, \forall i \in \mathcal{V}_j, \forall x_i \in S \end{aligned} \tag{4.23}$$

A corresponding subproblem involving the HOP can be solved using message passing algorithm presented in Chapter 3.

However, the biggest disadvantage here is that a corresponding inference algorithm becomes computationally expensive for larger number of nodes in the MRF.

We note that the presented approach by separating the high order term can also be applied to solve constrained optimization problem as in (4.14) by considering the following formulation

$$\begin{aligned} \underset{\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathcal{X}_G}{\text{minimize}} \quad & \boldsymbol{\theta}^\top \boldsymbol{\mu} + \mathbb{1}_\infty[A\boldsymbol{\nu} = \boldsymbol{b}] \\ \text{subject to} \quad & \mu_i(y_i) = \nu_i(y_i) \end{aligned} \tag{4.24}$$

This yields an approximation approach where the approximation is with respect to the energy while the corresponding solution is always feasible according to the constraint $A\boldsymbol{\mu} = \boldsymbol{b}$. Namely, we can always extract a feasible solution from the subproblem corresponding to the HOP. As mentioned earlier, we can (exactly) solve some instances of the problem (4.14) with respect to the set of parameters $\mathcal{B}$ by means of the Lagrangian relaxation as presented in the previous section. For the remaining parameter values we could use the above approach to find an approximate solution. The following lemma shows that the the above formulation results in a tighter lower bound for the problem (4.14).

Lemma 9. *The equality $g_{4.14}(\boldsymbol{u}) = g_{4.24}(A^\top \boldsymbol{u})$ holds, where $g_{4.14}$ and $g_{4.24}$ denote the dual functions of the problem (4.14) and (4.24), respectively. That is, $d^*_{4.14} \leqslant d^*_{4.24}$, where d^* denotes the optimal value of a corresponding dual problem.*

4.2.3 General Case

We now summarize how to apply Lagrangian relaxation for a more general case related to Problem 1 defined in the previous chapter. Here we drop the first assumption for F and $\boldsymbol{G}$ to have a bounded treewidth. Instead, we only assume that the maximal clique size for both mappings is sufficiently low (otherwise there is no other way as to consider all the possible combinations) but allow the treewidth to be a function of the graph size.

Using the standard overcomplete representation $\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{y})$ for a joint variable assignment $\boldsymbol{y} \in \mathcal{Y}$ we write F and $\boldsymbol{G}$ as linear mappings. To deal with an arbitrary wrapper function $H = H(F, \boldsymbol{G})$ we can use the approach presented in Section 4.2.1 where we set $F(\boldsymbol{y}) := \boldsymbol{\theta}^T \boldsymbol{\mu}(\boldsymbol{y})$ and $\boldsymbol{G}(\boldsymbol{y}) := A\boldsymbol{\mu}(\boldsymbol{y})$ for suitable $\boldsymbol{\theta}$ and A. As already mentioned, the presented approach can find optimal solutions subject to the constraint $\boldsymbol{G}(\boldsymbol{y}) = \boldsymbol{b}$ for all $\boldsymbol{b} \in \mathcal{B}$ as discussed in Section 4.2.1. For the values not in $\mathcal{B}$ we can use the approach presented in Section 4.2.2 (see problem (4.24)). After gathering all the solutions for each value of the statistics represented by $\boldsymbol{G}$ we can choose the best by evaluating the function H on the corresponding values of $F(\boldsymbol{y})$ and $\boldsymbol{G}(\boldsymbol{y})$. In the case where the global term is a cardinality potential we can separate it from the rest of the graph via DD.

4.3 Summary and Discussion

We presented the established frameworks of linear programming (LP) relaxation and dual decomposition (DD), two important MAP solvers for discrete MRFs. It is well known that these two methods have a strong connection both providing an optimal solution to the MAP problem when a corresponding LP relaxation is tight. However, less is known about their relationship in the opposite and more realistic case. Namely, while it is known that both methods have the same optimal objective value, an interesting question is if there are other properties they share. In particular, the connection between the solutions of LP relaxation and the assignments which can be extracted via DD, has not been clarified. For example, even if the solution of the LP relaxation is unique, there might be multiple optimal (but disagreeing) assignments via DD. What is the nature of this ambiguity? Are all these assignments equivalent or can we even extract an additional information about the optimal solutions from analyzing the disagreement behavior? These and other questions were the main motivation for the corresponding analysis presented in the first part of this chapter. Here we successfully provided a few novel findings explaining how the fully integral assignments obtained via DD agree with the optimal fractional assignments via LP relaxation when the latter is not tight. More specifically, we have proved:

- given an optimal (fractional) solution of the LP relaxation, there always exists an optimal variable assignment via DD which agrees with the integral part of the LP solution; this also holds for non binary models with arbitrary higher order potentials
- for binary pairwise MRFs (in a non degenerate case) the first result holds for every optimal assignment which can be extracted via DD
- for binary pairwise MRFs (in a non degenerate case) the unambiguous part among all optimal assignments from different trees in a decomposition coincides with the integral part of the (fractional) optimal assignment via LP relaxation

In particular, for binary pairwise MRFs the integral part of an optimal solution provided via LP relaxation is known to be strongly persistent. Therefore, due to the properties listed above, we can conclude that (for this case) both methods LP relaxation and DD share the partial optimality property of their solutions. Combined with the previous works this implies that the two methods are indeed equivalent.

Practically, it has the following implications. 1) If the goal is to find an accurate MAP assignment we can use LP relaxation first to fix the integral part and then apply an approximation algorithm (e.g. loopy belief propagation) to set the remaining variables in the fractional part – each fractional node has the value 0.5 showing no preference of a corresponding variable to be in a specific state. On the other hand, DD provides fully integral assignments, which agree with the integral part of LP relaxation. 2) If we are only interested in the persistent part, we can extract the corresponding partial assignment from DD by considering an intersection of all the optimal assignments to the individual trees in a decomposition. The unambiguous part then coincides with the strongly persistent part of LP relaxation. In particular, considering a decomposition over individual edges, is more beneficial in that case. Namely, finding all optimal assignments to trees becomes a trivial task.

To summarize, the LP relaxation is a popular method for discrete MAP inference in pairwise graphical models because of its appealing (partial) optimality properties. However, this method does not scale nicely due to the extensive memory requirements restricting its practical use for bigger problems. Here, DD provides an effective alternative via distributed optimization, which scales to problems of arbitrary size — we can always consider a decomposition over the individual edges. Finally, the results presented in this paper suggest, that when using DD instead of the LP relaxation we do not lose any of the nice properties of the latter. Both methods provide exactly the same information (for binary pairwise models) about their solutions.

The main restriction of Problem 1 defined in the previous chapter is the requirement on the treewidth for the MRFs associated with the mappings F and $\boldsymbol{G}$ to be bounded. To extend the range of handleable problems to models with unbounded treewidth we proposed to use the existing technique of Lagrangian relaxation, and more specifically, DD. In particular, the approach of DD has been extensively used for the discrete MAP inference in the previous works [KPT11; SGJ11; RC14; Eve63; Lue73]. Due to the generality of these methods there are various ways how they can be applied to the problem of interest. We analyzed the intricacies of these approximation techniques when applied to the task of (globally) augmented MAP inference and discussed pros and contras of the individual formulations. As a further result we proposed a simple bisection method for the special case of MAP inference subject to a global linear constraint and investigated its relationship to the existing methods including ES and cutting-plane algorithm.

Chapter 5

Applications

In this chapter we show how the exact inference algorithms (Algorithm 2 and 3) presented previously can be applied for training an SSVM on three different application examples including the task of sequence tagging, sequence segmentation, and natural language parsing with context free grammars. An additional difficulty we face in practice is that for some applications the output graphs $\boldsymbol{y}$ vary not only in the configuration of the labels of the individual nodes but also in their structure. Although, the number of possible output graph can be exponential in the size of the input $\boldsymbol{x}$, often the graphs are not completely arbitrary but share a significant amount of factor and variable nodes. In such cases we can adjust the presented idea for a fixed MRF to perform inference over different graphs using dynamic programming.

The main **contributions** in this chapter are the following:

- **For the first time**, we empirically analyze the effect of training an SSVM with various **non decomposable** loss functions (for margin **and** slack scaling) by using **exact** inference during the training procedure.
- We show on the example of base-NP chunking and constituency parsing how the **exact** inference algorithms (Algorithm 2 and 3) can be applied to problems with **varying** graph structure.
- For the task of constituency parsing we show how to perform loss augmented inference with measures like F_1-loss on the **unbinarized** trees **improving** the resulting prediction accuracy.
- We empirically assess the practical usefulness of the generalization bound for structured learning (given in Theorem 3) by evaluating it on the trained models for the presented applications.

Parts of this chapter including Sections 5.2 - 5.4.3 are based on:

> A. Bauer, N. Görnitz, F. Biegler, K.-R. Müller, and M. Kloft. Efficient Algorithms for Exact Inference in Sequence Labeling SVMs. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* © 2014 IEEE. [Bau+14]
>
> A. Bauer, M. Braun, and K.-R. Müller. Accurate Maximum-Margin Training for Parsing with Context-Free Grammars. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* © 2017 IEEE. [BBM17]

- A. Bauer, S. Nakajima, N. Görnitz and K.-R. Müller. Optimizing for Measure of Performance in Max-Margin Parsing. *arXiv preprint, 2017.* [Bau+17a]

First we give an overview of the most popular loss functions and show how their choice affects the resulting running time. After that we present an empirical analysis where for each application we give a compact description of the learning problem along with the task-related details on the corresponding inference algorithm followed by the experimental results at the end of each application section.

5.1 Compact Representation of Loss Functions

In the following we present a bunch of popular dissimilarity measures which our algorithm can handle, summarized in Table 5.1. The measures are given in the compact representation $\Delta(\boldsymbol{y}^*, \boldsymbol{y}) = \eta(\boldsymbol{G}(\boldsymbol{y}))$ based on the corresponding sufficient statistics. Column 3 and 4 show the form of $\boldsymbol{G}(\cdot)$ and η, respectively. Column 5 gives an upper bound R on the number of possible values of auxiliary variables – this is important for the resulting running time (see Theorem 2). Column 6 gives the information whether the corresponding loss function is a high order loss (HOL): "✓" for "yes" and "✗" for "no". In the "no"-case it means that the loss function is decomposable, which benefits the margin scaling formulation as it can be folded into the prediction function making standard inference algorithms accessible for the loss augmented inference. In general, we can handle any function that can be efficiently computed from a contingency table. We now derive an efficient representation of the presented measures with respect to the number R.

Here, $|\boldsymbol{y}| = M$ denotes the number of nodes of the output $\boldsymbol{y}$. TP, FP, and FN are the number of true positives, false positives, and false negatives, respectively. In general, the number of true positives for a prediction $\boldsymbol{y}$ and a true output $\boldsymbol{y}^*$ is given by the number of common nodes with the same label. The number of false positives corresponds to the number of nodes which are present in the output $\boldsymbol{y}$ but missing (or having other label) in the true output $\boldsymbol{y}^*$. Similarly, the number of false negatives is given by the number of nodes present in $\boldsymbol{y}^*$ but missing (or having other label) in $\boldsymbol{y}$. In particular, it holds $|\boldsymbol{y}^*| = TP + FN$.

We see in Table 5.1 that each element of $\boldsymbol{G}(\cdot)$ is a sum of binary variables, which significantly reduces the image of $\boldsymbol{G}(\cdot)$, despite the exponential variety of the output space $\mathcal{Y}$. Indeed, the image size grows only polynomially with the size of the outputs $\boldsymbol{y} \in \mathcal{Y}$. Finally, for the presented dissimilarity measures, the number R provides an upper bound on the image size of $\boldsymbol{G}(\cdot)$.

Zero-One Loss ($\Delta_{0/1}$)
This loss function takes on binary values $\{0, 1\}$ and is the most uninformative among existing performance measures, since it only checks if the prediction matches the ground truth to a $100\ \%$ and gives no partial quantification of the prediction quality in the opposite case. Technically, this measure is not decomposable since (in the most general case) it requires the numbers *FP* and *FN* to be evaluated via

$$\Delta_{0/1}(\boldsymbol{y}^*, \boldsymbol{y}) = \mathbb{1}_1[\max\{FP, FN\} > 0]. \tag{5.1}$$

Sometimes[1] the number *FN* (unlike *FP*) cannot be counted from individual nodes of the prediction. However, we can count the number *TP* instead an then use the relationship $|\boldsymbol{y}^*| = TP + FN$ to evaluate the loss function. We also note that an alternative (and faster) inference approach can be used here by modifying the prediction

[1]For example, if the outputs $\boldsymbol{y} \in \mathcal{Y}$ are sets with no ordering indication of the individual set elements, we need to know the whole set $\boldsymbol{y}$ in order to be able to compute *FN*. Therefore, computing *FN* from a partially constructed output is not possible.

Loss Function	Designation	$\boldsymbol{G}(\boldsymbol{y}; \boldsymbol{y}^*)$	$\eta(\boldsymbol{G})$	R	HOL
Zero-one loss	$\Delta_{0/1}$	$(TP, \mathbb{1}_1[FP > 0])$	$\mathbb{1}_1[\max\{M - G_1, G_2\} > 0]$	$2M$	✓
Hamming distance	Δ_{HD}	$\sum_{t=1}^{M} \mathbb{1}_1[y_t^* \neq y_t]$	G	M	✗
Hamming loss	Δ_{HL}	$\sum_{t=1}^{M} \mathbb{1}_1[y_t^* \neq y_t]$	G/M	M	✗
Weighted Hamming distance	Δ_{WHD}	$\{\#(s_1, s_2)\}_{s_1,s_2 \in \{1,\ldots,N\}}$	$\sum_{s_1,s_2} \text{weight}(s_1, s_2) \cdot G_{s_1,s_2}$	M^{N^2}	✗
False positives number	$\Delta_{\#FP}$	FP	G	M	✗
Recall	Δ_R	TP	$1 - G/\lvert\boldsymbol{y}^*\rvert$	M	✗
Precision	Δ_P	(TP, FP)	$1 - \frac{G_1}{G_1+G_2}$	M^2	✓
F_β-loss	Δ_{F_β}	(TP, FP)	$1 - \frac{(1+\beta^2)\cdot G_1}{\beta^2 \cdot \lvert\boldsymbol{y}^*\rvert + G_1 + G_2}$	M^2	✓
Intersection over union	$\Delta_{\cap/\cup}$	(TP, FP)	$1 - \frac{G_1}{\lvert\boldsymbol{y}^*\rvert + G_2}$	M^2	✓
Label-count loss	Δ_{LC}	$\sum_{t=1}^{M} y_t$	$\left\lvert G - \sum_{t=1}^{M} y_t^* \right\rvert$	M	✓
Crossing brackets number	$\Delta_{\#CB}$	$\#CB$	G	M	✗
Crossing brackets rate	Δ_{CBR}	$(\#CB, \lvert\boldsymbol{y}\rvert)$	G_1/G_2	M^2	✓
BLEU	Δ_{BLEU}	$(TP_1, FP_1, \ldots, TP_K, FP_K)$	$1 - BP(\cdot) \cdot \exp\left(\frac{1}{K}\sum_{k=1}^{K} \log p_k\right)$	M^{2K}	✓
ROUGE-K	$\Delta_{ROUGE\text{-}K}$	$\{\text{count}(k, X)\}_{k \in \text{grams}(Ref)}$	$1 - \frac{\sum_{S \in Ref} \sum_{k \in \text{grams}(S)} \min\{\text{count}(k,S), G_k\}}{\sum_{S \in Ref} \sum_{k \in \text{grams}(S)} \text{count}(k,S)}$	M^D	✗
ROUGE-LCS	$\Delta_{ROUGE\text{-}LCS}$	$\{LCS(X, S)\}_{S \in Ref}$	$1 - \frac{1}{\lvert Ref\rvert} \sum_{S \in Ref} \frac{(1+\beta^2)P(G_S)\cdot R(G_S)}{\beta^2 P(G_S) + R(G_S)}$	$M^{2\lvert Ref\rvert}$	✓

Table 5.1: Compact representation of popular dissimilarity measures based on the corresponding sufficient statistics $\boldsymbol{G}(\cdot)$.

algorithm to compute additionally the second best output – one of these two outputs is an optimal solution to the loss augmented problem.

Hamming Distance/Hamming Loss (Δ_{HD}, Δ_{HL})
Given a true $\boldsymbol{y}^*$ and a predicted sequence $\boldsymbol{y}$ of the same length, *Hamming distance* measures the number of states on which the two sequences disagree:

$$\Delta_{HD}(\boldsymbol{y}^*, \boldsymbol{y}) = \sum_{t=1}^{M} \mathbb{1}_1[\boldsymbol{y}_t^* \neq \boldsymbol{y}_t]. \tag{5.2}$$

By normalizing the Hamming distance we get *Hamming loss* which is independent of the length of the sequences. Both measures are decomposable and are often used in sequence labeling tasks.

Weighted Hamming Distance (Δ_{WHD})
Given a matrix of weights (denoted: weight $\in \mathbb{R}^{N \times N}$), *weighted Hamming distance* corresponds to $\Delta_{WHD}(\boldsymbol{y}^*, \boldsymbol{y}) = \sum_{t=1}^{M} \text{weight}(y_t^*, y_t)$. Keeping track of the accumulated sum of the weights until the current position t in a sequence is (unlike for the normal Hamming distance) in general intractable. However, we can use the following compact representation. Namely, it is sufficient to count the numbers of occurrences $(\boldsymbol{y}_t^*, \boldsymbol{y}_t)$ for each pair of states $\boldsymbol{y}_t^*, \boldsymbol{y}_t \in \{1, ..., N\}$ according to

$$\sum_{t=1}^{M} \text{weight}(y_t^*, y_t) = \sum_{s_1, s_2} \text{weight}(s_1, s_2) \sum_{t=1}^{M} \mathbb{1}_1[y_t^* = s_1 \wedge y_t = s_2]. \tag{5.3}$$

That is, each dimension of $\boldsymbol{G}$ (denoted G_{s_1,s_2}) corresponds to

$$G_{s_1,s_2}(\boldsymbol{y}; \boldsymbol{y}^*) = \sum_{t=1}^{M} \mathbb{1}_1[y_t^* = s_1 \wedge y_t = s_2]. \tag{5.4}$$

The image size of $\boldsymbol{G}(\cdot)$ can be upper bounded by considering an urn problem with N^2 distinguishable urns and M indistinguishable balls. The number of possible distributions of balls over the urns is given by $\binom{M+N^2-1}{M} \leqslant M^{N^2}$.

False Positives Number/Precision/Recall ($\Delta_{\#FP}, \Delta_P, \Delta_R$)
False positives number measures the discrepancy between outputs by counting the number of false positives in a prediction $\boldsymbol{y}$ with respect to the true output $\boldsymbol{y}^*$ and has been used for natural language parsing due to its simplicity. Precision and recall are measures often used in the information retrieval. We can convert them to a loss function by subtracting the corresponding values from one. Unlike for precision given by $TP/(TP + FP)$, recall effectively depends only on one parameter. Although it is originally parameterized by two parameters given as $TP/(TP + FN)$ we can use the fact that the value $|\boldsymbol{y}^*| = TP + FN$ is always known in advance during the inference. This also makes recall a decomposable measure.

F_β-Loss (Δ_{F_β})
$F_{\beta=1}$-score is appropriate for many structured prediction tasks and is often used to evaluate the resulting performance in various natural language processing applications. It is originally defined as the harmonic mean of precision and recall resulting

in

$$F_1 = \frac{2TP}{2TP + FP + FN}. \tag{5.5}$$

However, again due to the fact that the value $|\boldsymbol{y}^*| = TP + FN$ is always known in advance during the inference, F_β-score effectively depends only on two parameters (TP, FP). The corresponding loss function is given by $\Delta_{F_\beta} = 1 - F_\beta$.

Intersection Over Union ($\Delta_{\cap/\cup}$)
Intersection Over Union loss also known as *Jaccard similarity* is mostly used in image processing, in particular, for the task of image segmentation and object recognition and was used as performance measure in the Pascal Visual Object Classes Challenge. It is defined as $1 - area(\boldsymbol{y}^* \cap \boldsymbol{y})/area(\boldsymbol{y}^* \cup \boldsymbol{y})$. This value is easy to interpret in case where the outputs $\boldsymbol{y}^*$, $\boldsymbol{y}$ describe bounding boxes of pixels. The more the overlap of two boxes the smaller the loss value. In terms of contingency table this yields[1]

$$\Delta_{\cap/\cup} = 1 - \frac{TP}{(TP + FP + FN)}. \tag{5.6}$$

Since $|\boldsymbol{y}^*| = TP + FN$, the value $\Delta_{\cap/\cup}$ effectively depends only on two parameters instead of three. It also has nice theoretical property that, unlike F_β**-loss**, $\Delta_{\cap/\cup}$ defines a proper distance metric on sets.

Label-Count Loss (Δ_{LC})
Label-Count loss is a performance measure used in computer vision for the task of binary image segmentation and is defined as

$$\Delta(\boldsymbol{y}^*, \boldsymbol{y}) = \frac{1}{M} \left| \sum_{i=1}^{M} y_i - \sum_{i=1}^{M} y_i^* \right|. \tag{5.7}$$

This loss function prevents segmentation labellings with substantially different area compared to the ground truth to be assigned a low energy under the model.

Crossing Brackets Number/Crossing Brackets Rate ($\Delta_{\#CB}$, Δ_{CBR})
The Number of *Crossing Brackets* ($\#CB$) is a measure used to evaluate the performance in natural language parsing and gives the average of how many constituents in one tree $\boldsymbol{y}$ cross over constituents boundaries in the other tree $\boldsymbol{y}^*$. The normalized version (by $|\boldsymbol{y}|$) of this measure is called *Crossing Brackets (Recall) Rate*. Since the value $|\boldsymbol{y}|$ is not known in advance we need a further parameter for the size of $\boldsymbol{y}$.

Bilingual Evaluation Understudy (Δ_{BLEU})
Bilingual Evaluation Understudy or for short *BLEU* is a measure used to evaluate the quality of machine translations. It computes the geometric mean of the precision $p_k = TP_k/(TP_k + FP_k)$ of k-grams of various lengths (for $k = 1, ..., K$) between a hypothesis and a set of reference translations multiplied by a factor $BP(\cdot)$ that penalizes short sentences according to

$$\Delta_{BLEU}(\boldsymbol{y}^*, \boldsymbol{y}) = 1 - BP(\boldsymbol{y}) \cdot \exp\left(\frac{1}{K} \sum_{k=1}^{K} \log p_k \right). \tag{5.8}$$

[1]Note that in case of the binary image segmentation, for example, we have a different interpretation of true and false positives. In particular, it holds $TP+FN = P$, where P is the number of positive entries in $\boldsymbol{y}^*$.

Note that K is a constant rendering the term M^{2K} a polynomial in M.

Recall Oriented Understudy for Gisting Evaluation ($\Delta_{ROUGE\text{-}K}$, $\Delta_{ROUGE\text{-}LCS}$)
Recall Oriented Understudy for Gisting Evaluation or for short *ROUGE* is a measure used to evaluate the quality of a summary by comparing it to other summaries created by humans. More precisely, given a set of reference summaries Ref and a summary candidate X, ROUGE-K computes the percentage of k-grams from the reference summaries appearing in X according to

$$ROUGE\text{-}K(X, Ref) = \frac{\sum_{S \in Ref} \sum_{k \in \text{k-grams}(S)} \min\{\text{count}(k, X), \text{count}(k, S)\}}{\sum_{S \in Ref} \sum_{k \in \text{k-grams}(S)} \text{count}(k, S)} \tag{5.9}$$

where count(k, S) denotes the number of occurrences of a k-gram k in a summary S. The upper bound R on the image size of $\boldsymbol{G}(\cdot)$ can be estimated similarly to the derivation for the weighted Hamming distance above and is given by M^D where $D := |\text{grams(Ref)}|$ is the dimensionality of $\boldsymbol{G}(\cdot)$, that is, the number of unique k-grams occurring in the reference summaries — note that we do not need to count grams which do not occur in the references.

Another version of ROIUGE (ROUGE-LCS) is based on the concept of the longest common subsequence (LCS). More precisely, given two summaries X and Y, we first compute $LCS(X, Y)$, the length of the LCS, which we use to define some sort of precision and recall given by $LCS(X, Y)/|X|$ and $LCS(X, Y)/|Y|$, respectively. The latter two are used to evaluate a corresponding F-measure, that is,

$$\Delta_{ROUGE\text{-}LCS} = 1 - \frac{1}{|Ref|} \sum_{S \in Ref} \frac{(1 + \beta^2) P_{LCS(X,S)} \cdot R_{LCS(X,S)}}{\beta^2 P_{LCS(X,S)} + R_{LCS(X,S)}}. \tag{5.10}$$

That is, each dimension in $\boldsymbol{G}(\cdot)$ is index by a summary $S \in Ref$. ROUGE-LCS (unlike ROUGE-K) is non decomposable.

5.2 Sequence Tagging

In the general task of *label sequence learning* we aim at predicting a label sequence $\boldsymbol{y} = (y_1, ..., y_M)$ for a given observation sequence $\boldsymbol{x} = (x_1, ..., x_M)$ by learning a functional relationship $h : \mathcal{X} \to \mathcal{Y}$ between an input space $\mathcal{X}$ and an output space $\mathcal{Y}$, based on a training sample of input-output pairs, independently drawn from some fixed but unknown probability distribution over $\mathcal{X} \times \mathcal{Y}$. The practical applications range from different tasks in natural language processing like tagging individual words according to their syntactic or semantic role in a sentence to various applications in computational biology such as gene prediction. As a prominent example we consider here the task of tagging words in a sentence according to their part of speech (see Figure 5.1) which is a fundamental preprocessing step for many applications in the area of natural language processing.

5.2.1 Model Description

Given a training data of annotated sequences, a common approach is to learn a joint discriminant function that assigns each pair $(\boldsymbol{x}, \boldsymbol{y})$ with a real number encoding a degree of compatibility between an input sequence $\boldsymbol{x}$ and an output sequence $\boldsymbol{y}$. Hence, by maximizing this function over all $\boldsymbol{y} \in \mathcal{Y}$, we can determine, for a given

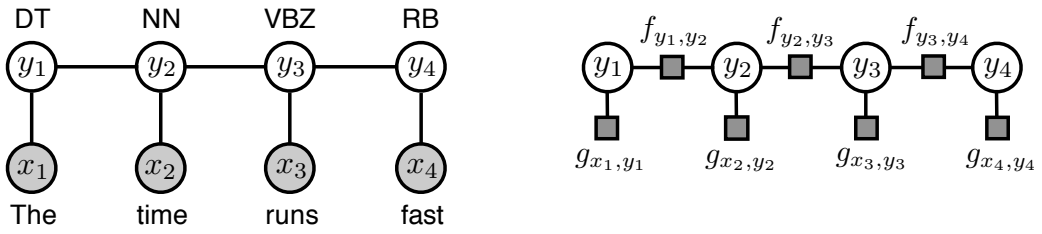


Figure 5.1: Graphical representation of the joint discriminant function $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$ for the task of part of speech tagging with $\boldsymbol{\Psi}$ given in (5.11). The individual words in an example sentence $\boldsymbol{x} =$ (The, time, runs, fast) have the following POS tags $\boldsymbol{y} =$ (DT, NN, VBZ, RB). The graph on the left represents an MRF and the graph on the right a corresponding factor graph. The factors are given by $g_{x_t,y_t} = \hat{w}_{x_t,y_t}$ and $f_{y_{t-1},y_t} = \bar{w}_{y_{t-1},y_t}$ where we decomposed the weight vector of the compatibility $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$ into two parts $\boldsymbol{w} = (\hat{\boldsymbol{w}}^\top, \bar{\boldsymbol{w}}^\top)^\top$ corresponding to the emission scores $\hat{\boldsymbol{w}}$ for the observation-label relation and transition scores $\bar{\boldsymbol{w}}$ for label-label interactions.

input $\boldsymbol{x}$, the most compatible output $h_{\boldsymbol{w}}(\boldsymbol{x}) = \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{\Psi}$ is a vector-valued mapping called joint feature map. One popular choice of $\boldsymbol{\Psi}$ is based on the concept of hidden Markov models [Rab89], which restrict possible dependencies between states in a label sequence by treating it as a Markov chain. This results in a formulation, which can be addressed by dynamic programming enabling an efficient inference. More precisely, we specify a joint feature map $\boldsymbol{\Psi}$ including the observation-label interactions between input features $\boldsymbol{\Phi}(x_t)$ and a state label y_t as well as the label-label interactions between nearby state variables y_{t-1}, y_t in the label sequence (cf. [Tso+05, p. 1475]):

$$\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) = \begin{pmatrix} \sum_{t=1}^{M} \boldsymbol{\Phi}(x_t) \otimes \boldsymbol{\Lambda}(y_t) \\ \sum_{t=2}^{M} \boldsymbol{\Lambda}(y_{t-1}) \otimes \boldsymbol{\Lambda}(y_t) \end{pmatrix} \tag{5.11}$$

where $\boldsymbol{\Lambda}(y_t) = (\delta(y_t, 1), ..., \delta(y_t, N))^\top \in \{0, 1\}^N$ and δ denotes the Kronecker delta function, yielding 1 if the arguments are equal and 0 otherwise. Here, $\otimes$ denotes the direct tensor product, $(a \otimes b)_{i+(j-1)\cdot dim(a)} = a_i \cdot b_j$. Figure 5.1 illustrates the corresponding discriminant function as an energy in an MRF where we use the notation $f_{y_{t-1},y_t} = f(y_{t-1}, y_t)$ to denote a value of a factor f and w_{y_{t-1},y_t} denotes the value of the dimension in $\boldsymbol{w}$ indexed by y_{t-1}, y_t. The inference problem inherent in the prediction function $h_{\boldsymbol{w}}(\boldsymbol{x})$ can be solved via dynamic programming in $O(M \cdot N^2)$ time by using the well known Viterbi's algorithm for hidden Markov models [For73; Rab89].

5.2.2 Algorithm for Loss Augmented Inference

As a reminder, *loss augmented inference* corresponds to maximizing (or minimizing) the joint discriminant function extended by a loss mapping Δ as follows:

$$\widehat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \; \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \odot \Delta(\boldsymbol{y}^*, \boldsymbol{y}), \quad \odot \in \{+, \cdot\} \tag{5.12}$$

where $\boldsymbol{y}^*$ is a fixed (true) output for a given input $\boldsymbol{x}$. In order to train an SSVM with margin or slack rescaling using the cutting-plane algorithm we have to solve an appropriate loss augmented inference problem. For the general case of an arbitrary loss function Δ it seems that we cannot do better than exhaustive search even if the joint feature map decomposes nicely. However, for the special case of the Hamming distance (see (5.2)) which additively decomposes over the sites of the sequences, we can

find an optimal solution for such inference problems in polynomial time. More precisely, the function $\Delta_{\text{HD}}(\boldsymbol{y}^*, \cdot)$ is 0-decomposable and the function $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \cdot)$ with $\boldsymbol{\Psi}$ as in (5.11) is 1-decomposable. That is, there are real-valued functions $f_2, ..., f_M$ with $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{t=2}^{M} f_t(y_{t-1}, y_t)$ such that we can define the following quantity

$$L_{t,k}(j) := \max_{y=(y_1,...,y_{t-1},j)\,:\,\Delta_{\text{HD}}(\boldsymbol{y}^*,\boldsymbol{y})=k} \sum_{i=2}^{t} f_i(y_{i-1}, y_i). \tag{5.13}$$

We interpret $L_{t,k}(j)$ as the optimal value of the compatibility function for subsequences $(x_1, ..., x_t)$, $(y_1, ..., y_t)$ with $y_t = j \in \{1, ..., N\}$ subject to the constraint that the value of a corresponding loss function is fixed to $k \in \{0, ..., t\}$ for $t \in \{2, ..., M\}$. In this special case, quantities $L_{t,k}(j)$ correspond to the messages $\mu_{f_{y_{t-1},y_t} \to y_t}(j, k)$ in Algorithm 2 where k is the value of a corresponding auxiliary variable for the loss statistics. The following holds:

$$\max_{1 \leqslant j \leqslant N,\, 0 \leqslant k \leqslant M} L_{M,k}(j) \odot k = \max_{\boldsymbol{y} \in \mathcal{Y}} \left[\sum_{t=2}^{M} f_t(y_{t-1}, y_t)\right] \odot \Delta_{\text{HD}}(\boldsymbol{y}^*, \boldsymbol{y}). \tag{5.14}$$

That is, if we know the values $L_{M,k}(j)$ for all $1 \leqslant j \leqslant N$ and $0 \leqslant k \leqslant M$, we can determine the optimal value of problem (5.12) by evaluating the left hand side of (5.14). The individual subproblems (or messages) can be computed recursively from smaller subproblems according to the following formula:

$$L_{t,k}(j) = \max_{1 \leqslant i \leqslant N} L_{t-1,k-\mathbb{1}_1[y_t \neq j]}(i) + f_t(i, j) \tag{5.15}$$

The formula in Eq. (5.15) reduces to Eq. (3.9) for the case of chain factor graphs after identifying:

$$L_{t,k}(j) = \mu_{f \to y_t}(j, k)$$

where f is a factor node with $\text{scope}(f_t) = \{y_{t-1}, y_t\}$. Altogether, we can solve the inference problem (5.12) algorithmically as follows. After the initialization part, we compute in a bottom-up manner (i.e. all combinations for t before $t + 1$) the values $L_{t,k}(j)$ using the formula in (5.15), so that we can determine the optimal value p^* according to the equation (5.14). Simultaneously, we store an optimal preceding state of a current state j in a structure S, which we use later to retrieve a corresponding optimal solution. That is, for all $t \in \{2, ..., M\}$, $k \in \{0, ..., t\}$ and $j \in \{1, ..., N\}$ we set $S_{t,k-\mathbb{1}_1[y_t^* \neq j]}(j) := i^*$, where i^* is a maximizing argument in the formula in (5.15). The last state $\widehat{y}_M$ of a corresponding optimal sequence $\widehat{\boldsymbol{y}}$ is given by

$$\widehat{y}_M = \operatorname*{argmax}_{1 \leqslant j \leqslant N} \left\{ \max_{0 \leqslant k \leqslant M} L_{M,k}(j) \right\}$$

and the remaining states can be computed in a reversed order from S. This way we obtain an efficient procedure for solving the optimization problem (5.12) (for Hamming distance), which is summarized in Algorithm 5. We provide a graphical simulation of this algorithm in Appendix C.1.

The running time of Algorithm 5 is dominated by the recursion part (lines 2 - 10). Since evaluating the formula in (5.15) requires $O(N)$ time, the overall time complexity is $O(M^2 \cdot N^2)$ where $M = |\boldsymbol{x}|$ is the sentence length. The correctness follows from the fact that Algorithm 5 is a special case of Algorithm 2. To relate to the running time of the latter, we note that here $R = M$. The following theorem summarizes this main result.

Algorithm 5 (Loss Augmented Sequence Tagging)

Initialization: set $L_{1,k}(j)$ according to the unary scores with respect to x_1 and y_1^* for all $1 \leqslant j \leqslant N$ and $k \in \{0, 1\}$; set all the remaining $L_{t,k}(j)$ to $-\infty$

{Recursion: compute in a bottom-up manner the values $L_{t,k}(j)$ and store the predecessors i of the state j in $S_{t,k}(j)$}

for $t = 2$ **to** M **do**
  for $1 \leqslant j \leqslant N$ **do**
    $\hat{k} \leftarrow \mathbb{1}_1[y_t^* \neq j]$
    for $k = \hat{k}$ **to** $t + \hat{k} - 1$ **do**
      set $L_{t,k}(j)$ using formula in (5.15)
      save a corresponding optimal predecessor of j in $S_{t,k}(j)$
    end for
  end for
end for

{Termination: determine the corresponding optimal value $\widehat{L}$ and the last state $\widehat{y}_M$ of an optimal sequence $\widehat{\boldsymbol{y}}$}

$\widehat{L} \leftarrow \max_{1 \leqslant j \leqslant N, 0 \leqslant k \leqslant M} L_{M,k}(j) \odot k$

$K \leftarrow \operatorname{argmax}_{1 \leqslant k \leqslant M} \left\{ \max_{1 \leqslant j \leqslant N} L_{M,k}(j) \odot k \right\}$

$\widehat{y}_M \leftarrow \operatorname{argmax}_{1 \leqslant j \leqslant N} L_{M,K}(j)$

{Back-Tracking: reconstruct the remaining states $\widehat{y}_t$ from $\widehat{y}_M$, S and K}

for $t = M - 1$ **to** 1 **do**
  $\widehat{y}_t \leftarrow S_{t,K}(\widehat{y}_{t+1})$
  $K \leftarrow K - \mathbb{1}_1[\widehat{y}_t \neq y_t^*]$
end for
return $\widehat{\boldsymbol{y}}$, $\widehat{L}$

Theorem 7 (Correctness and Complexity of Algorithm 5). *Algorithm 5 always finds an optimal solution of a corresponding problem in $O(M^2 \cdot N^2)$ time where $M = |\boldsymbol{x}|$ is the number of words in a sentence and N is the number of possible POS tags.*

It is also worth mentioning that for special case of margin scaling "$\odot = +$", we can use the same inference algorithm for evaluating the prediction function since the loss function decomposes in the same fashion as the feature map.

5.2.3 Experimental Results

In the following we present the experimental results for the task of part of speech tagging as an example of sequence tagging applications. Here, we used the Penn English Treebank-3 [Mar+94] as a benchmark data set, which provides a large corpus of annotated sentences from the Wall Street Journal. More specifically, we extracted the part of speech tags from the parse tree annotation according to the following standard data split: sections 0–18 for training (38219 sentences), sections 19–21 for validation (5527 sentences), and sections 22–24 for testing (5462 sentences).

In the experiments, an SSVM was trained via the cutting plane algorithm optimizing either for the zero-one loss function $\Delta_{0/1}$ or for the Hamming distance Δ_{HD}.

Table 5.2: Part of speech tagging experiment.

METHOD	TOACC$_{\text{VLD}}$	SEACC$_{\text{VLD}}$	TOACC$_{\text{TST}}$	SEACC$_{\text{TST}}$
SP	95.41	39.80	95.50	39.38
ASP	96.29	46.75	96.40	47.27
SVM$_{0/1}$	96.52	48.59	96.55	48.49
MS$_{\text{HD}}$	96.54	48.63	96.62	49.17
SS$_{\text{HD}}$	**96.60**	**49.06**	**96.73**	**50.20**

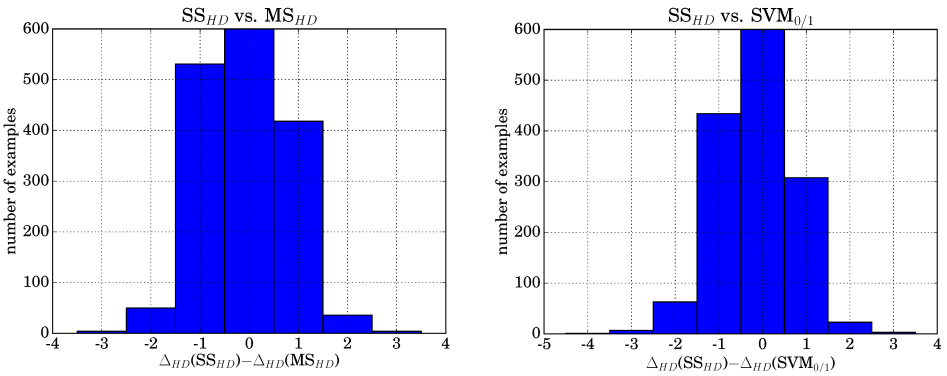


Figure 5.2: This figure illustrates the loss difference on the test set between SS$_{\text{HD}}$ and MS$_{\text{HD}}$ (on the left) and between SS$_{\text{HD}}$ and SVM$_{0/1}$ (on the right). The Wilcoxon signed-rank test confirms that this visually recognized difference is statistically significant. Namely, a corresponding null-hypothesis is that the measurement difference for a pair of methods follows a symmetric distribution around zero. Here, we can see that in both cases there is a clear shift of the corresponding distribution to the left favoring SS$_{\text{HD}}$ upon the other two methods.

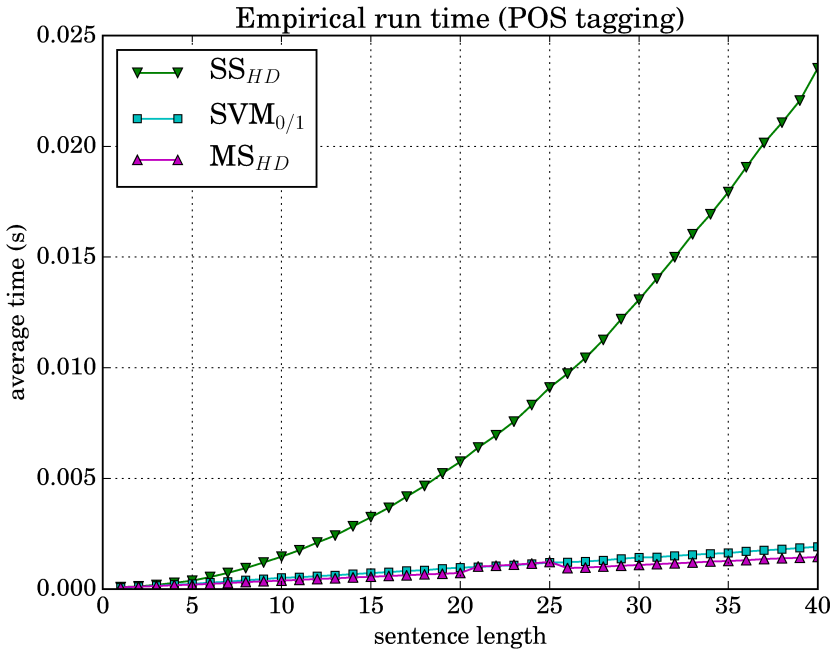


Figure 5.3: Empirical run time of Algorithm 5 as a function of the number of words in an input sentence.

The hyperparameter C was chosen by a cross-validation over a grid of values $\{10^i : i = 0, 1, ..., 5\}$.

Table 5.3: Evaluation of Generalization Bound in Theorem 3. The corresponding scores are given as accuracy percentage with respect to the Hamming loss averaged over the individual sentences in the corpus.

METHOD	$\text{ACC}_{\text{TRAIN}}$	ACC_{TEST}	BOUND
$\text{SVM}_{0/1}$	98.44	96.34	17.77
MS_{HD}	98.04	96.34	**39.53**
SS_{HD}	**98.91**	**96.56**	5.46

The resulting performance is evaluated in terms of accuracy percentage per token (ToAcc) and per sentence (SeAcc). The corresponding results on the validation set (second column) and on the test set (third column) are reported in Table 5.2 for margin scaling (MS_{HD}), slack scaling (SS_{HD}), and for training an SSVM with zero-one loss ($\text{SVM}_{0/1}$). Note that in the case of slack scaling we use the proposed exact inference algorithm (Algorithm 5) during the training procedure. Additionally, we report the results for structured perceptron (SP) and its averaged version (ASP) [Col02]. Here, slack scaling (marked in bold faces) slightly outperforms the other methods. Figure 5.2 visualizes in a histogram the loss difference of predictions on the test set for two methods SS_{HD} and MS_{HD} (on the left) and for SS_{HD} and $\text{SVM}_{0/1}$ (on the right). For the sake of better illustration we cut off the bar corresponding to the value zero at the mark 600 — the actual number of examples with zero difference is about several thousands. Note that in both cases the mode of the corresponding distribution is shifted to the left of zero. The Wilcoxon signed-rank test [RN11] confirms that this visually recognized difference is statistically significant.

An empirical run time evaluation of a corresponding inference algorithm (Algorithm 5) for training via slack scaling formulation (SS_{HD}) is given in Figure 5.3.
Note that for margin scaling (MS_{HD}) we used the same inference algorithm as for evaluating the prediction function by folding the loss and the compatibility function together. For $\text{SVM}_{0/1}$ we used the K-best Viterbi's algorithm [Wu+08] to compute the second-best solution.

Additionally, we evaluated the generalization bound in Theorem 3 for a few trained models. The results are given in Table 5.3. As we can see the corresponding numbers can significantly deviate from the actual performance on a specific data set reflecting the pessimistic assumptions of this generalization bound.

5.3 Sequence Segmentation

Unlike in a sequence tagging task, in sequence segmentation the goal is to divide a given input sequence into segments. The main difference here is that we can extract global features from whole segments rather than from single sites in a sequence. Typical applications matching this task originate from area of computational biology and natural language processing including gene finding, base-NP chunking, and named entity recognition.

5.3.1 Model Description

We use the notation $(j, s, e) \in \boldsymbol{y}$ to describe a segment in an output $\boldsymbol{y}$ for an input sequence $\boldsymbol{x} = (x_1, ..., x_{|\boldsymbol{x}|})$ where j denotes the label of the segment, s and e the start and the end position, respectively. A natural way to represent a segmented

input sequence $(\boldsymbol{x}, \boldsymbol{y})$ is by using a joint feature map counting different features $\boldsymbol{\Phi} = (\boldsymbol{\Phi}_c^\top, \boldsymbol{\Phi}_b^\top)^\top$ extracted from the content of the individual segments $\boldsymbol{\Phi}_c$ and from the borders between segments $\boldsymbol{\Phi}_b$. For example, for $\boldsymbol{\Phi}_c$ we could extract n-grams from the tokens $(x_s, ..., x_e)$ paired with the corresponding segment label. For $\boldsymbol{\Phi}_b$ we could do the same by considering a window of tokens around the border positions s and e to model the transition signals between different segments. That is,

$$\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) = \begin{pmatrix} \sum_{i=1}^{|\boldsymbol{y}|} \boldsymbol{\Phi}(\boldsymbol{x}, s_i, e_i) \otimes \boldsymbol{\Lambda}(j_i) \\ \sum_{i=2}^{|\boldsymbol{y}|} \boldsymbol{\Lambda}(j_{i-1}) \otimes \boldsymbol{\Lambda}(j_i) \end{pmatrix}. \tag{5.16}$$

We index the individual variable nodes in the outputs by the span boundaries of tokens. For example, $y_{s,e}$ means a segment $(y_{s,e}, s, e)$. For this formulation with only pairwise dependencies between neighboring segments the compatibility function $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$ is (for each restriction to a fixed graph structure defining the boundaries of segments) 1-decomposable. The parameters s and e in a segment (j, s, e) define the structure of the output while j conveys the label information.

5.3.2 Algorithm for Loss Augmented Inference

Our goal here is to derive an algorithm for solving the problem in (5.12) for sequence segmentation task. Although we also could optimize for Hamming distance Δ_{HD} as in the sequence tagging example, a more appropriate loss function here is F_β-loss ($\Delta_{F_\beta} = 1 - F_\beta$) on segments where

$$F_\beta = \frac{(1 + \beta^2) \cdot \text{TP}}{\beta^2 \cdot M^* + \text{TP} + \text{FP}}. \tag{5.17}$$

Here TP and FP denote the number of true and false positives with respect to the correct label $\boldsymbol{y}^*$ and $M^* = |\boldsymbol{y}^*|$ is the number of segments in the correct output. Unlike with Δ_{HD}, optimizing for Δ_{F_β} avoids giving credit to partially correct segment predictions.

The main idea also follows Algorithm 2, but needs some modification due to varying graph structure according to different segmentation boundaries. By introducing auxiliary variables for the loss parameters $(tp, fp) = \boldsymbol{G}(\boldsymbol{y}; \boldsymbol{y}^*)$ representing the numbers of true and false positives (with respect to $\boldsymbol{y}^*$), we can derive an algorithm which passes messages towards the root node over different graph structures according to varying segmentation boundaries. More precisely, sending a message corresponds to solving a subproblem

$$V_{t,j}^{tp,fp} := \max_{\boldsymbol{y}=\{(j_k,s_k,e_k):k=1,...,|\boldsymbol{y}|\}:\ \boldsymbol{G}(y)=(tp,fp)} \sum_{C \in \text{Cliques}(\boldsymbol{y})} f_C(\boldsymbol{y}_C) \tag{5.18}$$

where $f_C(\boldsymbol{y}_C)$ denote the individual factors of the compatibility function $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$. That is, the quantity $V_{t,j}^{tp,fp}$ corresponds to the value of an optimal configuration over all valid segmentations of a subsequence $(x_1, ..., x_t)$ with the label j for the last segment and a fixed loss value determined by the constraint $\boldsymbol{G}(\boldsymbol{y}) = (tp, fp)$.

After the initialization phase (lines 1–5 in Algorithm 6), the messages can be computed (lines 6–11) according to

$$V_{t,j}^{tp,fp} = \max_{1 \leqslant i \leqslant N; 1 \leqslant d \leqslant t-1} V_{t-d,i}^{\bar{tp},\bar{fp}} + f(i, j, t-d+1, t) \tag{5.19}$$

Algorithm 6 Loss Augmented Segmentation (with F_β-loss)

1: Set $V_{t,j}^{tp,fp} = -\infty$ for all t, j, tp, fp
2: **for** $1 \leqslant j \leqslant N$, $1 \leqslant e \leqslant |\boldsymbol{x}|$ **do**
3: $tp \leftarrow \mathbb{1}_1[(j,1,e) \in \boldsymbol{y}^*]$, $fp \leftarrow \mathbb{1}_1[(j,1,e) \notin \boldsymbol{y}^*]$
4: $V_{e,j}^{tp,fp} \leftarrow f(\cdot, j, 1, e)$
5: **end for**
6: **for** $t = 2$ **to** $|\boldsymbol{x}|$ **do**
7: **for** $1 \leqslant j \leqslant N$ and (tp, fp) in a feasible range **do**
8: set $V_{t,j}^{tp,fp}$ using Eq. (5.19)
9: save the maximizing arguments: $S_{t,j}^{tp,fp} := [i^*, d^*]$
10: **end for**
11: **end for**
12: $(tp^*, fp^*, j^*) \leftarrow \underset{tp,fp,j}{\arg\max}\ V_{|\boldsymbol{x}|,j}^{tp,fp} \odot \eta(tp, fp)$
13: starting at $S_{|\boldsymbol{x}|,j^*}^{tp^*,fp^*}$ recursively reconstruct an optimal segmentation according to the formula in Eq. (5.19)

where $\bar{tp} := tp - \mathbb{1}_1[(j, t-d+1, t) \in \boldsymbol{y}]$ and $\bar{fp} := fp - \mathbb{1}_1[(j, t-d+1, t) \notin \boldsymbol{y}]$. Note that the corresponding message passing is conducted over different segmentation boundaries by varying the start position $t - d + 1$ of the last segment ending at the position t. For a fixed d the formula in Eq. (5.19) reduces to Eq. (3.9) for the case of chain factor graphs after identifying:

$$V_{t,j}^{tp,fp} = \mu_{f \to y_{(t-d+1),t}}(j, (tp, fp)^\top)$$

where f is a factor node associated with the variable $y_{(t-d+1),t}$ and its neighbor variable to the left in the chain. The optimal value p^* of a corresponding instance of problem (5.12) can be computed from these quantities by maximizing over label $1 \leqslant j \leqslant N$ of the last segment and over the values of true and false positives $1 \leqslant tp, fp \leqslant |\boldsymbol{x}|$ according to

$$p^* := \max_{tp,fp,j} V_{|\boldsymbol{x}|,j}^{tp,fp} \odot \eta(tp, fp) \tag{5.20}$$

The whole procedure is summarized in Algorithm 6 supported by the following theorem.

Theorem 8. *Algorithm 6 always finds an optimal solution of a corresponding problem in $O(|\boldsymbol{x}|^4 \cdot N^2)$ time where $|\boldsymbol{x}|$ is the length of an input sequence $\boldsymbol{x}$ and N is the maximal number of possible labels for a segment.*

5.3.3 Experimental Results

We now consider the experimental results for the task of base-NP chunking as an example of sequence segmentation applications. Here, we used the Penn English Treebank-3 [Mar+94] as a benchmark data set, which provides a large corpus of annotated sentences from the Wall Street Journal. More precisely, we extracted the data from the parse trees basically selecting all noun phrases that contained no other nested noun phrases according to the following data split: sections 0–18 for training (15937 sentences), sections 22–24 for validation (2298 sentences), and sections 19–21

for testing (2335 sentences). Furthermore, we restrict ourselves to sentences only with the length $\leqslant 20$.

We used n-grams ($n \leqslant 3$) features of POS tags extracted (by using a pre-trained tagger) from each segment and a fixed-size window ($= 5$) around segment borders. Our target measure is F_1-score on segments. An SSVM was trained via the cutting plane algorithm. The hyperparameter C was chosen by a cross-validation over a grid of values $\{10^i : i = 0, 1, ..., 5\}$. The corresponding results for margin (MS) and slack scaling (SS), for various dissimilarity measures (see Table 5.1 for details) on the validation and test set are reported in Table 5.4 and Table 5.5, respectively. Here, we use a subscript (e.g. *HL* in SS_{HL}) to denote a dissimilarity measure optimized in training (see Table 5.1 for details). The resulting performance is evaluated in terms of precision (P), recall (R), F_1-score, intersection over union ($\cap/\cup$), Hamming accuracy (Acc HL), and zero-one prediction accuracy (0/1 Acc). Note that for MS_{F_1}, $\text{MS}_{\cap/\cup}$ and all methods based on SS we used the proposed exact inference algorithm (Algorithm 6) during the training procedure. Additionally, we report the results for

Table 5.4: Base-NP Chunking (validation set).

Method	P	R	F_1	$\cap/\cup$	HL Acc	0/1 Acc
SP	88.60	89.02	88.24	84.08	95.08	68.18
ASP	90.55	90.16	90.03	87.08	96.16	75.63
$\text{SVM}_{0/1}$	91.52	91.21	91.05	88.12	96.40	77.02
$\text{MS}_{\#FP}$	**93.10**	86.53	88.46	84.44	94.07	66.79
MS_{HL}	91.17	91.03	90.76	87.65	96.31	75.76
MS_{F_1}	91.57	91.22	91.03	87.99	96.23	76.06
$\text{MS}_{\cap/\cup}$	91.70	91.26	91.14	88.15	96.33	76.41
$\text{SS}_{\#FP}$	92.78	86.57	88.45	84.32	94.30	66.49
SS_{HL}	91.67	91.34	91.21	88.18	96.54	76.67
SS_{F_1}	92.16	**91.76**	**91.65**	**88.66**	**96.60**	**77.06**
$\text{SS}_{\cap/\cup}$	91.71	91.35	91.23	88.21	96.44	76.54

Table 5.5: © 2017 IEEE. Base-NP Chunking (test set).

Method	P	R	F_1	$\cap/\cup$	HL Acc	0/1 Acc
SP	89.94	90.40	89.72	85.92	95.73	71.52
ASP	92.52	92.47	92.24	89.55	96.88	79.05
$\text{SVM}_{0/1}$	92.34	92.52	92.19	89.53	96.84	79.35
$\text{MS}_{\#FP}$	**93.84**	88.80	90.42	86.86	95.08	71.04
MS_{HL}	92.55	92.46	92.24	89.49	96.86	78.88
MS_{F_1}	92.67	92.61	92.37	89.77	96.81	79.65
$\text{MS}_{\cap/\cup}$	92.76	92.69	92.46	89.90	96.83	79.91
$\text{SS}_{\#FP}$	93.69	89.29	90.76	87.18	95.43	71.69
SS_{HL}	92.60	92.65	92.38	89.72	96.97	79.57
SS_{F_1}	93.19	93.15	92.93	90.39	97.12	80.68
$\text{SS}_{\cap/\cup}$	93.31	**93.22**	**93.02**	**90.50**	**97.15**	**80.81**

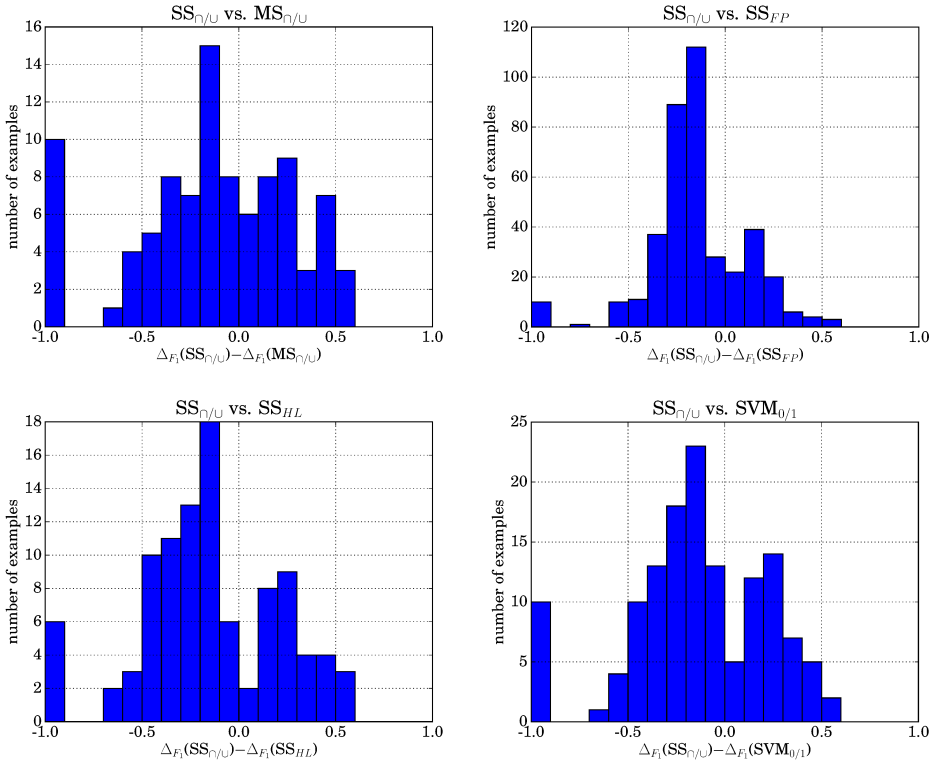


Figure 5.4: This figure illustrates the loss difference on the test set between a few chosen methods. The Wilcoxon signed-rank test confirms that this visually recognized difference is statistically significant. Namely, a corresponding null-hypothesis is that the measurement difference for a pair of methods follows a symmetric distribution around zero. Here, we can see that in each case there is a shift of the corresponding distribution to the left. Note that we do not count examples with zero difference in loss.

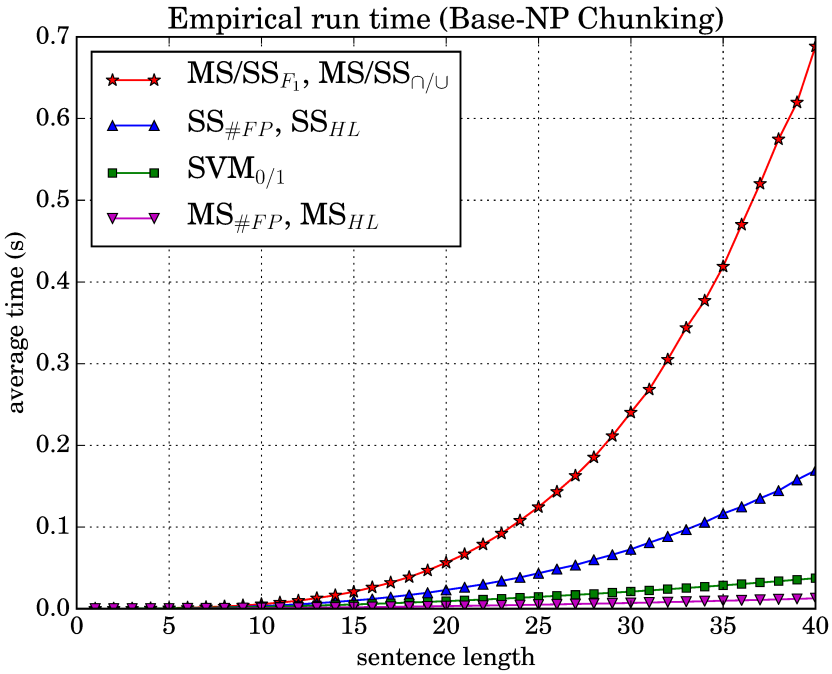


Figure 5.5: © 2017 IEEE. Empirical comparison of run times for the loss augmented inference (Algorithm 6) with various loss functions on a single input instance as a function of the sentence length.

Table 5.6: Evaluation of Generalization Bound in Theorem 3. The corresponding scores are given as accuracy percentage with respect to $\Delta_{\cap/\cup}$.

Method	$\text{Acc}_{\text{train}}$	Acc_{test}	Bound
$\text{SVM}_{0/1}$	91.32	89.53	46.10
$\text{MS}_{\#FP}$	88.99	86.86	20.11
MS_{F_1}	92.40	89.77	28.20
$\text{MS}_{\cap/\cup}$	92.12	89.90	29.95
$\text{SS}_{\#FP}$	90.72	87.18	–
SS_{F_1}	92.76	90.39	31.52
$\text{SS}_{\cap/\cup}$	93.08	90.50	26.48

structured perceptron (SP) and its averaged version (ASP) [Col02]. We see that adopting the training for optimizing high order loss functions corresponding to measures like F_1 or $\cap/\cup$ improves the resulting performance upon other loss functions. Also using SS consistently (except $\#FP$) results in a further performance boost upon MS. The Wilcoxon signed-rank test [RN11] confirms this observation. Figure 5.4 visualizes in a histogram the loss difference of predictions on the test set for a few chosen methods.

Furthermore, we report the averaged running times for the inference on a single input sentence as a function of the sentence length in Figure 5.5. Here, for all models based on SS and for $\text{MS}_{\cap/\cup}$ and MS_{F_1} we used a version of the proposed Algorithm 6 according to the chosen loss function. Note, that for MS with a decomposable dissimilarity measure we used the same inference algorithm as for evaluating the prediction function by folding the loss and the compatibility function together. For $\text{SVM}_{0/1}$ we used a version of the K-best Viterbi's algorithm [Wu+08] to compute the second-best solution.

Additionally, we evaluated the generalization bound in Theorem 3 for a few trained models. The results are given in Table 5.6. As we can see the corresponding numbers can significantly deviate from the actual performance on a specific data set reflecting the pessimistic assumptions of this generalization bound.

5.4 Constituency Parsing

The goal of natural language parsing is to analyze the syntactic structure of sentences according to some grammar formalism. In linguistics it is usually performed as a method to understand the exact meaning of a sentence by identifying the grammatical components, e.g. such as subject and predicate in English. More generally, it aims to analyze how the words in a sentence are grouped together building self-contained units like phrases and how these units are related to each other. This is usually done by placing tree structures over sentences, which due to recursive nature of languages are able to capture many relevant syntactic aspects. The work in formal syntax goes back to Noam Chomsky in 1950s [Cho55; Cho57]. Since then there has been an enormous amount of research in linguistics on describing the syntactic structure of different languages. Nowadays there is a broad range of existing formalisms for this task like lexical functional grammar (LFG) [KB82], head-driven phrase structure grammar (HPSG) [PS94], tree adjoining grammar (TAG) [Jos85; JLT75] and combinatory categorial grammar (CCG) [WJ88], just to mention a few. Here, we focus

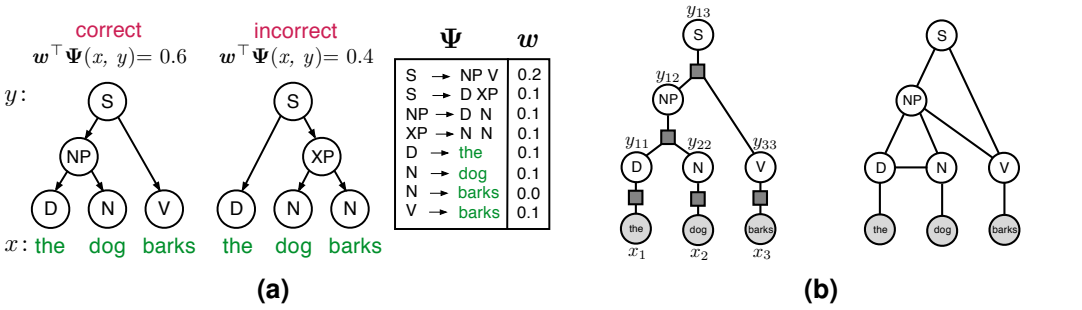


Figure 5.6: © 2017 IEEE. **(a)** Compatibility $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$ in natural language parsing for an input sentence $\boldsymbol{x} = (the, dog, barks)$ and two different parse trees. The weight vector $\boldsymbol{w}$ specifies how appropriate are the individual production rules from a corresponding grammar. The individual dimensions of $\boldsymbol{\Psi}$ are indexed by production rules and the corresponding entries show the number of occurrences of each rule in the tree. **(b)** Factor graph representation (on the left) and an MRF representation (on the right) for the leftmost parse tree in (a).

on the concept of the context-free grammar (CFG), which is very fundamental and builds, in some sense, the basis for all these modern formalisms.

A reliable and robust parser for natural languages is a very useful tool and there are many applications which could benefit from the ability to recover the underlying syntactic structure of a sentence. However, parsing is also a very difficult problem. Apart from the fact that natural languages are not completely context-free [Cho57], there is a high degree of ambiguity in their syntactic structure and the corresponding parsing methods, therefore, should be able to handle these difficulties. In the field of natural language processing, parsing is still one of the key challenges and its performance accuracy affects many other tasks like information extraction, machine translation, question answering and the task of natural language understanding. The most widely used machine learning approach to syntactic parsing is to treat it as a supervised learning problem, where we are given a labeled data consisting of sentences paired with the corresponding parse trees describing the syntactic phrasal structure of these sentences and the goal is to learn the underlying functional relationship by inspecting the training data. Each parse tree represents here a derivation of a corresponding sentence by iteratively applying production rules according to a given grammar. In order to deal with the ambiguity when and which production rules are applied to produce a given sentence, each one of them is assigned with a weight describing how likely it is for that rule to contribute to the parse tree. The simplest probabilistic model for this scenario is the probabilistic context free grammar (PCFG) [MS99; Joh98; Hee93; Cha97], which is just a CFG with probabilities added to each production rule indicating how likely different productions are. A corresponding discriminative approach is based on weighted context free grammars (WCFGs) [Tso+05; Tas+04], where we drop the requirement on the corresponding weights to be valid probabilities.

5.4.1 Model Description

Formally, our goal is to predict a parse tree $\widehat{\boldsymbol{y}}$ describing the phrase structure for a given sentence $\boldsymbol{x}$ by solving $\widehat{\boldsymbol{y}} = \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$. Each non-terminal node (together with its children) in a parse tree represents an application of a grammar rule. Here, joint feature mapping $\boldsymbol{\Psi}$ counts the number of occurrences for each production rule in the tree (see Figure 5.6a).

In order to achieve a cubic running time for prediction it is a conventional approach to binarize the grammar before training. Here, we can restrict our attention to the context-free grammars in the *Chomsky Normal Form* (CNF). That is, we consider only two types of productions $A \to \sigma$ and $A \to B\ C$, where $A, B, C \in \mathcal{N} = \{\rho_1, ..., \rho_{|\mathcal{N}|}\}$ are nonterminals and $\sigma \in \Sigma = \{\sigma_1, ..., \sigma_{|\Sigma|}\}$ are terminals. $\mathcal{S}$ is the start symbol for each derivation. A parse tree, thus, can be compactly represented in a vector form by using direct tensor product $(\boldsymbol{a} \otimes \boldsymbol{b})_{i+(j-1)\cdot dim(\boldsymbol{a})} = \boldsymbol{a}_i \cdot \boldsymbol{b}_j$ with a suitable binary vector representation for terminal and non-terminal symbols $\boldsymbol{\Phi} : \Sigma \to \{0,1\}^{|\Sigma|}$, $\sigma \mapsto (\delta(\sigma, \sigma_1), ..., \delta(\sigma, \sigma_{|\Sigma|}))^\top$ and $\boldsymbol{\Lambda} : \mathcal{N} \to \{0,1\}^{|\mathcal{N}|}$, $\rho \mapsto (\delta(\rho, \rho_1), ..., \delta(\rho, \rho_{|\mathcal{N}|}))^\top$ where δ is the generalized Kronecker delta function yielding 1 if the arguments are equal and 0 otherwise. The individual dimensions of $\boldsymbol{\Psi}$ are indexed by grammar productions, e.g., $\boldsymbol{\Psi}_{A\to\sigma}$ is given by $\boldsymbol{\Lambda}(A) \otimes \boldsymbol{\Phi}(\sigma)$ and $\boldsymbol{\Psi}_{A\to BC}$ by $\boldsymbol{\Lambda}(A) \otimes \boldsymbol{\Lambda}(B) \otimes \boldsymbol{\Lambda}(C)$. That is, for each rule in the grammar there is a feature encoding its presence (and frequency) in a parse tree $\boldsymbol{y}$:

$$\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) = \begin{pmatrix} \sum_{r \leqslant t < s} \boldsymbol{\Lambda}(y_{rs}) \otimes \boldsymbol{\Lambda}(y_{rt}) \otimes \boldsymbol{\Lambda}(y_{t+1\,s}) \\ \sum_{t=1}^{|\boldsymbol{x}|} \boldsymbol{\Lambda}(y_{tt}) \otimes \boldsymbol{\Phi}(x_t) \end{pmatrix}. \tag{5.21}$$

Here, a variable y_{rs} denotes the label of a tree node which spans the tokens $x_r, ..., x_s$. It is convenient to index the individual variable nodes by the corresponding span boundaries of tokens since it uniquely determines the position of a corresponding node in a tree (see Figure 5.6b). Since each production rule involves at most three variable nodes and the factor graphs of the outputs have a tree structure, the compatibility function $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$ (restricted to a fixed graph structure) is 2-decomposable. The dissimilarity measure we use here is the F_β-loss for which $\boldsymbol{G}$ in the compact representation $\eta(\boldsymbol{G}) = 1 - F_\beta(\boldsymbol{G})$ is $\boldsymbol{G}$ is 0-decomposable.

5.4.2 Algorithm for Loss Augmented Inference

Our goal here is to derive an algorithm for solving the problem in (5.12). The main idea follows Algorithm 2 extended to varying graph structure over binary trees. By introducing auxiliary variables for the loss parameters $(tp, fp) = \boldsymbol{G}(\boldsymbol{y}; \boldsymbol{y}^*)$, we can derive an algorithm, which passes messages towards the root node over different tree structures. More precisely, sending a message corresponds to solving a subproblem

$$\Pi_{i,j,A}^{tp,fp} := \max_{\boldsymbol{y} \in \mathcal{T}_{i,j,A} :\ \boldsymbol{G}(\boldsymbol{y}) = (tp, fp)} \sum_{C \in \text{Cliques}(\boldsymbol{y})} f_C(\boldsymbol{y}_C), \tag{5.22}$$

where $\mathcal{T}_{i,j,A}$ denotes the set of all valid subtrees spanning the tokens $(x_i, ..., x_j)$ and having the label A at the root. Cliques$(\boldsymbol{y})$ denotes the set of maximal cliques in an MRF associated with an output $\boldsymbol{y}$. $f_C(\boldsymbol{y}_C)$ are the individual factors of the compatibility function $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$. The parameters tp, fp encode the number of true and false positives, respectively. That is, the quantity $\Pi_{i,j,A}^{tp,fp}$ denotes the value of an optimal label configuration over the parse trees $\boldsymbol{y} \in \mathcal{T}_{i,j,A}$ which additionally results in a fixed value for true and false positives satisfying $\boldsymbol{G}(\boldsymbol{y}) = (tp, fp)$.

After the initialization phase (lines 1–5 in Algorithm 7), the message computation (lines 6–14) can be done according to the formula

$$\Pi_{i,j,A}^{tp,\,fp} = \max_{A \to BC,\ s,\ \hat{tp}, \hat{fp}} f(A \to BC) + \Pi_{i,s,B}^{\bar{tp},\,\bar{fp}} + \Pi_{s+1,j,C}^{\hat{tp},\hat{fp}} \tag{5.23}$$

where we maximize over all possible rules $A \to B\ C$ with fixed A, over all split points of subtrees $i \leqslant s < j$, and over possible distributions of the loss parameters

Algorithm 7 Loss Augmented Parsing (with F_β-loss)

Set $\Pi_{i,j,A}^{tp,fp} = -\infty$ for all i, j, A, tp, fp
for $1 \leqslant i \leqslant |\boldsymbol{x}|$, $A \in \mathcal{N}$ **do**
  $tp \leftarrow \mathbb{1}_1[A \rightarrow x_i \in \boldsymbol{y}^*]$, $fp \leftarrow \mathbb{1}_1[A \rightarrow x_i \notin \boldsymbol{y}^*]$
  $\Pi_{i,i,A}^{tp,fp} \leftarrow f(A \rightarrow x_i)$
end for
for $l = 1$ **to** $|\boldsymbol{x}| - 1$ **do**
  for $i = 1$ **to** $|\boldsymbol{x}| - l$ **do**
    $j \leftarrow i + l$
    for $A \in \mathcal{N}$ and (tp, fp) in a feasible range **do**
      set $\Pi_{i,j,A}^{tp,fp}$ using Eq. (5.23)
      save the best values: $S_{i,j,A}^{tp,fp} := [B, C, s, \hat{tp}, \hat{fp}]$
    end for
  end for
end for
$(tp^*, fp^*) \leftarrow \arg\max_{tp,fp} \Pi_{1,|\boldsymbol{x}|,\mathcal{S}}^{tp,fp} \odot \eta(tp, fp)$
starting at $S_{1,|\boldsymbol{x}|,\mathcal{S}}^{tp^*,fp^*}$ recursively reconstruct an optimal output tree according to the formula in Eq. (5.23)

$\hat{tp}$ and $\hat{fp}$ with $\bar{tp} := tp - \mathbb{1}_1([A]_{ij} \in \boldsymbol{y}^*) - \hat{tp}$ and $\bar{fp} := fp - \mathbb{1}_1([A]_{ij} \notin \boldsymbol{y}^*) - \hat{fp}$. The term $[A]_{ij}$ denotes a variable node that spans tokens $x_i, ..., x_j$ and has the label A. With a slight abuse of notation we write $[A]_{ij} \in \boldsymbol{y}^*$ to check if a node with the corresponding label is in a tree $\boldsymbol{y}^*$. f is the factor function for the clique associated with a current production rule.

Note that the corresponding message passing is conducted over different tree structures by varying the split point s. This parameter determines the boundary between spans of tokens for the children of a current node $\boldsymbol{y}_{ij}$. For a fixed s the formula in Eq. (5.23) reduces to Eq. (3.9) after identifying:

$$\Pi_{i,j,A}^{tp,\,fp} = \mu_{f \rightarrow y_{ij}}(A, (tp, fp)^\top)$$

where f is a factor node with $\text{scope}(f) = \{y_{ij}, y_{is}, y_{(s+1)j}\}$.

After computing all the messages, the optimal value p^* of the corresponding instance of problem (5.12) can be computed from quantities $\Pi_{1,|x|,\mathcal{S}}^{tp,fp}$ by maximizing over all possible values tp, fp (similar to Eq. (3.12)) according to

$$p^* = \max_{tp,fp} \Pi_{1,|x|,\mathcal{S}}^{tp,fp} \odot \eta(tp, fp). \tag{5.24}$$

The whole procedure is summarized in Algorithm 7 supported by the following theorem. We provide a graphical simulation of this algorithm in Appendix C.2.

Theorem 9. *Algorithm 7 always finds an optimal solution of a corresponding problem in $O(|\boldsymbol{x}|^7 \cdot |\mathcal{N}|^3)$ time, where $|\boldsymbol{x}|$ is the number of tokens in the input sentence and $|\mathcal{N}|$ is the number of non-terminal symbols in a grammar.*

Note, that the counts of true and false positives in the above computation scheme are with respect to the binarized tree representation. The resulting performance, however, is evaluated on the original tree representation after reversing the binarization.

Figure 5.7 illustrates the binarization procedure. It turns out that we can easily adjust the above computation scheme to keep track of the corresponding counts

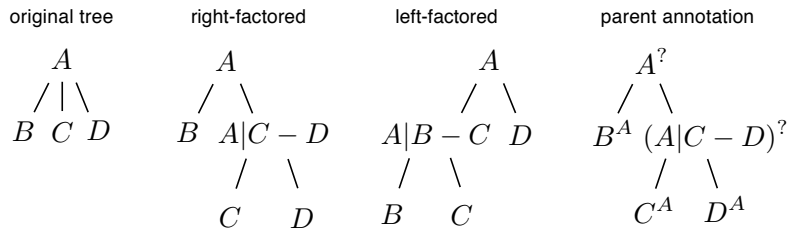


Figure 5.7: Grammar binarization by binarizing the trees. In the notation of artificial constituents $A|C-D$, A before $|$ denotes the parent in the original tree and $C-D$ the children nodes of A which are spanned by the current artificial constituent. $?$ denotes the label of the parent of A.

with respect to unbinarized trees. First note that in order to transform a binarized tree in the original form we need to remove all the artificial constituents. That is, the counts of true and false positives are not affected by their presence. Furthermore, after removing an artificial constituents we need to attach its children in a tree to its parent. In particular, the boundary indices of the corresponding spans of the children nodes do not change during this procedure. Finally we have to remove the additional annotation from the labels of the remaining nodes. To summarize, we can compute the counts of true and false positives with respect to unbinarized grammar from binarized trees if we completely ignore artificial nodes and the additional annotation (e.g. parent annotation). More precisely, we only need to replace the indicator function $\mathbb{1}_1$ for computing $\bar{tp}, \bar{fp}$ in (5.23) by

$$\bar{\mathbb{1}}([A^?]_{ij} \in \boldsymbol{y}^*) = \begin{cases} 0, & A \text{ is artificial,} \\ \mathbb{1}_1([A]_{ij} \in \boldsymbol{y}^*), & \text{else} \end{cases} \tag{5.25}$$

where $?$ denotes the parent annotation of $[A]_{ij}$ and $\boldsymbol{y}^*$ corresponds to the (unbinazised) ground truth. Similarly, we define

$$\bar{\mathbb{1}}([A^?]_{ij} \notin \boldsymbol{y}^*) = \begin{cases} 0, & A \text{ is artificial,} \\ \mathbb{1}_1([A]_{ij} \notin \boldsymbol{y}^*), & \text{else} \end{cases} \tag{5.26}$$

This way we ensure that the corresponding counts of true and false positives are with respect to the unbinarized trees. The overall computation scheme is provable correct, that is, it computes the optimal value of the problem in (5.12).

Again, in the case of margin scaling where the corresponding loss function is decomposable we can use the Cocke-Kasami-Younger (CKY) algorithm for evaluating the prediction function to find an optimal solution of the problem (5.12) by folding the loss function and the compatibility function together.

In the following we show that there can be a big similarity between loss functions which appear to be very different at a first glance. In particular, this applies to the case with binary trees. Namely, due to the fact that all considered parse trees are transformed in CNF, there is the following side effect.

Proposition 6. *Assume that all parse trees are given in CNF. Then for a given sentence $\boldsymbol{x}$ with the true parse tree $\boldsymbol{y}^*$ and a prediction $\boldsymbol{y}$ the number of true positives, the precision, the recall, and the F_1-score are uniquely determined by the number of false positives. In particular, the values of precision, recall, and F_1-score are equal.*

This means that for binary trees the corresponding measures are equivalent. For unbinarized trees, however, they lead to different results.

5.4.3 Experimental Results

We now present the experimental results for the task of constituency parsing on a corpus of english sentences. As training data we used a subset of the Wall Street Journal from the Penn English Treebank-3 containing all sentences of the length $\leqslant 20$. We used the standard data split: sections 2–21 for training (16667 sentences), section 22 for validation (725 sentences), and section 23 for testing (1034 sentences). All parse trees have been preprocessed in the standard way by removing functional tags and null elements.

We trained an SSVM via the cutting plane algorithm, where the corresponding hyperparameter C was chosen by cross-validation over a grid of values $\{10^i : i = 0, 1, ..., 5\}$. In all experiments we consistently used the same features – the productions in the parse trees annotated by the Stanford parser according to the model in [KM03b]. While evaluating the corresponding performance metrics we used the labeled brackets representation of parse trees. That is, each constituent corresponding

Table 5.7: Constituency Parsing (validation set).

Method	P	R	F_1	$\cap/\cup$	0/1 Acc
SP	85.97	86.57	86.20	77.56	15.46
ASP	89.10	88.69	88.85	81.68	23.20
$\text{SVM}_{0/1}$	89.77	90.01	89.85	83.21	27.09
$\text{MS}_{\#CB}$	90.40	90.42	90.37	83.95	25.97
$\text{MS}_{\#FP}$	90.96	90.75	90.81	84.72	**30.30**
MS_{F_1}	91.10	91.01	91.01	85.02	29.32
$\text{MS}_{\cap/\cup}$	**91.12**	**91.04**	**91.04**	**85.06**	29.32
$\text{SS}_{\#CB}$	90.19	90.20	90.15	83.66	25.97
$\text{SS}_{\#FP}$	90.82	90.64	90.69	84.53	29.05
SS_{F_1}	90.90	90.81	90.81	84.65	27.65
$\text{SS}_{\cap/\cup}$	90.89	90.82	90.81	84.67	27.65

Table 5.8: © 2017 IEEE. Constituency Parsing (test set).

Method	P	R	F_1	$\cap/\cup$	0/1 Acc
SP	85.74	86.17	85.88	77.22	16.39
ASP	89.55	89.06	89.25	82.38	25.12
$\text{SVM}_{0/1}$	88.86	89.27	89.00	82.13	27.32
$\text{MS}_{\#CB}$	89.87	89.89	89.81	83.30	26.34
$\text{MS}_{\#FP}$	90.39	90.29	90.28	83.93	28.99
MS_{F_1}	90.68	90.57	90.57	84.41	28.99
$\text{MS}_{\cap/\cup}$	90.83	90.69	90.71	84.61	**29.77**
$\text{SS}_{\#CB}$	90.06	90.02	89.98	83.47	26.34
$\text{SS}_{\#FP}$	90.43	90.34	90.32	84.02	27.91
SS_{F_1}	**90.92**	**90.82**	**90.82**	**84.77**	29.18
$\text{SS}_{\cap/\cup}$	90.78	90.75	90.71	84.65	29.28

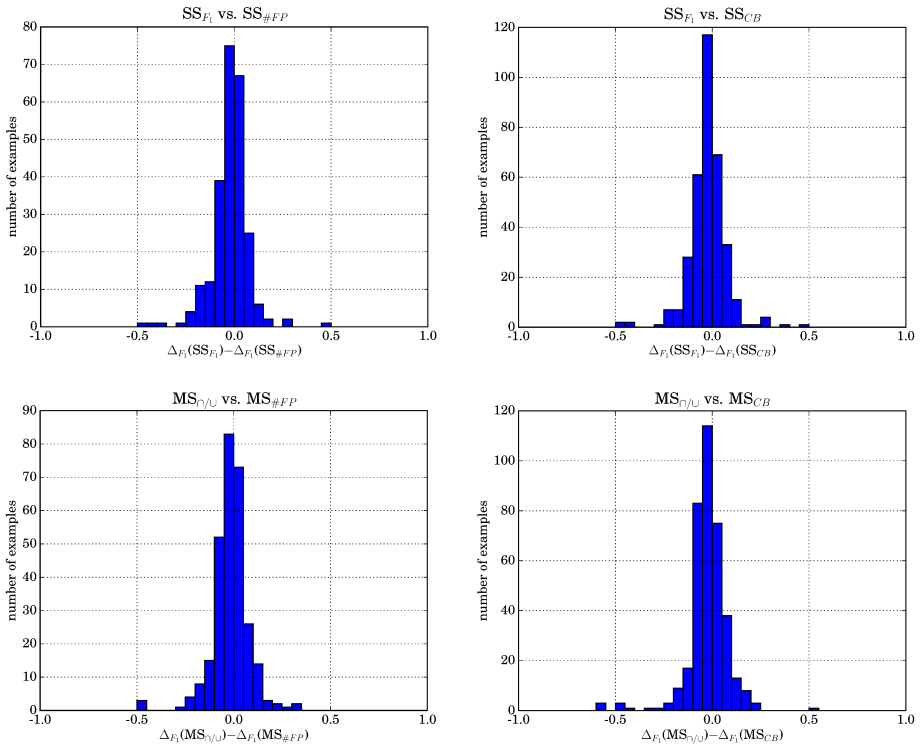


Figure 5.8: This figure illustrates the loss difference on the test set between a few chosen methods. The Wilcoxon signed-rank test confirms that this visually recognized difference is statistically significant. Namely, a corresponding null-hypothesis is that the measurement difference for a pair of methods follows a symmetric distribution around zero. Here, we can see that in each case there is a shift of the corresponding distribution to the left. Note that we do not count examples with zero difference in loss.

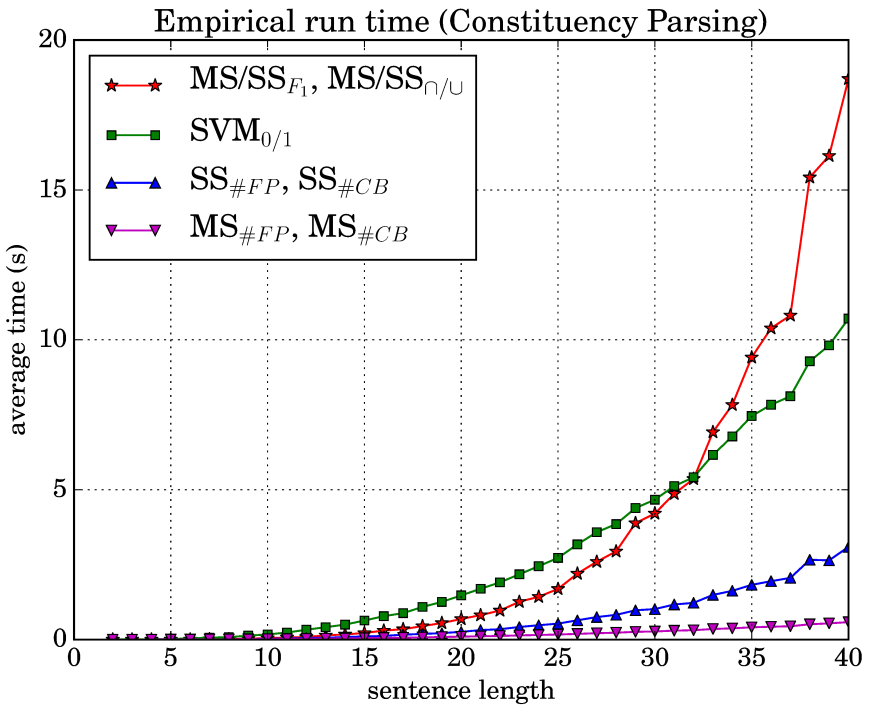


Figure 5.9: © 2017 IEEE. Empirical comparison of run times for the loss augmented inference (Algorithm 7) for various loss functions on a single input instance as a function of the sentence length.

to a node in a tree has been encoded as a triple (A, i, j) with a non-terminal A denoting the label of the node spanning the words $x_i, ..., x_j$ in an input sentence $\boldsymbol{x}$. Additionally we evaluated the performance of structured perceptron (SP) and its averaged version (ASP) [Col02]. The experimental results on the validation and test set are presented in Table 5.7 and Table 5.8, respectively. Here, we use a subscript (e.g. $\#FP$ in $\text{SS}_{\#FP}$) to denote a dissimilarity measure optimized in training (see Table 5.1 for details). The resulting performance is evaluated in terms of precision (P), recall (R), F_1-score, intersection over union ($\cap/\cup$), and zero-one prediction accuracy (0/1 Acc). We see that by adopting the training to optimize for measure like F_1 and $\cap/\cup$ improves the results upon other loss functions. According to the Wilcoxon signed-rank test [RN11] this result is statically significant, while the difference between SS and MS (for the same measure) is not. Figure 5.8 visualizes in a histogram the loss difference of predictions on the test set for a few chosen methods.

We report the averaged running times of the proposed inference algorithm for various loss functions as a function of the sentence length in Figure 5.9. Here, for all models based on SS and for $\text{MS}_{\cap/\cup}$ and MS_{F_1} we used a version of the proposed Algorithm 7 according to the chosen loss function. Note, that for MS with a decomposable dissimilarity measure (like $\#FP$ or $\#CB$) we used the same inference algorithm as for evaluating the prediction function by folding the loss and the compatibility function together. For $\text{SVM}_{0/1}$ we used a version of the K-best Viterbi's algorithm [Wu+08] to compute the second-best solution.

We also evaluated the values for the generalization bound in Theorem 3. However, the corresponding results contain only trivial bounds.

5.5 Summary and Discussion

In this chapter we showed how the generic inference algorithm (Algorithm 2 and 3) can be applied for structured learning on three different examples including sequence tagging (part-of-speech tagging), sequence segmentation (base-NP chunking), and natural language parsing with context free grammars (constituency parsing). In particular, for the first time we provided an empirical analysis of training an SSVM with various non decomposable loss functions (for margin and slack scaling) by using exact inference during the training procedure. The presented applications have been chosen according to their (increasing) complexity with respect to the structure of the outputs ranging from simple sequences to tree-structured graphs. This provides a template for many other applications in the areas like information retrieval, computational biology, and natural language processing.

Additionally, we presented a broad overview of the existing (structured) loss functions with the corresponding compact representation and the resulting complexity for the (augmented) inference algorithm. Interestingly, all of the considered loss functions satisfy the property that the number of states of the corresponding auxiliary variables is polynomially bounded in the size of the outputs, which ensures polynomial running time for the inference.

As a technical result we demonstrated how to adjust the inference algorithm on fixed MRFs to outputs with varying structures on the example of base-NP chunking and constituency parsing. Namely, the message passing procedure of Algorithm 2 and 3 can be combined with the corresponding dynamic programming algorithm for the prediction function.

In the task of constituency parsing the inference is performed on binarized trees but the loss function is evaluated on the original unbanarized representation. We extended the inference procedure to account for this difference. The result corresponds to the inference on unbinarized trees leading to a better prediction accuracy while keeping the computational advantage of binarized representation.

Finally, we used the (augmented) inference algorithm to assess the practical usefulness of the generalization bound for structured learning (given in Theorem 3) by evaluating it on the trained models for the presented applications. The corresponding results demonstrate that the value of this bound can significantly deviate from the actual performance on a data set.

Chapter 6

Conclusion

The main goal of the thesis is to address the computational challenges inherent in a family of discrete optimization problems which we here refer to as globally augmented MAP inference. In particular, our main focus is on the exact inference, which is known to be NP-hard in general. Here, junction tree algorithm [KF09; MC10] provides the most efficient technique known to perform exact MAP inference in discrete graphical models, but requires the corresponding treewidth to be bounded. However, adding a global potential to the model or imposing a global constraint on the variables results in a fully connected graph which renders the application of this algorithm intractable. We showed that for a considerable amount of cases when a corresponding global term (or constraint) has a suitable internal structure, we can perform exact inference efficiently. The main contribution of the thesis includes a formal definition (Problem 1) specifying a set of tractable problem instances and a corresponding generic message passing algorithm (Algorithm 3), which is guaranteed to find an optimal solution in polynomial time.

By relaxing some of the tractability conditions we extend the range of handleable problem instances but give up the optimality guarantee of the corresponding solutions. In particular, we (theoretically) analyze the intricacies of an accurate and efficient approximation framework of Lagrangian relaxation and dual decomposition when applied in the context of augmented MAP inference. An appealing property of these techniques is that they provide feasible solutions with a certificate of optimality which allows for an efficient verification whether a given solution is optimal. Furthermore, the corresponding computations can be performed in a distributed way. Related to this discussion we additionally investigated the connections between the two established optimization techniques for MAP inference, LP relaxation and dual decomposition. We successfully provided a few novel findings and showed that for the important practical case of binary pairwise MRFs (in a non degenerate case) both methods provide equivalent information about the potential solutions (see Lemma 1 and Theorem 6). This is a good news because LP relaxation is known to have nice theoretical properties but modest scaling behavior due to an extensive memory consumption. Dual decomposition, however, scales to problems of arbitrary size. By inheriting the nice properties of the LP relaxation it provides an effective alternative because of its distributed optimization scheme.

The most important practical implication of the presented theoretical analysis is that it provides a computational tool for training SSVMs by optimizing for popular non decomposable loss functions (see Table 5.1). The corresponding advantage of using task-dependent losses is that we can encode important prior knowledge about the model leading to a higher prediction accuracy with respect to the task loss as supported by the experimental results. This also includes training via slack scaling formulation. Since the original paper on SSVM [Tso+05] slack scaling has been conjectured to be more beneficial and accurate than margin scaling. We performed

a few experiments comparing both formulations which confirm that in some cases training via slack scaling tends to achieve more accurate solution.

Moreover, we demonstrated how the presented algorithmic idea can be used for evaluating generalization bounds for structured learning and performing MAP inference subject to global constraints on the prediction variables.

6.1 Summary of Contributions

6.1.1 Chapter 2

The main goal of Chapter 2 is to introduce necessary background and notation for the following discussions in the thesis. However, in the context of SSVMs we provide a noticeable new result regarding the properties of the margin scaling formulation. Namely, in the literature the slack scaling is believed to be in many ways better-behaved than margin scaling. In particular, a statement is widespread that one of the alleged disadvantages of margin scaling (compared to slack scaling) is not being scaling-invariant with respect to the loss function, which makes this approach less manageable. We showed (Proposition 3) that margin scaling also ensures invariance of some sort. Although we cannot tune the scale of the loss Δ by changing the values of the hyperparameter C in the same way as for slack scaling, this is not really a disadvantage, since the corresponding weight vectors $\boldsymbol{w}$ point in the same direction and differ only in their scaling. Therefore, as for the slack scaling, we also do not need to manage the scale of the loss function explicitly and can handle different scalings by means of the hyperparameter C.

6.1.2 Chapter 3

Given the general task of globally augmented MAP inference we defined a large class of tractable problem instances (Problem 1) which can be solved efficiently. For this tractable case we present a generic message passing algorithm (Algorithm 3) which is guaranteed to find an optimal solution in polynomial time with respect to the problem parameters. Additionally, we provided a rigorous theoretical analysis of the computational complexity and proved the correctness of the presented algorithm (Theorem 2, Proposition 5). Interestingly, the presented message passing algorithm can be seen as conventional message passing (equivalent to junction tree algorithm) on a clique tree without auxiliary variables, but where the size of the state space for each variable is now increased by a certain factor R being an upper bound on the number of states of the auxiliary variables. In other words, we can remove the global dependencies (imposed by the mapping H in Problem 1) reducing the overall treewidth by cleverly introducing auxiliary variables, but pay a price that the size of the label space becomes a function of the graph size. If R grows only polynomially with the graph size M – which is true in many practical cases – we can perform exact inference in polynomial time.

Additionally, we showed how the presented framework can be used in multiple application scenarios including loss augmented inference, evaluating PAC-Bayesian generalization bounds for structured prediction, and globally constrained MAP inference. To demonstrate further potential of the presented algorithmic idea, we used it to motivate a novel learning approach based a new structured AUC measure.

6.1.3 Chapter 4

We presented the established frameworks of linear programming (LP) relaxation and dual decomposition (DD), two important MAP solvers for discrete MRFs. It is well known that these two methods have a strong connection both providing an optimal solution to the MAP problem when a corresponding LP relaxation is tight. However, less is known about their relationship in the opposite and more realistic case. Namely, while it is known that both methods have the same optimal objective value, an interesting question is if there are other properties they share. In particular, the connection between the solutions of LP relaxation and the assignments which can be extracted via DD, has not been clarified. For example, even if the solution of the LP relaxation is unique, there might be multiple optimal (but disagreeing) assignments via DD. What is the nature of this ambiguity? Are all these assignments equivalent or can we even extract an additional information about the optimal solutions from analyzing the disagreement behavior? These and other questions were the main motivation for the corresponding analysis. Here, we successfully provided a few novel findings explaining how the fully integral assignments obtained via DD agree with the optimal fractional assignments via LP relaxation when the latter is not tight. More specifically, we have proved:

- given an optimal (fractional) solution of the LP relaxation, there always exists an optimal variable assignment via DD which agrees with the integral part of the LP solution; this also holds for non binary models with arbitrary higher order potentials
- for binary pairwise MRFs (in a non degenerate case) the first result holds for every optimal assignment which can be extracted via DD
- for binary pairwise MRFs (in a non degenerate case) the unambiguous part among all optimal assignments from different trees in a decomposition coincides with the integral part of the (fractional) optimal assignment via LP relaxation

In particular, for binary pairwise MRFs the integral part of an optimal solution provided via LP relaxation is known to be strongly persistent. Therefore, due to the properties listed above, we can conclude that (for this case) both methods LP relaxation and DD share the partial optimality property of their solutions.

To summarize, the LP relaxation is a popular method for discrete MAP inference in pairwise graphical models because of its appealing (partial) optimality properties. However, this method does not scale nicely due to the extensive memory requirements restricting its practical use for bigger problems. Here, DD provides an effective alternative via distributed optimization, which scales to problems of arbitrary size while inheriting all the (partial) optimality properties of the LP relaxation. We conclude that for binary pairwise models both methods provide exactly the same information about their solutions.

The main restriction of Problem 1 is the requirement on the treewidth for the MRFs associated with the mappings F and $\boldsymbol{G}$ to be bounded. We extend the range of handleable problems to models with unbounded treewidth by giving up the optimality guarantee of the corresponding solutions. More precisely, we proposed to use the technique of Lagrangian relaxation, and more specifically, DD. We analyzed the intricacies of these approximation techniques when applied to the task of (globally) augmented MAP inference and discussed pros and contras of the individual formulations. As a further result we proposed a simple bisection method for the

special case of MAP inference subject to a global linear constraint and investigated its relationship to the existing methods including ES and cutting-plane algorithm. The corresponding result shows significant savings in the number of calls to a corresponding inference algorithm, which builds the computational bottleneck during the optimization.

6.1.4 Chapter 5

We showed how the exact inference algorithm (Algorithm 3) can be applied for structured learning on three different examples including sequence tagging (part-of-speech tagging), sequence segmentation (base-NP chunking), and natural language parsing with context free grammars (constituency parsing). In particular, for the first time we provided an empirical analysis of training an SSVM with various non decomposable loss functions (for margin and slack scaling) by using exact inference during the training procedure. The presented applications have been chosen according to their (increasing) complexity with respect to the structure of the outputs ranging from simple sequences to tree-structured graphs. This provides a template for many other applications in the areas like information retrieval, computational biology, and natural language processing.

Additionally, we presented a broad overview of the existing (structured) loss functions with the corresponding compact representation and the resulting complexity for the (augmented) inference algorithm. Interestingly, all of the considered loss functions satisfy the property that the number of states of the corresponding auxiliary variables is polynomially bounded in the size of the outputs, which ensures polynomial running time for the inference.

As a further technical result we demonstrated how to adjust the inference algorithm on fixed MRFs to outputs with varying structures on the example of base-NP chunking and constituency parsing. Namely, the message passing procedure of the (augmented) inference algorithm can be combined with the corresponding dynamic programming algorithm for the prediction function.

Specifically, in the task of constituency parsing, inference is performed on binarized trees but the loss function is evaluated on the original unbanarized representation. We extended the inference procedure to account for this difference. The result corresponds to the inference on unbinarized trees leading to a higher prediction accuracy while keeping the computational advantage of binarized representation.

Finally, we used the (augmented) inference algorithm to assess the practical usefulness of the generalization bound for structured learning (given in Theorem 3) by evaluating it on the trained models for the presented applications. The corresponding results demonstrate that the value of this bound can significantly deviate from the actual performance on a data set.

Appendix A

Supplements (Chapter 3)

A.1 Proof of Theorem 1

The correctness proof is very similar to the proof of Theorem 2 presented in Chapter 3. However, we provide an explicit proof here for the sake of completeness. To show the correctness of the presented computations we first provide a semantic interpretation of messages as follows. Let $f_s - y_m$ be an edge between a factor node f_s and a variable node y_m. We denote by $\mathcal{F}_{\prec(f_s - y_m)}$ the set of all factors f_k of the mapping F on the f_s-th side of the factor graph and by $\mathcal{G}_{\prec(f_s - y_m)}$ a corresponding set of factors of the mapping $\boldsymbol{G}$. Without loss of generality we assume that for each factor f_k of the mapping F there is exactly one factor g_k in $\boldsymbol{G}$ with the same scope $\boldsymbol{y}_{C_k}$. Furthermore, we denote by $\mathcal{V}_{\prec(f_s - y_m)}$ the set of all variables appearing on the f_s. Intuitively, a message $\mu_{f_s \to y_m}(Y_m, \boldsymbol{l})$ sent from a factor node f_s to a variable node y_m corresponds to a sum of all factors contained in $\mathcal{F}_{\prec(f_s - y_m)}$ which is maximized (for fixed values of Y_m and $\boldsymbol{l}$) over the variables in $\mathcal{V}_{\prec(f_s - y_m)}$ subject to the constraint $\boldsymbol{l} = \sum_{\boldsymbol{g}_k \in \mathcal{G}_{\prec(f_s - y_m)}} \boldsymbol{g}_k(\boldsymbol{y}_{C_k})$. That is, we define the following induction hypothesis:

$$\mu_{f_s \to y_m}(Y_m, \boldsymbol{l}) = \max_{\mathcal{V}_{\prec(f_s - y_m)} \colon \boldsymbol{l} = \sum_{\boldsymbol{g}_k \in \mathcal{G}_{\prec(f_s - y_m)}} \boldsymbol{g}_k(\boldsymbol{y}_{C_k})} \sum_{f_k \in \mathcal{F}_{\prec(f_s - y_m)}} f_k(\boldsymbol{y}_{C_k}). \tag{A.1}$$

Similarly we define the sets $\mathcal{F}_{\prec(y_m - f_s)}, \mathcal{G}_{\prec(y_m - f_s)}, \mathcal{V}_{\prec(y_m - f_s)}$ and the induction hypothesis for the messages from a variable node to a factor node:

$$\mu_{y_m \to f_s}(Y_m, \boldsymbol{l}) = \max_{\mathcal{V}_{\prec(y_m - f_s)} \colon \boldsymbol{l} = \sum_{\boldsymbol{g}_k \in \mathcal{G}_{\prec(y_m - f_s)}} \boldsymbol{g}_k(\boldsymbol{y}_{C_k})} \sum_{f_k \in \mathcal{F}_{\prec(f_s - y_m)}} f_k(\boldsymbol{y}_{C_k}). \tag{A.2}$$

Checking the validity of messages from a leaf to neighbor node is straightforward. Consider now an edge $(f_s - y_m)$ such that f_s is not a leaf. Let $y_{i_1}, ..., y_{i_n}$ be the neighbor variables of f_s other than y_m. Since the factor graph is a tree the set $\mathcal{V}_{\prec(f_s - y_m)}$ is a disjoint union of $\mathcal{V}_{\prec(y_{i_k} - f_s)}$ for $k = 1, ..., n$. Similarly, $\mathcal{F}_{\prec(f_s - y_m)}$ is the disjoint union of the $\mathcal{F}_{\prec(y_{i_k} - f_s)}$ and $\{f_s\}$. Finally, $\mathcal{G}_{\prec(f_s - y_m)}$ is the disjoint union of the $\mathcal{G}_{\prec(y_{i_k} - f_s)}$ and $\{\boldsymbol{g}_s\}$. In the following, we abbreviate the term $\mathcal{V}_{\prec(y_{i_k} - f_s)} \colon \boldsymbol{l}_{i_k} = \sum_{\boldsymbol{g} \in \mathcal{G}_{\prec(y_{i_k} - f_s)}} \boldsymbol{g}$ describing a range of variables in $\mathcal{V}_{\prec(y_{i_k} - f_s)}$ subject to a corresponding equality constraint with respect to $\boldsymbol{l}_{i_k}$ by $\mathcal{V}_{\prec(i_k - s)} \colon \boldsymbol{l}_{i_k}$. Thus, the right hand side of Eq. (A.1) is equal to

$$\max_{\boldsymbol{y}_{C_s \setminus \{m\}}} \max_{\{\boldsymbol{l}_{i_k}\}_{k=1}^n} \max_{\mathcal{V}_{\prec(i_1 - s)} \colon \boldsymbol{l}_{i_1}} \cdots \max_{\mathcal{V}_{\prec(i_n - s)} \colon \boldsymbol{l}_{i_n}} \left(\sum_{f \in \mathcal{F}_{\prec(i_1 - s)}} f \right) + \cdots + \left(\sum_{f \in \mathcal{F}_{\prec(i_n - s)}} f \right) + f_s \tag{A.3}$$

where in the second $\max$ we maximize over all configurations of $\{\boldsymbol{l}_{i_k}\}_{k=1}^n$ subject to the constraint $\sum_{k=1}^n \boldsymbol{l}_{i_k} = \boldsymbol{l} - \boldsymbol{g}_s(\boldsymbol{y}_{C_s})$. Since all the corresponding sets are disjoint the term (A.3) is equal to

$$\max_{\boldsymbol{y}_{C_s\setminus\{m\}},\{\boldsymbol{l}_{i_k}\}_{k=1}^n} f_s + \underbrace{\max_{\mathcal{V}_{\prec(i_1-s)}\colon \boldsymbol{l}_{i_1}} \left(\sum_{f\in\mathcal{F}_{\prec(i_1-s)}} f\right)}_{\mu_{y_{i_1}\to f_s}(Y_{i_1},\boldsymbol{l}_{i_1})} + \cdots + \underbrace{\max_{\mathcal{V}_{\prec(i_n-s)}\colon \boldsymbol{l}_{i_n}} \left(\sum_{f\in\mathcal{F}_{\prec(i_n-s)}} f\right)}_{\mu_{y_{i_n}\to f_s}(Y_{i_n},\boldsymbol{l}_{i_n})} \tag{A.4}$$

where again the maximization over $\{\boldsymbol{l}_{i_k}\}_{k=1}^n$ is subject to the constraint $\sum_{k=1}^n \boldsymbol{l}_{i_k} = \boldsymbol{l} - \boldsymbol{g}_s(y_{C_s})$. Using the induction hypothesis (A.2) in the last expression we get the right hand side of Eq. (3.9) proving the claim in Eq. (A.1). The induction hypothesis (A.2) can be verified in a similar way.

Now look at Eq. (3.11). Using Eq. (A.1) and the fact that all involved sets of variables and factors for different messages are disjoint we conclude that the computed values $\mu(\boldsymbol{l})$ corresponds to the sum of all factors f for the mapping F over the variables in $\boldsymbol{y}$ which is maximized subject to the constraint $\boldsymbol{G}(\boldsymbol{y}) = \boldsymbol{l}$. Therefore, by performing maximization over all values $\boldsymbol{l}$ according to Eq. (3.12) we get the optimal value of Problem 1.

Consider now the formula for the message passing in Eq. (3.9). First a maximization is performed first over $|y_{C_s\setminus\{m\}}|$ variables for each configuration of Y_m resulting in a cost $O(N^{\tau+1})$. Then a maximization over $\{\boldsymbol{l}_k\}$ costs additionally $O(R^{\tau-1})$. Together with all possible values for $\boldsymbol{l}$ it yields $O(R^\tau)$. Therefore, sending a message for all possible configurations of $(Y_m, \boldsymbol{l})$ on the edge $(f_s - y_m)$ costs $O(N^{\tau+1}\cdot R^\tau)$ time. Since there are $O(M)$ such edges it results in a cost $O(M\cdot N^{\tau+1}\cdot R^\tau)$. Applying similar analysis to the formula in (3.10) we get an additional cost $O(M\cdot N\cdot R^{\nu-1})$ where ν is the maximal degree of a variable node. Therefore, the total complexity is upper bounded by $O(M\cdot N^{\tau+1}\cdot R^\tau + M\cdot N\cdot R^{\nu-1})$.
□

A.2 Proof of Proposition 5

Assume we are given a clique tree with treewidth τ. That is, every node in a clique tree has at most $\tau+1$ variables. Therefore, the number of all possible variable combinations for a sepset is given by $2^{\tau+1}-2$ where we exclude the empty set and the set containing all the variables in the corresponding clique.

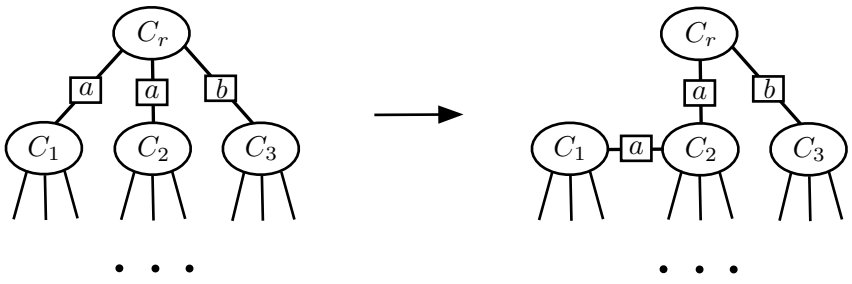


Figure A.1: Illustration of the reshaping procedure for a clique tree in the case where the condition $\nu \leqslant 2^{\tau+2}-4$ is violated. C_r is the root clique where a sepset a occurs at least two times. The number of neighbors of C_r can be reduced by removing the edge between C_1 and C_r and attaching C_1 to C_2. This way we can ensure that every node has at most one duplicate for every possible sepset. Furthermore, this procedure preserves the running intersection property.

Furthermore, we can deal with duplicates by rearranging the edges in the clique tree such that that for every sepset from the $2^{\tau+1}-2$ possibilities there is at most one duplicate resulting in $2^{\tau+2}-4$ possible sepsets. More precisely, we first choose a node having more than $2^{\tau+2}-4$ neighbors as root and then reshape the clique tree by propagating some of the neighbors towards the leaves as illustrated in Figure A.1. Due to these procedure the maximal number of repetition for every sepset is upper bounded by 2.
□

A.3 Structured AUC Optimization Framework

In the following we motivate a new learning approach for SSVMs. Based on the presented algorithmic idea, we first define a new loss function, which, for a given input, quantifies the prediction accuracy by taking in consideration a ranking of different outputs similarly to the area under the ROC curve (AUC) measure ([Faw06]). Then we show how to modify the framework of structured prediction to optimize for this new measure of performance. In some sense it generalizes the idea presented in ([Joa05]) to structured outputs.

For simplicity we consider the case where all the outputs $\boldsymbol{y} \in \mathcal{Y}$ are sequences of size M although the concept presented below generalizes to arbitrary structures. Let $r\colon \mathcal{Y} \to \mathbb{R}$ be any (partial) ranking function encoding some prior knowledge, e.g. a (negative) loss function like Hamming distance: $r(\boldsymbol{y}) = -\Delta_{HD}(\boldsymbol{y}^*, \boldsymbol{y})$. Our intuition here is that for a good weight vector $\boldsymbol{w}$, for any pair $(\boldsymbol{x}, \boldsymbol{y}^*)$ from the training set the following should hold true:

$$\forall \hat{\boldsymbol{y}}, \bar{\boldsymbol{y}} \in \mathcal{Y}\colon r(\hat{\boldsymbol{y}}) > r(\bar{\boldsymbol{y}}) \Rightarrow \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \hat{\boldsymbol{y}}) > \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \bar{\boldsymbol{y}}). \tag{A.5}$$

That is, outputs with a higher rank should also have a higher score with respect to the compatibility function. Instead of explicitly enforcing the above constraint (which is infeasible due to exponential size of $\mathcal{Y}$) we can introduce a penalty which counts the number of the violated constraints. Note that $r(\cdot)$ and $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \cdot)$ provide two different orderings (or rankings) on the outputs $\boldsymbol{y} \in \mathcal{Y}$. Therefore we can consider the number of swapped pairs

$$\#\text{SwappedPairs} := \left| \{(\hat{\boldsymbol{y}}, \bar{\boldsymbol{y}}) \in \mathcal{Y}(r, \boldsymbol{w})\colon \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \hat{\boldsymbol{y}}) > \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \bar{\boldsymbol{y}}) \text{ and } r(\hat{\boldsymbol{y}}) < r(\bar{\boldsymbol{y}})\} \right|, \tag{A.6}$$

where we only consider outputs with maximal score for each possible value of the ranking function. E.g., for $r(\boldsymbol{y}) = -\Delta_{HD}(\boldsymbol{y}^*, \boldsymbol{y})$ we get $\mathcal{Y}(r, \boldsymbol{w}) := \{\boldsymbol{y}^0, ..., \boldsymbol{y}^M\}$ where

$$\boldsymbol{y}^k := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}\colon r(\boldsymbol{y}) = -k} \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}), \quad k \in \{0, ..., M\}. \tag{A.7}$$

We define the structured AUC loss according to

$$\Delta_{\text{AUC}}(r) = \frac{\#\text{SwappedPairs}}{(M-1)M/2}. \tag{A.8}$$

We note that the above function generalizes the AUC measure for binary classification as has been shown in ([Joa05]).

Joint Feature Map

We now show how to modify the framework of the SSVMs to train by using the structured AUC measure in (A.8). Given an original joint feature map $\boldsymbol{\Psi}\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$, we now define a new joint feature map $\bar{\boldsymbol{\Psi}}\colon \mathcal{X} \times \mathcal{Y} \times \overline{\mathcal{Y}} \to \mathbb{R}^d$ with $\overline{\mathcal{Y}} := \{-1, 1\}^{(M-1)M/2}$ as

$$\bar{\boldsymbol{\Psi}}(\boldsymbol{x}, \boldsymbol{y}^*, \overline{\boldsymbol{y}}) = \sum_{\tilde{\boldsymbol{y}}, \hat{\boldsymbol{y}}\colon\, r(\tilde{\boldsymbol{y}}) < r(\hat{\boldsymbol{y}})} \overline{y}_{r(\tilde{\boldsymbol{y}}), r(\hat{\boldsymbol{y}})} (\boldsymbol{\Psi}(\boldsymbol{x}, \tilde{\boldsymbol{y}}) - \boldsymbol{\Psi}(\boldsymbol{x}, \hat{\boldsymbol{y}})) \tag{A.9}$$

We can interpret this new feature map as follows. The summation above goes over different outputs $\boldsymbol{y} \in \mathcal{Y}(r, \boldsymbol{w})$ corresponding to maximal violators for different values of the ranking function $r(\boldsymbol{y}) = -\Delta_{\text{HD}}(\boldsymbol{y}^*, \boldsymbol{y})$ including zero for the ground true label $\boldsymbol{y}^*$. The number of pairwise comparisons between these $M + 1$ outputs is $(M-1)M/2$. Given this new representation we can write the AUC loss as

$$\Delta_{\text{AUC}}(\overline{\boldsymbol{y}}^*, \overline{\boldsymbol{y}}) = \frac{1}{2} \sum_{i=1}^{(M-1)M/2} \overline{y}_i^* - \overline{y}_i \tag{A.10}$$

where $\overline{\boldsymbol{y}}^* = (1, ..., 1)^\top$. That is, $\Delta_{\text{AUC}}(\overline{\boldsymbol{y}}^*, \overline{\boldsymbol{y}})$ is equal to the number of negative entries in $\overline{\boldsymbol{y}}$. This corresponds to the number of swapped pairs. Note that we normalize the loss by dividing the number of swapped pairs by the total number of pairwise comparisons being $(M-1)M/2$.

The training procedure is similar to the conventional cutting plane approach. Namely, in each iteration for each training data point $(\boldsymbol{x}, \boldsymbol{y}^*)$ we compute the maximal violators $\hat{\boldsymbol{y}} \in \mathcal{Y}(r, \boldsymbol{w})$ (using Algorithm 3) for every value of the ranking function r. Given the maximal violators we can define a new constraint

$$\boldsymbol{w}^\top \bar{\boldsymbol{\Psi}}(\boldsymbol{x}, \bar{\boldsymbol{y}}^*) - \boldsymbol{w}^\top \bar{\boldsymbol{\Psi}}(\boldsymbol{x}, \bar{\boldsymbol{y}}) \geqslant \Delta_{\text{AUC}}(r) - \xi \tag{A.11}$$

by evaluating the AUC loss and computing the corresponding joint feature maps. Note that the prediction is still with respect to the original feature map $\boldsymbol{\Psi}$. That is,

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \; \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}). \tag{A.12}$$

The main difference of this approach to the conventional SSVM is that the constraints $\boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}^*) - \boldsymbol{w}^\top \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) \geqslant \Delta(\boldsymbol{y}^*, \boldsymbol{y}) - \xi$ aim at enforcing the margin for a specific loss value, while the new approach ignores the exact loss value and only focuses on a proper sorting of the prediction scores according to the loss function.

Appendix B

Supplements (Chapter 4)

B.1 Proof of Lemma 1

We show the direction $(i) \Rightarrow (iii)$. The remaining directions are straightforward. First note that the integer problems in (4.2) and (4.7) are equivalent. That is, if $\boldsymbol{\mu}^*$ is an optimal solution for (4.2), then $(\boldsymbol{\mu}^*, ..., \boldsymbol{\mu}^*)$ is an optimal solution for (4.7). Furthermore, since strong duality holds (because LP relaxation has an integral solution: $(i) \Rightarrow (ii)$) each optimal solution $(\boldsymbol{\mu}^*, ..., \boldsymbol{\mu}^*)$ is a minimizer of the corresponding Lagrangian (for the dual optimal) and also satisfies the agreement constraint.

B.2 Auxiliary Claims

The following two lemmas are required for the proof of Theorem 4.

Lemma 10. *Let $g_{4.7}$, $g_{4.8}$ be the dual functions of the problems (4.7) and (4.8) (presented in (4.9) and (4.10)), respectively. For any value of the dual variables $\boldsymbol{u}$ the equality $g_{4.7}(\boldsymbol{u}) = g_{4.8}(\boldsymbol{u})$ holds.*

Proof.

$$\begin{aligned} g_{4.8}(\boldsymbol{u}) &= \inf_{\boldsymbol{\mu}^1 \in L_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in L_{\mathcal{T}_m}} \left\{ \sum_{j=1}^{m} (\boldsymbol{\theta}^j + \boldsymbol{u}^j)^\top \boldsymbol{\mu}^j \right\} \\ &= \inf_{\boldsymbol{\mu}^1 \in L_{\mathcal{T}_1}} \{(\boldsymbol{\theta}^1 + \boldsymbol{u}^1)^\top \boldsymbol{\mu}^1\} + ... + \inf_{\boldsymbol{\mu}^m \in L_{\mathcal{T}_m}} \{(\boldsymbol{\theta}^m + \boldsymbol{u}^m)^\top \boldsymbol{\mu}^m\} \\ &= \inf_{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1}} \{(\boldsymbol{\theta}^1 + \boldsymbol{u}^1)^\top \boldsymbol{\mu}^1\} + ... + \inf_{\boldsymbol{\mu}^m \in \mathcal{X}_{\mathcal{T}_m}} \{(\boldsymbol{\theta}^m + \boldsymbol{u}^m)^\top \boldsymbol{\mu}^m\} \\ &= \inf_{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in \mathcal{X}_{\mathcal{T}_m}} \left\{ \sum_{j=1}^{m} (\boldsymbol{\theta}^j + \boldsymbol{u}^j)^\top \boldsymbol{\mu}^j \right\} = g_{4.7}(\boldsymbol{u}) \end{aligned}$$

In the third equation we used the known fact that for a tree-structured MRF the local consistency polytope coincides with the marginal polytope and that a corresponding objective for each subproblem is linear. □

Lemma 11. *For any tree-structured MRF $\mathcal{T}$ with the corresponding set of valid assignments $\mathcal{X}_{\mathcal{T}} \subseteq \mathbb{R}^d$ in the standard overcomplete representation (as defined in (4.3)) and for any $\mathcal{I} \subseteq \{1, ..., d\}$ the following equality holds*

$$conv\ \mathcal{X}_{\mathcal{T}} \cap \{\boldsymbol{\mu} \in \mathbb{R}^d \colon \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\} = conv\ \{\boldsymbol{\mu} \in \mathcal{X}_{\mathcal{T}} \colon \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\}, \tag{B.1}$$

where $\boldsymbol{\mu}^* \in \mathbb{R}^d$ is a point, for which there exists an assignment $\bar{\boldsymbol{\mu}} \in \mathcal{X}_{\mathcal{T}}$ with $\bar{\boldsymbol{\mu}}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}$.

Proof. " $\subseteq$ " : Let $\boldsymbol{v} \in \operatorname{conv} \mathcal{X}_{\mathcal{T}} \cap \{\boldsymbol{\mu} \in \mathbb{R}^d \colon \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\}$. We now show that $\boldsymbol{v}$ can be represented as a convex combination of points $\boldsymbol{\mu} \in \mathcal{X}_{\mathcal{T}}$ where $\boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}$ for each $\boldsymbol{\mu}$ in the combination. On the one hand, since $\boldsymbol{v} \in \operatorname{conv} \mathcal{X}_{\mathcal{T}}$ we can write it as a convex combination $\boldsymbol{v} = \sum_{\boldsymbol{\mu}} \alpha_{\boldsymbol{\mu}} \boldsymbol{\mu}$ for $\boldsymbol{\mu} \in \mathcal{X}_{\mathcal{T}}$, $\alpha_{\boldsymbol{\mu}} \in [0, 1]$ where we assume without loss of generality that the sum contains only $\alpha_{\boldsymbol{\mu}} > 0$. On the other hand, since $\boldsymbol{v} \in \{\boldsymbol{\mu} \in \mathbb{R}^d \colon \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\}$, it implies that $\sum_{\boldsymbol{\mu}} \alpha_{\boldsymbol{\mu}} \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}$ must hold. To prove the subset relationship it suffices to show that $\mu_i = \mu^*_i$ for all $i \in \mathcal{I}$ for each of the points $\boldsymbol{\mu}$ in the convex combination. Now let $\mu^*_i = 1$ for some $i \in \mathcal{I}$. Assuming that there is a $\hat{\boldsymbol{\mu}}$ in our combination with $\hat{\mu}_i = 0$ results in the following contradiction:

$$\sum_{\boldsymbol{\mu}} \alpha_{\boldsymbol{\mu}} \mu_i = \alpha_{\hat{\boldsymbol{\mu}}} \cdot 0 + \sum_{\boldsymbol{\mu} \neq \hat{\boldsymbol{\mu}}} \alpha_{\boldsymbol{\mu}} \underbrace{\mu_i}_{\leqslant 1} \leqslant \sum_{\boldsymbol{\mu} \neq \hat{\boldsymbol{\mu}}} \alpha_{\boldsymbol{\mu}} < 1 = \mu^*_i,$$

that is, $\sum_{\boldsymbol{\mu}} \alpha_{\boldsymbol{\mu}} \mu_i \neq \mu^*_i$. Analogously, considering the case $\mu^*_i = 0$ and assuming the existence of one $\hat{\mu}_i = 1$ gives rise to the following contradiction:

$$\sum_{\boldsymbol{\mu}} \alpha_{\boldsymbol{\mu}} \mu_i = \alpha_{\hat{\boldsymbol{\mu}}} \cdot 1 + \underbrace{\sum_{\boldsymbol{\mu} \neq \hat{\boldsymbol{\mu}}} \alpha_{\boldsymbol{\mu}} \mu_i}_{\geqslant 0} \geqslant \alpha_{\hat{\boldsymbol{\mu}}} > 0 = \mu^*_i,$$

that is, $\sum_{\boldsymbol{\mu}} \alpha_{\boldsymbol{\mu}} \mu_i \neq \mu^*_i$.

" $\supseteq$ " : This direction follows directly from $\{\boldsymbol{\mu} \in \mathcal{X}_{\mathcal{T}} \colon \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\} \subseteq \mathcal{X}_{\mathcal{T}}$ and $\{\boldsymbol{\mu} \in \mathcal{X}_{\mathcal{T}} \colon \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\} \subseteq \{\boldsymbol{\mu} \in \mathbb{R}^d \colon \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\}$ where $\{\boldsymbol{\mu} \in \mathbb{R}^d \colon \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\}$ is a convex set. □

B.3 Proof of Theorem 4

The following derivations imply the existence of a set of assignments $\boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^m$ for the individual subproblems according to the statement in the theorem:

$$\begin{aligned}
\inf_{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in \mathcal{X}_{\mathcal{T}_m}} \mathcal{L}(\boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^m, \bar{\boldsymbol{u}}) &\overset{L10}{=} \inf_{\boldsymbol{\mu}^1 \in L_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in L_{\mathcal{T}_m}} \mathcal{L}(\boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^m, \bar{\boldsymbol{u}}) \\
&= \inf_{\boldsymbol{\mu}^1 \in L_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in L_{\mathcal{T}_m} \colon \boldsymbol{\mu}^j_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \mathcal{L}(\boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^m, \bar{\boldsymbol{u}}) \\
&\overset{L11}{=} \inf_{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in \mathcal{X}_{\mathcal{T}_m} \colon \boldsymbol{\mu}^j_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \mathcal{L}(\boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^m, \bar{\boldsymbol{u}})
\end{aligned}$$

The first equality holds due to Lemma 10. The second equality is due to the following fact. Since strong duality holds for OP in (4.8), every optimal primal solution is a minimiser of the Lagrangian $\mathcal{L}(\cdot, ..., \cdot, \bar{\boldsymbol{u}})$. Therefore, the set of feasible solutions restricted by the constraints $\boldsymbol{\mu}^j_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}$ contains at least one optimal solution

$\boldsymbol{\mu}^1 := \boldsymbol{\mu}^*|_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m := \boldsymbol{\mu}^*|_{\mathcal{T}_m}$, where $\boldsymbol{\mu}^*|_{\mathcal{T}_j}$ denotes a projection to a subspace corresponding to a tree $\mathcal{T}_j$. The third equality can be shown using Lemma 11 as follows:

$$
\begin{aligned}
&\inf_{\boldsymbol{\mu}^1 \in L_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in L_{\mathcal{T}_m} \,:\, \boldsymbol{\mu}^j_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \mathcal{L}(\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_m, \bar{\boldsymbol{u}}) \\
&= \inf_{\boldsymbol{\mu}^1 \in L_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in L_{\mathcal{T}_m} \,:\, \boldsymbol{\mu}^j_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \left\{ \sum_{j=1}^{m} (\boldsymbol{\theta}_j + \bar{\boldsymbol{u}}_j)^\top \boldsymbol{\mu}^j \right\} \\
&= \inf_{\boldsymbol{\mu}^1 \in L_{\mathcal{T}_1} \,:\, \boldsymbol{\mu}^1_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \{(\boldsymbol{\theta}_1 + \bar{\boldsymbol{u}}_1)^\top \boldsymbol{\mu}^1\} + ... + \inf_{\boldsymbol{\mu}^m \in L_{\mathcal{T}_m} \,:\, \boldsymbol{\mu}^m_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \{(\boldsymbol{\theta}_m + \bar{\boldsymbol{u}}_m)^\top \boldsymbol{\mu}^m\} \\
&\overset{(a)}{=} \inf_{\boldsymbol{\mu}^1 \in \mathcal{M}_{\mathcal{T}_1} \,:\, \boldsymbol{\mu}^1_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \{(\boldsymbol{\theta}_1 + \bar{\boldsymbol{u}}_1)^\top \boldsymbol{\mu}^1\} + ... + \inf_{\boldsymbol{\mu}^m \in \mathcal{M}_{\mathcal{T}_m} \,:\, \boldsymbol{\mu}^m_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \{(\boldsymbol{\theta}_m + \bar{\boldsymbol{u}}_m)^\top \boldsymbol{\mu}^m\} \\
&\overset{(b)}{=} \inf_{\boldsymbol{\mu}^1 \in \text{ conv } \{\boldsymbol{\mu} \in \mathcal{X}_{\mathcal{T}_1} \,:\, \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\}} \{(\boldsymbol{\theta}_1 + \bar{\boldsymbol{u}}_1)^\top \boldsymbol{\mu}^1\} + ... + \inf_{\boldsymbol{\mu}^m \in \text{ conv } \{\boldsymbol{\mu} \in \mathcal{X}_{\mathcal{T}_m} \,:\, \boldsymbol{\mu}_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}\}} \{(\boldsymbol{\theta}_m + \bar{\boldsymbol{u}}_m)^\top \boldsymbol{\mu}^m\} \\
&\overset{(c)}{=} \inf_{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1} \,:\, \boldsymbol{\mu}^1_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \{(\boldsymbol{\theta}_1 + \bar{\boldsymbol{u}}_1)^\top \boldsymbol{\mu}^1\} + ... + \inf_{\boldsymbol{\mu}^m \in \mathcal{X}_{\mathcal{T}_m} \,:\, \boldsymbol{\mu}^m_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \{(\boldsymbol{\theta}_m + \bar{\boldsymbol{u}}_m)^\top \boldsymbol{\mu}^m\} \\
&= \inf_{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in \mathcal{X}_{\mathcal{T}_m} \,:\, \boldsymbol{\mu}^j_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \left\{ \sum_{j=1}^{m} (\boldsymbol{\theta}_j + \bar{\boldsymbol{u}}_j)^\top \boldsymbol{\mu}^j \right\} \\
&= \inf_{\boldsymbol{\mu}^1 \in \mathcal{X}_{\mathcal{T}_1}, ..., \boldsymbol{\mu}^m \in \mathcal{X}_{\mathcal{T}_m} \,:\, \boldsymbol{\mu}^j_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}} \mathcal{L}(\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_m, \bar{\boldsymbol{u}})
\end{aligned}
$$

where the step in (a) holds because for every tree-structured MRF $\mathcal{T}_j$ the marginal polytope $\mathcal{M}_{\mathcal{T}_j}$ coincides with the local consistency polytope $L_{\mathcal{T}_j}$; in step (b) we use $\mathcal{M}_{\mathcal{T}_j} = \text{conv } \mathcal{X}_{\mathcal{T}_j}$ and Lemma 11; finally, the step in (c) holds because a linear objective over a polytope always achieves its optimum at least at one of the extreme points (that is, corners) of the latter. □

Note that the agreement on the integral part holds also for edge marginals, that is, for every dimensions in $\boldsymbol{\mu}^*$ with an integral value, even if the nodes of an edge are fractional. Namely, we ca extend the constraint $\bar{\boldsymbol{\mu}}^j_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}$ in Theorem 4 to every dimension having an integral value and the proof still works.

B.4 Proof of Lemma 2

We denote by $\boldsymbol{\mu}^*|_{\mathcal{T}_j}$ a projection of a solution $\boldsymbol{\mu}^*$ over a graph $\mathcal{G}$ to a subspace corresponding to a subtree $\mathcal{T}_j$. Since strong duality holds for the problem (4.8) any primal optimal is a minimizer of the Lagrangian. Therefore, each restriction $\boldsymbol{\mu}^*|_{\mathcal{T}_j}$ is a minimizer of a corresponding subproblem over tree $\mathcal{T}_j$. Furthermore, Theorem 4 guarantees an existence of a minimizer $\bar{\boldsymbol{\mu}}^j$ that agrees with the integral part of $\boldsymbol{\mu}^*|_{\mathcal{T}_j}$ and differs from $\boldsymbol{\mu}^*$ only on the fractional entries. Note that this holds also for edge marginals. Namely, in the proof of Theorem 4 we can extend the constraints $\bar{\boldsymbol{\mu}}^j_{\mathcal{I}} = \boldsymbol{\mu}^*_{\mathcal{I}}$ to every dimension in $\boldsymbol{\mu}^*$ having an integral value (including edge marginals) and the proof still works. Any point on the line through these two solutions ($\boldsymbol{\mu}^*|_{\mathcal{T}_j}$ and $\bar{\boldsymbol{\mu}}^j$) is also optimal since the corresponding objective is linear. We now show an existence of a corresponding solution $\hat{\boldsymbol{\mu}}^j$ by construction. We define

$$
\hat{\mu}^j_i(x_i) := \begin{cases} \mu^*_i(x_i), & \text{if } i \in \mathcal{I} \\ 1 - \bar{\mu}^j_i(x_i), & \text{otherwise.} \end{cases} \tag{B.2}
$$

for each $i \in \mathcal{V}_j$ and

$$\hat{\mu}^j_{i,k}(x_i, x_k) := \begin{cases} \mu^*_{i,k}(x_i, x_k), & \text{if } \mu^*_{i,k}(x_i, x_k) \in \{0, 1\} \\ 1 - \bar{\mu}^j_{i,k}(x_i, x_k), & \text{if } \mu^*_{i,k}(x_i, x_k) = 0.5 \end{cases} \tag{B.3}$$

for each $(i, k) \in \mathcal{E}_j$. It is easy to see that the above definition of $\hat{\boldsymbol{\mu}}^j$ satisfies the equation (4.12). Furthermore, $\hat{\boldsymbol{\mu}}^j$ lies on the line through $\bar{\boldsymbol{\mu}}^j$ and the restriction $\boldsymbol{\mu}^*|_{\mathcal{T}_j}$ and is therefore optimal. We now show that it is feasible, that is, $\hat{\boldsymbol{\mu}}^j \in \mathcal{X}_{\mathcal{T}_j}$. Note that the variables $\hat{\boldsymbol{\mu}}^j$ are either equal to the variables in $\boldsymbol{\mu}^*_{\mathcal{I}}$ or have an opposite value to the variables in $\bar{\boldsymbol{\mu}}^j$. Therefore, they inherit the integrality constraints as well as the normalization constraints from $\boldsymbol{\mu}^*$ and $\bar{\boldsymbol{\mu}}^j$. Similar argument can be used for the marginalization constraints. More precisely, for the integral edges, where both end nodes are integral, the marginalization constraints hold true. We only need to check the cases with non integral edges. This can be done by considering all the cases listed in Lemma 12. We show exemplary one case – the remaining cases are straightforward. In the following we drop the superscript j denoting the subproblem and write only $\hat{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{\mu}}$. Consider the case (a) from Lemma 12. First, we have $\mu^*_{i,j}(0,0) = \mu^*_{i,j}(1,1) = 0.5$ and $\mu^*_{i,j}(0,1) = \mu^*_{i,j}(1,0) = 0$. That is, $\bar{\mu}_{i,j}(0,1) = \bar{\mu}_{i,j}(1,0) = 0$. Without loss of generality assume $\bar{\mu}_{i,j}(0,0) = 1$ and $\bar{\mu}_{i,j}(1,1) = 0$, that is, $\bar{\mu}_i(0) = \bar{\mu}_j(0) = 1$ and $\bar{\mu}_i(1) = \bar{\mu}_j(1) = 0$. Due to the construction in (B.2) we get $\hat{\mu}_i(0) = \hat{\mu}_j(0) = 0$ and $\hat{\mu}_i(1) = \hat{\mu}_j(1) = 1$. This corresponds to $\hat{\mu}_{i,j}(0,0) = \hat{\mu}_{i,j}(0,1) = \hat{\mu}_{i,j}(1,0) = 0$ and $\hat{\mu}_{i,j}(1,1) = 1$ which is exactly what we get from construction in (B.3). Therefore, we get a valid labeling of an edge and the marginalization constraints are satisfied. Finally, note that the equality $\boldsymbol{\mu}^*|_{\mathcal{T}_j} = \frac{1}{2}(\bar{\boldsymbol{\mu}}^j + \hat{\boldsymbol{\mu}}^j)$ also holds (including the edge marginals). □

B.5 Proof of Theorem 5

Let $\boldsymbol{\mu}^*$ be a unique optimal solution of the LP relaxation (4.5). We use the notation $\boldsymbol{\mu}^*|_{\mathcal{T}_j}$ to denote a reduction of $\boldsymbol{\mu}^*$ to a corresponding subtree. Theorem 4 guarantees an existence of minimizers $\bar{\boldsymbol{\mu}}^1 \in \mathcal{X}_{\mathcal{T}_1}, ..., \bar{\boldsymbol{\mu}}^m \in \mathcal{X}_{\mathcal{T}_m}$ of a corresponding Lagrangian $\mathcal{L}(\cdot, ..., \cdot, \bar{\boldsymbol{u}})$ where each $\bar{\boldsymbol{\mu}}^j$ agrees with the integral part of $\boldsymbol{\mu}^*|_{\mathcal{T}_j}$. We now assume that there is another minimizer $\hat{\boldsymbol{\mu}}^j$ for the j-th subproblem with $\bar{\boldsymbol{\mu}}^j_i(x_i) \neq \hat{\boldsymbol{\mu}}^j_i(x_i)$ for some $i \in \mathcal{I}$, $x_i \in S$ and show that this assumption leads to a contradiction. We do this by constructing another optimal solution of the LP relaxation different from $\boldsymbol{\mu}^*$.

Assume for simplicity a decomposition over individual edges. We consider the following relabeling procedure starting with an edge (i, k) corresponding to the j-th subproblem above. Since $\hat{\boldsymbol{\mu}}^j$ and $\bar{\boldsymbol{\mu}}^j$ both are minimizers for the corresponding subproblem, the average $\tilde{\boldsymbol{\mu}}^j := \frac{1}{2}(\hat{\boldsymbol{\mu}}^j + \bar{\boldsymbol{\mu}}^j)$ is also a minimizer (because the objective is linear) and $\tilde{\boldsymbol{\mu}}^j_i(x_i) = 0.5$ for $x_i \in S$. That is, the i-th node is now assigned with a fractional label 0.5. The remaining nodes x_r ($r \neq k$) adjacent to x_i can be relabeled in a consistent way to $x_i = 0.5$ such that a corresponding assignment $(0.5, x_r)$ is optimal for the edge (i, r) by using the weak tree agreement property[1]. Namely, since there are two optimal assignments for the edge (i, k) with both values for i, for every adjacent edge (i, k) there must also be optimal assignments with both values for i. Therefore, we can define a new labeling for each edge adjacent to i by computing

[1]Optimal assignments obtained via DD are known to satisfy the weak tree agreement (WTA) condition [KPT11]. In particular, for our purposes we use the following fact. Consider any two trees $\mathcal{T}_i$ and $\mathcal{T}_j$ which share a node x_k. Then for any optimal configuration $\boldsymbol{\mu}^i$ there exists an optimal configuration $\boldsymbol{\mu}^j$ with $\mu^i_k(x_k) = \mu^j_k(x_k)$.

the average of the corresponding assignments. During this procedure some nodes x_k can change their label. Note that this is possible only for nodes with integral value in $\boldsymbol{\mu}^*$. To validate this claim consider a fractional node x_k. Since x_i is integral in $\boldsymbol{\mu}^*$, there must be (due to lemma 2) optimal assignments $(x_i^*, 0)$ and $(x_i^*, 1)$ for edge (i, k), where x_i^* is the optimal label of x_i according to $\boldsymbol{\mu}^*$. Furthermore, because of $x_i = 0.5$ (due to relabeling $\tilde{\boldsymbol{\mu}}^j$) there also must be an optimal assignment for that edge of the form $(1 - x^*, 0)$ or $(1 - x^*, 1)$. In any case we can find an optimal average such that $x_i = x_k = 0.5$. That is, the value of fractional x_k does not change!

If a node x_k changes his label to 0.5 during this procedure, we then need to consider all its neighbors (except x_i) and proceed with the relabeling process. More precisely, we have the following cases:
We now consider an edge (i, k) where x_i has been relabeled to 0.5 in previous steps.
Case 1: $I \to I$
That is, x_i and x_k both have an integral value in $\boldsymbol{\mu}^*$.
(a) x_k does not change by computing a corresponding average, then there is nothing more to do.
(b) x_k changes. We label it with 0.5 and consider all adjacent cases (except x_i).
Case 2: $I \to F$
That is, x_i is integral in $\boldsymbol{\mu}^*$ and x_k is fractional. There must be (due to lemma 2) optimal assignments $(x_i^*, 0)$ and $(x_i^*, 1)$. Because $x_i = 0.5$ now there must be (due to WTA) an optimal assignment $(1 - x_i^*, 0)$ or $(1 - x_i^*, 1)$ such that a corresponding optimal average results in $x_i = x_k = 0.5$. So the label of x_k does not change.
Case 3: $I \to I/F$
That is, x_k has an integral value in $\boldsymbol{\mu}^*$ but has been relabeled to 0.5 previously. Due to the WTA there are always assignments such that a corresponding average results in $x_i = x_k = 0.5$.
Since only integral nodes can change their label during the above relabeling procedure, there are no other cases to consider. The relabeling procedure terminates with a new consistent joint labeling $\tilde{\boldsymbol{\mu}}$ different from $\boldsymbol{\mu}^*$. We can prove the statement for arbitrary tree decompositions (not only over edges) by using similar arguments. □

B.6 On the fractional solutions of LP relaxation

For binary pairwise MRFs the LP relaxation has the property that in every (extreme) optimal solution each fractional node is half integral ([DL97; Son10]). Furthermore, each edge marginal is either integral or has fractional values. More precisely, an edge marginal is integral only if both end nodes are integral. In fact, there are six further cases for fractional edge marginals as specified in the following lemma.

Lemma 12. *Each edge marginal $\mu_{i,j}(x_i, x_j)$ is either integral (if both end nodes x_i and x_j are integral) or*
(a) is equal to

$\mu_{i,j}(x_i, x_j)$	$x_j = 0$	$x_j = 1$
$x_i = 0$	0.5	0
$x_i = 1$	0	0.5

or

$\mu_{i,j}(x_i, x_j)$	$x_j = 0$	$x_j = 1$
$x_i = 0$	0	0.5
$x_i = 1$	0.5	0

if both x_i and x_j are fractional;
(b) is equal to

$\mu_{i,j}(x_i, x_j)$	$x_j = 0$	$x_j = 1$
$x_i = 0$	0.5	0
$x_i = 1$	0.5	0

or

$\mu_{i,j}(x_i, x_j)$	$x_j = 0$	$x_j = 1$
$x_i = 0$	0	0.5
$x_i = 1$	0	0.5

if x_i is fractional and x_j is integral ($x_j = 0$ on the left and $x_j = 1$ on the right);
(c) is equal to

$\mu_{i,j}(x_i, x_j)$	$x_j = 0$	$x_j = 1$
$x_i = 0$	0.5	0.5
$x_i = 1$	0	0

or

$\mu_{i,j}(x_i, x_j)$	$x_j = 0$	$x_j = 1$
$x_i = 0$	0	0
$x_i = 1$	0.5	0.5

if x_j is fractional and x_i is integral ($x_i = 0$ on the left and $x_i = 1$ on the right);

Proof. The integral case is clear. We now assume that a given edge is non integral, that is, at least one of the nodes is fractional. First we show that in every case a matrix corresponding to an edge assignment contains only two different values a and b.
Case (a):
Since every feasible solution $\boldsymbol{\mu} \in L_G$ is subject to the marginalization constraints $\sum_{x_i} \mu_{i,j}(x_i, x_j) = \mu_j(x_j)$ and $\sum_{x_j} \mu_{i,j}(x_i, x_j) = \mu_i(x_i)$ the following equations must hold

$$\begin{aligned} \mu_{i,j}(0,0) + \mu_{i,j}(0,1) &= \mu_i(0) \\ \mu_{i,j}(1,0) + \mu_{i,j}(1,1) &= \mu_i(1) \\ \mu_{i,j}(0,0) + \mu_{i,j}(1,0) &= \mu_j(0) \\ \mu_{i,j}(0,1) + \mu_{i,j}(1,1) &= \mu_j(1) \end{aligned} \tag{B.4}$$

Due to $\mu_i(0) = \mu_i(1) = \mu_j(0) = \mu_j(1) = 0.5$ it follows from (B.4) that $a := \mu_{i,j}(0,0) = \mu_{i,j}(1,1)$ and $b := \mu_{i,j}(0,1) = \mu_{i,j}(1,0)$. Now we argue that $a, b \in \{0, 0.5\}$. For this purpose assume that the edge marginal $\mu_{i,j}(x_i, x_j)$ contains other than half-integral values. So w.l.o.g. let $a \in (0, 0.5)$, then also $b \in (0, 0.5)$ (otherwise $a + b \neq 0.5$). We now define two different feasible solutions $\boldsymbol{\mu}^1$ and $\boldsymbol{\mu}^2$ which have the same entries as $\boldsymbol{\mu}$ except the entries for the marginal $\mu_{i,j}(x_i, x_j)$, which we define for $\boldsymbol{\mu}^1$ by $a_1 := a + \epsilon$, $b_1 := b - \epsilon$ and for $\boldsymbol{\mu}^2$ by $a_2 := a - \epsilon$ and $b_2 := b + \epsilon$, where ϵ is small enough such that $a_1, a_2, b_1, b_2 \in (0, 0.5)$. Furthermore, due to $a_1 + b_1 = a_2 + b_2 = 0.5$ a corresponding edge assignment is feasible, and therefore the solutions $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$. Since $\boldsymbol{\mu} = \frac{1}{2}(\boldsymbol{\mu}^1 + \boldsymbol{\mu}^2)$, the solution $\boldsymbol{\mu}$ is a convex combination of two different feasible solutions, and is therefore not extreme contradicting our assumption that $\boldsymbol{\mu}$ is a corner of the local polytope. So it must hold $a, b \in \{0, 0.5\}$. Finally, $a = 0$ implies $b = 0.5$ and vice versa due to $a + b = 0.5$. The remaining cases in (b) and (c) can be dealt with by using similar arguments as above. □

B.7 Proof of Lemma 5

The following holds:

$$\begin{aligned} g_{4.18}(u) =& \inf_{\boldsymbol{\mu}} \left\{ \boldsymbol{\theta}^\top \boldsymbol{\mu} + u((\boldsymbol{a}_1 - t\boldsymbol{a}_2)^\top \boldsymbol{\mu} - t \cdot d_2 + d_1) \right\} \\ =& \inf_{\boldsymbol{\mu}} \left\{ \boldsymbol{\theta}^\top \boldsymbol{\mu} + \begin{bmatrix} u \\ -t \cdot u \end{bmatrix}^\top (A\boldsymbol{\mu} - \boldsymbol{b}) \right\} \\ =& g_{4.14}((u, -tu)^\top) \end{aligned}$$

where in the second line we use:

$$\begin{bmatrix} u \\ -t \cdot u \end{bmatrix}^\top A\boldsymbol{\mu} = u(\boldsymbol{a}_1 - t\boldsymbol{a}_2)^\top \boldsymbol{\mu}$$

and

$$\begin{aligned} & t = (b_1 + d_1)/(b_2 + d_2) \\ \Rightarrow\ & d_1 - t \cdot d_2 = t \cdot b_2 - b_1 \\ \Rightarrow\ & u(d_1 - t \cdot d_2) = u(t \cdot b_2 - b_1) \\ \Rightarrow\ & u(d_1 - t \cdot d_2) = - \begin{bmatrix} u \\ -t \cdot u \end{bmatrix}^\top \boldsymbol{b} \end{aligned}$$

Finally,

$$\max_{u \in \mathbb{R}} g_{4.18}(u) = \max_{u \in \mathbb{R}} g_{4.14}((u, -tu)^\top) \leqslant \max_{\boldsymbol{u} \in \mathbb{R}^2} g_{4.14}(\boldsymbol{u})$$

implies: $d^*_{4.18} \leqslant d^*_{4.14}$. □

B.8 Proof of Lemma 8

The following holds:

$$\begin{aligned} g_{4.21}(\boldsymbol{u}) =& \inf_{\boldsymbol{\mu}, \boldsymbol{z}} \left\{ \boldsymbol{\theta}^\top \boldsymbol{\mu} + f(\boldsymbol{z}) + \boldsymbol{u}^\top (A\boldsymbol{\mu} - \boldsymbol{z}) \right\} \\ =& \inf_{\boldsymbol{\mu}, \boldsymbol{\nu}} \left\{ \boldsymbol{\theta}^\top \boldsymbol{\mu} + f(A\boldsymbol{\nu}) + \boldsymbol{u}^\top (A\boldsymbol{\mu} - A\boldsymbol{\nu}) \right\} \\ =& \inf_{\boldsymbol{\mu}, \boldsymbol{\nu}} \left\{ \boldsymbol{\theta}^\top \boldsymbol{\mu} + f(A\boldsymbol{\nu}) + (A^\top \boldsymbol{u})^\top (\boldsymbol{\mu} - \boldsymbol{\nu}) \right\} \\ =& g_{4.20}(A^\top \boldsymbol{u}) \end{aligned}$$

where in the second row we perform variable substitution $\boldsymbol{z} \to A\boldsymbol{\nu}$. Finally,

$$\max_{\boldsymbol{u} \in \mathbb{R}^r} g_{4.21}(\boldsymbol{u}) = \max_{\boldsymbol{u} \in \mathbb{R}^r} g_{4.20}(A^\top \boldsymbol{u}) \leqslant \max_{\hat{\boldsymbol{u}} \in \mathbb{R}^d} g_{4.20}(\hat{\boldsymbol{u}})$$

implies $d^*_{4.21} \leqslant d^*_{4.20}$, where in the last step the inequality holds since we are maximizing over a bigger set. Note that if A has a full rank, that is, A is quadratic and all columns are linearly independent, then the both formulations are equivalent. □

B.9 Proof of Lemma 9

The following holds:

$$\begin{aligned} g_{4.24}(A^\top \boldsymbol{u}) =& \inf_{\boldsymbol{\mu}, \boldsymbol{\nu}} \left\{ \boldsymbol{\theta}^\top \boldsymbol{\mu} + \mathbb{1}_\infty[A\boldsymbol{\nu} = \boldsymbol{b}] + (A^\top \boldsymbol{u})^\top (\boldsymbol{\mu} - \boldsymbol{\nu}) \right\} \\ =& \inf_{\boldsymbol{\mu}, \boldsymbol{\nu} \colon A\boldsymbol{\nu} = \boldsymbol{b}} \left\{ \boldsymbol{\theta}^\top \boldsymbol{\mu} + \boldsymbol{u}^\top (A\boldsymbol{\mu} - A\boldsymbol{\nu}) \right\} \\ =& \inf_{\boldsymbol{\mu}} \left\{ \boldsymbol{\theta}^\top \boldsymbol{\mu} + \boldsymbol{u}^\top (A\boldsymbol{\mu} - \boldsymbol{b}) \right\} \\ =& g_{4.14}(\boldsymbol{u}) \end{aligned}$$

Furthermore,

$$\max_{\boldsymbol{u} \in \mathbb{R}^r} g_{4.14}(\boldsymbol{u}) = \max_{\boldsymbol{u} \in \mathbb{R}^r} g_{4.24}(A^\top \boldsymbol{u}) \leqslant \max_{\hat{\boldsymbol{u}} \in \mathbb{R}^d} g_{4.24}(\hat{\boldsymbol{u}})$$

implies $d^*_{4.14} \leqslant d^*_{4.24}$. □

Appendix C

Supplements (Chapter 5)

C.1 Simulation of Algorithm 5

Figure C.1 provides a simulation of Algorithm 5 for the case "$\odot = \cdot$", $M = 4$, $N = 3$ and $\boldsymbol{y}^* = (3, 2, 1, 3)$. It is illustrated, how the variables $L_{t,k}(j)$ and $S_{t,k}(j)$ evolve over the iterations $t = 1, ..., M$. The first block illustrates the initialization. Each block corresponding to a value $t \in \{2, 3, 4\}$ in the recursion part represents one iteration step of the outer **for**-loop in lines 2-10 and depicts the content of the variable S represented by the edges between nodes with red edges marking the decision in a current iteration step: the nodes $(i, t-1)$ and (j, t) belonging to the same loss value $k \in \{1, 2, 3, 4\}$ are connected by an edge if and only if $S_{t,k}(j) = i$ [1]. For example, after the second iteration (t = 2) we have : $S_{2,1}(1) = 3$, $S_{2,1}(2) = 1$, $S_{2,1}(3) = 3$, $S_{2,2}(1) = 1$ and $S_{2,2}(3) = 2$. The corresponding factors $f_t(i, j)$ are represented in a matrix form: for example $f_2(1, 2) = 5$ and $f_2(2, 1) = 2$. The circled numbers represent the current values $L_{t,k}(j)$, where the values of the missing circles are not defined (initialized with $-\infty$). The following example illustrates how the recursion rule in (5.15) is applied during the simulation in Figure C.1.

Example 1. *After the initialization part we have a situation, where* $L_{1,1}(1) = 2$, $L_{1,1}(2) = 1$. *In the first iteration of the outer* ***for****-loop (t = 2) we get the following values of the variables* $L_{t,k}(j)$ *and* $S_{t,k}(j)$*:*

Case 1: $(t, j, k) = (2, 1, 1)$ *i.e.* $j \neq y_t^*$

$L_{2,1}(1) = f_1(3) + f_2(3, 1) = 3 + 1 = 4$

$S_{2,1}(1) = 3$

Case 2: $(t, j, k) = (2, 1, 2)$ *i.e.* $j \neq y_t^*$

$L_{2,2}(1) = \max\limits_{1 \leqslant i \leqslant N} L_{1,1}(i) + f_2(i, 1) = L_{1,1}(1) + f_2(1, 1) = 2 + 4 = 6$

$S_{2,2}(1) = 1$

Case 3: $(t, j, k) = (2, 2, 1)$ *i.e.* $j = y_t^*$

$L_{2,1}(2) = \max\limits_{1 \leqslant i \leqslant N} L_{1,1}(i) + f_2(i, 2) = L_{1,1}(1) + f_2(1, 2) = 2 + 5 = 7$

$S_{2,1}(2) = 1$

Case 4: $(t, j, k) = (2, 3, 1)$ *i.e.* $j \neq y_t^*$

$L_{2,1}(3) = f_1(3) + f_2(3, 3) = 3 + 3 = 6$

$S_{2,1}(3) = 3$

[1] We omitted the value $k = 0$ in Figure C.1 for simplicity, since for that case the algorithm constructs only one path reproducing the true sequence $\boldsymbol{y}^*$.

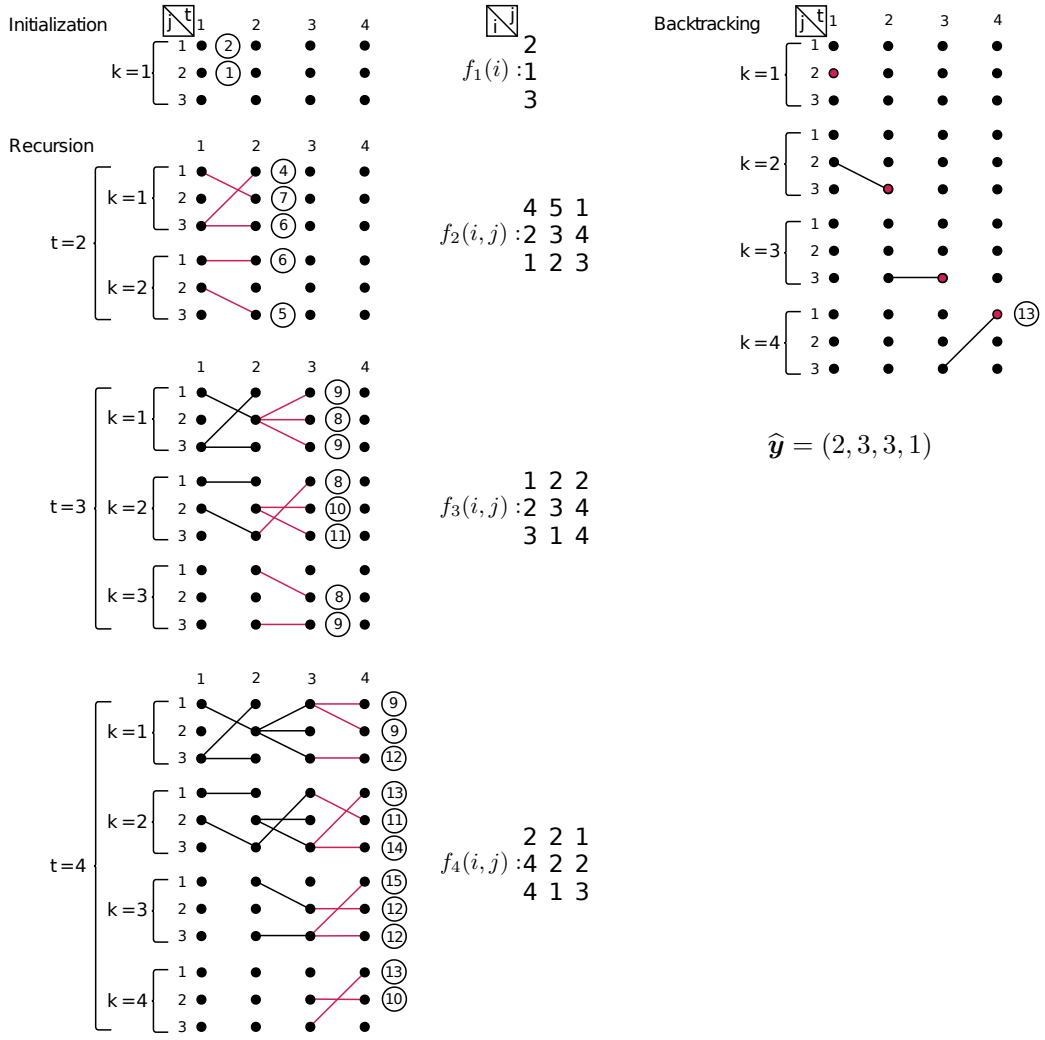


Figure C.1: © 2014 IEEE. Simulation of Algorithm 5 for the case "$\odot = \cdot$", $M = 4$, $N = 3$ and $\boldsymbol{y}^* = (3,2,1,3)$. The left part of the first column illustrates the values $S_{t,k}(j)$, where the edges between nodes denote the predecessor-relationship. The red edges mark the decision in a current iteration step and the circled numbers correspond to the values $L_{t,k}(j)$. In the right part of of the first column the values of factors $f_t(i,j)$ are shown in a matrix form. The second column illustrates the back-tracking part of the simulation, where the red nodes mark an optimal solution $\widehat{\boldsymbol{y}} = (2,3,3,1)$. A corresponding optimal compatibility score is 13 yielding the optimal value $L^* = 13 \cdot 4 = 13 \cdot 4 = 52$.

Case 5: $(t,j,k) = (2,3,2)$ *i.e.* $j \neq y_t^*$

$$L_{2,2}(3) = \max_{1 \leqslant i \leqslant N} L_{1,1}(i) + f_2(i,3) = L_{1,1}(2) + f_2(2,3) = 1 + 4 = 5$$

$$S_{2,2}(3) = 2$$

The remaining iterations ($t = 3$ and $t = 4$) are handled in a similar way. The corresponding back-tracking part yields an optimal solution $\widehat{y} = (2,3,3,1)$ with the optimal compatibility score 13*. The resulting optimal value is $L^* = 13 \cdot 4 = 52$.*

C.2 Simulation of Algorithm 7

A conventional chart parser computes the optimal parse score for a given sentence by filling cells of a table ($\sim \Pi_{i,j,A}^{tp,fp}$), called chart or well-formed substring table, with

edges (i.e. data structures encoding information about a particular parsing step)

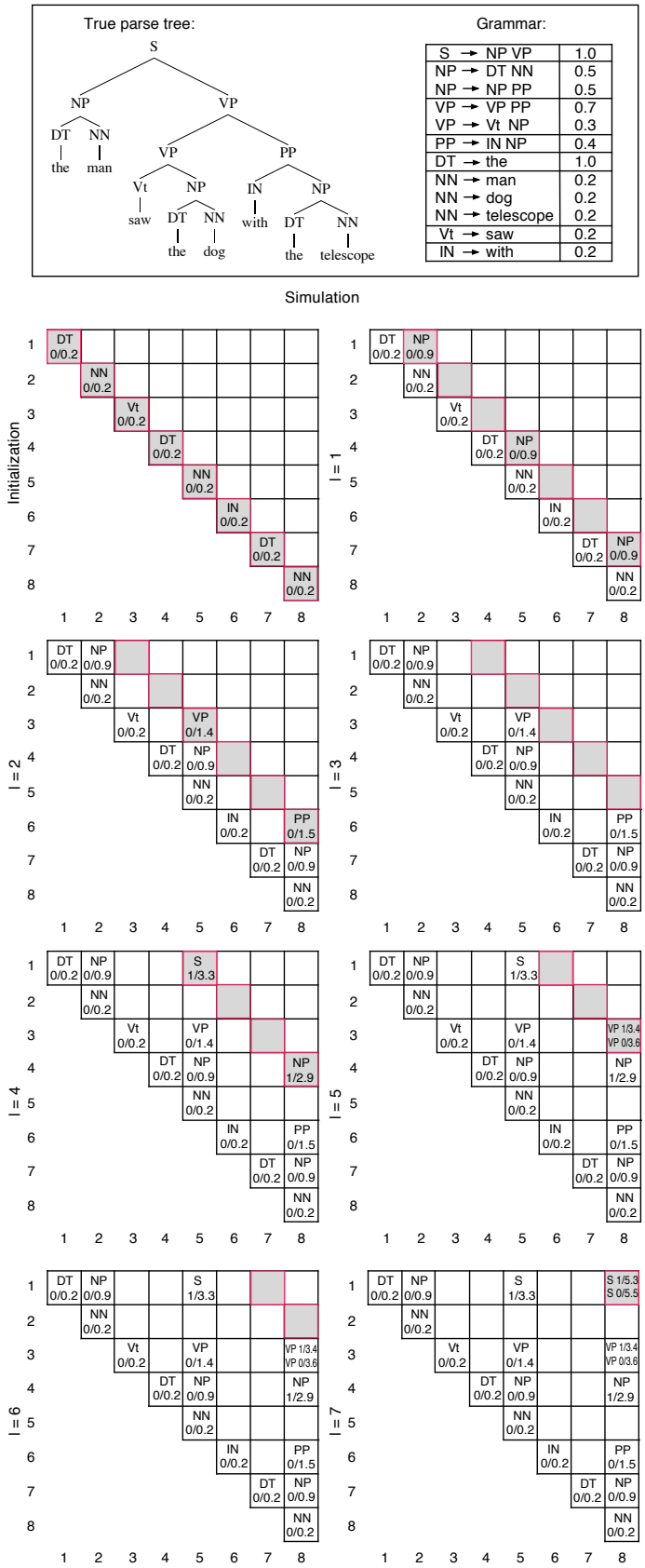


Figure C.2: © 2017 IEEE. Simulation of Algorithm 7: $\odot = \cdot$, $|\boldsymbol{x}| = 8$, and Δ counting the number of nodes in a prediction missing in the true parse tree.

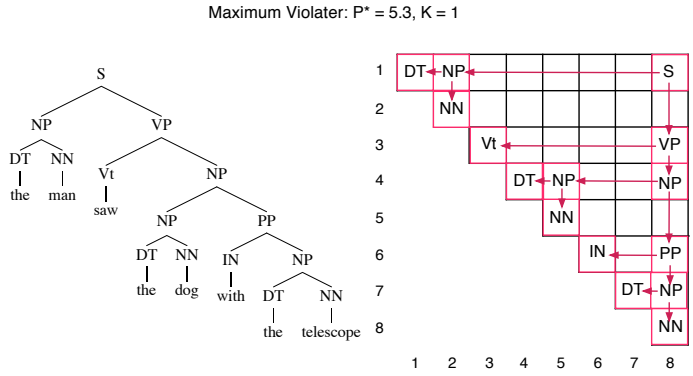


Figure C.3: © 2017 IEEE. Solution of the simulation of Algorithm 7. The maximum violating tree has the optimal value $p^* = 5.3$ and corresponds to the loss value $K = 1$ w.r.t. the true output tree.

according to some filling strategy where each such table can be represented as one triangular half of a two-dimensional array. We present in Figure C.2 a simulation of Algorithm 7 for the case $\odot = \cdot$, $|\boldsymbol{x}| = 8$ and loss function Δ counting the number of nodes in a prediction tree, which are missing in the true parse tree. In the first row the true output tree (on the left) is given with a grammar description (on the right) containing a list of production rules with corresponding weights. Each triangular table corresponds to one iteration of the most outer loop in lines 6–14, which iterates over the length $l = 1, ..., |\boldsymbol{x}| - 1$ of the token spans. The individual cells are indexed by the corresponding span boundaries i, j and filled with info corresponding to the values $\Pi^k_{i,j,A}$. Here we use the notation — $A\ k/\Pi^k_{i,j,A}$ — to represent the values $\Pi^k_{i,j,A}$. For example, the value $\Pi^0_{4,5,\text{NP}} = 0.9$ is presented in the cell with coordinates (4,5) by NP $0/0.9$.

In the following we exemplary present the computations made in one iteration corresponding to $l = 4$ for the algorithm simulation in Figure C.2. Note, that although in our simple example each table cell contains at most two elements, usually there are much more.

Example 2 (Iteration $l = 4$ of the simulation in Figure C.2)**.** *In this iteration only two new values are produced. In particular, in each of these two cases the set we are maximizing over (in the formula in (5.23)) contains only one element.*

Case 1: $(i, j, k) = (1, 5, 1)$

$$\begin{aligned}\Pi^1_{1,5,S} &= q(S \to NP\ VP) + \Pi^{1-\mathbb{1}_1[[S]_{1,5} \notin \boldsymbol{y}^*]-0}_{1,2,NP} + \Pi^0_{3,5,VP} \\ &= 1.0 + 0.9 + 1.4 = 3.3\end{aligned}$$

Case 2: $(i, j, k) = (4, 8, 1)$

$$\begin{aligned}\Pi^1_{4,8,NP} &= q(NP \to NP\ PP) + \Pi^{1-\mathbb{1}_1[[NP]_{4,8} \notin \boldsymbol{y}^*]-0}_{4,5,NP} + \Pi^0_{6,8,PP} \\ &= 0.5 + 0.9 + 1.5 = 2.9\end{aligned}$$

The corresponding solution, which we obtain by back-tracing the optimal decisions of the intermediate computations stored in $S^k_{i,j,\cdot}$ is given in Figure C.3.

Bibliography

[AW09] Kurt M. Anstreicher and Laurence A. Wolsey. “Two "well-known" properties of subgradient optimization”. In: *Math. Program.* 120.1 (2009), pp. 213–220.

[Bak+07] Gökhan Bakir et al. *Predicting Structured Data*. The MIT Press, 2007.

[Bat+12] Dhruv Batra et al. “Diverse M-Best Solutions in Markov Random Fields”. In: *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*. 2012, pp. 1–16.

[Bau+14] Alexander Bauer et al. “Efficient Algorithms for Exact Inference in Sequence Labeling SVMs”. In: *IEEE Trans. Neural Netw. Learning Syst.* 25.5 (2014), pp. 870–881.

[Bau+17a] Alexander Bauer et al. “Optimizing Measure of Performance in Max-Margin Parsing”. In: *arXiv preprint arXiv:submit/1997771* (2017).

[Bau+17b] Alexander Bauer et al. “Partial Optimality of Dual Decomposition for MAP Inference in Pairwise MRFs”. In: *arXiv preprint arXiv:1708.03314* (2017).

[BBM17] Alexander Bauer, Mikio L. Braun, and Klaus-Robert Müller. “Accurate Maximum-Margin Training for Parsing With Context-Free Grammars”. In: *IEEE Trans. Neural Netw. Learning Syst.* 28.1 (2017), pp. 44–56.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[BJ01] Yuri Boykov and Marie-Pierre Jolly. “Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images”. In: *ICCV*. 2001, pp. 105–112.

[BM09] José Bento and Andrea Montanari. “Which graphical models are difficult to learn?” In: *CoRR* abs/0910.5761 (2009).

[BMK12] Alexander Binder, Klaus-Robert Müller, and Motoaki Kawanabe. “On Taxonomies for Multi-class Image Categorization”. In: *International Journal of Computer Vision* 99.3 (2012), pp. 281–301.

[BNM16] Alexander Bauer, Shinichi Nakajima, and Klaus-Robert Müller. “Efficient Exact Inference With Loss Augmented Objective in Structured Learning”. In: *IEEE Trans. Neural Netw. Learning Syst.; In Press* (2016).

[Bod+11] Nathan Bodenstab et al. “Beam-Width Prediction for Efficient Context-Free Parsing”. In: *Proc. 49th ACL*. 2011, pp. 440–449.

[Bod93] H. Bodlaender. “A Tourist Guide Through Treewidth”. In: *Acta Cybernetica* 11.1–2 (1993).

[Bor+07] Antoine Bordes et al. “Solving multiclass support vector machines with LaRank”. In: *Proc. 24th ICML*. Corvallis, Oregon, USA, 2007, pp. 89–96.

[BSS15] P. Balamurugan, Shirish Krishnaj Shevade, and S. Sundararajan. "A Simple Label Switching Algorithm for Semisupervised Structural SVMs". In: *Neural Computation* 27.10 (2015), pp. 2183–2206.

[BT97] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*. Athena scientific series in optimization and neural computation. Belmont (Mass.): Athena Scientific, 1997.

[BV06] Yuri Boykov and Olga Veksler. "Graph Cuts in Vision and Graphics: Theories and Applications". In: *Handbook of Mathematical Models in Computer Vision*. 2006, pp. 79–96.

[CC04] Stephen Clark and James R. Curran. "Parsing the WSJ Using CCG and Log-Linear Models". In: *Proc. 42nd ACL*. Barcelona, Spain, 2004, pp. 103–110.

[CC12] Shay B. Cohen and Michael Collins. "Tensor Decomposition for Fast Parsing with Latent-Variable PCFGs". In: *Proc. 25th NIPS*. Nevada, USA, 2012, pp. 2528–2536.

[CCK08] Xavier Carreras, Michael Collins, and Terry Koo. "TAG, Dynamic Programming, and the Perceptron for Efficient, Feature-Rich Parsing". In: *Proc. 12th CoNLL*. Manchester, UK, 2008, pp. 9–16.

[Cha+10] Ming-Wei Chang et al. "Discriminative Learning over Constrained Latent Representations". In: *Proc. 11th NAACL HLT*. 2010, pp. 429–437.

[Cha00] Eugene Charniak. "A Maximum-Entropy-Inspired Parser". In: *6th Applied Natural Language Processing Conference, ANLP 2000, Seattle, Washington, USA, April 29 - May 4, 2000*. 2000, pp. 132–139.

[Cha93] Eugene Charniak. *Statistical Language Learning*. Cambridge, MA: The MIT Press, 1993.

[Cha97] Eugene Charniak. "Statistical Parsing with a Context-Free Grammar and Word Statistics". In: *Proc. 40th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence*. 1997, pp. 598–603.

[Cho55] Noam Chomsky. *The logical structure of linguistic theory*. MIT Humanities Library, 1955.

[Cho57] Noam Chomsky. *Syntactic Structures*. The Hague: Mouton, 1957.

[CK05] Michael Collins and Terry Koo. "Discriminative Reranking for Natural Language Parsing". In: *Computational Linguistics* 31.1 (2005), pp. 25–70.

[Col01] Michael Collins. "Parameter Estimation for Statistical Parsing Models: Theory and Practice of Distribution-Free Methods". In: *Proc. 7th International Workshop on Parsing Technologies*. Beijing, China, 2001.

[Col02] Michael Collins. "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms". In: *Proc. EMNLP*. 2002.

[Col03] Michael Collins. "Head-Driven Statistical Models for Natural Language Parsing". In: *Computational Linguistics* 29.4 (2003), pp. 589–637.

[Col96] Michael Collins. "A new statistical parser based on bigram lexical dependencies". In: *Proc. 34th ACL*. 1996, pp. 184–191.

[Cor+09] Thomas H. Cormen et al. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[CSC13] Shay B. Cohen, Giorgio Satta, and Michael Collins. "Approximate PCFG Parsing Using Tensor Decomposition". In: *Proc. NAACL HLT*. Atlanta, Georgia, USA, 2013, pp. 487–496.

[CSH08] Venkat Chandrasekaran, Nathan Srebro, and Prahladh Harsha. "Complexity of Inference in Graphical Models". In: *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*. 2008, pp. 70–78.

[CSH12] Venkat Chandrasekaran, Nathan Srebro, and Prahladh Harsha. "Complexity of Inference in Graphical Models". In: *CoRR* abs/1206.3240 (2012).

[DL97] M. M. Deza and M. Laurent. *Geometry of Cuts and Metrics*. Algorithms and Combinatorics, vol. 15. Springer, 1997.

[Eve+15] Mark Everingham et al. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (2015), pp. 98–136.

[Eve63] Hugh Everett. "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources". In: *Operations Research* 11.3 (1963), pp. 399–417. ISSN: 1526-5463.

[Faw06] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (2006), pp. 861–874.

[FJ08] Thomas Finley and Thorsten Joachims. "Training Structural SVMs when Exact Inference is Intractable". In: *Proc. 25th ICML*. 2008, pp. 304–311.

[For73] George David Forney. "The Viterbi algorithm". In: *Proc IEEE*. Vol. 61. 1973, pp. 268–278.

[GDS07] Rahul Gupta, Ajit A. Diwan, and Sunita Sarawagi. "Efficient inference with cardinality-based clique potentials". In: *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*. 2007, pp. 329–336.

[GJ02] Stuart Geman and Mark Johnson. "Dynamic programming for parsing and estimation of stochastic unification-based grammars". In: *Proc. 40th ACL*. 2002, pp. 279–286.

[GKB13] Abner Guzmán-Rivera, Pushmeet Kohli, and Dhruv Batra. "DivMCuts: Faster Training of Structural SVMs with Diverse M-Best Cutting-Planes". In: *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*. 2013, pp. 316–324.

[GLS88] Martin Grötschel, L?szl? Lov?sz, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Algorithms and combinatorics. Berlin, New York: Springer-Verlag, 1988.

[Gus80] Dan Gusfield. "Sensitivity analysis for combinatorial optimization". PhD thesis. UC Berkeley, 1980.

[Hee93] J. S. Heemskerk. "A Probabilistic Context-free Grammar for Disambiguation in Morphological Parsing". In: *Proc. EACL*. 1993, pp. 183–192.

[HHS84] Peter L. Hammer, Pierre Hansen, and Bruno Simeone. "Roof duality, complementation and persistency in quadratic 0-1 optimization". In: *Math. Program.* 28.2 (1984), pp. 121–155.

[Hön+17] János Höner et al. “Minimizing Trust Leaks for Robust Sybil Detection”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1520–1528.

[IM05] Hal Daumé Iii and Daniel Marcu. “Learning as search optimization: Approximate large margin methods for structured prediction”. In: *ICML*. 2005, pp. 169–176.

[JFY09] T. Joachims, T. Finley, and Chun-Nam Yu. “Cutting-Plane Training of Structural SVMs”. In: *Machine Learning* 77.1 (2009), pp. 27–59.

[JLT75] Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. “Tree adjunct grammars”. In: *Journal of Computer and System Sciences* 10.1 (1975), pp. 136 –163.

[Joa+09] Thorsten Joachims et al. “Predicting Structured Objects with Support Vector Machines”. In: *Communications of the ACM - Scratch Programming for All* 52, no. 11 (2009), pp. 97–104.

[Joa05] T. Joachims. “A Support Vector Method for Multivariate Performance Measures”. In: *Proc. 22nd ICML*. 2005, pp. 377–384.

[Joh+99] Mark Johnson et al. “Estimators for Stochastic Unification-based Grammars”. In: *Proc. 37th ACL*. 1999, pp. 535–541.

[Joh01] Mark Johnson. “Joint and Conditional Estimation of Tagging and Parsing Models”. In: *Proc. 39th ACL and 10th EACL*. 2001, pp. 314–321.

[Joh98] Mark Johnson. “PCFG Models of Linguistic Tree Representations”. In: *Computational Linguistics* 24.4 (1998), pp. 613–632.

[Jos85] Aravind K. Joshi. “Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions?” In: *Natural Language Parsing*. Cambridge University Press, 1985.

[JYJ09] Chun nam John Yu and Thorsten Joachims. “Learning Structural SVMs with Latent Variables”. In: *ICML*. 2009, pp. 1169–1176.

[Kap+04] Ronald M. Kaplan et al. “Speed and Accuracy in Shallow and Deep Stochastic Parsing”. In: *Proc. HLT-NAACL*. 2004, pp. 97–104.

[Kap+15] Jörg H. Kappes et al. “A Comparative Study of Modern Inference Techniques for Structured Discrete Energy Minimization Problems”. In: *International Journal of Computer Vision* 115.2 (2015), pp. 155–184.

[KB82] R. M. Kaplan and J. Bresnan. “Lexical-Functional Grammar: A Formal System for Grammatical Representation”. In: *The Mental Representation of Grammatical Relations*. Cambridge, MA: The MIT Press, 1982.

[KBR07] Vladimir Kolmogorov, Yuri Boykov, and Carsten Rother. “Applications of parametric maxflow in computer vision”. In: *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*. 2007, pp. 1–8.

[Kel60] J. E. Kelley. “The Cutting-Plane Method for Solving Convex Programs”. In: *Journal of the Society for Industrial and Applied Mathematics* 8.4 (1960), pp. 703–712.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[KM03a] Dan Klein and Christopher D. Manning. "A* Parsing: Fast Exact Viterbi Parse Selection". In: *Proc. HLT-NAACL*. 2003, pp. 119–126.

[KM03b] Dan Klein and Christopher D. Manning. "Accurate Unlexicalized Parsing". In: *Proc. 41st ACL*. Sapporo, Japan, 2003, pp. 423–430.

[Kol06] Vladimir Kolmogorov. "Convergent Tree-Reweighted Message Passing for Energy Minimization". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 28.10 (2006), pp. 1568–1583.

[Kol99] Daphne Koller. "Probabilistic Relational Models". In: *Inductive Logic Programming, 9th International Workshop, ILP-99, Bled, Slovenia, June 24-27, 1999, Proceedings*. 1999, pp. 3–13.

[KP07] Alex Kulesza and Fernando Pereira. "Structured Learning with Approximate Inference". In: *Proc. 20th NIPS*. 2007, pp. 785–792.

[KP09] Nikos Komodakis and Nikos Paragios. "Beyond pairwise energies: Efficient optimization for higher-order MRFs". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. 2009, pp. 2985–2992.

[KPT11] Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. "MRF Energy Minimization and Beyond via Dual Decomposition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.3 (2011), pp. 531–552.

[KW05] Vladimir Kolmogorov and Martin J. Wainwright. "On the Optimality of Tree-reweighted Max-product Message-passing". In: *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*. 2005, pp. 316–323.

[KW12] Vladimir Kolmogorov and Martin J. Wainwright. "On the optimality of tree-reweighted max-product message-passing". In: *CoRR* abs/1207.1395 (2012).

[KZ04] Vladimir Kolmogorov and Ramin Zabih. "What Energy Functions Can Be Minimized via Graph Cuts?" In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.2 (2004), pp. 147–159.

[Lac+13] Simon Lacoste-Julien et al. "Block-Coordinate Frank-Wolfe Optimization for Structural SVMs". In: *Proc. 30th ICML*. 2013, pp. 53–61.

[Laf01] John Lafferty. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: *ICML*. Morgan Kaufmann, 2001, pp. 282–289.

[Lin04] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of summaries". In: *Proc. ACL workshop on Text Summarization Branches Out*. 2004.

[LJK14] Yongsub Lim, Kyomin Jung, and Pushmeet Kohli. "Efficient Energy Minimization for Enforcing Label Statistics". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 36.9 (2014), pp. 1893–1899.

[LS90] S. L. Lauritzen and D. J. Spiegelhalter. "Readings in Uncertain Reasoning". In: San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990. Chap. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems, pp. 415–448.

[Lue73] David G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, 1973.

[Mar+94] Mitchell P. Marcus et al. "The Penn Treebank: Annotating Predicate Argument Structure". In: *Human Language Technology, Proceedings of a Workshop held at Plainsboro, New Jerey, USA, March 8-11, 1994*. 1994.

[MC10] Julian John McAuley and Tibério S. Caetano. "Exploiting Within-Clique Factorizations in Junction-Tree Algorithms". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. 2010, pp. 525–532.

[MC11] Julian John McAuley and Tibério S. Caetano. "Faster Algorithms for Max-Product Message-Passing". In: *Journal of Machine Learning Research* 12 (2011), pp. 1349–1388.

[McA06] David McAllester. "Generalization bounds and consistency for structured labeling". In: *Predicting Structured Data*. Ed. by Bakir H. Gökhan et al. The MIT Press, 2006, 247–262.

[Mes+10] Ofer Meshi et al. "Learning Efficiently with Approximate Inference via Dual Losses". In: *Proc. 27th ICML*. 2010, pp. 783–790.

[Mez+13] Elad Mezuman et al. "Tighter Linear Program Relaxations for High Order Graphical Models". In: *Proc. of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*. 2013.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: The MIT Press, 1999.

[MTG13] Qi Mao, Ivor Wai-Hung Tsang, and Shenghua Gao. "Objective-Guided Image Annotation". In: *IEEE Trans. Image Processing* 22.4 (2013), pp. 1585–1597.

[Mül+01] Klaus-Robert Müller et al. "An introduction to kernel-based learning algorithms". In: *IEEE Trans. Neural Networks* 12.2 (2001), pp. 181–201.

[NO09] Angelia Nedic and Asuman E. Ozdaglar. "Approximate Primal Solutions and Rate Analysis for Dual Subgradient Methods". In: *SIAM Journal on Optimization* 19.4 (2009), pp. 1757–1780.

[Now+14] Sebastian Nowozin et al. *Advanced Structured Prediction*. The MIT Press, 2014.

[Pap+02] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*. 2002, pp. 311–318.

[Pea89] Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989.

[Pet+06] Slav Petrov et al. "Learning Accurate, Compact, and Interpretable Tree Annotation". In: *Proc. COLING-ACL*. Sydney, Australia, 2006, pp. 433–440.

[PK07] Slav Petrov and Dan Klein. "Improved Inference for Unlexicalized Parsing". In: *Proc. HLT-NAACL*. Rochester, New York, 2007, pp. 404–411.

[PS94] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press, 1994.

[Rab89] Lawrence R. Rabiner. "A tutorial on hidden markov models and selected applications in speech recognition". In: *Proceedings of the IEEE*. 1989, pp. 257–286.

[RBZ07a] Nathan D. Ratliff, J. Andrew Bagnell, and Martin Zinkevich. "(Approximate) Subgradient Methods for Structured Prediction". In: *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS 2007, San Juan, Puerto Rico, March 21-24, 2007*. 2007, pp. 380–387.

[RBZ07b] Nathan D. Ratliff, J. Andrew Bagnell, and Martin Zinkevich. "(Approximate) Subgradient Methods for Structured Prediction". In: *Proc. 11th AISTATS*. San Juan, Puerto Rico, 2007, pp. 380–387.

[RC12] Alexander M. Rush and Michael Collins. "A Tutorial on Dual Decomposition and Lagrangian Relaxation for Inference in Natural Language Processing". In: *Journal of Artificial Intelligence Research* 45 (2012), pp. 305–362.

[RC14] Alexander M. Rush and Michael Collins. "A Tutorial on Dual Decomposition and Lagrangian Relaxation for Inference in Natural Language Processing". In: *CoRR* abs/1405.5208 (2014).

[RD06] Matthew Richardson and Pedro M. Domingos. "Markov logic networks". In: *Machine Learning* 62.1-2 (2006), pp. 107–136.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. ""GrabCut": interactive foreground extraction using iterated graph cuts". In: *ACM Trans. Graph.* 23.3 (2004), pp. 309–314.

[RMW10] Mani Ranjbar, Greg Mori, and Yang Wang. "Optimizing Complex Loss Functions in Structured Prediction". In: *Proc. 11th ECCV*. Heraklion, Crete, Greece, 2010, pp. 580–593.

[RN11] Denise Rey and Markus Neuhäuser. "Wilcoxon-Signed-Rank Test". In: *International Encyclopedia of Statistical Science*. 2011, pp. 1658–1659.

[RS07] Gunnar Rätsch and Sören Sonnenburg. "Large Scale Hidden Semi-Markov SVMs". In: *Proc. 19 NIPS*. 2007, pp. 1161–1168.

[Rus+12] Alexander M. Rush et al. "Improved Parsing and POS Tagging Using Inter-Sentence Consistency Constraints". In: *Proc. EMNLP-CoNLL*. Jeju Island, Korea, 2012, pp. 1434–1444.

[RVM12] Mani Ranjbar, Arash Vahdat, and Greg Mori. "Complex loss optimization via dual decomposition". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*. 2012, pp. 2304–2311.

[SB09] Christina Sauper and Regina Barzilay. "Automatically Generating Wikipedia Articles: A Structure-Aware Approach". In: *Proc. 47th ACL and 4th IJCNLP*. 2009, pp. 208–216.

[SG08] Sunita Sarawagi and Rahul Gupta. "Accurate max-margin training for structured output spaces". In: *Proc. 25th ICML*. 2008, pp. 888–895.

[SGJ11] David Sontag, Amir Globerson, and Tommi Jaakkola. "Introduction to Dual Decomposition for Inference". In: *Optimization for Machine Learning*. MIT Press, 2011.

[She+11] Shirish K. Shevade et al. "A Sequential Dual Method for Structural SVMs". In: *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28-30, 2011, Mesa, Arizona, USA*. 2011, pp. 223–234.

[She+13] Liu Sheng et al. "Approximated Slack Scaling for Structural Support Vector Machines in Scene Depth Analysis". In: *Mathematical Problems in Engineering* (2013).

[SL90] David J. Spiegelhalter and Steffen L. Lauritzen. "Sequential updating of conditional probabilities on directed graphical structures". In: *Networks* 20.5 (1990), pp. 579–605.

[SM14] Vivek Srikumar and Christopher D Manning. "Learning Distributed Representations for Structured Output Prediction". In: *Proc. 27th NIPS*. 2014, pp. 3266–3274.

[Son+12] David Sontag et al. "Tightening LP Relaxations for MAP using Message Passing". In: *CoRR* abs/1206.3288 (2012).

[Son10] David Sontag. "Approximate Inference in Graphical Models using LP Relaxations". PhD thesis. Department of Electrical Engineering and Computer Science: Massachusetts Institute of Technology, 2010.

[SVL07] Alexander J. Smola, S. V. N. Vishwanathan, and Quoc V. Le. "Bundle Methods for Machine Learning". In: *Proc. 21st NIPS*. Vancouver, British Columbia, Canada, 2007, pp. 1377–1384.

[SZ13] Shai Shalev-Shwartz and Tong Zhang. "Stochastic dual coordinate ascent methods for regularized loss". In: *Journal of Machine Learning Research* 14.1 (2013), pp. 567–599.

[TAK13] Ben Taskar, Pieter Abbeel, and Daphne Koller. "Discriminative Probabilistic Models for Relational Data". In: *CoRR* abs/1301.0604 (2013).

[Tar+12] Daniel Tarlow et al. "Fast Exact Inference for Recursive Cardinality Models". In: *Proc. 28th UAI*. 2012.

[Tas+04] Ben Taskar et al. "Max-Margin Parsing". In: *Proc. EMNLP*. Barcelona, Spain, 2004, pp. 1–8.

[Tas+05] Benjamin Taskar et al. "Learning structured prediction models: a large margin approach". In: *Proc. 22nd ICML*. Bonn, Germany, 2005, pp. 896–903.

[Teo+10] Choon Hui Teo et al. "Bundle Methods for Regularized Risk Minimization". In: *Journal of Machine Learning Research* 11 (2010), pp. 311–365.

[TGK03] Benjamin Taskar, Carlos Guestrin, and Daphne Koller. "Max-Margin Markov Networks". In: *Proc. 16th NIPS*. 2003, pp. 25–32.

[TGZ10] Daniel Tarlow, Inmar E Givoni, and Richard S Zemel. "HOP-MAP: Efficient message passing with high order potentials". In: *Proc. 13th AISTATS*. 2010.

[TLJ06] Benjamin Taskar, Simon Lacoste-Julien, and Michael I. Jordan. "Structured Prediction, Dual Extragradient and Bregman Projections". In: *Journal of Machine Learning Research* 7 (2006), pp. 1627–1653.

[Tso+05] I. Tsochantaridis et al. "Large Margin Methods for Structured and Interdependent Output Variables". In: *Journal of Machine Learning Research* 6 (2005), pp. 1453–1484.

[TT04] Yoshimasa Tsuruoka and Jun'ichi Tsujii. "Iterative CKY Parsing for Probabilistic Context-Free Grammars". In: *Proc. 1st IJCNLP*. Hainan Island, China, 2004, pp. 52–60.

[TZ12] Daniel Tarlow and Richard S. Zemel. "Structured Output Learning with High Order Loss Functions". In: *Proc. 15th AISTATS*. La Palma, Canary Islands, 2012, pp. 1212–1220.

[WJ08] Martin J. Wainwright and Michael I. Jordan. "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends in Machine Learning* 1.1-2 (2008), pp. 1–305.

[WJ88] David J. Weir and Aravind K. Joshi. "Combinatory Categorial Grammars: Generative Power and Relationship to Linear Context-Free Rewriting Systems". In: *Proc. 26th ACL*. New York, USA, 1988, pp. 278–285.

[WJW05] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. "MAP estimation via agreement on trees: message-passing and linear programming". In: *IEEE Trans. Information Theory* 51.11 (2005), pp. 3697–3717.

[Wu+08] Yi Hsuan Wu et al. "Early-Pruned K-Best Sphere Decoding Algorithm Based on Radius Constraints". In: *Proceedings of IEEE International Conference on Communications, ICC 2008, Beijing, China, 19-23 May 2008*. 2008, pp. 4496–4500.

[WY14] Junyan Wang and Sai-Kit Yeung. "A Compact Linear Programming Relaxation for Binary Sub-modular MRF". In: *Energy Minimization Methods in Computer Vision and Pattern Recognition - 10th International Conference, EMMCVPR 2015, Hong Kong, China, January 13-16, 2015. Proceedings*. 2014, pp. 29–42.

[You67] Daniel H. Younger. "Recognition and Parsing of Context-Free Languages in Time nˆ3". In: *Information and Control* 10.2 (1967), pp. 189–208.

[Zho+10] Wenliang Zhong et al. "Incorporating the loss function into discriminative clustering of structured outputs". In: *IEEE Transactions on Neural Networks and Learning Systems* 21, no. 10 (2010), pp. 1564–1575.