Efficient Methods for Learning and Inference in Structured Output Prediction

Thesis submitted for the degree of
"Doctor of Philosophy"

By Ofer Meshi

Submitted to the Senate of the Hebrew University of Jerusalem

August 2013

This work was carried out under the supervision of:

Professor Amir Globerson, and
Professor Nir Friedman

Acknowledgements

First, I would like to thank my advisor Amir Globerson for giving me a chance and guiding me patiently through the past few years. I am truly honored to be Amir's first student, I have learned so much from him. His wide and deep knowledge, his graciousness, his kindness, and his excellence are a source of inspiration and I feel lucky that I had a chance to work with him. I would also like to thank Nir Friedman for giving me my first chance of graduate studies by taking me as a master's student and then as a Ph.D. student. Nir is a brilliant scientist and I feel privileged that I had the opportunity to learn from him about the scientific practice. I would also like to thank Nir for his kindness and for his willingness to listen and give advice about anything.

Second, I would like to thank my collaborators: Tommi Jaakkola, David Sontag, Gal Elidan, Elad Eban, and Ariel Jaimovich. It has been a great pleasure to work with such talented smart people.

Third, I am indebted to the committee members: Nati Linial, Gal Elidan, and Shai Shalev-Shwartz. They have patiently provided invaluable advice and insightful comments which have improved the quality of this work.

Next, I would like to thank the past and present students in the machine learning group. It was so much fun to work besides them. There was always someone with whom I could scribble some ideas on the board, take a lunch break, have a cup of coffee, or simply chat. In particular I would like to thank Ariel Jaimovich, Elad Eban, Nir Rosenfeld, and Cobi Cario for the close collaboration and exciting moments. This is also a good opportunity to thank Moran Yassour for being such a wonderful roommate. I really hope those friendships will last for years to come.

I would also like to acknowledge the great and open atmosphere in the machine learning group inspired by the faculty members. In particular, I would like to thank Shai Shalev-Shwartz and Ami Wiesel for opening their doors, providing good advice, and calmly answering my questions.

Finally, I want to thank my family for their endless support. My wonderful parents, Zvika and Amalia, who provided constant mental and material support. Without them I could not have done this. My children: Noam, Eden, and Tom, whom I love so much. I thank them for giving me a real life outside research. They have given me the chance to take the right break at the right time. Lastly, I would like to thank my beloved wife and partner Avital for her infinite support and encouragement. Without her all this would be meaningless.

Abstract

Binary classification has dominated machine learning for decades. However, many modern applications of machine learning require modeling complex high-dimensional outputs. Although the output space in such problems is typically very large, it often has intrinsic structure which can be exploited to construct efficient algorithms. Indeed, in recent years *structured output prediction* has shown state-of-the-art results in many real-world problems from computer vision, natural language processing, computational biology, and other fields. However, for this kind of models to remain useful, they must perform prediction efficiently and scale well to handle large data sets. This raises many interesting and challenging questions in machine learning and optimization.

In this setting there are two main tasks, namely prediction and learning. In the prediction (a.k.a. inference) task the goal is to jointly predict a set of interdependent output variables, given a model. Since the space of possible outputs may be prohibitively large, finding the optimal output is generally intractable. Therefore, it is helpful to assume that the dependencies between output variables obey a particular structure. For some special types of structures the prediction task can be solved exactly and efficiently. However, this task is NP-hard in general. In the learning task the goal is to learn the predictive model from empirical data. This can be done using formulations that generalize binary Support Vector Machines (SVMs) to structured outputs. However, these formulations include a loss-augmented prediction problem for every training example, and are also computationally hard in general.

In this thesis we propose and analyze algorithms that address the prediction and learning tasks for structured outputs. The proposed methods provide efficient approximate solutions to both problems. They achieve high accuracy with fast runtime and good scaling with problem size. Empirical evaluations on various benchmark datasets show that they compare favorably with other state-of-the-art approaches.

In particular, one type of approximations that has proved particularly successful for many structured prediction applications is Linear Programming (LP) relaxation. For the prediction task, in this thesis we propose a new algorithm for efficient and scalable prediction based on an augmented Lagrangian formulation of the dual LP relaxation. We also study the convergence properties of a family of inference algorithms which perform block coordinate descent on a smoothed variant of the dual LP. In addition, we address here the learning problem. We present an efficient algorithm for training structured predictors from labeled data. Our algorithm is based on a reformulation of the learning objective which replaces the primal LP relaxation with its dual. This reformulation has several computational advantages. Finally, we study the problem of learning tree-structured predictors, which is interesting since for tree-structured models prediction is then fast and exact. We show that this is a hard problem and propose an efficient and highly accurate approximation scheme.

Contents

Chapter 1

Introduction

Numerous applications involve prediction of complex multivariate outputs. For example, in natural language parsing one is given an input sentence and the task is to output a parse tree [Taskar et al., 2004b]. In computer vision, a typical task is to assign labels to all pixels of a given image [Szeliski et al., 2008, Felzenszwalb et al., 2010]. In such problems the output consists of multiple interdependent variables. One may simply train a classifier for each variable independently, however, exploiting interdependence between the variables often improves prediction quality. Moreover, in some cases the output variables must satisfy some global constraints, and thus cannot be predicted independently. For these reasons *structured output prediction* has been proposed as a framework to handle such complex multivariate output spaces. In this approach the goal is to jointly predict all variables simultaneously, taking into account their mutual dependence. Indeed, using this framework has resulted in state-of-the-art results in applications from various fields [Bakir et al., 2007].

Formally, structured prediction problems can be cast as mapping a real-valued input vector x to an output vector $y = (y_1, \dots, y_n)$, where y_i are discrete variables (e.g., part-of-speech tags of words, or depth levels of pixels). This mapping is commonly done via a linear discrimination rule:

$$y(x) = \operatorname*{argmax}_y w^\top \phi(x, y) \equiv \operatorname*{argmax}_y \theta(y; x, w) \tag{1.1}$$

where $\phi(x, y)$ is a function that maps input-output pairs to a feature vector, w is the corresponding weight vector, and $\theta(y; x, w)$ is the resulting score function over outputs.[1] For example, in part-of-speech tagging x is an input sentence, y is a set of tags – one for each word, and $\phi(x, y)$ could have an entry which counts the number of times occurrences of the word *"the"* in the sentence are assigned the tag *"determiner"* in y.

Since the space of possible outputs y may be exponential in size, one cannot naively enumerate all possible outputs to find the maximizing assignment. Therefore, it is helpful

[1]In conditional random fields [Lafferty et al., 2001] the score is normalized to define a probability distribution over outputs: $p(y|x) \propto \exp \theta(y; x, w)$. In this thesis we do not require such a probabilistic interpretation.

to assume that the score function *decomposes* into simpler score functions (a.k.a. factors):

$$\theta(y; x, w) = \sum_c \theta_c(y_c; x, w) \tag{1.2}$$

where y_c are subsets of output variables (to simplify notation we sometimes drop the dependence of θ on x and w in the sequel). One such decomposition that is commonly used in many applications consists of scores over single and pairs of variables that correspond to nodes and edges of a graph G:

$$\theta(y) = \sum_{ij \in E(G)} \theta_{ij}(y_i, y_j) + \sum_{i \in V(G)} \theta_i(y_i) \tag{1.3}$$

For example, in an image segmentation problem each y_i may represent whether the i'th pixel belongs to the background ($y_i = 0$) or foreground ($y_i = 1$). In this case the singleton function θ_i assigns a score for each pixel which depends on the image itself (e.g., by the color of that pixel), while θ_{ij} assigns scores to pairs of neighboring pixels (e.g., it may assign a high score if both pixels are assigned to the background). Many other decompositions were found to be useful for a variety of applications [e.g., Martins et al., 2009a]. For some special types of decompositions the prediction task can be solved exactly and efficiently (see Section 1.1.1). However this combinatorial optimization problem is NP-hard in general [Shimony, 1994].[2] In the literature on probabilistic graphical models, the prediction task is known as the *MAP inference* problem [e.g., Koller and Friedman, 2009, Wainwright and Jordan, 2008].

Structured predictors can be learned from data using formulations like Max-Margin Markov Networks (M^{3}N) [Taskar et al., 2003], Structured Support Vector Machines (SSVM) [Tsochantaridis et al., 2005], and conditional random fields (CRF) [Lafferty et al., 2001]. In this learning task one is given training data consisting of input-output pairs $\{(x^m, y^m)\}_{m=1}^M$, and the goal is to find a weight vector w that correctly classifies the training examples: $y^m = \text{argmax}_y\, w^\top \phi(x^m, y)$. In practice, the training set may not be separable and therefore this requirement is encouraged softly through an *empirical risk minimization (ERM)* framework. Specifically, we focus here on the M^{3}N formulation in which w is learned by minimizing the following regularized structured hinge loss:

$$\ell(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_m h^m(w) \tag{1.4}$$

where

$$h^m(w) = \max_y \left(w^\top \phi(x^m, y) + \Delta(y, y^m) \right) - w^\top \phi(x^m, y^m) \ , \tag{1.5}$$

and $\Delta(y, y^m)$ is a label-loss function measuring the cost of predicting y when the true label is y^m (e.g., 0/1 or Hamming distance). To gain some intuition on this formulation, notice that a positive loss is incurred whenever the score of the true output $w^\top \phi(x^m, y^m)$ is smaller than the score of the highest scoring assignment $\max_y w^\top \phi(x^m, y)$ (plus a margin determined by Δ). This is a generalization of binary SVM to structured outputs. We have recently shown that

[2]For example, max-cut can be easily seen as an instance of maximizing Eq. (1.3).

the learning task for structured prediction is also NP-hard in general [Sontag et al., 2010], and therefore one must resort to approximations or restrict models to tractable families.

Early research on structured prediction focused on the case where prediction and learning were computationally tractable [Lafferty et al., 2001, Collins, 2002, Taskar et al., 2003]. However, when one wishes to model rich dependencies between output variables, then prediction and learning become computationally expensive. Hence, the setting we have is that the desired outputs are maxima of a function (Eq. (1.1)) whose optimization is hard and whose parameters (w) we do not know. Clearly, some approximations are necessary. But what can be said about these? How can we perform them efficiently? How can we integrate them into the learning procedure? As we see, this setting gives rise to many interesting and challenging questions in machine learning and optimization. In this thesis I address this challenge by developing and analyzing efficient approximation algorithms for the prediction and learning tasks for structured outputs.

1.1 The Prediction Problem

Under the decomposition assumption of Eq. (1.2), prediction (i.e., Eq. (1.1) and Eq. (1.5)) requires solving the following optimization problem:

$$\max_y \sum_c \theta_c(y_c) \tag{1.6}$$

As mentioned, due to its combinatorial nature this problem is generally hard to solve. In this section we review some of the commonly used methods which have been proposed for solving this problem. We begin by inspecting a few special cases, where the optimization problem in Eq. (1.6) can be solved exactly and efficiently. We then proceed to survey the main approximation techniques for the more general case.

1.1.1 Exact inference

Tree Structured Graphs: One type of tractable models is *tree-structured* graphs. In these models the score decomposes over a graph as in Eq. (1.3), and the graph contains no cycles. In this case a MAP assignment can be found efficiently using a dynamic programming algorithm known as *max-product belief propagation (BP)* [Pearl, 1988, Kschischang et al., 2001]. The algorithm begins by defining one of the variables (nodes) as root, and then proceeds by sending *messages* from the leaf nodes up the tree until the root, and then back downwards to the leafs. The updates take the following form:

$$\delta_{ji}(y_i) \propto \max_{y_j} \left(\exp(\theta_{ij}(y_i, y_j) + \theta_j(y_j)) \prod_{k \in N(j) \setminus i} \delta_{kj}(y_j) \right) \qquad \text{for all } y_i \tag{1.7}$$

where δ_{ji} is the message from node j to node i, and $N(i)$ is the set of neighbors of node i in the graph. Notice that this update uses the fact that the messages are invariant to scaling, so they can be normalized. Also, in order to avoid numerical issues, messages are often stored and computed in log-space. Finally, a MAP assignment is easily obtained from the messages around each node by:

$$\forall i \quad y_i = \underset{y_i'}{\operatorname{argmax}} \prod_{k \in N(i)} \delta_{ki}(y_i')$$

Notice that the total runtime of the max-product algorithm is linear in the number of variables n, which makes it extremely efficient in practice. Indeed, this has motivated us to study the problem of learning tree-structured models for structured outputs (see Section 2.4). A similar dynamic programming approach can be applied not only to trees, but also to *low treewidth* graphs [Koller and Friedman, 2009], where the treewidth of a graph is defined as the size of the largest clique in a chordal completion of that graph. The runtime of the max-product algorithm is known to be exponential in the treewidth, so when this size is small enough then inference remains tractable.

Submodular Scores: A second class of models which facilitate exact inference is when the local edge scores are *submodular*. More specifically, this class consists of models where the variables are *binary*, the score decomposes over a graph, and the pairwise scores are restricted as follows:

$$\theta_{ij}(0,0) + \theta_{ij}(1,1) \geq \theta_{ij}(0,1) + \theta_{ij}(1,0) \quad \forall ij \in E(G)$$

Intuitively, this kind of score encourages the corresponding variables to take the same assignment. In this case the MAP problem can be formulated as a minimum cut problem in an appropriately constructed graph [Hammer, 1965, Greig et al., 1989]. Thus, the MAP problem can be solved exactly by standard maximum flow algorithms (e.g., Ford and Fulkerson [1956], Dinitz [1970]) in polynomial time (e.g., $O(n^3)$ for the Dinitz algorithm). There are also extensions of this framework to general convex edge scores and non-binary variables [Ishikawa, 2003, Flach and Schlesinger, 2004].

Planar Graphs: A third class of tractable models is *planar* graphs. A graph is called planar if it can be drawn in the plane without any crossing edges. It turns out that when variables are *binary*, singleton scores are all zero ($\theta_i(y_i) = 0$ for all i, y_i), and pairwise socres take the form: $\theta_{ij} = \begin{pmatrix} w_{ij} & 0 \\ 0 & w_{ij} \end{pmatrix}$, then exact MAP inference over planar graphs can be solved efficiently [Kasteleyn, 1963, Fisher, 1966, Plummer and Lovász, 1986, Deza and Laurent, 1997]. This is achieved by converting the problem into a maximum-weight perfect matching problem over a transformed graph. Several algorithms have been proposed for solving maximum-weight perfect matching in polynomial time [Edmonds, 1965, Cook and Rohe, 1999] (runtime can be as low as $O(n^3 \log n)$).

Finally, there are few other types of models where exact inference is possible [e.g., Darwiche, 2003, Dechter and Mateescu, 2007, Poon and Domingos, 2011, Jebara, 2013]. However, for many applications limiting the model class to tractable families is too restrictive. Instead, we may want to give up exactness of the solution in order to allow richer more expressive models.

1.2 Approximate MAP inference

As mentioned earlier, in many real problems restricting the model to be tractable is not suitable. Since performing prediction is still desirable, even when the model is intractable, there has been considerable amount of research on approximation schemes. A simple approach to approximate MAP inference is to use a probabilistic interpretation of the model. In particular, Eq. (1.2) can be used to define a probability distribution over outputs by taking the score exponent and then dividing by a normalization constant.[3] One can then use various methods to approximate the marginal probabilities of single variables [Koller and Friedman, 2009], and then an approximate MAP solution can be easily obtained by assigning each variable its maximizing value. One shortcoming of this procedure is that it requires computing marginal probabilities, a hard task in itself. In addition, this simple approach is often outperformed by approaches which approximate the MAP directly. In the rest of this section I review some of the most popular of these approaches.

1.2.1 Loopy Belief propagation

The max-product BP algorithm described in Section 1.1.1 is exact for tree-structured graphs. However, when the graph contains cycles the messages in Eq. (1.7) are still well defined. Therefore, we can perform message updates iteratively in some order (schedule), in a similar manner to the original algorithm. This variant is called *loopy belief propagation*. Unfortunately, this algorithm is no longer guaranteed to converge and even if it does converge, the returned solution may not be optimal. However, it has shown excellent performance in applications from various domains [Murphy et al., 1999]. Moreover, Tarlow et al. [2011] have shown that under a particular scheduling and message averaging (known as 'damping'), loopy BP is equivalent to graph cuts, and thus optimal for submodular scores (see previous section).

1.2.2 Sampling

Given the probabilistic interpretation of the model mentioned above, sampling methods can be used to draw samples from the underlying distribution. One method which is commonly used for this task is *Markov chain Monte Carlo (MCMC)*. The basic idea is to construct a Markov chain that has as its stationary distribution the target distribution. Then by simulating the Markov

[3]This normalization constant, a.k.a. the 'partition function', is hard to compute as it requires summing over all possible outputs.

chain we can obtain a sample from the stationary distribution. These samples can then be used to approximate expectations of functions under the distribution of interest (e.g., marginal probabilities). A popular way to construct such Markov chains is through the *Gibbs sampler* [Geman and Geman, 1984]. The algorithm proceeds by drawing samples from conditional distributions over small subsets of variables, conditioned on the states of the other variables, which can be done efficiently.

Perhaps the main drawback of this approach is that the mixing time of the Markov chain is often very long, and therefore the obtained samples may not be from the correct distribution. Nevertheless, sampling methods have been used for approximate inference in many applications with good empirical results.

Sampling can be used to obtain an approximate MAP solution through *simulated annealing* [Kirkpatrick et al., 1983]. The idea is to run MCMC while gradually decreasing a temperature parameter which controls the skewness of the distribution. When the temperature reaches zero then samples are drawn only from the mode of the distribution, corresponding to MAP assignments. If the annealing schedule (i.e., rate of temperature decrease) is slow enough, then the probability of sampling the maximizing assignment y^* tends to 1 as the number of iterations grows [Geman and Geman, 1984]. However, this requires a schedule which is too slow to be useful in practice, so instead, faster rates are used, which do not enjoy the optimality guarantee.

1.2.3 Local search methods

An alternative scheme used for MAP inference is based on search methods [see, e.g., Dechter, 2003]. Search algorithms begin with some solution and iteratively attempt to improve it. This is done by considering a set of neighboring states and taking the best move, until no improvement can be achieved. Therefore, the returned solution is a local optimum w.r.t. the search space. Typically, the number of considered neighbor states is kept small so each iteration can be computed efficiently.

Perhaps the best known example of a local search procedure is the *iterated conditional modes (ICM)* algorithm [Besag, 1986]. The algorithm begins by choosing some complete assignment (randomly or in a predefined manner). In each iteration one of the variables is picked, and then the objective (Eq. (1.6)) is maximized over all of its possible assignments while fixing all other variables to their current state. This local optimization problem can be solved efficiently by enumeration.[4]

More advanced search methods have also been explored. Notably, Boykov et al. [2001] proposed two neighborhoods, *α-expansion* and *α-β-swap*. In α-expansion any variable may change its assignment to a particular state (label) α. Similarly, in α-β-swap two states are chosen and then any variable labeled with either of these states may change its assignment to the other state. In both cases, finding the optimal move reduces to solving a minimum cut problem, which can be done efficiently. Moreover, Komodakis and Tziritas [2007] have shown

[4]This calculation is very similar to the one used in the Gibbs sampler which was mentioned before.

that α-expansion can be viewed as a primal-dual algorithm for solving a specific form of LP relaxation (Section 1.2.4). This approach became very popular in computer vision applications due to its efficiency and high solution quality. More recently, Kumar et al. [2011] proposed to generalize these search techniques by performing 'Range Moves', which explore a larger search space by considering a range of states rather than one or two at a time.

1.2.4 LP relaxation and dual decomposition

In this thesis we focus on one type of approximations that has proved particularly successful for many structured prediction applications, namely *linear programming (LP) relaxation* [Wainwright et al., 2005, Werner, 2007]. In particular, under the score decomposition assumption (Eq. (1.2)) each prediction problem can be cast as the following LP:[5]

$$\begin{aligned}
&\max_y \sum_c \theta_c(y_c) + \sum_i \theta_i(y_i) \\
&= \max_y \sum_c \sum_{y_c'} \mathbb{I}\{y_c = y_c'\}\theta_c(y_c) + \sum_i \sum_{y_i'} \mathbb{I}\{y_i = y_i'\}\theta_i(y_i) \\
&\leq \max_{\mu\in\mathcal{L}(G)} \sum_c \sum_{y_c} \mu_c(y_c)\theta_c(y_c) + \sum_i \sum_{y_i} \mu_i(y_i)\theta_i(y_i) \\
&\equiv \max_{\mu\in\mathcal{L}(G)} \mu\cdot\theta
\end{aligned} \tag{1.8}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function, μ are relaxed variables corresponding to local marginal probabilities,[6] and $\mathcal{L}(G)$ is the relaxed set of constraints known as the "local marginal polytope" which enforces local agreement between marginals:

$$\mathcal{L}(G) = \left\{ \mu \geq 0 : \begin{array}{ll} \sum_{y_{c\setminus i}} \mu_c(y_c) = \mu_i(y_i) & \forall c, i \in c, y_i \\ \sum_{y_i} \mu_i(y_i) = 1 & \forall i \end{array} \right\} \tag{1.9}$$

If μ is constrained to be integral, then the integer LP of Eq. (1.8) is in fact exact (i.e., holds with equality). However, in order to obtain a tractable LP, the integrality requirement is relaxed, which allows for fractional solutions and yields an upper bound on the optimum value. The basic relaxation can be further tightened by introducing higher-order agreement constraints [Sontag et al., 2008, Werner, 2008, Komodakis and Paragios, 2008].

Although the LP in Eq. (1.8) is tractable, standard solvers (e.g., based on the simplex algorithm [Murty, 1983]) perform poorly and do not scale well with problem size [Yanover et al., 2006]. Therefore, in recent years significant effort has been put into designing efficient and scalable solvers for this LP.

Ravikumar et al. [2010] proposed to optimize the LP via a proximal point method. Their

[5]We add factors for individual variables for notational convenience, those are not needed for generality.

[6]The notation $\mu_c(y_c)$ is standard in the literature and can be understood as the marginal probability of the subset of variables y_c [see Wainwright and Jordan, 2008].

algorithm is guaranteed to converge to the optimum of the LP, however it has the disadvantage of using a double loop scheme where every update involves an iterative algorithm for projecting onto the local polytope. More recently, Martins et al. [2011] presented an algorithm based on the alternating direction method of multipliers (ADMM) [Glowinski and Marrocco, 1975, Gabay and Mercier, 1976, Boyd et al., 2011]. However, their framework is restricted to binary pairwise models.[7] In Section 2.1 we present a related approach, which is also based on ADMM.

In what follows we focus on solvers that optimize the *dual LP* of Eq. (1.8). In particular, the Lagrangian dual of the LP in Eq. (1.8) has the following form:

$$\min_{\delta} \left[\sum_c \max_{y_c} \left(\theta_c(y_c) - \sum_{i:i\in c} \delta_{ci}(y_i) \right) + \sum_i \max_{y_i} \left(\theta_i(y_i) + \sum_{c:i\in c} \delta_{ci}(y_i) \right) \right] \tag{1.10}$$

where $\delta_{ci}(y_i)$ are the dual variables / messages / Lagrange multipliers corresponding to the agreement constraints between factor c and variable i on the assignment y_i from Eq. (1.9) [see Sontag et al., 2011, for a review]. In this dual problem, the local functions being maximized can be thought of as *reparameterizations*: adjusted versions of the original factors θ [Wainwright et al., 2003]. Each such reparameterization provides an upper bound on the optimal solution to the LP relaxation Eq. (1.8), and we are seeking to minimize this bound. In fact, if there exists a solution δ such that all the local maximizations are consistent (i.e., they assign the same values to variables in their intersections), then we are guaranteed to have found the exact solution to the original problem Eq. (1.6) (see Weiss et al. [2007], and further discussion at the end of this section). The dual in Eq. (1.10) is an unconstrained piecewise-linear convex optimization problem and can be solved by various methods.

Komodakis et al. [2007] presented a *subgradient* algorithm, which has very simple updates, but may take a long time to converge. Specifically, its convergence rate is upper bounded by $O(1/\epsilon^2)$. Namely, after $O(1/\epsilon^2)$ iterations of the subgradient algorithm the solution δ^t is guaranteed to be worse than the optimal solution by at most ϵ. A number of authors proposed a *block coordinate descent* algorithm for this optimization problem [Globerson and Jaakkola, 2008, Werner, 2007, Kolmogorov, 2006, Sontag and Jaakkola, 2009]. These methods differ slightly from each other in the size of the block which is optimized at each iteration. The updates can be elegantly understood as message-passing in a corresponding *factor graph* [Kschischang et al., 2001]. The algorithms in this class are monotonically decreasing and often converge fast, however, since the objective function is non-smooth, they might get stuck and return suboptimal solutions.

To remedy this shortcoming, several authors suggested to *smooth* the LP and then run coordinate descent on the smooth dual LP [Hazan and Shashua, 2010, Werner, 2009, Johnson, 2008]. The resulting algorithm is guaranteed to converge to the global optimum of the smoothed LP. In Section 2.2 we show that the convergence rate of this approach is upper bounded by $O(\tau/\epsilon)$, where τ is a temperature parameter which controls the amount of smoothing of the

[7]Several specific global factors can also be handled.

LP. To get an ϵ-optimal solution, τ is set to $O(1/\epsilon)$. This yields a rate similar to that of the subgradient algorithm, however coordinate descent algorithms often converge faster in practice.

An alternative optimization scheme for the smooth dual LP, based on Nesterov's *accelerated gradient* method [Nesterov, 2005], was presented by Jojic et al. [2010] and improved by Savchynskyy et al. [2011]. This approach enjoys a faster rate guarantee of $O(1/\epsilon)$, however, each iteration of the algorithm requires computation of the full gradient, which might be expensive in some applications.

More recently we have shown how to apply *ADMM* to the non-smooth dual LP (see Section 2.1). The advantage of our approach is that the updates are simple and the algorithm is guaranteed to globally converge w.r.t. the original non-smooth LP relaxation. Moreover, it was recently shown that it has fast convergence rate of $O(1/\epsilon)$ [He and Yuan, 2012, Wang and Banerjee, 2012], which is similar to the accelerated gradient method, but does not require smoothing of the objective. Finally, here we surveyed some of the more popular algorithms, however, our list is by no means exhaustive. This is an active research area and new solvers for the relaxed LP keep appearing [e.g., Schmidt et al., 2011, Kappes et al., 2012, Schwing et al., 2012, Kappes et al., 2013].

Working in the dual LP not only allows for efficient algorithms, but can also sometimes provide a certificate of optimality. For any dual solution δ the objective of Eq. (1.10) upper bounds the relaxed optimum (Eq. (1.8)), which in turn upper bounds the non-relaxed optimum (Eq. (1.6)). Therefore, a certificate of optimality can be achieved by decoding an integral solution and checking the difference between the dual and primal values. If there is no gap then we are guaranteed to have an optimal solution to the original (non-relaxed) optimization problem. When the LP is not tight then the dual and (decoded) primal values never meet (i.e., Eq. (1.8) holds with strong inequality). In such cases we may want to map the dual solution to a feasible *fractional* primal solution (e.g., in order to have a good stopping criterion). We have recently proposed a simple and efficient way to compute such a mapping (see Section 2.2).

1.2.5 Quadratic programming relaxation

In the case of pairwise factors Ravikumar and Lafferty [2006] formulate the MAP problem as a *quadratic program (QP)* silimlar to the LP formulation:

$$\begin{aligned}
&\max_y \sum_{ij} \theta_{ij}(y_i, y_j) + \sum_i \theta_i(y_i) \\
&= \max_y \sum_{ij} \sum_{y_i'} \sum_{y_j'} \mathbb{I}\{y_i = y_i'\}\mathbb{I}\{y_j = y_j'\}\theta_{ij}(y_i, y_j) + \sum_i \sum_{y_i'} \mathbb{I}\{y_i = y_i'\}\theta_i(y_i) \\
&= \max_\mu \sum_{ij} \mu_i^\top \Theta_{ij} \mu_j + \sum_i \mu_i^\top \theta_i \qquad (1.11) \\
&\text{s.t.} \qquad 0 \le \mu_i(y_i) \le 1 \;\; \forall i, y_i; \qquad \sum_{y_i} \mu_i(y_i) = 1 \;\; \forall i
\end{aligned}$$

where Θ_{ij} is the matrix realization of $\theta_{ij}(\cdot,\cdot)$. Notice that Eq. (1.11) holds with equality, namely this relaxation is in fact tight [see Ravikumar and Lafferty, 2006]. In addition, the representation is more concise than the LP since the pairwise marginals μ_{ij} are not represented explicitly, and the number of constraints is also smaller.

Whenever Θ is negative-semidefinite then this QP is convex and can be solved efficiently in various ways, however, in the general case this problem is non-convex. Ravikumar and Lafferty [2006] propose to use a convex relaxation. An alternative approach is to solve the non-convex problem directly using a convex-concave procedure [Kappes and Schnörr, 2008, Kumar et al., 2012], however, this scheme may get stuck in a local optimum.

Finally, despite the flexibility in the form of the objective and constraints offered by QP relaxation, it has been proved that the LP relaxation strictly dominates (i.e., provides a better approximation than) QP relaxation [Kumar et al., 2009].

To conclude, in addition to the approaches reviewed above, various other approximations have been proposed for MAP inference [e.g. Rother et al., 2007, Kumar and Zilberstein, 2010]. New and better algorithms for the prediction task for structured outputs are the subject of ongoing research.

1.3 The Learning Problem

We now proceed to address the learning problem for structured output prediction. Specifically, we focus on the M^{3}N formulation in Eq. (1.4), which we restate here for convenience:

$$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_m \max_y \left(w^\top \phi(x^m, y) + \Delta(y, y^m)\right) - w^\top \phi(x^m, y^m) \tag{1.12}$$

In terms of the weights w, this is a convex optimization problem which can be solved in various ways. However, in order to calculate the objective or learn a classifier, one has to solve a loss-augmented prediction problem for each training sample: $\max_y \left(w^\top \phi(x^m, y) + \Delta(y, y^m)\right)$. In particular, the structured hinge loss terms form a piecewise linear function of w with an exponential number of segments (one for each possible assignment y), which complicates the optimization. Typically, an iterative learning algorithm would need to solve the (generally intractable) maximization over y in order to update w. As we shall see later, whenever this maximization (i.e., inference) can be solved efficiently then standard convex optimization methods can be readily applied to this learning problem. Indeed, numerous algorithms have been suggested for this case, including: structured perceptron [Collins, 2002], sequential minimal optimization (SMO) [Taskar et al., 2003], cutting-plane [Tsochantaridis et al., 2005], stochastic subgradient [Ratliff et al., 2007], extra-gradient [Taskar et al., 2006], bundle methods [Teo et al., 2010], the Frank-Wolfe algorithm [Lacoste-Julien et al., 2013], exponentiated gradient[Collins et al., 2008] and more. We next give some details on a few of the most popular of these approaches and then proceed to discuss the learning problem in case of intractable inference.

1.3.1 Subgradient descent

One of the simplest ways to optimize a convex function is to follow a descent direction. However, since the objective in Eq. (1.12) is non-differentiable the simple gradient descent method is not applicable. Luckily, the *subgradient descent* method, which generalizes gradient descent, can be applied in this case [Shor et al., 1985]. The subgradient of the objective in Eq. (1.12) is obtained by:

$$g = \lambda w + \frac{1}{M}\sum_m \phi(x^m, \hat{y}^m) - \phi(x^m, y^m) \tag{1.13}$$

where $\hat{y}^m \in \text{argmax}_y \left(w^\top \phi(x^m, y) + \Delta(y, y^m)\right)$. The algorithm proceeds by taking steps in the inverse direction of the subgradient using decreasing step-sizes (see below). Notice that in order to obtain $\hat{y}^m$ one has to optimize the prediction score and the label loss *together* [see, e.g., McAllester et al., 2010, for some complex loss terms]. This is known as the loss-augmented prediction task. Notice that each time we update the weights w we need to solve the loss-augmented prediction for all training examples. Even when prediction is tractable, this may become very expensive if the dataset is large. Fortunately, the subgradient algorithm can also be implemented in a *stochastic* manner [Bertsekas et al., 2003, Shalev-Shwartz et al., 2007, Ratliff et al., 2007]. In particular, it is enough to obtain at iteration t a random estimate $\hat{g}_t$, such that $\mathbb{E}[\hat{g}_t] \in \partial\ell(w_t)$. This means we can consider only a single training example m in each iteration:

$$\hat{g} = \lambda w + \phi(x^m, \hat{y}^m) - \phi(x^m, y^m) \tag{1.14}$$

Setting a learning rate η, the resulting algorithm is summarized below.

Algorithm 1 Stochastic subgradient descent (SGD)

```
Input: $w_1$, $\eta$
for $t = 1, \dots, T$ do
    Pick a training example $m$ uniformly at random
    $\hat{y}^m = \text{argmax}_y \left(w_t^\top \phi(x^m, y) + \Delta(y, y^m)\right)$
    $w_{t+1} = w_t - \frac{\eta}{t}\left(\lambda w_t + \phi(x^m, \hat{y}^m) - \phi(x^m, y^m)\right)$
end for
return $w_T$
```

The convergence rate of SGD for a strongly-convex non-smooth objective (like Eq. (1.12)) has been recently analyzed by Shamir and Zhang [2013]. They show that the suboptimality of the solution at iteration t, namely $\ell(w_t) - \ell(w^*)$, is upper bounded by $O(\log(t)/t)$. This rate can be further improved to $O(1/t)$ by a simple averaging scheme [Shamir and Zhang, 2013]. In fact, this improved rate is optimal as it achieves an information-theoretic $\Omega(1/t)$ lower bound [Agarwal et al., 2012].

1.3.2 Cutting plane

Another approach that gained significant popularity for solving the learning problem is the cutting plane method. This approach is based on the following equivalent formulation of the learning problem:

$$\begin{aligned} &\min_{w,\xi\geq 0} \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_m \xi_m \\ &\text{s.t.} \quad \xi_m \geq w^\top\phi(x^m, y) + \Delta(y, y^m) - w^\top\phi(x^m, y^m) \quad \forall m, y \end{aligned} \tag{1.15}$$

where ξ_m's are called 'slack variables'. This optimization problem has the form of a quadratic program, but this program has an exponential number of constraints per training example, one for each possible configuration y. It turns out that in this case there exists a small set of constraints which suffices to determine the optimal solution. Intuitively, if we knew that set of relevant constraints in advance we could solve the optimization problem in Eq. (1.15) efficiently by using only this small set of constraints. *Cutting plane* is a constraint generation technique which is based precisely on this idea. The algorithm alternates between finding new constraints which are violated for the current weight vector, and optimizing the weights with respect to the active set of constraints. If at some point no violating constraints are found, then the optimal solution has been found and the algorithm terminates. This procedure results in the following algorithm:

Algorithm 2 The cutting plane algorithm

```
for $m = 1, \dots, M$ do
    Set: $A^m = \emptyset$
end for
repeat
    for $m = 1, \dots, M$ do
        $\hat{y}^m = \text{argmax}_y \left(w^\top\phi(x^m, y) + \Delta(y, y^m)\right)$
        if $\xi_m + \epsilon < w^\top\phi(x^m, \hat{y}^m) + \Delta(\hat{y}^M, y^m) - w^\top\phi(x^m, y^m)$ then
            $A^m = A^m \cup \{\hat{y}^m\}$
        end if
        $(w, \xi) \leftarrow$ solve Eq. (1.15) using constraints from $A$
    end for
until $A$ has not changed during the last iteration
return $w$
```

Notice that finding the most violated constraint boils down to solving the loss-augmented prediction problem, as in the stochastic subgradient algorithm. In addition, to use this algorithm one has to solve quadratic programs over the active set of constraints. As long as this constraint set is not too large, general-purpose QP solvers can be used.

In terms of computational complexity, it has been shown that adding $O(M/\epsilon^2)$ constraints is sufficient to obtain an ϵ-accurate solution. This yields a convergence rate of $O(\sqrt{M/t})$, where t is the number of iterations of Algorithm 2 [Tsochantaridis et al., 2005]. Notice that this rate is significantly slower than that of SGD, and moreover, it depends on the size of the training set M which may be large. The convergence rate can be improved by replacing the sum over slack variables in Eq. (1.15) with a single slack variable. This formulation yields a better $O(1/t)$ rate [Joachims et al., 2009], however each iteration requires a pass over the entire training set, so it still depends linearly on M.

1.3.3 The stochastic Frank-Wolfe algorithm

More recently, Lacoste-Julien et al. [2013] proposed to solve the learning problem Eq. (1.12) using a stochastic version of the classic Frank-Wolfe algorithm [Frank and Wolfe, 1956]. The algorithm essentially performs stochastic coordinate ascent on the dual function. In particular, the Lagrange dual of the problem in Eq. (1.12) is the following quadratic program:

$$\begin{aligned} &\min_{\alpha \geq 0} \frac{\lambda}{2} \|A\alpha\|^2 - b^\top \alpha \\ &\text{s.t.} \quad \sum_y \alpha_m(y) = 1 \quad \forall m \end{aligned} \tag{1.16}$$

In this dual problem, each variable $\alpha_m(y)$ corresponds to a training example m and possible output y, and these have to form a distribution over outputs (i.e., obey simplex constraints per training example). Notice that the number of dual variables is usually very large, so we will generally not be able to explicitly represent them in memory. The (m, y)'th column of the matrix A consists of the vector $\frac{1}{\lambda M}(\phi(x^m, y^m) - \phi(x^m, y))$, and finally, the (m, y)'th element of the vector b is given by $\frac{1}{M}\Delta(y, y^m)$. Using standard Lagrangian duality, it is possible to obtain a mapping from dual variables to primal variables by: $w = A\alpha$.

The Frank-Wolfe algorithm [Frank and Wolfe, 1956] is an iterative procedure for constrained optimization problems such as Eq. (1.16). In each iteration the algorithm first finds a linearization of the convex function at the current point α, it then minimizes the linear objective subject to the constraints to obtain the solution s, and finally, the next point is found as a convex combination of α and s with some step size.

Lacoste-Julien et al. [2013] show that this algorithm can be applied in a stochastic manner, where the weight updates depend on single training examples. Moreover, due to the simple mapping from dual to primal variables, the algorithm can be executed by maintaining only primal variables, which can be done efficiently. Finally, it turns out that the optimal step-size can be found analytically in the case of the dual problem Eq. (1.16). The resulting algorithm is described in Algorithm 3.

As in the previous learning algorithms, each iteration requires solving the loss-augmented prediction problem (finding $\hat{y}^m$). However, this procedure has two main advantages over SGD. First, it provides a computable duality gap between the dual and primal objectives, which can be used as a sound stopping criterion. Specifically, the duality gap is given by: $\sum_m \lambda(w_t - w_t^m)^\top w_t - l_t + l_t^m$. Second, the optimal step size (γ) can be computed analytically, which can

Algorithm 3 Stochastic Frank-Wolfe algorithm

```
Set: $w_0 = w_0^m = 0$
Set: $l_0 = l_0^m = 0$
for $t = 1, \dots, T$ do
    Pick a training example $m$ uniformly at random
    $\hat{y}^m = \text{argmax}_y \left( w_t^\top \phi(x^m, y) + \Delta(y, y^m) \right)$
    $w_s = \frac{1}{\lambda M}(\phi(x^m, y^m) - \phi(x^m, \hat{y}^m))$ and $l_s = \frac{1}{M}\Delta(\hat{y}^m, y^m)$
    $\gamma = \frac{\lambda(w_t^m - w_s)^\top w_t - l_t^m + l_s}{\lambda \|w_t^m - w_s\|^2}$ and clip to $[0, 1]$
    Update: $w_{t+1}^m = (1-\gamma) w_t^m + \gamma w_s$ and $l_{t+1}^m = (1-\gamma) l_t^m + \gamma l_s$
    Update: $w_{t+1} = w_t + w_{t+1}^m - w_t^m$ and $l_{t+1} = l_t + l_{t+1}^m - l_t^m$
end for
return $w_T$
```

help practitioners save the laborious search for effective values.

The convergence rate of this stochastic version of the Frank-Wolfe algorithm has been shown to be upper bounded by $O(1/t)$ [Lacoste-Julien et al., 2013]. This rate is similar to that of SGD, although empirically they show that for several problems it converges much faster.

1.3.4 Approximate learning

All of the learning algorithms described above require solving the loss-augmented prediction problem in each iteration. When one wishes to use rich, expressive, complex models this task usually becomes hard and requires resorting to some kind of approximation. Thus, one of the main challenges in structured output prediction is to balance tractability and expressivity. At one extreme, we can independently predict each output variable. In this case both inference and learning are computationally cheap. In fact, if errors are measured using the Hamming distance (the number of misclassified variables), then this simple strategy is sufficient for obtaining an optimal predictor [Dembczynski et al., 2010]. However, such an approach may result in poor prediction quality when other error measures are used, such as the zero-one loss. At the other extreme, one can model complex dependencies between many of the variables, which may result in a potentially accurate predictor.[8] Even if we had some magical way to learn the exact optimal parameters for such a model,[9] we would still need to perform predictions with that model. If we eventually use approximate inference for prediction then it is generally not the case that this optimal model would have the best performance [Wainwright, 2006]. In fact, it will often be better to use the same approximate inference scheme when training the model. This is actually intuitive: assuming that the inference algorithm will make mistakes at test time, we would like the learned model to be aware of those mistakes and possibly try to correct them.

Perhaps the simplest approach is to treat inference as a black-box within the learning procedure and simply call an approximate inference routine, such as loopy BP. However, it

[8]For a thorough comparison between these two extremes see Liang et al. [2008].

[9]In fact, when the data is 'nice' then there is a way to learn the optimal model efficiently [Sontag et al., 2010].

turns out that even when the inference algorithm enjoys guaranteed approximation quality,[10] it can still lead to arbitrarily bad learning results. In particular, Kulesza and Pereira [2008] show that the combination of Perceptron learning with loopy belief propagation inference can lead to divergence even when inference yields a 1.1-approximation and the data is separable. On the other hand, they provide a generalization bound for the case of solving Eq. (1.12) using LP relaxation for approximate inference. These results highlight the importance of choosing compatible inference and learning procedures.

Finley and Joachims [2008] also study the learning problem when exact inference is intractable. They distinguish between *under-generating* and *over-generating* techniques. Under-generating techniques, like local search methods or loopy BP, search only over a subset of possible labels. In contrast, over-generating techniques, like LP-relaxation, search over a set that is larger than the original output space (e.g., includes also fractional solutions of the LP). The difference between the two can be best understood by considering the slack formulation of the learning problem (Eq. (1.15)). In particular, the number of constraints on w in under-generating approximations is smaller than that of the exact form (Eq. (1.15)), which in turn is smaller than the number of constraints in over-generating approximations. Notice that learning with over-generating approximations makes the learning problem harder than it actually is, which can potentially improve the generalization power of the learned model. Indeed, Finley and Joachims [2008] report good empirical performance for this approach.

The usage of LP relaxation for approximate learning is of particular interest in this thesis. In terms of approximation quality, it has been shown that the LP relaxation is often tight for real world problems, namely the optimal solution to the LP at test time happens to be integral [Martins et al., 2009a, Koo et al., 2010]. In contrast, training with exact inference and using LP relaxation at test time often yields poor results. Formalizing and understanding these phenomena is the subject of ongoing research [see for example Chandrasekaran and Jordan, 2012]. In addition, it has been observed that training with LP relaxation yields faster test time prediction than with exact training, even when using exact inference at test time [Martins et al., 2009a]. Intuitively, since LP relaxation is conservative (by over-constraining w), it tends to make things easier for other prediction methods as well [Finley and Joachims, 2008]. In terms of computational cost of learning with LP relaxation, we show in Section 2.3 that using the dual LP Eq. (1.10) instead of the primal LP during training yields a more convenient optimization problem, which results in significant speedups over alternative approaches.

We have also studied the usage of under-generating techniques. In particular, we have shown that using a very simple and intuitive search strategy, which we termed *pseudo-max*, leads to learning the optimal (exact) model under mild assumptions and when enough training data is available [Sontag et al., 2010].[11]

An alternative approach to the above is to restrict the learned model to belong to a tractable family. Although this limits model expressiveness, for some problems a restricted model that

[10]Such guarantee is generally NP-hard to obtain.

[11]That work is not included in this thesis.

considers a subset of all possible dependencies between output variables may be adequate. Some examples of tractable families which have been learned in the literature are: submodular scores [Taskar et al., 2004a], arithmetic circuits [Lowd and Domingos, 2008], and tree-structured graphs [Bradley and Guestrin, 2010, Chechetka and Guestrin, 2010]. In Section 2.4 we address the problem of learning tree-strucutured models when optimizing a max-margin objective. We show that this problem is NP-hard and propose an approximation scheme. Identifying new tractable families and efficient ways for learning them is also the focus of current research on structured prediction.

Chapter 2

Results

In this chapter I present the main results of this thesis. The chapter is organized as follows: Sections 2.1 and 2.2 address the prediction problem, where Section 2.1 presents a new MAP inference algorithm based on applying the alternating direction method of multipliers to the dual LP relaxation, and Section 2.2 provides convergence rate analysis for coordinate descent algorithms on the smooth dual LP relaxation. Sections 2.3 and 2.4 address the learning problem, where Section 2.3 proposes an efficient learning algorithm based on the dual LP relaxation, and Section 2.4 addresses the problem of learning tree-structured predictors.

For convenience, here is the list of the papers and venues in which they were published:

- Section 2.1: An alternating direction method for dual MAP LP relaxation. Ofer Meshi and Amir Globerson. European Conference on Machine Learning (ECML), 2011.

- Section 2.2: Convergence Rate Analysis of MAP Coordinate Minimization Algorithms. Ofer Meshi, Tommi Jaakkola and Amir Globerson. Neural Information Processing Systems (NIPS), 2012.

- Section 2.3: Learning efficiently with approximate inference via dual losses. Ofer Meshi, David Sontag, Tommi Jaakkola and Amir Globerson. International Conference on Machine Learning (ICML), 2010.

- Section 2.4: Learning Max-Margin Tree Predictors. Ofer Meshi, Elad Eban, Gal Elidan and Amir Globerson. Uncertainty in Artificial intelligence (UAI), 2013.

For background, I also provide a list of my other publications during the doctoral studies which are not included in this thesis:

- Convexifying the Bethe Free Energy. Ofer Meshi, Ariel Jaimovich, Amir Globerson and Nir Friedman. Uncertainty in Artificial Intelligence (UAI), 2009.

- FastInf: An efficient approximate inference library. Ariel Jaimovich, Ofer Meshi, Ian McGraw and Gal Elidan. Journal of Machine Learning Research, 2010.

- More data means less inference: A pseudo-max approach to structured learning. David Sontag, Ofer Meshi, Tommi Jaakkola and Amir Globerson. Neural Information Processing Systems (NIPS), 2010.

2.1 An Alternating Direction Method for Dual MAP LP Relaxation

An Alternating Direction Method for Dual MAP LP Relaxation

Ofer Meshi and Amir Globerson

The School of Computer Science and Engineering,
The Hebrew University of Jerusalem, Jerusalem, Israel
{meshi,gamir}@cs.huji.ac.il

Abstract. Maximum a-posteriori (MAP) estimation is an important task in many applications of probabilistic graphical models. Although finding an exact solution is generally intractable, approximations based on linear programming (LP) relaxation often provide good approximate solutions. In this paper we present an algorithm for solving the LP relaxation optimization problem. In order to overcome the lack of strict convexity, we apply an augmented Lagrangian method to the dual LP. The algorithm, based on the alternating direction method of multipliers (ADMM), is guaranteed to converge to the global optimum of the LP relaxation objective. Our experimental results show that this algorithm is competitive with other state-of-the-art algorithms for approximate MAP estimation.

Keywords: Graphical Models, Maximum a-posteriori, Approximate Inference, LP Relaxation, Augmented Lagrangian Methods

1 Introduction

Graphical models are widely used to describe multivariate statistics for discrete variables, and have found widespread applications in numerous domains. One of the basic inference tasks in such models is to find the *maximum a-posteriori* (MAP) assignment. Unfortunately, this is typically a hard computational problem which cannot be solved exactly for many problems of interest. It has turned out that *linear programming* (LP) relaxations provide effective approximations to the MAP problem in many cases (e.g., see [15, 21, 24]).

Despite the theoretical computational tractability of MAP-LP relaxations, solving them in practice is a challenge for real world problems. Using off-the-shelf LP solvers is typically inadequate for large models since the resulting LPs have too many constraints and variables [29]. This has led researchers to seek optimization algorithms that are tailored to the specific structure of the MAP-LP [7, 13, 14, 16, 20, 28]. The advantage of such methods is that they work with very simple local updates and are therefore easy to implement in the large scale setting.

The suggested algorithms fall into several classes, depending on their approach to the problem. The TRW-S [14], MSD [28] and MPLP [7] algorithms employ coordinate descent in the dual of the LP. While these methods typically show good empirical behavior, they are not guaranteed to reach the global optimum of the LP relaxation. This is a result of non strict-convexity of the dual LP and the fact that block coordinate descent might get stuck in suboptimal points under these conditions. One way to avoid this problem is to use a *soft-max* function which is smooth and strictly convex, hence this results in globally convergent algorithms [6, 10, 12]. Another class of algorithms [13, 16] uses the same dual objective, but employs variants of subgradient descent to it. While these methods are guaranteed to converge globally, they are typically slower in practice than the coordinate descent ones (e.g., see [13] for a comparison). Finally, there are also algorithms that optimize the primal LP directly. One example is the proximal point method of Ravikumar et al. [20]. While also globally convergent, it has the disadvantage of using a double loop scheme where every update involves an iterative algorithm for projecting onto the local polytope.

More recently, Martins et al. [17] proposed a globally convergent algorithm for MAP-LP based on the *alternating direction method of multipliers* (ADMM) [8, 5, 4, 2]. This method proceeds by iteratively updating primal and dual variables in order to find a saddle point of an *augmented Lagrangian* for the problem. They suggest to use an augmented Lagrangian of the *primal* MAP-LP problem. However, their formulation is restricted to binary pairwise factors and several specific global factors. In this work, we propose an algorithm that is based on the same key idea of ADMM, however it stems from augmenting the Lagrangian of the *dual* MAP-LP problem instead. An important advantage of our approach is that the resulting algorithm can be applied to models with *general local factors* (non-pairwise, non-binary). We also show that in practice our algorithm converges much faster than the primal ADMM algorithm and that it compares favorably with other state-of-the-art methods for MAP-LP optimization.

2 MAP and LP relaxation

Markov Random Fields (MRFs) are probabilistic graphical models that encode the joint distribution of a set of discrete random variables $\mathcal{X} = \{X_1, ..., X_n\}$. The joint probability is defined by combining a set C of local functions $\theta_c(x_c)$, termed *factors*. The factors depend only on (small) subsets of the variables ($X_c \subseteq \mathcal{X}$) and model the direct interactions between them (to simplify notation we drop the variable name in $X_c = x_c$; see [27]). The joint distribution is then given by: $P(x) \propto \exp\left(\sum_i \theta_i(x_i) + \sum_{c\in C} \theta_c(x_c)\right)$, where we have included also singleton factors over individual variables [27]. In many applications of MRFs we are interested in finding the maximum probability assignment (MAP assignment). This yields the optimization problem:

$$\arg\max_x \sum_i \theta_i(x_i) + \sum_{c\in C} \theta_c(x_c)$$

Due to its combinatorial nature, this problem is NP-hard for general graphical models, and tractable only in isolated cases such as tree structured graphs. This has motivated research on approximation algorithms.

One of the most successful approximation schemes has been to use LP relaxations of the MAP problem. In this approach the original combinatorial problem is posed as a LP and then some of the constraints are relaxed to obtain a tractable LP problem that approximates the original one. In our case, the resulting MAP-LP relaxation problem is:

$$\max_{\mu \in L(G)} \sum_i \sum_{x_i} \mu_i(x_i)\theta_i(x_i) + \sum_c \sum_{x_c} \mu_c(x_c)\theta_c(x_c) \tag{1}$$

where μ are auxiliary variables that correspond to (pseudo) marginal distributions, and $L(G)$ is the reduced set of constraints called the *local polytope* [27], defined by:

$$L(G) = \left\{ \mu \geq 0 \,\middle|\, \begin{array}{ll} \sum_{x_{c\setminus i}} \mu_c(x_{c\setminus i}, x_i) = \mu_i(x_i) & \forall c, i : i \in c, x_i \\ \sum_{x_i} \mu_i(x_i) = 1 & \forall i \end{array} \right\}$$

In this paper we use the dual problem of Eq. (1), which takes the form:

$$\min_{\delta} \sum_i \max_{x_i} \left(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i) \right) + \sum_c \max_{x_c} \left(\theta_c(x_c) - \sum_{i:i\in c} \delta_{ci}(x_i) \right) \tag{2}$$

where δ are dual variables corresponding to the marginalization constraints in $L(G)$ (see [22, 28, 23]).[1] This formulation offers several advantages. First, it minimizes an upper bound on the true MAP value. Second, it provides an optimality certificate through the duality gap w.r.t. a decoded primal solution [23]. Third, the resulting problem is unconstrained, which facilitates its optimization. Indeed, several algorithms have been proposed for optimizing this dual problem. The two main approaches are block coordinate descent [14, 28, 7] and subgradient descent [16], each with its advantages and disadvantages. In particular, coordinate descent algorithms are typically much faster at minimizing the dual, while the subgradient method is guaranteed to converge to the global optimum (see [23] for in-depth discussion).

Recently, Jojic et al. [13] presented an accelerated dual decomposition algorithm which stems from adding strongly convex smoothing terms to the subproblems in the dual function Eq. (2). Their method achieves a better convergence rate over the standard subgradient method ($O\left(\frac{1}{\epsilon}\right)$ vs. $O\left(\frac{1}{\epsilon^2}\right)$). An alternative approach, that is also globally convergent, has been recently suggested by Martins et al. [17]. Their approach is based on an augmented Lagrangian method, which we next discuss.

3 The Alternating Direction Method of Multipliers

We now briefly review ADMM for convex optimization [8, 5, 4, 2].

[1] An equivalent optimization problem can be derived via a dual decomposition approach [23].

Consider the following optimization problem:

$$\text{minimize } f(x) + g(z) \quad \text{s.t. } Ax = z \tag{3}$$

where f and g are convex functions. The ADMM approach begins by adding the function $\frac{\rho}{2}\left\|Ax - z\right\|^2$ to the above objective, where $\rho > 0$ is a penalty parameter. This results in the optimization problem:

$$\text{minimize } f(x) + g(z) + \frac{\rho}{2}\left\|Ax - z\right\|^2 \quad \text{s.t. } Ax = z \tag{4}$$

Clearly the above has the same optimum as Eq. (3) since when the constraints $Ax = z$ are satisfied, the added quadratic term equals zero. The Lagrangian of the augmented problem Eq. (4) is given by:

$$\mathcal{L}_\rho(x, z, \nu) = f(x) + g(z) + \nu^\top(Ax - z) + \frac{\rho}{2}\left\|Ax - z\right\|^2 \tag{5}$$

where ν is a vector of Lagrange multipliers. The solution to the problem of Eq. (4) is given by $\max_\nu \min_{x,z} \mathcal{L}_\rho(x, z, \nu)$. The ADMM method provides an elegant algorithm for finding this saddle point. The idea is to combine subgradient descent over ν with coordinate descent over the x and z variables. The method applies the following iterations:

$$\begin{aligned} x^{t+1} &= \arg\min_x \mathcal{L}_\rho(x, z^t, \nu^t) \\ z^{t+1} &= \arg\min_z \mathcal{L}_\rho(x^{t+1}, z, \nu^t) \\ \nu^{t+1} &= \nu^t + \rho\left(Ax^{t+1} - z^{t+1}\right) \end{aligned} \tag{6}$$

The algorithm consists of primal and dual updates, where the primal update is executed sequentially, minimizing first over x and then over z. This split retains the decomposition of the objective that has been lost due to the addition of the quadratic term.

The algorithm is run either until the number of iterations exceeds a predefined limit, or until some termination criterion is met. A commonly used such stopping criterion is: $\left\|Ax - z\right\|^2 \leq \epsilon$ and $\left\|z^{t+1} - z^t\right\|^2 \leq \epsilon$. These two conditions can serve to bound the suboptimality of the solution.

The ADMM algorithm is guaranteed to converge to the global optimum of Eq. (3) under rather mild conditions [2]. However, in terms of convergence rate, the worst case complexity of ADMM is $O(\frac{1}{\epsilon^2})$. Despite this potential caveat, ADMM has been shown to work well in practice (e.g., [1, 26]). Recently, accelerated variants on the basic alternating direction method have been proposed [9]. These faster algorithms are based on linearization and come with improved convergence rate of $O(\frac{1}{\epsilon})$, achieving the theoretical lower bound for first-order methods [19]. In this paper we focus on the basic ADMM formulation and leave derivation of accelerated variants to future work.

4 The Augmented Dual LP Algorithm

In this section we derive our algorithm by applying ADMM to the dual MAP-LP problem of Eq. (2). The challenge is to design the constraints in a way that facilitates efficient closed-form solutions for all updates.

To this end, we duplicate the dual variables δ and denote the second copy by $\bar{\delta}$. We then introduce additional variables λ_c corresponding to the summation of δ's pertaining to factor c. These agreement constraints are enforced through $\bar{\delta}$, and thus we have a constraint $\delta_{ci}(x_i) = \bar{\delta}_{ci}(x_i)$ for all $c, i : i \in c, x_i$, and $\lambda_c(x_c) = \sum_{i:i\in c} \bar{\delta}_{ci}(x_i)$ for all c, x_c.

Following the ADMM framework, we add quadratic terms and obtain the augmented Lagrangian for the dual MAP-LP problem of Eq. (2):

$$
\begin{aligned}
&\mathcal{L}_\rho(\delta, \lambda, \bar{\delta}, \gamma, \mu) = \\
&\sum_i \max_{x_i} \left(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i) \right) + \sum_c \max_{x_c} \left(\theta_c(x_c) - \lambda_c(x_c) \right) \\
&+ \sum_c \sum_{i:i\in c} \sum_{x_i} \gamma_{ci}(x_i) \left(\delta_{ci}(x_i) - \bar{\delta}_{ci}(x_i) \right) + \frac{\rho}{2} \sum_c \sum_{i:i\in c} \sum_{x_i} \left(\delta_{ci}(x_i) - \bar{\delta}_{ci}(x_i) \right)^2 \\
&+ \sum_c \sum_{x_c} \mu_c(x_c) \left(\lambda_c(x_c) - \sum_{i:i\in c} \bar{\delta}_{ci}(x_i) \right) + \frac{\rho}{2} \sum_c \sum_{x_c} \left(\lambda_c(x_c) - \sum_{i:i\in c} \bar{\delta}_{ci}(x_i) \right)^2
\end{aligned}
$$

To see the relation of this formulation to Eq. (5), notice that (δ, λ) subsume the role of x, $\bar{\delta}$ subsumes the role of z (with $g(z) = 0$), and the multipliers (γ, μ) correspond to ν.

The updates of our algorithm, which stem from Eq. (6), are summarized in Alg. 1 (a detailed derivation appears in Appendix A). In Alg. 1 we define $N(i) = \{c : i \in c\}$, and the subroutine $w = \text{TRIM}(v, d)$ that serves to clip the values in the vector v at some threshold t (i.e., $w_i = \min\{v_i, t\}$) such that the sum of removed parts equals $d > 0$ (i.e., $\sum_i v_i - w_i = d$). This can be carried out efficiently in linear time (in expectation) by partitioning [3].

Notice that all updates can be computed efficiently so the cost of each iteration is similar to that of message passing algorithms like MPLP [7] or MSD [28], and to that of dual decomposition [13, 16]. Furthermore, significant speedup is attained by caching some results for future iterations. In particular, the threshold in the TRIM subroutine (the new maximum) can serve as a good initial guess at the next iteration, especially at later iterations where the change in variable values is quite small. Finally, many of the updates can be executed in parallel. In particular, the δ update can be carried out simultaneously for all variables i, and likewise all factors c can be updated simultaneously in the λ and $\bar{\delta}$ updates. In addition, δ and λ can be optimized independently, since they appear in different parts of the objective. This may result in considerable reduction in runtime when executed on parallel architecture.[2]

[2] In our experiments we used sequential updates.

Algorithm 1 The Augmented Dual LP Algorithm (ADLP)

```
for $t = 1$ to $T$ do
    Update $\delta$: for all $i = 1, ..., n$
    Set $\bar{\theta}_i = \theta_i + \sum_{c:i\in c}(\bar{\delta}_{ci} - \frac{1}{\rho}\gamma_{ci})$
    $\bar{\theta}'_i = \text{TRIM}(\bar{\theta}_i, \frac{|N(i)|}{\rho})$
    $q = (\bar{\theta}_i - \bar{\theta}'_i)/|N(i)|$
    Update $\delta_{ci} = \bar{\delta}_{ci} - \frac{1}{\rho}\gamma_{ci} - q \quad \forall c : i \in c$

    Update $\lambda$: for all $c \in C$
    Set $\bar{\theta}_c = \theta_c - \sum_{i:i\in c}\bar{\delta}_{ci} + \frac{1}{\rho}\mu_c$
    $\bar{\theta}'_c = \text{TRIM}(\bar{\theta}_c, \frac{1}{\rho})$
    Update $\lambda_c = \theta_c - \bar{\theta}'_c$

    Update $\bar{\delta}$: for all $c \in C, i : i \in c, x_i$
    Set $v_{ci}(x_i) = \delta_{ci}(x_i) + \frac{1}{\rho}\gamma_{ci}(x_i) + \sum_{x_{c\setminus i}}\lambda_c(x_{c\setminus i}, x_i) + \frac{1}{\rho}\sum_{x_{c\setminus i}}\mu_c(x_{c\setminus i}, x_i)$
    $\bar{v}_c = \frac{1}{1+\sum_{k:k\in c}|X_{c\setminus k}|}\sum_{k:k\in c}|X_{c\setminus k}|\sum_{x_k}v_{ck}(x_k)$
    Update $\bar{\delta}_{ci}(x_i) = \frac{1}{1+|X_{c\setminus i}|}\left[v_{ci}(x_i) - \sum_{j:j\in c, j\neq i}|X_{c\setminus\{i,j\}}|\left(\sum_{x_j}v_{cj}(x_j) - \bar{v}_c\right)\right]$

    Update the multipliers:
    $\gamma_{ci}(x_i) \leftarrow \gamma_{ci}(x_i) + \rho\left(\delta_{ci}(x_i) - \bar{\delta}_{ci}(x_i)\right)$    for all $c \in C, i : i \in c, x_i$
    $\mu_c(x_c) \leftarrow \mu_c(x_c) + \rho\left(\lambda_c(x_c) - \sum_{i:i\in c}\bar{\delta}_{ci}(x_i)\right)$ for all $c \in C, x_c$
end for
```

5 Experimental Results

To evaluate our augmented dual LP (ADLP) algorithm (Alg. 1) we compare it to two other algorithms for finding an approximate MAP solution. The first is MPLP of Globerson and Jaakkola [7], which minimizes the dual LP of Eq. (2) via block coordinate descent steps (cast as message passing). The second is the accelerated dual decomposition (ADD) algorithm of Jojic et al. [13].[3] We conduct experiments on protein design problems from the dataset of Yanover et al. [29]. In these problems we are given a 3D structure and the goal is to find a sequence of amino-acids that is the most stable for that structure. The problems are modeled by singleton and pairwise factors and can be posed as finding a MAP assignment for the given model. This is a demanding setting in which each problem may have hundreds of variables with 100 possible states on average [29, 24].

Figure 1 shows two typical examples of protein design problems. It plots the objective of Eq. (2) (computed using δ variables only) as a function of the execution time for all algorithms. First, in Figure 1 (left) we observe that the coordinate descent algorithm (MPLP) converges faster than the other algorithms,

[3] For both algorithms we used the same `C++` implementation used by Jojic et al. [13], available at `http://ai.stanford.edu/~sgould/svl`. Our own algorithm was implemented as an extension of their package.

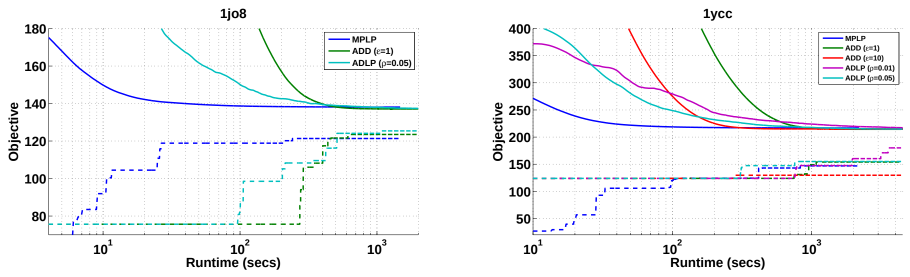


Fig. 1. Comparison of three algorithms for approximate MAP estimation: our augmented dual LP algorithm (ADLP), accelerated dual decomposition algorithm (ADD) by Jojic et al. [13], and the dual coordinate descent MPLP algorithm [7]. The figure shows two examples of protein design problems, for each the dual objective of Eq. (2) is plotted as a function of execution time. Dashed lines denote the value of the best decoded primal solution.

however it tends to stop prematurely and yield suboptimal solutions. In contrast, ADD and ADLP take longer to converge but achieve the globally optimal solution to the approximate objective. Second, it can be seen that the convergence times of ADD and ADLP are very close, with a slight advantage to ADD. The dashed lines in Figure 1 show the value of the decoded primal solution (assignment) [23]. We see that there is generally a correlation between the quality of the dual objective and the decoded primal solution, namely the decoded primal solution improves as the dual solution approaches optimality. Nevertheless, we note that there is no dominant algorithm in terms of decoding (here we show examples where our decoding is superior). In many cases MPLP yields better decoded solutions despite being suboptimal in terms of the dual objective (not shown; this is also noted in [13]).

We also conduct experiments to study the effect of the penalty parameter ρ. Our algorithm is guaranteed to globally converge for all $\rho > 0$, but its choice affects the actual rate of convergence. In Figure 1 (right) we compare two values of the penalty parameter $\rho = 0.01$ and $\rho = 0.05$. It shows that setting $\rho = 0.01$ results in somewhat slower convergence to the optimum, however in this case the final primal solution (dashed line) is better than that of the other algorithms. In practice, in order to choose an appropriate ρ, one can run a few iterations of ADLP with several values and see which one achieves the best objective [17]. We mention in passing that ADD employs an accuracy parameter ϵ which determines the desired suboptimality of the final solution [13]. Setting ϵ to a large value results in faster convergence to a lower accuracy solution. On the one hand, this trade-off can be viewed as a merit of ADD, which allows to obtain coarser approximations at reduced cost. On the other hand, an advantage of our method is that the choice of penalty ρ affects only the rate of convergence and does not impose additional reduction in solution accuracy over that of the LP relaxation. In Figure 1 (left) we use $\epsilon = 1$, as in Jojic et al., while in Figure 1

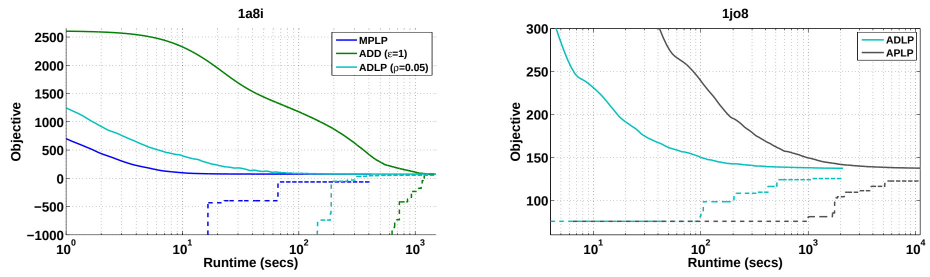


Fig. 2. (Left) Comparison for a side chain prediction problem similar to Figure 1 (left). (Right) Comparison of our augmented dual LP algorithm (ADLP) and a generalized variant (APLP) of the ADMM algorithm by Martins et al. [17] on a protein design problem. The dual objective of Eq. (2) is plotted as a function of execution time. Dashed lines denote the value of the best decoded primal solution.

(right) we compare two values $\epsilon = 1$ and $\epsilon = 10$ to demonstrate the effect of this accuracy parameter.

We next compare performance of the algorithms on a side chain prediction problem [29]. This problem is the inverse of the protein design problem, and involves finding the 3D configuration of rotamers given the backbone structure of a protein. Figure 2 (left) shows a comparison of MPLP, ADD and ADLP on one of the largest proteins in the dataset (812 variables with 12 states on average). As in the protein design problems, MPLP converges fast to a suboptimal solution. We observe that here ADLP converges somewhat faster than ADD, possibly because the smaller state space results in faster ADLP updates.

As noted earlier, Martins et al. [17] recently presented an approach that applies ADMM to the primal LP (i.e., Eq. (1)). Although their method is limited to binary pairwise factors (and several global factors), it can be modified to handle non-binary higher-order factors, as the derivation in Appendix B shows. We denote this variant by APLP. As in ADLP, in the APLP algorithm all updates are computed analytically and executed efficiently. Figure 2 (right) shows a comparison of ADLP and APLP on a protein design problem. It illustrates that ADLP converges significantly faster than APLP (similar results, not shown here, are obtained for the other proteins).

6 Discussion

Approximate MAP inference methods based on LP relaxation have drawn much attention lately due to their practical success and attractive properties. In this paper we presented a novel globally convergent algorithm for approximate MAP estimation via LP relaxation. Our algorithm is based on the augmented Lagrangian method for convex optimization, which overcomes the lack of strict convexity by adding a quadratic term to smooth the objective. Importantly, our algorithm proceeds by applying simple to implement closed-form updates, and it is highly scalable and parallelizable. We have shown empirically that our algorithm compares favorably with other state-of-the-art algorithms for approximate MAP estimation in terms of accuracy and convergence time.

Several existing globally convergent algorithms for MAP-LP relaxation rely on adding local entropy terms in order to smooth the objective [6, 10, 12, 13]. Those methods must specify a temperature control parameter which affects the quality of the solution. Specifically, solving the optimization subproblems at high temperature reduces solution accuracy, while solving them at low temperature might raise numerical issues. In contrast, our algorithm is quite insensitive to the choice of such control parameters. In fact, the penalty parameter ρ affects the rate of convergence but not the accuracy or numerical stability of the algorithm. Moreover, despite lack of fast convergence rate guarantees, in practice the algorithm has similar or better convergence times compared to other globally convergent methods in various settings. Note that [17] also show an advantage of their primal based ADMM method over several baselines.

Several improvements over our basic algorithm can be considered. One such improvement is to use smart initialization of the variables. For example, since MPLP achieves larger decrease in objective at early iterations, it is possible to run it for a limited number of steps and then take the resulting variables δ for the initialization of ADLP. Notice, however, that for this scheme to work well, the Lagrange multipliers γ and μ should be also initialized accordingly. Another potential improvement is to use an adaptive penalty parameter ρ_t (e.g., [11]). This may improve convergence in practice, as well as reduce sensitivity to the initial choice of ρ. On the downside, the theoretical convergence guarantees of ADMM no longer hold in this case. Martins et al. [17] show that the ADMM framework is also suitable for handling certain types of global factors, which include a large number of variables in their scope (e.g., XOR factor). Using an appropriate formulation, it is possible to incorporate such factors in our dual LP framework as well.[4] Finally, it is likely that our method can be further improved by using recently introduced accelerated variants of ADMM [9]. Since these variants achieve asymptotically better convergence rate, the application of such methods to MAP-LP similar to the one we presented here will likely result in faster algorithms for approximate MAP estimation.

In this paper, we assumed that the model parameters were given. However, in many cases one wishes to learn these from data, for example by minimizing a prediction loss (e.g., hinge loss [25]). We have recently shown how to incorporate dual relaxation algorithms into such learning problems [18]. It will be interesting to apply our ADMM approach in this setting to yield an efficient learning algorithm for structured prediction problems.

Acknowledgments. We thank Ami Wiesel and Elad Eban for useful discussions and comments on this manuscript. We thank Stephen Gould for his SVL code. Ofer Meshi is a recipient of the Google European Fellowship in Machine Learning, and this research is supported in part by this Google Fellowship.

[4] The auxiliary variables λ_c are not used in this case.

A Derivation of Augmented Dual LP Algorithm

In this section we derive the ADMM updates for the augmented Lagrangian of the dual MAP-LP which we restate here for convenience:

$$\begin{aligned}
&\mathcal{L}_\rho(\delta, \lambda, \bar{\delta}, \gamma, \mu) = \\
&\sum_i \max_{x_i} \left(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i) \right) + \sum_c \max_{x_c} \left(\theta_c(x_c) - \lambda_c(x_c)\right) \\
&+ \sum_c \sum_{i:i\in c} \sum_{x_i} \gamma_{ci}(x_i) \left(\delta_{ci}(x_i) - \bar{\delta}_{ci}(x_i)\right) + \frac{\rho}{2} \sum_c \sum_{i:i\in c} \sum_{x_i} \left(\delta_{ci}(x_i) - \bar{\delta}_{ci}(x_i)\right)^2 \\
&+ \sum_c \sum_{x_c} \mu_c(x_c) \left(\lambda_c(x_c) - \sum_{i:i\in c} \bar{\delta}_{ci}(x_i) \right) + \frac{\rho}{2} \sum_c \sum_{x_c} \left(\lambda_c(x_c) - \sum_{i:i\in c} \bar{\delta}_{ci}(x_i) \right)^2
\end{aligned}$$

Updates:

– **The δ update:**
For each variable $i = 1, ..., n$ consider a block δ_i which consists of δ_{ci} for all $c : i \in c$. For this block we need to minimize the following function:

$$\max_{x_i} \left(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i) \right) + \sum_{c:i\in c} \sum_{x_i} \gamma_{ci}(x_i)\delta_{ci}(x_i) + \frac{\rho}{2} \sum_{c:i\in c} \sum_{x_i} \left(\delta_{ci}(x_i) - \bar{\delta}_{ci}(x_i)\right)^2$$

Equivalently, this can be written more compactly in vector notation as:

$$\min_{\delta_i} \frac{1}{2}\|\delta_i\|^2 - (\bar{\delta}_i - \frac{1}{\rho}\gamma_i)^\top \delta_i + \frac{1}{\rho} \max_{x_i}(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i))$$

where $\bar{\delta}_i$ and γ_i are defined analogous to δ_i. The closed-form solution to this QP is given by the update in Alg. 1. It is obtained by inspecting KKT conditions and exploiting the structure of the summation inside the max (for a similar derivation see [3]).

– **The λ update:**
For each factor $c \in C$ we seek to minimize the function:

$$\max_{x_c} \left(\theta_c(x_c) - \lambda_c(x_c)\right) + \sum_{x_c} \mu_c(x_c)\lambda_c(x_c) + \frac{\rho}{2} \sum_{x_c} \left(\lambda_c(x_c) - \sum_{i:i\in c} \bar{\delta}_{ci}(x_i) \right)^2$$

In equivalent vector notation we have the problem:

$$\min_{\lambda_c} \frac{1}{2} \|\lambda_c\|^2 - \left(\sum_{i:i\in c} \bar{\delta}_{ci} - \frac{1}{\rho}\mu_c \right)^\top \lambda_c + \frac{1}{\rho} \max_{x_c}(\theta_c(x_c) - \lambda_c(x_c))$$

This QP is very similar to that of the δ update and can be solved using the same technique. The resulting closed-form update is given in Alg. 1.

- **The $\bar{\delta}$ update:**
  For each $c \in C$ we consider a block which consists of $\bar{\delta}_{ci}$ for all $i : i \in c$. We seek a minimizer of the function:
$$
\begin{aligned}
&-\sum_{i:i\in c}\sum_{x_i}\gamma_{ci}(x_i)\bar{\delta}_{ci}(x_i) + \frac{\rho}{2}\sum_{i:i\in c}\sum_{x_i}\left(\delta_{ci}(x_i) - \bar{\delta}_{ci}(x_i)\right)^2 \\
&-\sum_{x_c}\mu_c(x_c)\sum_{i:i\in c}\bar{\delta}_{ci}(x_i) + \frac{\rho}{2}\sum_{x_c}\left(\lambda_c(x_c) - \sum_{i:i\in c}\bar{\delta}_{ci}(x_i)\right)^2
\end{aligned}
$$
  Taking partial derivative w.r.t. $\bar{\delta}_{ci}(x_i)$ and setting to 0 yields:
$$
\bar{\delta}_{ci}(x_i) = \frac{1}{1 + |X_{c\setminus i}|}\left(v_{ci}(x_i) - \sum_{j:j\in c, j\neq i}|X_{c\setminus\{i,j\}}|\sum_{x_j}\bar{\delta}_{cj}(x_j)\right)
$$
  where: $v_{ci}(x_i) = \delta_{ci}(x_i) + \frac{1}{\rho}\gamma_{ci}(x_i) + \sum_{x_{c\setminus i}}\lambda_c(x_{c\setminus i}, x_i) + \frac{1}{\rho}\sum_{x_{c\setminus i}}\mu_c(x_{c\setminus i}, x_i)$. Summing this over x_i and $i : i \in c$ and plugging back in, we get the update in Alg. 1.
- Finally, the multipliers update is straightforward.

B Derivation of Augmented Primal LP Algorithm

We next derive the algorithm for optimizing Eq. (1) with general local factors.

Consider the following formulation which is equivalent to the primal MAP-LP problem of Eq. (1). Define:
$$
f_i(\mu_i) = \begin{cases} \sum_{x_i}\mu_i(x_i)\theta_i(x_i) & \mu_i(x_i) \geq 0 \text{ and } \sum_{x_i}\mu_i(x_i) = 1 \\ -\infty & \text{otherwise} \end{cases}
$$
$$
f_c(\mu_c) = \begin{cases} \sum_{x_c}\mu_c(x_c)\theta_c(x_c) & \mu_c(x_c) \geq 0 \text{ and } \sum_{x_c}\mu_c(x_c) = 1 \\ -\infty & \text{otherwise} \end{cases}
$$

f accounts for the non-negativity and normalization constraints in $L(G)$. We add the marginalization constraints via copies of μ_c for each $i \in c$, denoted by $\bar{\mu}_{ci}$. Thus we get the augmented Lagrangian:
$$
\begin{aligned}
&\mathcal{L}_\rho(\mu, \bar{\mu}, \delta, \beta) = \\
&\sum_i f_i(\mu_i) + \sum_c f_c(\mu_c) \\
&-\sum_c\sum_{i:i\in c}\sum_{x_i}\delta_{ci}(x_i)\left(\bar{\mu}_{ci}(x_i) - \mu_i(x_i)\right) - \frac{\rho}{2}\sum_c\sum_{i:i\in c}\sum_{x_i}\left(\bar{\mu}_{ci}(x_i) - \mu_i(x_i)\right)^2 \\
&-\sum_c\sum_{i:i\in c}\sum_{x_c}\beta_{ci}(x_c)\left(\bar{\mu}_{ci}(x_c) - \mu_c(x_c)\right) - \frac{\rho}{2}\sum_c\sum_{i:i\in c}\sum_{x_c}\left(\bar{\mu}_{ci}(x_c) - \mu_c(x_c)\right)^2
\end{aligned}
$$

where $\bar{\mu}_{ci}(x_i) = \sum_{x_{c\setminus i}} \bar{\mu}_{ci}(x_{c\setminus i}, x_i)$.

To draw the connection with Eq. (5), in this formulation μ subsumes the role of x, $\bar{\mu}$ subsumes the role of z (with $g(z) = 0$), and the multipliers (δ, β) correspond to ν. We next show the updates which result from applying Eq. (6) to this formulation.

– **Update μ_i** for all $i = 1, ..., n$:

$$\mu_i \leftarrow \arg\max_{\mu_i \in \Delta_i} \mu_i^\top \left(\theta_i + \sum_{c:i\in c} (\delta_{ci} + \rho M_i \bar{\mu}_{ci}) \right) - \frac{1}{2} \mu_i^\top (\rho |N(i)| I) \mu_i$$

where $M_i \bar{\mu}_{ci} = \sum_{x_{c\setminus i}} \bar{\mu}_{ci}(x_{c\setminus i}, \cdot)$.
We have to maximize this QP under simplex constraints on μ_i. Notice that the objective matrix is diagonal, so this can be solved in closed form by shifting the target vector and then truncating at 0 such that the sum of positive elements equals 1 (see [3]). The solution can be computed in linear time (in expectation) by partitioning [3].

– **Update μ_c** for all $c \in C$:

$$\mu_c \leftarrow \arg\max_{\mu_c \in \Delta_c} \mu_c^\top \left(\theta_c + \sum_{i:i\in c} (\beta_{ci} + \rho \bar{\mu}_{ci}) \right) - \frac{1}{2} \mu_c^\top (\rho |N(c)| I) \mu_c$$

where $N(c) = \{i : i \in c\}$.
Again we have a projection onto the simplex with diagonal objective matrix, which can be done efficiently.

– **Update $\bar{\mu}_{ci}$** for all $c \in C, i : i \in c$:

$$\bar{\mu}_{ci} \leftarrow \arg\max_{\bar{\mu}_{ci}} \bar{\mu}_{ci}^\top \left(M_i^\top (\rho \mu_i - \delta_{ci}) - \beta_{ci} + \rho \mu_c \right) - \frac{\rho}{2} \bar{\mu}_{ci}^\top \left(M_i^\top M_i + I \right) \bar{\mu}_{ci}$$

Here we have an unconstrained QP, so the solution is obtained by $H^{-1}v$. Further notice that the inverse H^{-1} can be computed in closed form. To see how, $M_i^\top M_i$ is a block-diagonal matrix with blocks of ones with size $|X_i|$. Therefore, $H = \rho \left(M_i^\top M_i + I \right)$ is also block-diagonal. It follows that the inverse H^{-1} is a block-diagonal matrix where each block is the inverse of the corresponding block in H. Finally, it is easy to verify that the inverse of a block $\rho \left(1_{|X_i|} + I_{|X_i|} \right)$ is given by $\frac{1}{\rho} \left(I_{|X_i|} - \frac{1}{|X_i|+1} 1_{|X_i|} \right)$.

– **Update the multipliers**:

$$\begin{aligned} \delta_{ci}(x_i) &\leftarrow \delta_{ci}(x_i) + \rho \left(\bar{\mu}_{ci}(x_i) - \mu_i(x_i) \right) && \text{for all } c \in C, i : i \in c, x_i \\ \beta_{ci}(x_c) &\leftarrow \beta_{ci}(x_c) + \rho \left(\bar{\mu}_{ci}(x_c) - \mu_c(x_c) \right) && \text{for all } c \in C, i : i \in c, x_c \end{aligned}$$

Bibliography

[1] M. Afonso, J. Bioucas-Dias, and M. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *Image Processing, IEEE Transactions on*, 19(9):2345 –2356, sept. 2010.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[3] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279, 2008.

[4] J. Eckstein and D. P. Bertsekas. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318, June 1992.

[5] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. *Computers and Mathematics with Applications*, 2:17–40, 1976.

[6] K. Gimpel and N. A. Smith. Softmax-margin crfs: training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736, 2010.

[7] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems*, pages 553–560, 2008.

[8] R. Glowinski and A. Marrocco. Sur lapproximation, par elements finis dordre un, et la resolution, par penalisation-dualité, dune classe de problems de dirichlet non lineares. *Revue Française d'Automatique, Informatique, et Recherche Opérationelle*, 9:4176, 1975.

[9] D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. Technical report, UCLA CAM, 2010.

[10] T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *Information Theory, IEEE Transactions on*, 56(12):6294 –6316, Dec. 2010.

[11] B. S. He, H. Yang, and S. L. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and Applications*, 106:337–356, 2000.

[12] J. Johnson. *Convex Relaxation Methods for Graphical Models: Lagrangian and Maximum Entropy Approaches*. PhD thesis, EECS, MIT, 2008.

[13] V. Jojic, S. Gould, and D. Koller. Fast and smooth: Accelerated dual decomposition for MAP inference. In *Proceedings of International Conference on Machine Learning*, 2010.

[14] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1568–1583, 2006.

[15] N. Komodakis and N. Paragios. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *10th European Conference on Computer Vision*, pages 806–820, 2008.

[16] N. Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33:531–552, March 2011.

[17] A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. An augmented lagrangian approach to constrained map inference. In *International Conference on Machine Learning*, June 2011.

[18] O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *Proceedings of the 27th International Conference on Machine Learning*, pages 783–790, 2010.

[19] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005.

[20] P. Ravikumar, A. Agarwal, and M. Wainwright. Message-passing for graph-structured linear programs: proximal projections, convergence and rounding schemes. In *Proc. of the 25th International Conference on Machine Learning*, pages 800–807, 2008.

[21] A. M. Rush, D. Sontag, M. Collins, and T. Jaakkola. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2010.

[22] M. I. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4:113–130, 1976.

[23] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2011.

[24] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Proc. of the 24th Annual Conference on Uncertainty in Artificial Intelligence*, pages 503–510, 2008.

[25] B. Taskar, C. Guestrin, and D. Koller. Max margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press, Cambridge, MA, 2004.

[26] S. Tosserams, L. Etman, P. Papalambros, and J. Rooda. An augmented lagrangian relaxation for analytical target cascading using the alternating direction method of multipliers. *Structural and Multidisciplinary Optimization*, 31:176–189, 2006.

[27] M. J. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[28] T. Werner. A linear programming approach to max-sum problem: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29:1165–1179, 2007.

[29] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

2.2 Convergence Rate Analysis of MAP Coordinate Minimization Algorithms

Convergence Rate Analysis of MAP Coordinate Minimization Algorithms

Ofer Meshi [*]
meshi@cs.huji.ac.il

Tommi Jaakkola [†]
tommi@csail.mit.edu

Amir Globerson [*]
gamir@cs.huji.ac.il

Abstract

Finding maximum a posteriori (MAP) assignments in graphical models is an important task in many applications. Since the problem is generally hard, linear programming (LP) relaxations are often used. Solving these relaxations efficiently is thus an important practical problem. In recent years, several authors have proposed message passing updates corresponding to coordinate descent in the dual LP. However, these are generally not guaranteed to converge to a global optimum. One approach to remedy this is to smooth the LP, and perform coordinate descent on the smoothed dual. However, little is known about the convergence rate of this procedure. Here we perform a thorough rate analysis of such schemes and derive primal and dual convergence rates. We also provide a simple dual to primal mapping that yields feasible primal solutions with a guaranteed rate of convergence. Empirical evaluation supports our theoretical claims and shows that the method is highly competitive with state of the art approaches that yield global optima.

1 Introduction

Many applications involve simultaneous prediction of multiple variables. For example, we may seek to label pixels in an image, infer amino acid residues in protein design, or find the semantic role of words in a sentence. These problems can be cast as maximizing a function over a set of labels (or minimizing an energy function). The function typically decomposes into a sum of local functions over overlapping subsets of variables. These local functions can be usually learned from data.

Such maximization problems are nevertheless typically hard. Even for simple decompositions (e.g., subsets correspond to pairs of variables), maximizing over the set of labels is often provably NP-hard. One approach would be to reduce the problem to a tractable one, e.g., by constraining the model to a low tree-width graph. However, empirically, using more complex interactions together with approximate inference methods is often advantageous. A popular family of approximate methods are based on linear programming (LP) relaxations. Although these LPs are generally tractable, general purpose LP solvers do not typically adequately exploit the problem structure [28]. Instead, a great deal of effort has gone into designing solvers that are specifically tailored to typical MAP-LP relaxations. These include, for example, cut based algorithms [2], accelerated gradient methods [8], and augmented Lagrangian methods [10, 12]. One class of particularly simple algorithms, which we will focus on here, are coordinate minimization based approaches. Examples include max-sum-diffusion [25], MPLP [5] and TRW-S [9]. These work by first taking the dual of the LP and then optimizing the dual in a block coordinate fashion [see 21, for a thorough review]. In many cases, the coordinate block operations can be performed in closed form resulting in updates quite similar to the max-product message passing algorithm. By coordinate minimization we mean that at each step a set of coordinates is chosen, all other coordinates are fixed, and the chosen coordinates are

[*]School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel
[†]CSAIL, MIT, Cambridge, MA

set to their optimal value given the rest. This is different from a coordinate descent strategy where instead a gradient step is performed on the chosen coordinates (rather than full optimization).

A main caveat of the coordinate minimization approach is that it will not necessarily find the global optimum of the LP (although in practice it often does). This is a direct result of the LP objective not being strictly convex. Several authors have proposed to smooth the LP with entropy terms and employ variants of coordinate minimization [7, 26]. However, the convergence rates of these methods have not been analyzed. Moreover, since the algorithms work in the dual, there is no simple procedure to map the result back into primal feasible variables. We seek to address all these shortcomings: we present a convergence rate analysis of dual coordinate minimization algorithms, provide a simple scheme for generating primal variables from dual ones, and analyze the resulting primal convergence rates.

Convergence rates for coordinate minimization are not common in the literature. While asymptotic convergence is relatively well understood [22], finite rates have been harder to obtain. Recent work [17] provides rates for rather limited settings which do not hold in our case. On the other hand, for coordinate descent methods, some rates have been obtained for greedy and stochastic update schemes [16, 20]. These do not apply directly to our case. Our bounds are derived for the full coordinate minimization case. A related analysis of MAP-LP using smoothing appeared in [3]. However, their approach is specific to LDPC codes, and does not apply to general MAP problems as we analyze here.

2 MAP and LP relaxations

Consider a set of n discrete variables $x_1, \dots, x_n$, and a set C of subsets of these variables (i.e., $c \in C$ is a subset of $\{1, \dots, n\}$). We consider maximization problems over functions that decompose according to these subsets. In particular, each subset c is associated with a local function or factor $\theta_c(x_c)$ and we also include factors $\theta_i(x_i)$ for each individual variable.[1] The MAP problem is to find an assignment $x = (x_1, \dots, x_n)$ to all the variables which maximizes the sum of these factors:

$$\text{MAP}(\theta) = \max_x \sum_{c \in C} \theta_c(x_c) + \sum_{i=1}^{n} \theta_i(x_i) \tag{1}$$

Linear programming relaxations are a popular approach to approximating combinatorial optimization problems [6, 23, 25]. For example, we obtain a relaxation of the discrete optimization problem given in Eq. (1) by replacing it with the following linear program:[2]

$$PMAP : \max_{\mu \in \mathcal{M}_L} P(\mu) = \max_{\mu \in \mathcal{M}_L} \left\{ \sum_c \sum_{x_c} \theta_c(x_c)\mu_c(x_c) + \sum_i \sum_{x_i} \theta_i(x_i)\mu_i(x_i) \right\} = \max_{\mu \in \mathcal{M}_L} \mu \cdot \theta \tag{2}$$

where $P(\mu)$ is the primal (linear) objective and the *local marginal polytope* $\mathcal{M}_L$ enforces basic consistency constraints on the marginals $\{\mu_i(x_i), \forall x_i\}$ and $\{\mu_c(x_c), \forall x_c\}$. Specifically,

$$\mathcal{M}_L = \left\{ \mu \geq 0 : \begin{array}{ll} \sum_{x_{c \setminus i}} \mu_c(x_c) = \mu_i(x_i) & \forall c, i \in c, x_i \\ \sum_{x_i} \mu_i(x_i) = 1 & \forall i \end{array} \right\} \tag{3}$$

If the maximizer of $PMAP$ has only integral values (i.e., 0 or 1) it can be used to find the MAP assignment (e.g., by taking the x_i that maximizes $\mu_i(x_i)$). However, in the general case the solution may be fractional [24] and the maximum of $PMAP$ is an upper bound on $\text{MAP}(\theta)$.

2.1 Smoothing the LP

As mentioned earlier, several authors have considered a smoothed version of the LP in Eq. (2). As we shall see, this offers several advantages over solving the LP directly. Given a smoothing parameter $\tau > 0$, we consider the following smoothed primal problem:

$$PMAP_\tau : \quad \max_{\mu \in \mathcal{M}_L} P_\tau(\mu) = \max_{\mu \in \mathcal{M}_L} \left\{ \mu \cdot \theta + \frac{1}{\tau} \sum_c H(\mu_c) + \frac{1}{\tau} \sum_i H(\mu_i) \right\} \tag{4}$$

[1]Although singleton factors are not needed for generality, we keep them for notational convenience.

[2]We use μ and θ to denote vectors consisting of all μ and θ values respectively.

where $H(\mu_c)$ and $H(\mu_i)$ are local entropy terms. Note that as $\tau \to \infty$ we obtain the original primal problem. In fact, a stronger result can be shown. Namely, that the optimal value of $PMAP$ is $O(\frac{1}{\tau})$ close to the optimal value of $PMAP_\tau$. This justifies using the smoothed objective P_τ as a proxy to P in Eq. (2). We express this in the following lemma (which appears in similar forms in [7, 15]).

Lemma 2.1. *Denote by μ^* the optimum of problem $PMAP$ in Eq. (2) and by $\hat{\mu}^*$ the optimum of problem $PMAP_\tau$ in Eq. (4). Then:*

$$\hat{\mu}^* \cdot \theta \leq \mu^* \cdot \theta \leq \hat{\mu}^* \cdot \theta + \frac{H_{\max}}{\tau} \tag{5}$$

where $H_{\max} = \sum_c \log |x_c| + \sum_i \log |x_i|$. In other words, the smoothed optimum is an $O(\frac{1}{\tau})$-optimal solution of the original non-smoothed problem.

We shall be particularly interested in the dual of $PMAP_\tau$ since it facilitates simple coordinate minimization updates. Our dual variables will be denoted by $\delta_{ci}(x_i)$, which can be interpreted as the messages from subset c to node i about the value of variable x_i. The dual variables are therefore indexed by (c, i, x_i) and written as $\delta_{ci}(x_i)$. The dual objective can be shown to be:

$$F(\delta) = \sum_c \frac{1}{\tau} \log \sum_{x_c} \exp\left(\tau\theta_c(x_c) - \tau \sum_{i:i\in c} \delta_{ci}(x_i)\right) + \sum_i \frac{1}{\tau} \log \sum_{x_i} \exp\left(\tau\theta_i(x_i) + \tau \sum_{c:i\in c} \delta_{ci}(x_i)\right) \tag{6}$$

The dual problem is an unconstrained smooth minimization problem:

$$DMAP_\tau : \min_\delta F(\delta) \tag{7}$$

Convex duality implies that the optima of $DMAP_\tau$ and $PMAP_\tau$ coincide.

Finally, we shall be interested in transformations between dual variables δ and primal variables μ (see Section 5). The following are the transformations obtained from the Lagrangian derivation (i.e., they can be used to switch from *optimal* dual variables to *optimal* primal variables).

$$\mu_c(x_c; \delta) \propto \exp\left(\tau\theta_c(x_c) - \tau \sum_{i:i\in c} \delta_{ci}(x_i)\right) \quad , \quad \mu_i(x_i; \delta) \propto \exp\left(\tau\theta_i(x_i) + \tau \sum_{c:i\in c} \delta_{ci}(x_i)\right) \tag{8}$$

We denote the vector of all such marginals by $\mu(\delta)$. For the dual variables δ that minimize $F(\delta)$ it holds that $\mu(\delta)$ are feasible (i.e., $\mu(\delta) \in \mathcal{M}_L$). However, we will also consider $\mu(\delta)$ for non optimal δ, and show how to obtain primal feasible approximations from $\mu(\delta)$. These will be helpful in obtaining primal convergence rates.

It is easy to see that: $(\nabla F(\delta^t))_{c,i,x_i} = \mu_i(x_i; \delta^t) - \mu_c(x_i; \delta^t)$, where (with some abuse of notation) we denote: $\mu_c(x_i) = \sum_{x_{c\setminus i}} \mu_c(x_{c\setminus i}, x_i)$. The elements of the gradient thus correspond to inconsistency between the marginals $\mu(\delta^t)$ (i.e., the degree to which they violate the constraints in Eq. (3)). We shall make repeated use of this fact to link primal and dual variables.

3 Coordinate Minimization Algorithms

In this section we propose several coordinate minimization procedures for solving $DMAP_\tau$ (Eq. (7)). We first set some notation to define block coordinate minimization algorithms. Denote the objective we want to minimize by $F(\delta)$ where δ corresponds to a set of N variables. Now define $\mathcal{S} = \{S_1, \ldots, S_M\}$ as a set of subsets, where each subset $S_i \subseteq \{1, \ldots, N\}$ describes a coordinate block. We will assume that $S_i \cap S_j = \emptyset$ for all i, j and that $\cup_i S_i = \{1, \ldots, N\}$.

Block coordinate minimization algorithms work as follows: at each iteration, first set $\delta^{t+1} = \delta^t$. Next choose a block S_i and set:

$$\delta_{S_i}^{t+1} = \arg\min_{\delta_{S_i}} F_i(\delta_{S_i}; \delta^t) \tag{9}$$

where we use $F_i(\delta_{S_i}; \delta^t)$ to denote the function F restricted to the variables δ_{S_i} and where all other variables are set to their value in δ^t. In other words, at each iteration we fully optimize only over the variables δ_{S_i} while fixing all other variables. We assume that the minimization step in Eq. (9) can be solved in closed form, which is indeed the case for the updates we consider.

Regarding the choice of an update schedule, several options are available:

- **Cyclic**: Decide on a fixed order (e.g., $S_1, \dots, S_M$) and cycle through it.
- **Stochastic**: Draw an index i uniformly at random[3] at each iteration and use the block S_i.
- **Greedy**: Denote by $\nabla_{S_i} F(\delta^t)$ the gradient $\nabla F(\delta^t)$ evaluated at coordinates S_i only. The greedy scheme is to choose S_i that maximizes $\|\nabla_{S_i} F(\delta^t)\|_\infty$. In other words, choose the set of coordinates that correspond to maximum gradient of the function F. Intuitively this corresponds to choosing the block that promises the maximal (local) decrease in objective. Note that to find the best coordinate we presumably must process all sets S_i to find the best one. We will show later that this can be done rather efficiently in our case.

In our analysis, we shall focus on the Stochastic and Greedy cases, and analyze their rate of convergence. The cyclic case is typically hard to analyze, with results only under multiple conditions which do not hold here (e.g., see [17]).

Another consideration when designing coordinate minimization algorithms is the choice of block size. One possible choice is all variables $\delta_{ci}(\cdot)$ (for a specific pair ci). This is the block chosen in the max-sum-diffusion (MSD) algorithm (see [25] and [26] for non-smooth and smooth MSD). A larger block that also facilitates closed form updates is the set of variables $\delta_{\cdot i}(\cdot)$. Namely, all messages into a variable i from c such that $i \in c$. We call this a *star* update. The update is used in [13] for the non-smoothed dual (but the possibility of applying it to the smoothed version is mentioned).

For simplicity, we focus here only on the star update, but the derivation is similar for other choices. To derive the star update around variable i, one needs to fix all variables except $\delta_{\cdot i}(\cdot)$ and then set the latter to minimize $F(\delta)$. Since $F(\delta)$ is differentiable this is pretty straightforward. The update turns out to be:[4]

$$\delta_{ci}^{t+1}(x_i) = \delta_{ci}^t(x_i) + \frac{1}{\tau}\log \mu_c^t(x_i) - \frac{1}{N_i+1}\cdot\frac{1}{\tau}\log\left(\mu_i^t(x_i)\cdot\prod_{c':i\in c'}\mu_{c'}^t(x_i)\right) \tag{10}$$

where $N_i = |\{c : i \in c\}|$. It is interesting to consider the improvement in $F(\delta)$ as a result of the star update. It can be shown to be exactly:

$$F(\delta^t) - F(\delta^{t+1}) = -\frac{1}{\tau}\log\left(\sum_{x_i}\left(\mu_i^t(x_i)\cdot\prod_{c:i\in c}\mu_c^t(x_i)\right)^{\frac{1}{N_i+1}}\right)^{N_i+1}$$

The RHS is known as *Matusita's divergence measure* [11], and is a generalization of the Bhattacharyya divergence to several distributions. Thus the improvement can be easily computed before actually applying the update and is directly related to how consistent the $N_i + 1$ distributions $\mu_c^t(x_i), \mu_i^t(x_i)$ are. Recall that at the optimum they all agree as $\mu \in \mathcal{M}_L$, and thus the expected improvement is zero.

4 Dual Convergence Rate Analysis

We begin with the convergence rates of the dual F using greedy and random schemes described in Section 3. In Section 5 we subsequently show how to obtain a primal feasible solution and how the dual rates give rise to primal rates. Our analysis builds on the fact that we can lower bound the improvement at each step, as a function of some norm of the block gradient.

4.1 Greedy block minimization

Theorem 4.1. *Define B_1 to be a constant such that $\|\delta^t - \delta^*\|_1 \le B_1$ for all t. If* coordinate minimization *of each block S_i satisfies:*

$$F(\delta^t) - F(\delta^{t+1}) \ge \frac{1}{k}\|\nabla_{S_i} F(\delta^t)\|_\infty^2 \tag{11}$$

for all t, then for any $\epsilon > 0$ after $T = \frac{kB_1^2}{\epsilon}$ iterations of the greedy algorithm, $F(\delta^T) - F(\delta^*) \le \epsilon$.

[3]Non uniform schedules are also possible. We consider the uniform for simplicity.

[4]The update is presented here in additive form, there is an equivalent absolute form [21].

Proof. Using Hölder's inequality we obtain the bound:

$$F(\delta^t) - F(\delta^*) \leq \nabla F(\delta^t)^\top(\delta^t - \delta^*) \leq \|\nabla F(\delta^t)\|_\infty \cdot \|\delta^t - \delta^*\|_1 \tag{12}$$

Implying: $\|\nabla F(\delta^t)\|_\infty \geq \frac{1}{B_1}\left(F(\delta^t) - F(\delta^*)\right)$. Now, using the condition on the improvement and the greedy nature of the update, we obtain a bound on the improvement:

$$\begin{aligned} F(\delta^t) - F(\delta^{t+1}) &\geq \frac{1}{k}\|\nabla_{S_i} F(\delta^t)\|_\infty^2 = \frac{1}{k}\|\nabla F(\delta^t)\|_\infty^2 \\ &\geq \frac{1}{kB_1^2}\left(F(\delta^t) - F(\delta^*)\right)^2 \geq \frac{1}{kB_1^2}\left(F(\delta^t) - F(\delta^*)\right)\left(F(\delta^{t+1}) - F(\delta^*)\right) \end{aligned}$$

Hence,

$$\frac{1}{kB_1^2} \leq \frac{F(\delta^t) - F(\delta^*) - \left(F(\delta^{t+1}) - F(\delta^*)\right)}{\left(F(\delta^t) - F(\delta^*)\right)\left(F(\delta^{t+1}) - F(\delta^*)\right)} = \frac{1}{F(\delta^{t+1}) - F(\delta^*)} - \frac{1}{F(\delta^t) - F(\delta^*)} \tag{13}$$

Summing over t we obtain:

$$\frac{T}{kB_1^2} \leq \frac{1}{F(\delta^T) - F(\delta^*)} - \frac{1}{F(\delta^0) - F(\delta^*)} \leq \frac{1}{F(\delta^T) - F(\delta^*)} \tag{14}$$

and the desired result follows. □

4.2 Stochastic block minimization

Theorem 4.2. *Define B_2 to be a constant such that $\|\delta^t - \delta^*\|_2 \leq B_2$ for all t. If coordinate minimization of each block S_i satisfies:*

$$F(\delta^t) - F(\delta^{t+1}) \geq \frac{1}{k}\|\nabla_{S_i} F(\delta^t)\|_2^2 \tag{15}$$

for all t, then for any $\epsilon > 0$ after $T = \frac{k|\mathcal{S}|B_2^2}{\epsilon}$ iterations of the stochastic algorithm we have that $\mathbb{E}[F(\delta^T)] - F(\delta^*) \leq \epsilon$.[5]

The proof is similar to Nesterov's analysis (see Theorem 1 in [16]). The proof in [16] relies on the improvement condition in Eq. (15) and not on the precise nature of the update. Note that since the cost of the update is roughly linear in the size of the block then this bound does not tell us which block size is better (the cost of an update times the number of blocks is roughly constant).

4.3 Analysis of $DMAP_\tau$ block minimization

We can now obtain rates for our coordinate minimization scheme for optimizing $DMAP_\tau$ by finding the k to be used in conditions Eq. (15) and Eq. (11). The result for the star update is given below.

Proposition 4.3. *The star update for x_i satisfies the conditions in Eqs. 15 and 11 with $k = 4\tau N_i$.*

This can be shown using Equation 2.4 in [14], which states that if $F_i(\delta_{S_i};\delta)$ (see Eq. (9)) has Lipschitz constant L_i then Eq. (15) is satisfied with $k = 2L_i$. We can then use the fact that the Lipschitz constant of a star block is at most $2\tau N_i$ (this can be calculated as in [18]) to obtain the result.[6] To complete the analysis, it turns out that B_1 and B_2 can be bounded via a function of θ by bounding $\|\delta\|_1$ (see supplementary, Lemma 1.2). We proceed to discuss the implications of these bounds.

4.4 Comparing the different schemes

The results we derived have several implications. First, we see that both stochastic and greedy schemes achieve a rate of $O(\frac{\tau}{\epsilon})$. This matches the known rates for regular (non-accelerated) gradient descent on functions with Lipschitz continuous gradient (e.g., see [14]), although in practice coordinate minimization is often much faster.

[5]Expectation is taken with respect to the randomization of blocks.

[6]We also provide a direct proof in the supplementary, Section 2.

The main difference between the greedy and stochastic rates is that the factor $|\mathcal{S}|$ (the number of blocks) does not appear in the greedy rate, and does appear in the stochastic one. This can have a considerable effect since $|\mathcal{S}|$ is either the number of variables n (in the star update) or the number of factors $|C|$ (in MPLP). Both can be significant (e.g., $|C|$ is the number of edges in a pairwise MRF model). The greedy algorithm does pay a price for this advantage, since it has to find the optimal block to update at each iteration. However, for the problem we study here this can be done much more efficiently using a priority queue. To see this, consider the star update. A change in the variables $\delta_{\cdot i}(\cdot)$ will only affect the blocks that correspond to variables j that are in c such that $i \in c$. In many cases this is small (e.g., low degree pairwise MRFs) and thus we will only have to change the priority queue a small number of times, and this cost would be negligible when using a Fibonacci heap for example.[7] Indeed, our empirical results show that the greedy algorithm consistently outperforms the stochastic one (see Section 6).

5 Primal convergence

Thus far we have considered only dual variables. However, it is often important to recover the primal variables. We therefore focus on extracting primal feasible solutions from current δ, and characterize the degree of primal optimality and associated rates. The primal variables $\mu(\delta)$ (see Eq. (8)) need not be feasible in the sense that the consistency constraints in Eq. (3) are not necessarily satisfied. This is true also for other approaches to recovering primal variables from the dual, such as averaging subgradients when using subgradient descent (see, e.g., [21]).

We propose a simple two-step algorithm for transforming any dual variables δ into primal feasible variables $\tilde{\mu}(\delta) \in \mathcal{M}_L$. The resulting $\tilde{\mu}(\delta)$ will also be shown to converge to the optimal primal solution in Section 5.1. The procedure is described in Algorithm 1 below.

Algorithm 1 Mapping to feasible primal solution

```
Step 1: Make marginals consistent.
For all $i$ do: $\bar{\mu}_i(x_i) = \frac{1}{1+\sum_{c:i\in c}\frac{1}{|X_{c\setminus i}|}}\left(\mu_i(x_i) + \sum_{c:i\in c}\frac{1}{|X_{c\setminus i}|}\mu_c(x_i)\right)$
For all $c$ do: $\bar{\mu}_c(x_c) = \mu_c(x_c) - \sum_{i:i\in c}\frac{1}{|X_{c\setminus i}|}\left(\mu_c(x_i) - \bar{\mu}_i(x_i)\right)$
Step 2: Make marginals non-negative.
$\lambda = 0$
for $c \in C, x_c$ do
    if $\bar{\mu}_c(x_c) < 0$ then
        $\lambda = \max\left\{\lambda, \frac{-\bar{\mu}_c(x_c)}{-\bar{\mu}_c(x_c)+\frac{1}{|X_c|}}\right\}$
    else if $\bar{\mu}_c(x_c) > 1$ then
        $\lambda = \max\left\{\lambda, \frac{\bar{\mu}_c(x_c)-1}{\bar{\mu}_c(x_c)-\frac{1}{|X_c|}}\right\}$
    end if
end for
for $\ell = 1, \dots, n; c \in C$ do
    $\tilde{\mu}_\ell(x_\ell) = (1-\lambda)\bar{\mu}_\ell(x_\ell) + \lambda\frac{1}{|X_\ell|}$
end for
```

Importantly, all steps consist of cheap elementary local calculations in contrast to other methods previously proposed for this task (compare to [18, 27]). The first step performs a Euclidian projection of $\mu(\delta)$ to consistent marginals $\bar{\mu}$. Specifically, it solves:

$$\min_{\bar{\mu}} \quad \frac{1}{2}\|\mu(\delta) - \bar{\mu}\|^2 \quad , \quad \text{s.t. } \bar{\mu}_c(x_i) = \bar{\mu}_i(x_i), \text{ for all } c, i \in c, x_i \quad , \quad \sum_i \bar{\mu}_i(x_i) = 1, \text{ for all } i$$

Note that we did not include non-negativity constraints above, so the projection might result in negative $\bar{\mu}$. In the second step we "pull" $\bar{\mu}$ back into the feasible regime by taking a convex combination

[7] This was also used in the residual belief propagation approach [4], which however is less theoretically justified than what we propose here.

with the uniform distribution u (see [3] for a related approach). In particular, this step solves the simple problem of finding the smallest $\lambda \in [0, 1]$ such that $0 \le \tilde{\mu} \le 1$ (where $\tilde{\mu} = (1 - \lambda)\bar{\mu} + \lambda u$). Since this step interpolates between two distributions that satisfy consistency and normalization constraints, $\tilde{\mu}$ will be in the local polytope $\mathcal{M}_L$.

5.1 Primal convergence rate

Now that we have a procedure for obtaining a primal solution we analyze the corresponding convergence rate. First, we show that if we have δ for which $\|\nabla F(\delta)\|_\infty \le \epsilon$ then $\tilde{\mu}(\delta)$ (after Algorithm 1) is an $O(\epsilon)$ primal optimal solution.

Theorem 5.1. *Denote by P^*_τ the optimum of the smoothed primal $PMAP_\tau$. For any set of dual variables δ, and any $\epsilon \in R(\tau)$ (see supp. for definition of $R(\tau)$) it holds that if $\|\nabla F(\delta)\|_\infty \le \epsilon$ then $P^*_\tau - P_\tau(\tilde{\mu}(\delta)) \le C_0\epsilon$. The constant C_0 depends only on the parameters θ and is independent of τ.*

The proof is given in the supplementary file (Section 1). The key idea is to break $F(\delta) - P_\tau(\tilde{\mu}(\delta))$ into components, and show that each component is upper bounded by $O(\epsilon)$. The range $R(\tau)$ consists of $\epsilon \ge O(\frac{1}{\tau})$ and $\epsilon \le O(e^{-\tau})$. As we show in the supplementary this range is large enough to guarantee any desired accuracy in the non-smoothed primal. We can now translate dual rates into primal rates. This can be done via the following well known lemma:

Lemma 5.2. *Any convex function F with Lipschitz continuous gradient and Lipschitz constant L satisfies $\|\nabla F(\delta)\|_2^2 \le 2L\left(F(\delta) - F(\delta^*)\right)$.*

These results together with the fact that $\|\nabla F(\delta)\|_2^2 \ge \|\nabla F(\delta)\|_\infty^2$, and the Lipschitz constant of $F(\delta)$ is $O(\tau)$, lead to the following theorem.

Theorem 5.3. *Given any algorithm for optimizing $DMAP_\tau$ and $\epsilon \in R(\tau)$, if the algorithn is guaranteed to achieve $F(\delta^t) - F(\delta^*) \le \epsilon$ after $O(g(\epsilon))$ iterations, then it is guaranteed to be ϵ primal optimal, i.e., $P^*_\tau - P_\tau(\tilde{\mu}(\delta^t)) \le \epsilon$ after $O(g(\frac{\epsilon^2}{\tau}))$ iterations.*[8]

The theorem lets us directly translate dual convergence rates into primal ones. Note that it applies to any algorithm for $DMAP_\tau$ (not only coordinate minimization), and the only property of the algorithm used in the proof is $F(\delta^t) \le F(0)$ for all t. Put in the context of our previous results, any algorithm that achieves $F(\delta^t) - F(\delta^*) \le \epsilon$ in $t = O(\tau/\epsilon)$ iterations, then it is guaranteed to achieve $P^*_\tau - P_\tau(\tilde{\mu}(\delta^{t'})) \le \epsilon$ in $t' = O(\tau^2/\epsilon^2)$ iterations.

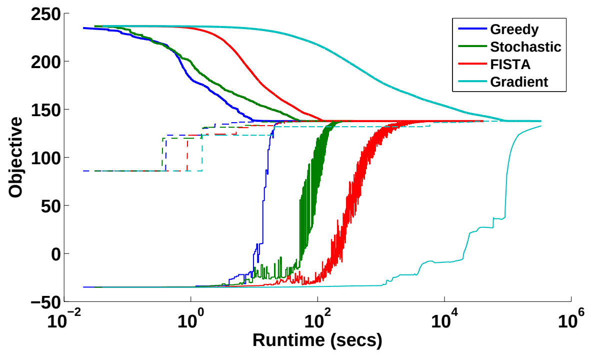
6 Experiments

In this section we evaluate coordinate minimization algorithms on a MAP problem, and compare them to state-of-the-art baselines. Specifically, we compare the running time of greedy coordinate minimization, stochastic coordinate minimization, full gradient descent, and FISTA – an accelerated gradient method [1] (details on the gradient-based algorithms are provided in the supplementary, Section 3). Gradient descent is known to converge in $O\left(\frac{1}{\epsilon}\right)$ iterations while FISTA converges in $O\left(\frac{1}{\sqrt{\epsilon}}\right)$ iterations [1]. We compare the performance of the algorithms on protein side-chain prediction problems from the dataset of Yanover et al. [28]. These problems involve finding the 3D configuration of rotamers given the backbone structure of a protein. The problems are modeled by singleton and pairwise factors and can be posed as finding a MAP assignment for the given model.

Figure 1(a) shows the objective value for each algorithm over time. We first notice that the greedy algorithm converges faster than the stochastic one. This is in agreement with our theoretical analysis. Second, we observe that the coordinate minimization algorithms are competitive with the accelerated gradient method FISTA and are much faster than the gradient method. Third, as Theorem 5.3 predicts, primal convergence is slower than dual convergence (notice the logarithmic timescale). Finally, we can see that better convergence of the dual objective corresponds to better convergence of the primal objective, in both fractional and integral domains. In our experiments the quality of the decoded integral solution (dashed lines) significantly exceeds that of the fractional solution. Although sometimes a fractional solution can be useful in itself, this suggests that if only an integral solution is sought then it could be enough to decode directly from the dual variables.

[8]We omit constants not depending on τ and ϵ.

(a)



(b) t_{alg}/t_{greedy}

Greedy	1
Stochastic	8.6 ± 0.6
FISTA	814.2 ± 38.1
Gradient	13849.8 ± 6086.5

Figure 1: Comparison of coordinate minimization, gradient descent, and the accelerated gradient algorithms on protein side-chain prediction task. Figure (a) shows a typical run of the algorithms. For each algorithm the dual objective of Eq. (6) is plotted as a function of execution time. The value (Eq. (4)) of the feasible primal solution of Algorithm 1 is also shown (lower solid line), as well as the objective (Eq. (1)) of the best decoded integer solution (dashed line; those are decoded directly from the dual variables δ). Table (b) shows the ratio of runtime of each algorithm w.r.t. the greedy algorithm. The mean ratio over the proteins in the dataset is shown followed by standard error.

The table in Figure 1(b) shows overall statistics for the proteins in the dataset. Here we run each algorithm until the duality gap drops bellow a fixed desired precision ($\epsilon = 0.1$) and compare the total runtime. The table presents the ratio of runtime of each algorithm w.r.t. the greedy algorithm (t_{alg}/t_{greedy}). These results are consistent with the example in Figure 1(a).

7 Discussion

We presented the first convergence rate analysis of dual coordinate minimization algorithms on MAP-LP relaxations. We also showed how such dual iterates can be turned into primal feasible iterates and analyzed the rate with which these primal iterates converge to the primal optimum. The primal mapping is of considerable practical value, as it allows us to monitor the distance between the upper (dual) and lower (primal) bounds on the optimum and use this as a stopping criterion. Note that this cannot be done without a primal feasible solution.[9]

The overall rates we obtain are of the order $O(\frac{\tau}{\epsilon})$ for the $DMAP_\tau$ problem. If one requires an ϵ accurate solution for $PMAP$, then τ needs to be set to $O(\frac{1}{\epsilon})$ (see Eq. (5)) and the overall rate is $O(\frac{1}{\epsilon^2})$ in the dual. As noted in [8, 18], a faster rate of $O(\frac{1}{\epsilon})$ may be obtained using accelerated methods such as Nesterov's [15] or FISTA [1]. However, these also have an extra factor of N which does not appear in the greedy rate. This could partially explain the excellent performance of the greedy scheme when compared to FISTA (see Section 6).

Our analysis also highlights the advantage of using greedy block choice for MAP problems. The advantage comes from the fact that the choice of block to update is quite efficient since its cost is of the order of the other computations required by the algorithm. This can be viewed as a theoretical reinforcement of selective scheduling algorithms such as Residual Belief Propagation [4].

Many interesting questions still remain to be answered. How should one choose between different block updates (e.g., MSD vs star)? What are lower bounds on rates? Can we use acceleration as in [15] to obtain better rates? What is the effect of adaptive smoothing (see [19]) on rates? We plan to address these in future work.

Acknowledgments: This work was supported by BSF grant 2008303. Ofer Meshi is a recipient of the Google Europe Fellowship in Machine Learning, and this research is supported in part by this Google Fellowship.

[9]An alternative commonly used progress criterion is to decode an integral solution from the dual variables, and see if its value is close to the dual upper bound. However, this will only work if $PMAP$ has an integral solution and we have managed to decode it.

References

[1] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, Mar. 2009.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 1999.

[3] D. Burshtein. Iterative approximate linear programming decoding of ldpc codes with linear complexity. *IEEE Transactions on Information Theory*, 55(11):4835–4859, 2009.

[4] G. Elidan, I. Mcgraw, and D. Koller. Residual belief propagation: informed scheduling for asynchronous message passing. In *UAI*, 2006.

[5] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NIPS 20*. MIT Press, 2008.

[6] M. Guignard and S. Kim. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39(2):215–228, 1987.

[7] T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*, 56(12):6294–6316, 2010.

[8] V. Jojic, S. Gould, and D. Koller. Fast and smooth: Accelerated dual decomposition for MAP inference. In *Proceedings of International Conference on Machine Learning (ICML)*, 2010.

[9] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[10] A. L. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. An augmented lagrangian approach to constrained map inference. In *ICML*, pages 169–176, 2011.

[11] K. Matusita. On the notion of affinity of several distributions and some of its applications. *Annals of the Institute of Statistical Mathematics*, 19:181–192, 1967. 10.1007/BF02911675.

[12] O. Meshi and A. Globerson. An alternating direction method for dual map lp relaxation. In *ECML PKDD*, pages 470–483. Springer-Verlag, 2011.

[13] O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *ICML*, pages 783–790, New York, NY, USA, 2010. ACM.

[14] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Kluwer Academic Publishers, 2004.

[15] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Prog.*, 103(1):127–152, May 2005.

[16] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. Core discussion papers, Universit catholique de Louvain, 2010.

[17] A. Saha and A. Tewari. On the finite time convergence of cyclic coordinate descent methods, 2010. preprint arXiv:1005.2146.

[18] B. Savchynskyy, S. Schmidt, J. Kappes, and C. Schnorr. A study of Nesterov's scheme for lagrangian decomposition and map labeling. *CVPR*, 2011.

[19] B. Savchynskyy, S. Schmidt, J. H. Kappes, and C. Schnörr. Efficient mrf energy minimization via adaptive diminishing smoothing. In *UAI*, 2012.

[20] S. Shalev-Shwartz and A. Tewari. Stochastic methods for l1-regularized loss minimization. *J. Mach. Learn. Res.*, 12:1865–1892, July 2011.

[21] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*, pages 219–254. MIT Press, 2011.

[22] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization 1. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[23] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.

[24] M. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA, 2008.

[25] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007.

[26] T. Werner. Revisiting the decomposition approach to inference in exponential families and graphical models. Technical Report CTU-CMP-2009-06, Czech Technical University, 2009.

[27] T. Werner. How to compute primal solution from dual one in MAP inference in MRF? In *Control Systems and Computers (special issue on Optimal Labeling Problems in Structual Pattern Recognition)*, 2011.

[28] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

Supplementary material for: Convergence Rate Analysis of MAP Coordinate Minimization Algorithms

Ofer Meshi meshi@cs.huji.ac.il
Tommi Jaakkola tommi@csail.mit.edu
Amir Globerson gamir@cs.huji.ac.il

1 Primal Convergence Rate

For clarity, we define

$$\begin{aligned}\mu\cdot\theta &= \sum_i\sum_{x_i}\mu_i(x_i)\theta_i(x_i)+\sum_c\sum_{x_c}\mu_c(x_c)\theta_c(x_c) && (1)\\ H(\mu) &= \sum_i H(\mu_i(\cdot))+\sum_c H(\mu_c(\cdot)) && (2)\end{aligned}$$

Theorem 1.1. *Denote by P_τ^* the optimum of the smoothed primal $PMAP_\tau$. Then for any set of dual variables δ, if $\|\nabla F(\delta)\|_\infty \le \epsilon \in R(\tau)$ (for a range of values $R(\tau)$), then $P_\tau^* - P_\tau(\tilde{\mu}) \le C_0\epsilon$, where C_0 is a constant that depends only on the parameters θ, independent of τ, and $\tilde{\mu}$ represents the set of locally consistent marginals from Algorithm 1 in response to $\mu=\mu(\delta)$.*

Proof. $\|\nabla F(\delta)\|_\infty \le \epsilon$ guarantees that $\mu=\mu(\delta)$ are ϵ-consistent in the sense that $|\mu_i(x_i)-\mu_c(x_i)|\le\epsilon$ for all $c, i\in c$ and x_i. Algorithm 1 maps any such ϵ-consistent μ to locally consistent marginals $\tilde{\mu}$ such that

$$|\mu_i(x_i)-\tilde{\mu}_i(x_i)|\le 3\epsilon N_{\max},\quad |\mu_c(x_c)-\tilde{\mu}_c(x_c)|\le 2\epsilon N_{\max}^2, \qquad (3)$$

for all i, x_i, c, and x_c, where $N_{\max}=\max\{\max_i N_i, \max_c N_c\}$. In other words, $\|\mu-\tilde{\mu}\|_\infty \le K\epsilon$. This can be easily derived from the update in Algorithm 1 and the fact that $|\mu_i(x_i)-\mu_c(x_i)|\le\epsilon$.

Next, it can be shown that $F(\delta)=P_\tau(\mu(\delta))$. And it follows that $P_\tau^*\le F(\delta)\le P_\tau(\mu)$, where the first inequality follows from weak duality.

Thus we have:

$$\begin{aligned}P_\tau^*\le P_\tau(\mu) &= \mu\cdot\theta+\frac{1}{\tau}H(\mu)=(\tilde{\mu}+\mu-\tilde{\mu})\cdot\theta+\frac{1}{\tau}H(\tilde{\mu})+\frac{1}{\tau}(H(\mu)-H(\tilde{\mu})) && (4)\\ &\le P_\tau(\tilde{\mu})+\|\mu-\tilde{\mu}\|_\infty\|\theta\|_1+\frac{1}{\tau}(H(\mu)-H(\tilde{\mu})) && (5)\\ &\le P_\tau(\tilde{\mu})+K\epsilon\|\theta\|_1+\frac{1}{\tau}(H(\mu)-H(\tilde{\mu})) && (6)\end{aligned}$$

Where we have used Holder's inequality for the first inequality and Eq. (3) for the second inequality.

It remains to bound $\frac{1}{\tau}(H(\mu)-H(\tilde{\mu}))$ by a linear function of ϵ. We note that it is impossible to achieve such a bound in general (e.g., see [1]). However, since the entropy is bounded the difference is also bounded. Now, if we also restrict ϵ to be large enough $\epsilon\ge\frac{1}{\tau}$, then we obtain the bound:

$$\frac{1}{\tau}(H(\mu)-H(\tilde{\mu}))\le\frac{1}{\tau}H_{\max}\le\epsilon H_{\max} \qquad (7)$$

We thus obtain that Eq. (6) is of the form $P_\tau(\tilde{\mu})+O(\epsilon)$ and the result follows.

For the high-accuracy regime (small ϵ) we provide a similar bound for the case $\epsilon \le O(e^{-\tau})$. Let $v = \mu - \tilde{\mu}$, so we have:

$$\begin{aligned} H(\mu) - H(\tilde{\mu}) &= H(\tilde{\mu} + v) - H(\tilde{\mu}) \\ &\le H(\tilde{\mu}) + \nabla H(\tilde{\mu})^\top v - H(\tilde{\mu}) \\ &= -\sum_i \sum_{x_i} v_i(x_i) \log \tilde{\mu}_i(x_i) - \sum_c \sum_{x_c} v_c(x_c) \log \tilde{\mu}_c(x_c) \end{aligned}$$

where the inequality follows from the concavity of entropy, and the second equality is true because $\sum_{x_i} v_i(x_i) = 0$ and similarly for $v_c(x_c)$. Now, from the definition of $\mu_i(x_i;\delta)$ we obtain the following bound:

$$\mu_i(x_i;\delta) = \frac{1}{Z_i} e^{\tau(\theta_i(x_i) + \sum_{c:i\in c} \delta_{ci}(x_i))} \ge \frac{1}{|X_i|} e^{-2\tau(\|\theta_i\|_\infty + \|\delta_i\|_1)}$$

We will show below (Lemma 1.2) that $\|\delta_i\|_1$ remains bounded by a constant A independent of τ. Thus we can write:

$$\mu_i(x_i;\delta) \ge \frac{1}{|X_{\max}|} e^{-2\tau(\|\theta_i\|_\infty + A)}$$

where $|X_{\max}| = \max\{\max_i |X_i|, \max_c |X_c|\}$. We define $\gamma_0 = \frac{1}{(2|X_{\max}|)^\tau} e^{-2\tau(\|\theta_i\|_\infty + A)}$, and thus for any $\tau \ge 1$ we have that $\mu_i(x_i;\delta)$ is bounded away from zero by $2^\tau \gamma_0$. Since we assume that $\epsilon \le \gamma_0$, we can bound $\tilde{\mu}$ from below by γ_0. As a result, since $\|v_i\|_\infty \le K\epsilon$,

$$-\frac{1}{\tau} \sum_i \sum_{x_i} v_i(x_i) \log \tilde{\mu}_i(x_i) \quad \le \quad -\frac{1}{\tau} (\log \gamma_0) |X_i| K\epsilon = (2(\|\theta_i\|_\infty + A) + \log(2|X_{\max}|)) |X_i| K\epsilon$$

and similarly for the other entropy terms.

Again, we obtain that Eq. (6) is of the form $P_\tau(\tilde{\mu}) + O(\epsilon)$ and the result holds.

In conclusion, we have shown that if $\|\nabla F(\delta)\|_\infty \le \epsilon$, then for large values $\epsilon \ge \frac{1}{\tau}$ and small values $\epsilon \le \frac{1}{(2|X_{\max}|)^\tau} e^{-2\tau(\|\theta_i\|_\infty + A)}$ we have that: $P^*_\tau - P_\tau(\tilde{\mu}) \le O(\epsilon)$. Our analysis does not cover values in the middle range, but we next argue that the covered range is useful. □

The allowed range of ϵ (namely $\epsilon \in R(\tau)$) seems like a restriction. However, as we argue next taking $\epsilon \ge \frac{1}{\tau}$ (i.e., $\epsilon \in R(\tau)$) is all we need in order to obtain a desired accuracy in the non-smoothed primal.

Suppose one wants to solve the original problem $PMAP$ to within accuracy ϵ'. There are two sources of inaccuracy, namely the smoothing and suboptimality. To ensure the desired accuracy, we require that $P^*_\tau - P^* \le \alpha\epsilon'$ and likewise $P_\tau(\tilde{\mu}) - P^*_\tau \le (1-\alpha)\epsilon'$. In other words, we allow $\alpha\epsilon'$ suboptimality due to smoothing and $(1-\alpha)\epsilon'$ due to suboptimality.

For the first condition, it is enough to set the smoothing constant as: $\tau \ge \frac{H_{\max}}{\alpha\epsilon'}$. The second condition will be satisfied as long as we use an ϵ such that: $\epsilon \le \frac{(1-\alpha)\epsilon'}{(K\|\theta\|_1 + H_{\max})}$ (see Eq. (6) and Eq. (7)). If we choose $\alpha = \frac{H_{\max}}{K\|\theta\|_1 + 2H_{\max}}$ we obtain that this ϵ satisfies $\epsilon \ge \frac{1}{\tau}$ and therefore $\epsilon \in R(\tau)$.

Lemma 1.2. *Assume δ is a set of dual variables satisfying $F(\delta) \le F(0)$ where $F(0)$ is the dual value corresponding to $\delta = 0$. We can require $\sum_{c:i\in c} \delta_{ci}(x_i) = 0$ since $F(\delta)$ is invariant to constant shifts. Then it holds that:*

$$\sum_{c,i,x_i} |\delta_{ci}(x_i)| = \|\delta\|_1 \le A \tag{8}$$

where

$$A = 2\max_i |X_i| \left(F(0) + \sum_i \max_{x_i} |\theta_i(x_i)| + \sum_c \max_{x_c} |\theta_c(x_c)| \right) \tag{9}$$

Proof. To show this, we bound

$$\begin{aligned}\max_{\delta} & \sum_{c,i,x_i} r_{ci}(x_i)\delta_{ci}(x_i) \\ \text{s.t. } & F(\delta) \le F(0) \\ & \sum_{c:i\in c} \delta_{ci}(x_i) = 0\end{aligned} \tag{10}$$

For any $r_{ci}(x_i) \in [-1,1]$. The dual problem turns out to be:

$$\begin{array}{ll}\min\limits_{\mu,\gamma,\alpha} & \alpha(F(0) - \sum\limits_{c,x_c} \mu_c(x_c)\theta_c(x_c) - \sum\limits_{i,x_i} \mu_i(x_i)\theta_i(x_i) - \sum\limits_i H(\mu_i(x_i)) - \sum\limits_c H(\mu_c(x_c)) \\ \text{s.t.} & \mu_i(x_i) - \mu_c(x_i) = \frac{r_{ci}(x_i) - \gamma_{ci}}{\alpha} \\ & \mu_i(x_i) \ge 0, \mu_c(x_c) \ge 0 \\ & \sum_{x_i} \mu_i(x_i) = 1, \sum_{x_c} \mu_c(x_c) = 1 \\ & \alpha \ge 0\end{array} \tag{11}$$

We will next upper bound this minimum with a constant independent of r and thus obtain an upper bound that holds for all r. To do this, we will present a feasible assignment to the variables α, μ, γ above and use the value they attain. First, we set $\alpha = \hat{\alpha} = 2\max_i |X_i|$. Next, we note that for this $\hat{\alpha}$, the objective of Eq. (11) is upper bounded by A (as defined in Eq. (9)). Thus we only need to show that $\hat{\alpha} = 2\max_i |X_i|$ is indeed a feasible value, and this will be done by showing feasible values for the other variables denoted by $\hat{\mu}, \hat{\gamma}$. First, we set:

$$\hat{\mu}_i(x_i) \quad = \quad \frac{1}{|X_i|}$$

and:

$$\hat{\gamma}_{ci} = \frac{1}{|X_i|}\sum_{x_i} r_{ci}(x_i) \tag{12}$$

Next, we define $\nu_{ci}(x_i)$ (for all c, i, x_i) as follows:

$$\nu_{ci}(x_i) = \hat{\mu}_i(x_i) - \frac{r_{ci}(x_i) - \hat{\gamma}_{ci}}{\hat{\alpha}} \tag{13}$$

It can easily be shown that $\nu_{ci}(x_i)$ is a valid distribution over x_i (i.e., non negative and sums to one). Thus we can define:

$$\hat{\mu}_c(x_c) = \prod_{i\in c} \nu_{ci}(x_i) \tag{14}$$

Since $\hat{\mu}_c(x_c)$ is a product of distributions over the variables in c, it is also a valid distribution. Thus it follows that all constraints in Eq. (11) are satisfied by $\hat{\alpha}, \hat{\gamma}, \hat{\mu}$, and the desired bound holds.

□

2 Star Improvement Bound

We prove the following proposition:

Proposition 2.1. *The star update for variable x_i satisfies:*

$$F(\delta^t) - F(\delta^{t+1}) \ge \frac{1}{4\tau N_i}\|\nabla_{S_i} F(\delta^t)\|_2^2$$

Proof. First, we know that the improvement associated with the star update for variable x_i is:

$$F(\delta^t) - F(\delta^{t+1}) = -\frac{1}{\tau}\log\left(\sum_{x_i}\left(\mu_i^t(x_i)\cdot\prod_{c:i\in c}\mu_c^t(x_i)\right)^{\frac{1}{N_i+1}}\right)^{N_i+1}$$

Therefore, for any probability distributions $p, q^{(1)}, ..., q^{(m)}$ we want to prove that:

$$\left(\sum_i\left(p_i\cdot\prod_k q_i^{(k)}\right)^{\frac{1}{m+1}}\right)^{m+1} \quad \le \quad \exp\left(-\frac{1}{4m}\sum_k\sum_i\left(p_i - q_i^{(k)}\right)^2\right)$$

Lemma 2.2. *For any probability distributions* $p, q^{(1)}, ..., q^{(m)}$ *the following holds:*

$$\left(\sum_i \left(p_i \cdot \prod_k q_i^{(k)}\right)^{\frac{1}{m+1}}\right)^{m+1} \leq 1 - \frac{1}{4m}\sum_k \left(\sum_i |p_i - q_i^{(k)}|\right)^2$$

Proof.

$$\begin{aligned}
\sum_k \left(\sum_i |p_i - q_i^{(k)}|\right)^2 &\leq \sum_k \left(\sum_i (\sqrt{p_i} - \sqrt{q_i^{(k)}})^2 \cdot \sum_i (\sqrt{p_i} + \sqrt{q_i^{(k)}})^2\right) \\
&= \sum_k \left(4 - 4\left(\sum_i \sqrt{p_i q_i^{(k)}}\right)^2\right) \\
&= 4m - 4\sum_k \left(\sum_i \sqrt{p_i q_i^{(k)}}\right)^2 \\
&\leq 4m - 4\sum_k \left(\sum_i \left(p_i \cdot \prod_{k'} q_i^{(k')}\right)^{\frac{1}{m+1}}\right)^{m+1} \\
&= 4m - 4m\left(\sum_i \left(p_i \cdot \prod_{k'} q_i^{(k')}\right)^{\frac{1}{m+1}}\right)^{m+1} \\
\Rightarrow \left(\sum_i \left(p_i \cdot \prod_{k'} q_i^{(k')}\right)^{\frac{1}{m+1}}\right)^{m+1} &\leq 1 - \frac{1}{4m}\sum_k \left(\sum_i |p_i - q_i^{(k)}|\right)^2
\end{aligned}$$

For the first transition see [3] (also in [2] p. 57). The second inequality follows from Theorem 1 in [4]. □

Using Lemma 2.2 the desired result follows since:

$$\begin{aligned}
\left(\sum_i \left(p_i \cdot \prod_k q_i^{(k)}\right)^{\frac{1}{m+1}}\right)^{m+1} &\leq 1 - \frac{1}{4m}\sum_k \left(\sum_i |p_i - q_i^{(k)}|\right)^2 \\
&\leq 1 - \frac{1}{4m}\sum_k \sum_i \left(p_i - q_i^{(k)}\right)^2 \\
&\leq \exp\left(-\frac{1}{4m}\sum_k \sum_i \left(p_i - q_i^{(k)}\right)^2\right)
\end{aligned}$$

□

3 Gradient-Based Algorithms

In this section we describe the gradient descent and FISTA algorithms used in the experiments.

Algorithm 1: Gradient descent

```
1: for $t = 1, \ldots$ do
2:     $\delta^{t+1} = \delta^t - \frac{1}{L}\nabla F(\delta^t)$
3: end for
```

Algorithm 2: FISTA

```
1: $\bar{\delta}^1 = \delta^0, \quad \alpha^1 = 1$
2: for $t = 1, \ldots$ do
3:     $\delta^t = \bar{\delta}^t - \frac{1}{L}\nabla F(\bar{\delta}^t)$
4:     $\alpha^{t+1} = \frac{1+\sqrt{1+4(\alpha^t)^2}}{2}$
5:     $\bar{\delta}^{t+1} = \delta^t + \left(\frac{\alpha^t - 1}{\alpha^{t+1}}\right)\left(\delta^t - \delta^{t-1}\right)$
6: end for
```

References

[1] D. Berend and A. Kontorovich. A reverse pinsker inequality. *CoRR*, abs/1206.6544, 2012.

[2] T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *Communication Technology, IEEE Transactions on*, 15(1):52 –60, february 1967.

[3] C. Kraft. Some conditions for consistency and uniform consistency of statistical procedures. In *Univ. of California Publ. in Statistics, vol. 1*, pages 125–142. Univ. of California, Berkeley, 1955.

[4] K. Matusita. On the notion of affinity of several distributions and some of its applications. *Annals of the Institute of Statistical Mathematics*, 19:181–192, 1967. 10.1007/BF02911675.

2.3 Learning Efficiently with Approximate Inference via Dual Losses

Learning Efficiently with Approximate Inference via Dual Losses

Ofer Meshi MESHI@CS.HUJI.AC.IL
David Sontag DSONTAG@CSAIL.MIT.EDU
Tommi Jaakkola TOMMI@CSAIL.MIT.EDU
Amir Globerson GAMIR@CS.HUJI.AC.IL

Abstract

Many structured prediction tasks involve complex models where inference is computationally intractable, but where it can be well approximated using a linear programming relaxation. Previous approaches for learning for structured prediction (e.g., cutting-plane, subgradient methods, perceptron) repeatedly make predictions for some of the data points. These approaches are computationally demanding because each prediction involves solving a linear program to optimality. We present a scalable algorithm for learning for structured prediction. The main idea is to instead solve the dual of the structured prediction loss. We formulate the learning task as a convex minimization over both the weights and the dual variables corresponding to each data point. As a result, we can begin to optimize the weights even before completely solving any of the individual prediction problems. We show how the dual variables can be efficiently optimized using coordinate descent. Our algorithm is competitive with state-of-the-art methods such as stochastic subgradient and cutting-plane.

1. Introduction

In many prediction problems we are interested in predicting multiple labels $y_1, \dots, y_d$ from an input $\boldsymbol{x}$ rather than a single label as in multiclass prediction. This setting is referred to as structured prediction, and has found many applications in various domains, from natural language processing to computational biology (Bakir et al., 2007). A naive approach to the problem is to predict each label y_i individually, ignoring possible correlations between the labels. A better approach would be to explicitly model the interactions between the labels, which then results in the labels being jointly predicted. Structured prediction models do this by using classifiers of the form $\boldsymbol{y} = \arg\max_{\hat{\boldsymbol{y}}} \boldsymbol{w} \cdot f(\boldsymbol{x}, \hat{\boldsymbol{y}})$, where $f(\boldsymbol{x}, \boldsymbol{y})$ is a given function and $\boldsymbol{w}$ are weights to be learned from data.

Much of the early work on structured prediction (Lafferty et al., 2001; Taskar et al., 2004) focused on the case where prediction (i.e., maximization over $\boldsymbol{y}$) could be done using efficient combinatorial algorithms such as dynamic programming or maximum-weight matching. However, this restricted the types of interactions that these models were capable of capturing to tractable structures such as tree graphs. Recent work on graphical models has shown that even when the maximization over $\boldsymbol{y}$ is not known a priori to be tractable, linear programming (LP) relaxations often succeed at finding the true maximum, even giving certificates of optimality (Sontag et al., 2008). This strongly motivates learning structured prediction models which use LP relaxations for prediction, and indeed several recent works show that this yields empirically effective results (Finley and Joachims, 2008; Martins et al., 2009).

Learning with large scale data necessitates efficient algorithms for finding the optimal weight vector $\boldsymbol{w}$. Although several such algorithms have been proposed for structured prediction, these have primarily focused on settings where the maximization over $\boldsymbol{y}$ is performed using combinatorial optimization. Some examples are structured perceptron (Collins, 2002), stochastic subgradient (Ratliff et al., 2007), extra-gradient (Taskar et al., 2006), and cutting-plane algorithms (Joachims et al., 2009). All of these approaches require making a prediction at every iteration. When LP relaxations are used, this corresponds to repeatedly solving an LP to optimality, significantly reducing the scalability of the overall learning algorithm.

These earlier approaches have two potential sources of inefficiency. First, it is likely not necessary to solve the LPs to optimality to obtain an approximately cor-

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

rect update for the weights, particularly in the early iterations of the algorithms. Second, these approaches typically re-solve the LPs from scratch at each iteration. However, particularly in later iterations when there are only small changes to the weight vector, we would like to be able to "warm start" using the previous iteration's solution to the LP for a data point.

In this paper we introduce a novel method for learning structured prediction models using LP relaxations. Whereas previous learning approaches involved repeatedly solving the computationally intensive LP problem per data point, our new formulation replaces the standard LP with its dual. This turns the entire problem into a minimization over the weights $\boldsymbol{w}$ and auxiliary dual variables δ. The latter can be updated via a simple closed form message passing scheme that decreases the overall objective at each iteration. We combine these with stochastic subgradient updates on $\boldsymbol{w}$ and thus our scheme has an online flavor similar to Shalev-Shwartz et al. (2007).

We show empirically that avoiding the LP solution indeed results in much improved convergence time when compared to previous methods. This effect becomes more pronounced the larger the label space is, and the method is thus expected to enhance performance on many large scale structured prediction problems.

2. Problem Formulation

We begin by reviewing the maximum margin Markov network formulation (M^3N) (Taskar et al., 2004), and its LP relaxation. We consider a labelled dataset $\{\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\}_{i=1}^{n}$ containing n samples. We seek a function $h(\boldsymbol{x};\boldsymbol{w})$ that will predict $\boldsymbol{y}$ from $\boldsymbol{x}$. It is assumed to be of the form

$$h(\boldsymbol{x};\boldsymbol{w}) = \arg\max_{\boldsymbol{y}} \boldsymbol{w}\cdot f(\boldsymbol{x},\boldsymbol{y}) \tag{1}$$

where $f(\boldsymbol{x},\boldsymbol{y})$, the feature vector, is given by a fixed, known, vector-valued function of both $\boldsymbol{x}$ and $\boldsymbol{y}$. In what follows we assume that $\boldsymbol{y}$ is multivariate and has d variables denoted by $y_1,\ldots,y_d$. Furthermore, f is assumed to decompose into pairwise and singleton factors on $\boldsymbol{y}$, so that:

$$\boldsymbol{w}\cdot f(\boldsymbol{x},\boldsymbol{y}) = \sum_{ij\in E} f_{ij}(y_i,y_j,\boldsymbol{x})\cdot\boldsymbol{w}_{ij} + \sum_i f_i(y_i,\boldsymbol{x})\cdot\boldsymbol{w}_i \tag{2}$$

where E is a set of edges in a graph G, and the vectors $\boldsymbol{w}_{ij},\boldsymbol{w}_i$ are the elements of the weight vector corresponding to each one of the factors (some weights may be shared across edges).[1]

[1]We use factors of size one and two for notational convenience only; our approach generalizes to larger factors.

In M^3N the weight vector is found by minimizing the following regularized hinge loss:

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_m \ell_h(\boldsymbol{x}^{(m)},\boldsymbol{y}^{(m)};\boldsymbol{w}) \tag{3}$$

where:

$$\ell_h(\boldsymbol{x}^{(m)},\boldsymbol{y}^{(m)};\boldsymbol{w}) = \max_{\boldsymbol{y}} \boldsymbol{w}\cdot\Delta f^{(m)}(\boldsymbol{y}) + e^{(m)}(\boldsymbol{y}) \tag{4}$$

Here $e^{(m)}(\boldsymbol{y})$ is the discrepancy between the true labelling $\boldsymbol{y}^{(m)}$ and $\boldsymbol{y}$, and is assumed to decompose as $e^{(m)}(\boldsymbol{y}) = \sum_i e_i^{(m)}(\boldsymbol{y})$.[2] We also define $\Delta f^{(m)}(\boldsymbol{y}) = f(\boldsymbol{x}^{(m)},\boldsymbol{y}) - f(\boldsymbol{x}^{(m)},\boldsymbol{y}^{(m)})$.

The problems in Eq. 1 and Eq. 4 involve finding an assignment $\boldsymbol{y}$ that maximizes a function of the form $\sum_{ij}\theta_{ij}(y_i,y_j) + \sum_i \theta_i(y_i)$. This problem, commonly referred to as the MAP problem in the graphical models literature, is intractable (NP hard) for general graphs G, and tractable only in isolated cases such as tree structured graphs. However, linear programming (LP) relaxations are often effective as approximations, and have been incorporated into M^3N by several authors, which we review further in Section 4.

In MAP-LP relaxations one replaces maximization over $\boldsymbol{y}$ of $\sum_{ij}\theta_{ij}(y_i,y_j)+\sum_i\theta_i(y_i)$ with the following linear program:[3] $\max_{\boldsymbol{\mu}\in\mathcal{M}_L(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta}$, where $\mathcal{M}_L(G)$ enforces consistency between the pairwise distributions $\mu_{ij}(y_i,y_j)$:

$$\mathcal{M}_L(G) = \left\{ \boldsymbol{\mu}\geq 0 \;\middle|\; \begin{array}{l} \sum_{y_j}\mu_{ij}(y_i,y_j) = \mu_i(y_i) \\ \sum_{y_i}\mu_{ij}(y_i,y_j) = \mu_j(y_j) \\ \sum_{y_i}\mu_i(y_i) = 1 \end{array} \right\}. \tag{5}$$

To introduce the LP relaxation into M^3N, we use the notation $\boldsymbol{\theta}(f,e,\boldsymbol{w})$ to denote the vector of parameters, where the pairwise elements are $\theta_{ij}(y_i,y_j) = f_{ij}(y_i,y_j)\cdot\boldsymbol{w}_{ij}$ and the singleton elements are $\theta_i(y_i) = f_i(y_i)\cdot\boldsymbol{w}_i + e_i(y_i)$. We shall also use: $\boldsymbol{\theta}^{(m)}(\boldsymbol{w}) = \boldsymbol{\theta}(\Delta f^{(m)}, e^{(m)}, \boldsymbol{w})$. Finally, we will denote the singleton and pairwise elements of $\boldsymbol{\theta}^{(m)}(\boldsymbol{w})$ by $\boldsymbol{\theta}^{(m)}(y_i;\boldsymbol{w})$ and $\boldsymbol{\theta}^{(m)}(y_i,y_j;\boldsymbol{w})$ respectively.

We then have the following approximation of the loss ℓ_h from Eq. 4:

$$\hat{\ell}_h(\boldsymbol{x}^{(m)},\boldsymbol{y}^{(m)};\boldsymbol{w}) = \max_{\boldsymbol{\mu}\in\mathcal{M}_L(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta}^{(m)}(\boldsymbol{w}) \tag{6}$$

[2]The loss could also be defined along the edges, but we omit this for notational convenience.

[3]We use the notation $\boldsymbol{\mu}\cdot\boldsymbol{\theta} = \sum_{ij\in E}\sum_{y_i,y_j}\mu_{ij}(y_i,y_j)\theta_{ij}(y_i,y_j) + \sum_i\sum_{y_i}\mu_i(y_i)\theta_i(y_i)$

It is straightforward to show that the relaxed loss $\hat{\ell}_h$ provides an upper bound on the true loss ℓ_h (Finley and Joachims, 2008).

The final classifier is obtained by solving the maximization $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}_L(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta}(f,0,\boldsymbol{w})$ and returning $y_i = \arg\max_{\hat{y}_i} \mu_i(\hat{y}_i)$. To summarize the above, we are interested in solving the optimization problem:

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_m \max_{\boldsymbol{\mu}\in\mathcal{M}_L(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta}^{(m)}(\boldsymbol{w}) \tag{7}$$

3. Optimization via Dual Losses

In this section we present our algorithm for solving the optimization problem in Eq. 7. We begin by using convex duality to replace the internal maximization in Eq. 7 with minimization of a piecewise-linear objective. Numerous duals have been suggested for the MAP LP relaxation problem (e.g., Globerson and Jaakkola, 2008; Komodakis et al., 2007; Werner, 2007). We use the formulation discussed in Werner (2007). The dual of $\max\limits_{\boldsymbol{\mu}\in\mathcal{M}_L(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta}$ is thus:

$$\begin{aligned}\min\nolimits_\delta \quad & \sum_i \max\nolimits_{y_i}\left[\theta_i(y_i) + \sum\nolimits_{k\in N(i)}\delta_{ki}(y_i)\right] + \\ & \sum_{ij}\max\nolimits_{y_i,y_j}\left[\theta_{ij}(y_i,y_j) - \delta_{ij}(y_j) - \delta_{ji}(y_i)\right]\end{aligned} \tag{8}$$

Denote the objective of the above dual by $g(\delta;\boldsymbol{\theta})$. Then we have that Eq. 7 equals:

$$\min_{\boldsymbol{w},\delta^{(1)},\ldots,\delta^{(n)}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_m g(\delta^{(m)};\boldsymbol{\theta}^{(m)}(\boldsymbol{w})), \tag{9}$$

where we now minimize over the dual variables $\delta^{(m)}$ in addition to the weights $\boldsymbol{w}$. Because the dual always upper bounds the primal, the function $g(\delta^{(m)};\boldsymbol{\theta}^{(m)}(\boldsymbol{w}))$ is a convex upper bound on the relaxed loss $\hat{\ell}_h(\boldsymbol{x}^{(m)},\boldsymbol{y}^{(m)};\boldsymbol{w})$ for every value of $\delta^{(m)}$. This bound can be tightened by minimizing it over $\delta^{(m)}$. By LP duality, the minimal $\delta^{(m)}$ value gives us exactly $\hat{\ell}_h(\boldsymbol{x}^{(m)},\boldsymbol{y}^{(m)};\boldsymbol{w})$.

The key advantage of Eq. 9 is that it removes the difficult inner maximization from Eq. 7. Moreover, Eq. 9 is jointly convex in its variables (namely $\boldsymbol{w}$ and the $\delta^{(m)}$s), and furthermore is unconstrained. Thus, we can employ a variety of minimization algorithms to it without the need for a "black-box" solver of the maximization problem.

In the next three sections we describe our algorithm for optimizing Eq. 9. Our approach has two components: one is to decrease the objective via message passing updates on δ, corresponding to coordinate descent on δ. The other is stochastic subgradient updates on $\boldsymbol{w}$ that process each example separately and are thus ideally suited for large data sets.

3.1. Dual minimization via coordinate descent

Notice that, unlike the $\boldsymbol{w}$ variables, the $\delta^{(m)}$ variables are only dependent on the m^{th} sample in the dual formulation Eq. 9. Block coordinate descent of $g(\delta^{(m)};\boldsymbol{\theta}^{(m)}(\boldsymbol{w}))$ can be performed in closed form, as has been noted by several authors (e.g., Globerson and Jaakkola, 2008; Werner, 2007), and as we review next.

Suppose that at iteration t we have a given value of the $\delta^{(m)}$ variables, denoted by $\delta^{(m,t)}$. Now assume we fix all $\delta^{(m)}$ variables except $\delta^{(m)}_{ij},\delta^{(m)}_{ji}$ and seek the optimal value of $\delta^{(m)}_{ij},\delta^{(m)}_{ji}$. The closed form solution is given by:

$$\begin{aligned}\delta^{(m,t+1)}_{ij}(y_j) = \; & -\tfrac{1}{2}\boldsymbol{\theta}^{(m)}(y_j;\boldsymbol{w}) - \tfrac{1}{2}\sum\nolimits_{k\in N(j)\setminus i}\delta^{(m,t)}_{kj}(y_j) \\ & +\tfrac{1}{2}\max\nolimits_{y_i}\left[\boldsymbol{\theta}^{(m)}(y_i,y_j;\boldsymbol{w}) - \delta^{(m,t)}_{ji}(y_i)\right],\end{aligned} \tag{10}$$

and analogously for $\delta^{(m,t+1)}_{ji}(y_i)$. This update is commonly referred to as max-sum diffusion, or MSD (see Werner, 2007, and references within).

We use a more efficient block coordinate descent step where we simultaneously update all of the dual variables $\delta_{ij}(y_j)$ going into variable y_j (see Globerson and Jaakkola, 2008, for a similar update). It is equivalent to iterating the MSD updates for the corresponding edges until convergence, and is given by (we drop the m superscript for brevity):

$$\delta_{ij}(y_j) = -\frac{1}{1+d_j}\boldsymbol{\theta}^{(m)}(y_j;\boldsymbol{w}) - \frac{1}{1+d_j}\gamma_j(y_j) + \gamma_{ij}(y_j) \tag{11}$$

where d_j is the degree of node j in the graph and:

$$\gamma_{ij}(y_j) = \max_{y_i}\left[\boldsymbol{\theta}^{(m)}(y_i,y_j;\boldsymbol{w}) - \delta_{ji}(y_i)\right] \tag{12}$$

and $\gamma_j(y_j) = \sum_{k\in N(j)}\gamma_{kj}(y_j)$. The messages $\delta_{ij}(y_j)$ need to be updated simultaneously for all neighbors of j. The derivation is very similar to that in Globerson and Jaakkola (2008) and is not repeated here. We note that even more efficient updates can be performed, for example by simultaneously updating all δ's that correspond to a tree (Sontag and Jaakkola, 2009).

Because the objective g is not strictly convex, the MSD updates may get trapped in a sub-optimal point (Kolmogorov, 2006). This problem does not occur for binary variables however, as shown in e.g., Globerson and Jaakkola (2008). One way to avoid this is to replace the max function in Eq. 8 with a *soft-max func-*

tion, namely:

$$\max_{y_i} f(y_i) \leq \frac{1}{K} \log \sum_{y_i} e^{Kf(y_i)} \qquad (13)$$

The upper bound becomes tight as $K \to \infty$. Note that g with the soft-max is also a convex upper bound on the original loss. Thus we can use g with a sufficiently high K and the technical issue of non-optimal fixed points is alleviated. It also turns out (Johnson et al., 2007) that the MSD updates when the soft-max is used are exactly as in Eq. 10 and Eq. 11, only with the soft-max replacing the max function. The convergence rate of such updates has recently been analyzed in the context of LDPC codes (Burshtein, 2009). In practice we have found that using the original max does not sacrifice optimality, so we do not use the soft-max in practice.

3.2. Subgradient optimization over $\boldsymbol{w}$

The previous section showed how to update the $\delta^{(m)}$ variables such that the objective is decreased at every iteration. We now turn to the update steps on the weight vector $\boldsymbol{w}$. One method that has proven very useful for losses as in Eq. 9 is stochastic subgradient descent (SSD) (Shalev-Shwartz et al., 2007; Ratliff et al., 2007). In SSD, the vector $\boldsymbol{w}$ is changed in the direction of the subgradient of $g(\delta^{(m)}; \boldsymbol{\theta}^{(m)}(\boldsymbol{w}))$ for each sample m. This strategy is especially effective for large datasets since the exact subgradient involves summation over all the sample points.

In the Pegasos method (Shalev-Shwartz et al., 2007; Shalev-Shwartz and Srebro, 2009), the stochastic subgradient is followed by a projection onto a ball of radius $\sqrt{C}$. This is justified by the fact that the optimal $\boldsymbol{w}$ is known to be inside this ball, and results in improved rates of convergence. We follow the same procedure here, since $\boldsymbol{w}$ in our case satisfies the same condition (assuming that the label loss is upper bounded by one, which it is in our case since we use the normalized Hamming loss). The key difference between Pegasos and the method we propose is that we introduce additional variables $\delta^{(m)}$ and minimize with respect to these as well. In the original Pegasos algorithm, one views the objective as a function of $\boldsymbol{w}$ alone, and therefore has to calculate exact subgradients w.r.t. $\boldsymbol{w}$, which requires solving an LP problem at every sample point. As we show in the experiments, this can have a major effect on runtime.

3.3. The DLPW algorithm and convergence

To summarize the above two sections, we propose to solve the structured prediction problem in Eq. 7 by casting it as a joint minimization problem over $\delta^{(m)}$ and $\boldsymbol{w}$ (Eq. 9) and performing coordinate descent updates on $\delta^{(m)}$ together with stochastic subgradient updates on $\boldsymbol{w}$. The overall algorithm, which we call DLPW for *Dual Loss Primal Weights* is described in Algorithm 1. When processing the m^{th} sample point, the algorithm first updates its $\delta^{(m)}$ variables by improving the objective using coordinate descent updates (Eq. 11). Each $\delta_{ij}^{(m)}(y_j)$ should be updated at least once, but in practice it is preferable to perform R passes over the graph, where R is a small number (we use $R = 10$ in our experiments).

Our scheme combines two minimization approaches: stochastic subgradient and coordinate descent. Each is known to converge to the global optimum if used alone (under appropriate conditions on the objective). Although it is not obvious that using them together would have the same guarantees, we show in Appendix A that the combined method will in fact converge to the global optimum. Since the MSD updates for non-binary y_i may get trapped in suboptimal $\delta^{(m)}$, we show convergence for either binary y_i or a soft-max with any K (which in the limit is equivalent to the max function).

Algorithm 1 The DLPW algorithm

```
Initialize: Choose $\boldsymbol{w}_1$ s.t. $\|\boldsymbol{w}_1\| \leq \sqrt{C}$
for $t = 1$ to $T$ do
    Pick a sample point $m$
    Perform $R$ coordinate descent iterations on all
    variables $\delta^{(m,t)}$ via the updates in Eq. 11. Denote the new values by $\delta^{(m,t+1)}$.
    Set: $\boldsymbol{w}_{t+\frac{1}{2}} = \boldsymbol{w}_t - \frac{1}{t}\partial_{\boldsymbol{w}_t} g_m(\delta^{(m,t+1)}, \boldsymbol{\theta}^{(m)}(\boldsymbol{w}))$
    Set: $\boldsymbol{w}_{t+1} = \min\left\{1, \frac{\sqrt{C}}{\|\boldsymbol{w}_{t+\frac{1}{2}}\|}\right\} \boldsymbol{w}_{t+\frac{1}{2}}$
end for
```

4. Previous Approaches

Most algorithmic effort in structured prediction has focused on the case where maximization over the label space y is tractable. Key examples are when the graph G corresponds to a tree (Taskar et al., 2004; Collins, 2002), or where labels correspond to a combinatorial object such as graphs or matchings (Taskar et al., 2006). Below we review these approaches, and highlight their applicability to LP approximations.

In Taskar et al. (2004) the authors noted that although the primal hinge loss involves maximizing over an exponentially large set, its dual has a simpler structure. Specifically, for singly connected graphs G the dual involves only a polynomial number of constraints. They suggested a dual algorithm similar to the SMO algorithm in Platt (1998), where the dual variables are updated by switching probability mass between two labels y_1, y_2. We note that this dual is different from ours since it also involves dualizing over $\boldsymbol{w}$. This method can also be applied to the LP relaxation case as noted in Taskar et al. (2004, Section 4).

Another dual approach is to use exponentiated gradient steps (Collins et al., 2008). However, these are tailored for the case when marginal inference is tractable and do not seem easily transferable to LP approximations. It is also possible to adapt the perceptron update to the LP case (Kulesza and Pereira, 2008) but this again requires solving the LP at every iteration, and is only exact in the separable case.

Primal methods (such as the one we propose here) operate by updating $\boldsymbol{w}$ directly, and seem to have been used more frequently for structured prediction with LP approximations. One natural approach is to use stochastic (or incremental) subgradient descent on the objective in Eq. 3 (e.g., Shalev-Shwartz and Srebro, 2009; Ratliff et al., 2007). The main drawback of this approach is that calculating the subgradient requires solving the LP approximation after every sample point. This can be quite costly, and is precisely what we avoid doing in the current paper. Another popular primal approach is based on cutting planes (Joachims et al., 2009; Finley and Joachims, 2008). Here one incrementally adds constraints that correspond to vertices of the relaxed polytope. It can be shown that a polynomial number of constraints are sufficient to achieve a given optimization accuracy, but in practice this number may be large. The method we present here avoids this growth in problem size.

The work closest to ours is Taskar et al. (2005), where the dual of only the LP over $\boldsymbol{\mu}$ is taken and the $\boldsymbol{w}$ is kept intact. However, this is done only for problems where the LP is exact (has only integral vertices) and the authors suggest solving the problem via a standard QP solver, as opposed to the efficient coordinate descent message passing procedure we employ here.

5. Experiments

To evaluate our proposed algorithm, we compare its performance on multi-label classification tasks to some of the alternative approaches discussed in Section 4. We will show that DLPW often outperforms the other algorithms, and that it scales well with problem size.

In this multi-label classification setting, each label y_i is a binary random variable indicating whether the i'th label is 'on', and these form a fully connected graph over all label variables. Our model is equivalent to the one used by Finley and Joachims (2008) except that we use an overcomplete representation for feature functions f ($f_i(y_i, \boldsymbol{x})$ is defined for all values y_i and similarly for $f_{ij}(y_i, y_j, \boldsymbol{x})$). The inputs $\boldsymbol{x}$ are vectors in $\mathbb{R}^s$. The feature $f_i(y_i, \boldsymbol{x})$ is a $|y_i| * s$ dimensional vector, i.e., $|y_i|$ concatenated vectors of dimension s. The value of $f_i(y_i, \boldsymbol{x})$ will be $\boldsymbol{x}$ for the vector corresponding to the label y_i and zero elsewhere. The edge feature functions $f_{ij}(y_i, y_j, \boldsymbol{x})$ are indicator vectors, so the length of $\boldsymbol{w}_{ij}$ is $|y_i| * |y_j|$ (4 in the binary case). We use a normalized Hamming loss with $e_i^{(m)}(\boldsymbol{y}) = 1\{y_i^{(m)} \neq y_i\}/d$.

We focus on three datasets of real-world domains taken from the LIBSVM collection (Chang and Lin, 2001) including **Yeast** [14 labels, 1500 training samples, 103 features in $\boldsymbol{x}$] (Elisseeff and Weston, 2001), **Scene** [6 labels, 1211 samples, 294 features] (Boutell et al., 2004), and **Reuters** (subset 1 [3000 samples, 47236 features]) (Lewis et al., 2004). We use a reduction of the Reuters dataset to the 30 most frequent labels. For each dataset, we train a classifier using DLPW and two other algorithms. The first is a cutting-plane algorithm (Finley and Joachims, 2008) and the second is the Pegasos algorithm which uses an LP solver to obtain the approximate MAP at each iteration.[4] The results are shown in Fig. 1(a-c).

As we mentioned in Section 3.3, we limited the number of iterations (MSD updates to all nodes) in DLPW to $R = 10$. We tried a couple of other values for R in this regime and found that these gave similar overall performance. Generally, decreasing R results in faster iterations, but each has a smaller improvement in the objective. On the other hand, if we set no limit on R and allow the MSD algorithm to converge, we get similar performance to that of Pegasos. This makes sense as the LP would be solved completely at each iteration by both algorithms.

Figure 1 shows the objective of Eq. 7, evaluated using the weights found at each iteration, as a function of runtime for each algorithm.[5] For the Yeast dataset we can first see that both subgradient algorithms converge to the optimum much faster than the cutting-plane algorithm. Furthermore, we see that DLPW is

[4] Implementation details: we have implemented all algorithms in C++. The cutting-plane code is taken from `svm_struct` (`http://svmlight.joachims.org/svm_struct.html`) and adapted for the LP case. We use the `GLPK` library (`http://www.gnu.org/software/glpk/glpk.html`) to solve the relaxed LP in both cutting-plane and Pegasos algorithms. We run the experiments on a Dual-Core AMD 2.6 GHz Linux machine.

[5] Not including the time to evaluate Eq. 7.

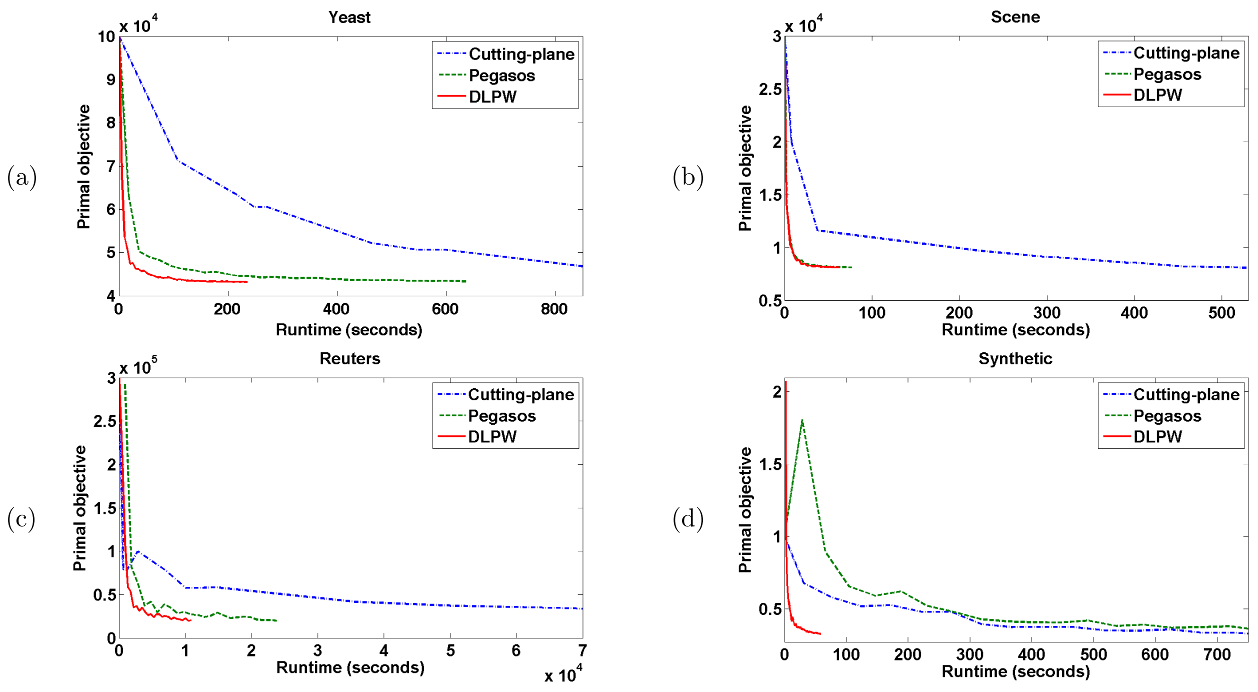


Figure 1. Comparing quality of solution as a function of runtime for various datasets. The x-axis in each subfigure represents the runtime in seconds while the y-axis represents the objective of Eq. 7. Each subfigure shows a line for each of the tested algorithms. Some of the traces were truncated to show more details in the interesting range.

significantly more efficient than Pegasos, which solves the primal LP at each iteration. Specifically, on this dataset DLPW runs 6 times faster than Pegasos and 43 times faster than cutting-plane. In the Scene dataset we see that again the subgradient methods converge much faster than cutting-plane, but here there is only a small advantage for DLPW over Pegasos. This is presumably because the graph is rather small (6 nodes vs. 14 nodes in Yeast) so the LP solver becomes quite efficient. Finally, for the Reuters dataset we observe once more the improved efficiency of the subgradient algorithms. Here DLPW takes less than half the time to converge than Pegasos. We note that when reducing the Reuters dataset to only the 10 most frequent labels, rather than 30, DLPW converges only slightly faster than Pegasos (not shown), which demonstrates the improved scalability of our method as the graph grows in size.

For the multi-label binary models, one could also use graph cut based methods for inference, which would typically be much faster than a general LP solver (e.g., see Finley and Joachims, 2008). However, our method generalizes to the non-binary setting where cut based methods are less effective at solving LPs. Indeed, Figure 1(d) shows results for such a case. For this we use synthetic data similar to the multi-label setting, but with $f_i(y_i, \boldsymbol{x})$ holding x_i in position y_i rather than the whole vector $\boldsymbol{x}$ ($\boldsymbol{x}$ and $\boldsymbol{y}$ are assumed to be of the same length). We run all algorithms on a fully connected graph with $d = 20$, each y_i has 4 states, and the training set consists of 100 samples (these were generated by randomly sampling $\boldsymbol{w}, \boldsymbol{x}$ and obtaining $\boldsymbol{y}$ via iterative conditional modes). We can see that in this setting the cutting-plane algorithm outperforms Pegasos, however DLPW is significantly faster than both.

6. Discussion

We have presented an algorithm for efficient scalable optimization of structured prediction problems that employ approximate inference. Our algorithm dualizes the LP approximation and thus avoids the need to completely solve it at each iteration. The dual can be viewed as an upper bound on the hinge loss (hence the term dual-loss) which can be tightened via auxiliary variables $\delta^{(m)}$. An interesting future direction would be to further explore this notion of tunable surrogate losses further, and study its effects on generalization.

Our empirical results show that our DLPW algorithm improves on methods that employ LP solvers as a black box, and that the improvement is increased as the number of labels grow. A natural question might be why we could not have simply used the MSD updates within these solvers to replace the black-box LP. There are two problems with this approach. First, we would still be required to iterate the MSD to convergence (since the LP needs to be solved exactly in these methods). Second, there would be an extra step of obtaining a primal solution from the $\delta^{(m)}$ variables, which requires extra work for non-binary labels.

In many structured prediction problems, part of the instance may have special structure for which more efficient inference algorithms are known. In these cases we can make global moves to optimize the $\delta^{(m)}$ variables, e.g. tree block coordinate descent (Sontag and Jaakkola, 2009). Our techniques can also be applied to structured prediction problems other than graphical models, such as parsing, by varying the dual decomposition of the optimization problem used for prediction.

The convergence results presented here are asymptotic. It would be desirable to also derive rate of convergence results. It is possible to show that if the $\delta^{(m)}$ are updated at each iteration until they minimize the dual loss then a rate of $O(\frac{1}{\epsilon})$ is obtained. A similar result can be given for minimization up to a certain accuracy. It thus seems likely we can obtain rate results by employing convergence rates for the $\delta^{(m)}$ updates. Recent work (Burshtein, 2009) has analyzed convergence for a related variant of MSD, and can probably be used in this context.

Finally, our results are easily extended to functions $f(x, y)$ that involve larger cliques on y. This can be done via generalizations of MSD type updates to this case (e.g., the GMPLP algorithm in Globerson and Jaakkola, 2008). Furthermore, such cliques can be introduced to improve the accuracy of the LP approximation (Sontag et al., 2008). We thus expect DLPW to allow improved prediction accuracy for a variety of large scale structured prediction problems.

Acknowledgements

This work was supported by BSF grant 2008303. D.S. was supported by a Google PhD Fellowship.

A. Convergence Proof

To simplify the derivation we assume we only have two delta variables per sample point, and denote those by $\delta_1^{(m)}, \delta_2^{(m)}$. All arguments generalize to $\delta^{(m)}$ with more variables. Our objective thus has the form:

$$h(\boldsymbol{w}, \boldsymbol{\delta}) = \sum_m h_m(\boldsymbol{w}, \delta_1^{(m)}, \delta_2^{(m)}) \tag{14}$$

where h includes the $L2$ regularization on $\boldsymbol{w}$. We assume that h is strictly convex w.r.t. its variables. Strict convexity w.r.t. the δ variables is obtained if we use the soft-max loss (see Eq. 13) for any finite value of K. h is strictly convex w.r.t. $\boldsymbol{w}$ because of the $L2$ regularization. The proof also applies (with minor modifications) to the case where y_i are binary, since in this case the max-sum-diffusion updates do not have non-optimal fixed-points and the strict convexity w.r.t. δ is then not needed in the proof.

We wish to show that our algorithm converges to the global minimum of $h(\boldsymbol{w}, \boldsymbol{\delta})$. The updates of our algorithm are as follows. At iteration t choose the next sample point m (for simplicity we shall assume that the sample points are picked according to the same order at every iteration. We also implicitly assume that m is dependent on t but for brevity drop it from the notation) and:

- Choose $\delta_1^{(m,t+1)}$ to minimize $f(\boldsymbol{w}_t, \delta_1^{(m)}, \delta_2^{(m,t)})$
- Choose $\delta_2^{(m,t+1)}$ to minimize $f(\boldsymbol{w}_t, \delta_1^{(m,t+1)}, \delta_2^{(m)})$
- Set $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha_t \partial_{\boldsymbol{w}} h_m(\boldsymbol{w}_t, \delta_1^{(m,t+1)}, \delta_2^{(m,t+1)})$

Note that the update on $\boldsymbol{w}$ may be viewed as a subgradient on the function $\partial_{\boldsymbol{w}} h_m(\boldsymbol{w}_t, \delta_1^{(m,t+1)}, \delta_2^{(m,t+1)})$ when the latter is understood as a function of $\boldsymbol{w}$.

Using the same derivations as in Nedic and Bertsekas (2001, Lemma 2.1 therein) we arrive at the following inequality, which holds at iteration t for every $\boldsymbol{w}$:

$$\begin{aligned}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}\|^2 \leq \;& \|\boldsymbol{w}_t - \boldsymbol{w}\|^2 + \alpha_t^2 D^2 \\ & -2\alpha_t \left[h(\boldsymbol{w}_t, \delta^{(m,t+1)}) - h(\boldsymbol{w}, \delta^{(m,t+1)})\right]\end{aligned} \tag{15}$$

where D is an upper bound on the norm of the subgradient (which is indeed bounded by $\sqrt{C} + \hat{R}$ where $\hat{R}$ is the maximum norm of a feature vector $f(\boldsymbol{x}, \boldsymbol{y})$ as in Shalev-Shwartz et al., 2007). Specifically, the above holds for $\boldsymbol{w} = \boldsymbol{w}(\delta^{(m,t+1)}) = \arg\min_{\hat{\boldsymbol{w}}} h_m(\hat{\boldsymbol{w}}, \delta^{(m,t+1)})$ (this $\boldsymbol{w}$ is unique due to the strict convexity of $h(\boldsymbol{w}, \boldsymbol{\delta})$ in $\boldsymbol{w}$ as a result of the L2 regularization). For this $\boldsymbol{w}$ the difference in h objectives above is always non-negative so that by Eq. 15 every iteration brings $\boldsymbol{w}_t$ closer to its optimal values for the current $\delta^{(m,t)}$, provided that the stepsize α_t is sufficiently small. We now wish to take $t \to \infty$ and analyze the limit point $\bar{\boldsymbol{w}}, \bar{\delta}$. It can be shown via standard methods that such a point exists. Furthermore, using similar arguments to Correa and Lemaréchal (1993, Proposition 1.2) we can conclude that: $h(\boldsymbol{w}(\bar{\delta}), \bar{\delta}) = h(\bar{\boldsymbol{w}}, \bar{\delta})$ and thus $\bar{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} h(\boldsymbol{w}, \bar{\delta})$.

We now wish to prove that $\bar{\delta}$ is optimal for $\bar{\boldsymbol{w}}$. This is done as in standard coordinate descent analyses (e.g., Bertsekas, 1995, page 274). The update of $\delta_1^{(m,t+1)}$ requires $h_m(\boldsymbol{w}_t, \delta_1^{(m,t+1)}, \delta_2^{(m,t)}) \leq h_m(\boldsymbol{w}_t, \delta_1^{(m)}, \delta_2^{(m,t)})$ for all values of $\delta_1^{(m)}$. Taking $t \to \infty$ we have $h_m(\bar{\boldsymbol{w}}, \bar{\delta}_1^{(m)}, \bar{\delta}_2^{(m)}) \leq h_m(\bar{\boldsymbol{w}}, \delta_1^{(m)}, \bar{\delta}_2^{(m)})$. This implies that the limit value $\bar{\delta}_1^{(m)}$ is optimal with respect to all other coordinate (δ and $\boldsymbol{w}$). We can show this *local optimality* for all coordinates of δ, and by the properties of strictly convex functions we obtain that $\bar{\delta} = \arg\min_\delta h(\bar{\boldsymbol{w}}, \delta)$.

Taken together, the above arguments show that $\bar{\boldsymbol{w}}, \bar{\delta}$ are the minimizers of $h(\boldsymbol{w}, \delta)$ when one of the two variables is fixed to either $\bar{\boldsymbol{w}}$ or $\bar{\delta}$. Strict convexity of h then implies that $\bar{\boldsymbol{w}}, \bar{\delta}$ is the global optimum of $h(\boldsymbol{w}, \boldsymbol{\delta})$.

References

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. The MIT Press, 2007.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

M. Boutell, J. Luo, X. Shen, and C. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37 (9):1757–1771, 2004.

D. Burshtein. Iterative approximate linear programming decoding of ldpc codes with linear complexity. *IEEE Trans. on Information Theory*, 55(11):4835–4859, 2009.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8, 2002.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *JMLR*, 9:1775–1822, 2008.

R. Correa and C. Lemaréchal. Convergence of some algorithms for convex minimization. *Math. Program.*, 62(2): 261–275, 1993.

A. Elisseeff and J. Weston. Kernel methods for multi-labelled classification and categorical regression problems. In *NIPS 14*, pages 681–687, 2001.

T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *ICML 25*, pages 304–311, New York, NY, USA, 2008. ACM.

A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS 20*, pages 553–560. 2008.

T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

J. K. Johnson, D. Malioutov, and A. S. Willsky. Lagrangian relaxation for map estimation in graphical models. In *45th Annual Allerton Conference on Communication, Control and Computing*, September 2007.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on PAMI*, 28(10):1568–1583, 2006.

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.

A. Kulesza and F. Pereira. Structured learning with approximate inference. In *NIPS 20*, pages 785–792. 2008.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 18*, pages 282–289, 2001.

D. Lewis, , Y. Yang, T. Rose, and F. Li. RCV1: a new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

A. F. T. Martins, N. A. Smith, and E. P. Xing. Polyhedral outer approximations with application to natural language parsing. In *ICML 26*, pages 713–720, 2009.

A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. on Optimization*, 12(1):109–138, 2001.

J. C. Platt. Fast training of Support Vector Machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

N. Ratliff, J. A. D. Bagnell, and M. Zinkevich. (Online) subgradient methods for structured prediction. In *AISTATS*, 2007.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML 24*, pages 807–814, New York, NY, 2007. ACM Press.

S. Shalev-Shwartz and N. Srebro. Theory and practice of support vector machines optimization. In J. Keshet and S. Bengio, editors, *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*. 2009.

D. Sontag and T. Jaakkola. Tree block coordinate descent for MAP in graphical models. In *AISTATS 12*, 2009.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *UAI 24*, pages 503–510, 2008.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: a large margin approach. In *ICML 22*, pages 896–903, 2005.

B. Taskar, C. Guestrin, and D. Koller. Max margin Markov networks. In *NIPS 16*, pages 25–32. 2004.

B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction, dual extragradient and Bregman projections. *JMLR*, pages 1627–1653, 2006.

T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1165–1179, 2007.

2.4 Learning Max-Margin Tree Predictors

Learning Max-Margin Tree Predictors

Ofer Meshi†* **Elad Eban**†* **Gal Elidan**‡ **Amir Globerson**†

† School of Computer Science and Engineering
‡ Department of Statistics
The Hebrew University of Jerusalem, Israel

Abstract

Structured prediction is a powerful framework for coping with joint prediction of interacting outputs. A central difficulty in using this framework is that often the correct label dependence structure is unknown. At the same time, we would like to avoid an overly complex structure that will lead to intractable prediction. In this work we address the challenge of learning tree structured predictive models that achieve high accuracy while at the same time facilitate efficient (linear time) inference. We start by proving that this task is in general NP-hard, and then suggest an approximate alternative. Our CRANK approach relies on a novel Circuit-RANK regularizer that penalizes non-tree structures and can be optimized using a convex-concave procedure. We demonstrate the effectiveness of our approach on several domains and show that its accuracy matches that of fully connected models, while performing prediction substantially faster.

1 Introduction

Numerous applications involve joint prediction of complex outputs. For example, in document classification the goal is to assign the most relevant (possibly multiple) topics to each document; in gene annotation, we would like to assign each gene a set of relevant functional tags out of a large set of possible cellular functions; in medical diagnosis, we would like to identify all the diseases a given patient suffers from. Although the output space in such problems is typically very large, it often has intrinsic *structure* which can be exploited to construct efficient predictors. Indeed, in recent years using *structured output prediction* has resulted in state-of-the-art results in many real-worlds problems from computer vision, natural language processing, computational biology, and other fields [Bakir et al., 2007]. Such predictors can be learned from data using formulations such as Max-Margin Markov Networks (M^3N) [Taskar et al., 2003, Tsochantaridis et al., 2006], or conditional random fields (CRF) [Lafferty et al., 2001].

While the prediction and the learning tasks are generally computationally intractable [Shimony, 1994, Sontag et al., 2010], for some models they can be carried out efficiently. For example, when the model consists of pairwise dependencies between output variables, and these form a tree structure, prediction can be computed efficiently using dynamic programming at a linear cost in the number of output variables [Pearl, 1988]. Moreover, despite their simplicity, tree structured models are often sufficiently expressive to yield highly accurate predictors. Accordingly, much of the research on structured prediction focused on this setting [e.g., Lafferty et al., 2001, Collins, 2002, Taskar et al., 2003, Tsochantaridis et al., 2006].

Given the above success of tree structured models, it is unfortunate that in many scenarios, such as a document classification task, there is no obvious way in which to choose the most beneficial tree. Thus, a natural question is how to find the tree model that best fits a given structured prediction problem. This is precisely the problem we address in the current paper. Specifically, we ask what is the tree structure that is optimal in terms of a max-margin objective [Taskar et al., 2003]. Somewhat surprisingly, this optimal tree problem has received very little attention in the context of discriminative structured prediction (the most relevant work is Bradley and Guestrin [2010] which we address in Section 6).

Our contributions are as follows. We begin by proving that it is NP-hard in general to find the optimal max-margin predictive tree, in marked contrast to

*Authors contributed equally.

the generative case where the optimal tree can be learned efficiently [Chow and Liu, 1968]. To cope with this theoretical barrier, we propose an approximation scheme that uses regularization to penalize non-tree models. Concretely, we propose a regularizer that is based on the *circuit rank* of a graph [Berge, 1962], namely the minimal number of edges that need to be removed from the graph in order to obtain a tree. Minimization of the resulting objective is still difficult, and we further approximate it using a difference of continuous convex envelopes. The resulting objective can then be readily optimized using the convex concave procedure [Yuille and Rangarajan, 2003].

We apply our method to synthetic and varied real-world structured output prediction tasks. First, we show that the learned tree model is competitive with a fully connected max-margin model that is substantially more computationally demanding at prediction time. Second, we show that our approach is superior to several baseline alternatives (e.g., greedy structure learning) in terms of generalization performance and running time.

2 The Max-margin Tree

Let x be an input vector (e.g., a document) and y a discrete output vector (e.g., topics assigned to the document, where $y_i = 1$ when topic i is addressed in x). As in most structured prediction approaches, we assume that inputs are mapped to outputs according to a linear discrimination rule: $y(x;w) = \text{argmax}_{y'} w^\top \phi(x,y')$, where $\phi(x,y)$ is a function that maps input-output pairs to a feature vector, and w is the corresponding weight vector. We will call $w^\top \phi(x,y')$ the *score* that is assigned to the prediction y' given an input x.

Assume we have a set of M labeled pairs $\{(x^m, y^m)\}_{m=1}^M$, and would like to learn w. In the M^3N formulation proposed by Taskar et al. [2003], w is learned by minimizing the following (regularized) structured hinge loss:

$$\ell(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_m h^m(w),$$

where

$$h^m(w) = \max_y \left[w^\top \phi(x^m, y) + \Delta(y, y^m) \right] - w^\top \phi(x^m, y^m), \tag{1}$$

and $\Delta(y, y^m)$ is a label-loss function measuring the cost of predicting y when the true label is y^m (e.g., 0/1 or Hamming distance). Thus, the learning problem involves a loss-augmented prediction problem for each training example.

Since the space of possible outputs may be quite large, maximization of y can be computationally intractable. It is therefore useful to consider score functions that decompose into simpler ones. One such decomposition that is commonly used consists of scores over single variables and pairs of variables that correspond to nodes and edges of a graph G, respectively:

$$w^\top \phi(x,y) = \sum_{ij \in E(G)} w_{ij}^\top \phi_{ij}(x, y_i, y_j) + \sum_{i \in V(G)} w_i^\top \phi_i(x, y_i). \tag{2}$$

Importantly, when the graph G has a tree structure then the maximization over y can be solved exactly and efficiently using dynamic programming algorithms (e.g., Belief Propagation [Pearl, 1988]).

As mentioned above, we consider problems where there is no natural way to choose a particular tree structure, and our goal is to learn the optimal tree from training data. We next formalize this objective.

In a tree structured model, the set of edges ij in Eq. (2) forms a tree. This is equivalent to requiring that the vectors w_{ij} in Eq. (2) be non-zero only on edges of some tree. To make this precise, we first define, for a *given* spanning tree T, the set $\mathcal{W}_T$ of weight vectors that "agree" with T:[1]

$$\mathcal{W}_T = \{w : \ ij \notin T \implies w_{ij} = 0\}. \tag{3}$$

Next we consider the set $\mathcal{W}_\cup$ of weight vectors that agree with *some* spanning tree. Denote the set of all spanning trees by $\mathcal{T}$, then: $\mathcal{W}_\cup = \bigcup_{T \in \mathcal{T}} \mathcal{W}_T$. The problem of finding the optimal max-margin tree predictor is therefore:

$$\min_{w \in \mathcal{W}_\cup} \ell(w). \tag{4}$$

We denote this as the $M^{Tree}N$ problem. In what follows, we first show that this problem is NP-hard, and then present an approximation scheme.

3 Learning M^3N Trees is NP-hard

We start by showing that learning the optimal tree in the discriminative max-margin setting is NP-hard. As noted, this is somewhat of a surprise given that the best tree is easily learned in the generative setting [Chow and Liu, 1968], and that tree structured models are often used due to their computational advantages.

In particular, we consider the problem of deciding whether there exists a tree structured model that correctly labels a given dataset (i.e., deciding whether the

[1]Note that weights corresponding to single node features are not restricted.

dataset is separable with a tree model). Formally, we define the $M^{Tree}N$ *decision problem* as determining whether the following set is empty:

$$\left\{w \in \mathcal{W}_{\cup} \middle| w^\top \phi(x^m, y^m) \geq w^\top \phi(x^m, y) + \Delta(y, y^m)\ \forall m, y\right\}. \tag{5}$$

To facilitate the identifiability of the model parameters that is later needed, we adopt the formalism of Sontag et al. [2010] and define the score:

$$\begin{aligned} & S(y; x, T, w) \\ & = \sum_{ij \in T} w_{ij}^\top \phi_{ij}(x, y_i, y_j) + \sum_i (w_i^\top \phi_i(x, y_i) + x_i(y_i)) \\ & \equiv \sum_{ij \in T} \theta_{ij}(y_i, y_j) + \sum_i \theta_i(y_i) + \sum_i x_i(y_i), \end{aligned} \tag{6}$$

where $x_i(y_i)$ is a bias term which does not depend on w,[2] and for notational convenience we have dropped the dependence of θ on x and w. We can now reformulate the set in Eq. (5) as:

$$\left\{T, w \middle| S(y^m; x^m, T, w) \geq \max_y S(y; x^m, T, w)\ \forall m\right\}, \tag{7}$$

where, for simplicity, we omit the label loss $\Delta(y, y^m)$. This is valid since the bias terms already ensure that the trivial solution $w = 0$ is avoided. With this reformulation, we can now state the hardness result.

Theorem 3.1. *Given a set of training examples $\{(x^m, y^m)\}_{m=1}^M$, it is NP-hard to decide whether there exists a tree T and weights w such that $\forall m, y^m = \mathrm{argmax}_y\, S(y; x^m, T, w)$.*

Proof. We show a reduction from the NP-hard *bounded-degree spanning tree* (BDST) problem to the $M^{Tree}N$ decision problem defined in Eq. (7).[3] In the BDST problem, given an undirected graph G and an integer D, the goal is to decide whether there exists a spanning tree with maximum degree D (for $D = 2$ this is the Hamiltonian path problem, hence the NP-hardness).

Given an instance of BDST we construct an instance of problem Eq. (7) on the same graph G as follows. First, we define variables $y_1, \ldots, y_n$ that take values in $\{0, 1, \ldots, n\}$, where $n = |V(G)|$. Second, we will define the parameters $\theta_i(y_i)$ and $\theta_{ij}(y_i, y_j)$ and bias terms $x_i(y_i)$ in such a way that solving the $M^{Tree}N$ decision problem will also solve the BDST problem. To complement this, we will define a set of training examples which are separable only by the desired parameters. For clarity of exposition, we defer the proof that these parameters are identifiable using a polynomial number of training examples to App. A.

The singleton parameters are

$$\theta_i(y_i) = \begin{cases} D & i = 1, y_1 = 0 \\ 0 & \text{otherwise}, \end{cases} \tag{8}$$

and the pairwise parameters for $ij \in E(G)$ are:

$$\theta_{ij}(y_i, y_j) = \begin{cases} -n^2 & y_i \neq y_j \\ 0 & y_i = y_j = 0 \\ 1 & y_i = y_j = i \\ 1 & y_i = y_j = j \\ 0 & \text{otherwise}. \end{cases} \tag{9}$$

Now consider the case where the bias term $x_i(y_i)$ is identically zero. In this case the score for (non-zero) uniform assignments equals the degree of the vertex in T. That is, $S(i, \ldots, i; 0, T, \theta) = \deg_T(i)$ since $\theta_{ij}(i, i) = 1$ for all $j \in N(i)$ and the other parameter values are zero. The score for the assignment $y = 0$ is $S(0, \ldots, 0; 0, T, \theta) = D$, and for non-uniform assignments we get a negative score $S(y; 0, T, \theta) < 0$. Therefore, the maximization over y in Eq. (7) reduces to a maximization over n uniform states (each corresponding to a vertex in the graph). The maximum value is thus the maximum between D and the maximum degree in T: $\max_y S(y; 0, T, \theta) = \max\{D, \max_i \deg_T(i)\}$.

It follows that, if we augment the training set that realizes the above parameters (see App. A) with a single training example where $x^m = y^m = (0, \ldots, 0)$, the set of Eq. (7) can now be written as:

$$\left\{T \middle| D \geq \max_i \deg_T(i)\right\}.$$

Thus, we have that the learning problem is separable if and only if there exists a bounded degree spanning tree in G. This concludes the reduction, and we have shown that the decision problem in Theorem 3.1 is indeed NP-hard. □

The above hardness proof illustrates a striking difference between generative learning of tree models (i.e., Chow Liu) and discriminative learning (our NP-hard setting). Clearly, we do not want to abandon the discriminative setting and trees remain computationally appealing at test time. Thus, in the rest of the paper, we propose practical approaches for learning a tree structured predictive model and demonstrate empirically that our method is competitive.

[2] As usual, the bias term can be removed by fixing some elements of w to 1.

[3] A related reduction is given in Aissi et al. [2005], Theorem 8.

4 Tree-Inducing Regularization

Due to the hardness of the tree learning problem, we next develop an approximation scheme for it. We begin with the exact formulation of the problem, and then introduce an approximate formulation along with an optimization procedure. Our construction relies on several properties of submodular functions and their convex envelopes.

4.1 Exact Tree Regularization

As described in Section 2, we would like to find a tree structured weight vector $w \in \mathcal{W}_{\cup}$ that minimizes the empirical hinge loss. The main difficulty in doing so is that the sparsity pattern of w needs to obey a fairly complex constraint, namely being tree structured. This is in marked contrast to popular sparsity constraints such as an upper bound on the number of non-zero values, a constraint that does not take into account the resulting global structure.

To overcome this difficulty, we will formulate the exact learning problem via an appropriate regularization. We begin by defining a function that maps w to the space of edges: $\pi : \mathbb{R}^d \mapsto \mathbb{R}^{|E|}$, where E corresponds to all edges of the full graph. Specifically, the component in π corresponding to the edge ij is:

$$\pi_{ij}(w) = \|w_{ij}\|_1.$$

Now, denote by $Supp(\pi(w))$ the set of coordinates in $\pi(w)$ that are non-zero. We would like the edges corresponding to these coordinates to form a tree graph. Thus, we wish to define a set function $F(Supp(\pi(w)))$ which will be equal to zero if $Supp(\pi(w))$ conforms to *some* tree structure, and a positive value otherwise. If we then add $\beta F(Supp(\pi(w)))$ to the objective in Eq. (4) with β large enough, the resulting w will be tree structured. The optimization problem is then:

$$\min_w \ell(w) + \beta F(Supp(\pi(w))). \qquad (10)$$

In what follows, we define a set function $F(A)$ with the desired properties. That is, we seek a function that takes a set of edges A and outputs zero if they correspond to a tree, and a positive number otherwise. Intuitively, it is also desirable to define the function such that its value increases as the graph becomes less "tree-like". To make this concrete we define a measure for "treeness" as the minimum number of edges that need to be removed from A in order to reduce it to a tree structure. This measure is also known as the *circuit-rank* of the graph [Berge, 1962]. Formally:

$$r = |A| + c(A) - n,$$

where $c(A)$ is the number of connected components in the graph, and n is the number of vertices. We note that the circuit rank is also the co-rank of the graphic matroid, and is hence supermodular.

Putting it all together, we have that our desired tree-inducing function is given by:

$$F(Supp(\pi(w))) = |Supp(\pi(w))| + c(Supp(\pi(w))) - n.$$

Of course, given our hardness result, optimizing Eq. (10) with the above $F(A)$ is still computationally hard. From an optimization perspective, the difficulty comes from the non-convexity of the the above function in w. Optimization is further complicated by the fact that it is highly non-smooth, similarly to the ℓ_0 norm. In the next section we suggest a smoothed approximation that is more amenable to optimization.

We also note that in Eq. (10) w is generally not tree structured, so the maximization over y in Eq. (1) is not necessarily tractable. Therefore, we replace the hinge loss in Eq. (1) with its *overgenerating* approximation [Finley and Joachims, 2008], known as linear programming (LP) relaxation [e.g., see Meshi et al., 2010]. This is achieved by formulating the optimization in Eq. (1) as an integer LP and then relaxing the integrality requirement, allowing fractional solutions. Importantly, for tree structured graphs this approximation is in fact exact [Wainwright and Jordan, 2008]. This implies that if there exists a tree model that separates the data, it will be found even when using this relaxation in Eq. (10). With slight abuse of notation, we keep referring to the approximate objective as $\ell(w)$.

4.2 Approximate Tree Regularization

The function $F(A)$ is a set function applied to the support of $\pi(w)$. Bach [2010] (Proposition 1) shows that when F is submodular *and* non-decreasing, the convex envelope of $F(Supp(\pi(w)))$ can be calculated efficiently. This is desirable since the convex envelope then serves as a convex regularizer. Furthermore, this convex envelope can be elegantly understood as the Lovász extension f of F, applied to $|\pi(w)|$ (in our case, $|\pi(w)| = \pi(w)$). Unfortunately, the circuit-rank r does not satisfy these conditions, since it is supermodular and non-decreasing (in fact, its convex envelope is a constant).

To overcome this difficulty, we observe that $F(A)$ can be decomposed in a way that allows us to use the result of Bach [2010]. Specifically, we can write $F(A) = F_1(A) - F_2(A)$, where

$$F_1(A) = |A| \quad , \quad F_2(A) = n - c(A). \qquad (11)$$

$F_1(A)$ is simply the cardinality function which is modular and increasing. Furthermore, $F_2(A)$ is the rank of the graphic matroid [Oxley, 2006], and is hence submodular. It is also easy to see that $F_2(A)$ is non-decreasing. Thus, both functions satisfy the conditions of Proposition 1 in Bach [2010] and their convex envelopes can be found in closed form, as characterized in the following corollaries.

Corollary 4.1. *The convex envelope of* $F_1(Supp(\pi(w))$ *is* $f_1(\pi(w)) = \sum_{ij} \pi_{ij}(w) = \|w\|_1$.

Proof. Follows directly from Prop. 1 in Bach [2010] and the fact that the Lovász extension of the cardinality function is the ℓ_1 norm. □

Corollary 4.2. *The convex envelope of* $F_2(Supp(\pi(w))$ *is* $f_2(\pi(w))$, *defined as follows. Sort the elements of* $\pi(w)$ *in decreasing order, and construct a maximum-spanning-tree with this ordering as in Kruskal's algorithm [Kruskal, 1956]. Denoting the resulting tree by* $T(\pi(w))$, *we obtain*

$$f_2(\pi(w)) = \sum_{ij\in T(\pi(w))} \pi_{ij}(w) = \sum_{ij\in T(\pi(w))} \|w_{ij}\|_1$$

Proof. Let $(ij)_k$ denote the k^{th} edge when sorting $\pi(w)$, then the Lovász extension f_2 of F_2 at $\pi(w)$ is:

$$\sum_{k=1}^{|E|} \pi_{(ij)_k}(w)[F_2(\{(ij)_1,\ldots,(ij)_k) - F_2(\{(ij)_1,\ldots,(ij)_{k-1})]$$
$$= \sum_{ij\in T(\pi(w))} \pi_{ij}(w),$$

where we have used Eq. (11) and the fact that the number of connected components decreases by one only when introducing edges in Kruskal's tree. The desired result follows from Prop. 1 in [Bach, 2010]. □

We now approximate $F(Supp(\pi(w)))$ as a difference of the two corresponding convex envelopes, denoted by $f(\pi(w))$:

$$f(\pi(w)) \equiv f_1(\pi(w)) - f_2(\pi(w)) = \sum_{ij\notin T(\pi(w))} \|w_{ij}\|_1 \tag{12}$$

This function has two properties that make it computationally and conceptually appealing:

- $f(\pi(w))$ is a difference of two convex functions so that a local minimum can be easily found using the convex concave procedure (see Section 4.3).
- The set $\{ij \notin T(\pi(w))\}$ are precisely these edges that form a cycle when added according to the order implied by $\pi(w)$. Thus, the penalty we use corresponds to the magnitude of $\|w_{ij}\|_1$ on the edges that form cycles, namely the non-tree edges.

4.3 Optimizing with Approximate Tree Regularization

Using the tree inducing regularizer from the previous section, our overall optimization problem becomes:

$$\min_w \ell(w) + \beta f_1(\pi(w)) - \beta f_2(\pi(w)).$$

Since the function $f_2(\pi(w))$ is rather elaborate, optimizing the above objective still requires care. In what follows, we introduce a simple procedure for doing so that utilizes the convex concave procedure (CCCP) [Yuille and Rangarajan, 2003].[4] Recall that CCCP is applicable for an objective function (to be minimized) that is a sum of a convex and concave functions, and proceeds via linearization of the concave part.

To use CCCP for our problem we observe that from the discussion of the previous section it follows that our objective can indeed be decomposed into the following convex and concave components:

$$h_\cup(w) = \ell(w) + \beta f_1(\pi(w)), \quad h_\cap(w) = -\beta f_2(\pi(w)),$$

where $\cap$ and $\cup$ correspond to the convex and concave parts, respectively. To linearize $h_\cap(w)$ around a point w^t, we need to find its subgradient at that point. The next proposition, which follows easily from Hazan and Kale [2009], gives the subgradient of $f_2(\pi(w))$:

Proposition 4.3. *The subgradient of* $f_2(\pi(w))$ *is given by the vector* v *defined as follows.*[5] *The coordinates in* v *corresponding to* w_{ij} *are given by:*

$$v_{ij} = \begin{cases} sign(w_{ij}) & ij \in T(\pi(w)) \\ 0 & otherwise \end{cases}$$

where sign is taken element wise. The other coordinates of v *(corresponding to* w_i*) are zero.*

We can now specify the resulting algorithm, which we call CRANK for circuit-rank regularizer.

Algorithm 1 The CRANK algorithm

```
Input: $w^1, \beta$
for $t = 1, \ldots$ do
    $h^t(w) = \ell(w) + \beta\|w\|_1 - \beta v(w^t)^\top w$
    $w^{t+1} = \text{argmin}_w h^t(w)$
end for
```

The objective $h^t(w)$ to be minimized at each iteration is a convex function, which can be optimized using any convex optimization method. In this work we use the

[4] To be precise, we are using the more general DC programming framework [Horst and Thoai, 1999], which can be applied to non differentiable functions.

[5] In cases where several w_{ij} are equal, there are multiple subgradients.

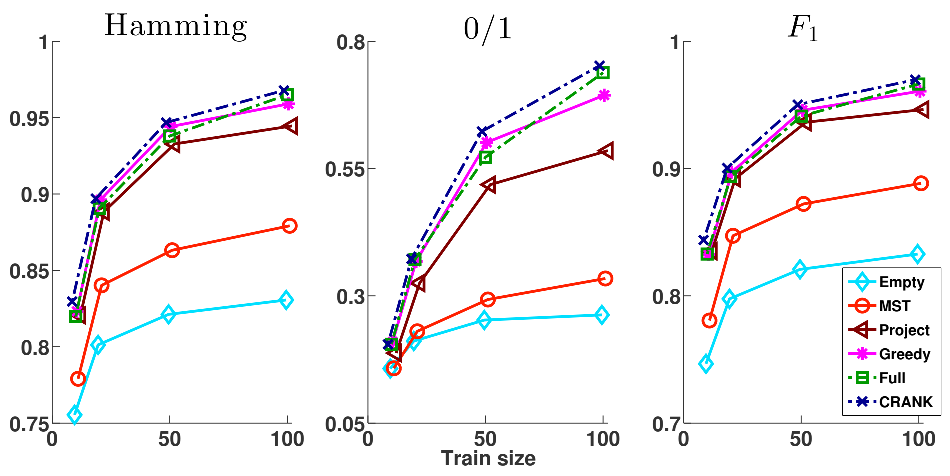


Figure 1: Average test performance as a function of the number of training samples for the synthetic datasets.

stochastic Frank-Wolfe procedure recently proposed by Lacoste-Julien et al. [2013].[6] The advantage of this approach is that the updates are simple, and it generates primal and dual bounds which help monitor convergence. In practice, we do not solve the inner optimization problems exactly, but rather up to some primal-dual gap.

5 Experiments

In this section we evaluate the proposed algorithm on multi-label classification tasks and compare its performance to several baselines. In this task the goal is to predict the subset of labels which best fits a given input. We use the model presented in Finley and Joachims [2008], where each possible label $y_i \in \{0, 1\}$ is associated with a weight vector w_i, the singleton scores are given by $w_i^\top x y_i$, and the pairwise scores are simply $w_{ij} y_i y_j$ (i.e., w_{ij} is scalar).

We compare our **CRANK** algorithm to the following baselines: The **Empty** model learns each label prediction independently of the other labels; The **Full** model learns a model that can use all pairwise dependencies; The **Greedy** trainer starts with the empty model and at each iteration adds the edge which achieves the largest gain in objective while not forming a cycle, until no more edges can be added; The **Project** algorithm, runs CRANK starting from the weights learned by the Full algorithm, and using a large penalty β; [7] The final baseline is an **MST** algorithm, which calculates the gain in objective for each edge separately, takes a maximum-spanning-tree over these weights, and then re-trains the resulting tree. Since CCCP may be sensitive to its starting point, we restart CRANK from 10 random points and choose the one with lowest objective (we run those in parallel). We apply the stochastic Frank-Wolfe algorithm [Lacoste-Julien et al., 2013] to optimize the weights in all algorithms. The Full and CRANK algorithms operate on non-tree graphs, and thus use an LP relaxation within the training loss (see Section 4.1). Since the model trained with Full is not tree structured, we also needs to use LP relaxation at test time (see implications on runtime below). We used the GLPK solver for solving the LPs.

To measure the performance of the algorithms, we consider three accuracy measures: **Hamming**, **zero-one**, and $\boldsymbol{F_1}$ (averaged over samples). See [Zhang and Schneider, 2012, Dembczynski et al., 2010] for similar evaluation schemes. Regularization coefficients were chosen using cross-validation. The parameter β in CRANK was gradually increased until a tree structure was obtained.

Synthetic Data: We first show results for synthetic data where $x \in \mathbb{R}^4$. The data was created as follows: a random tree T over $n = 10$ variables was picked and corresponding weights $w \in \mathcal{W}_T$ were sampled. Train and test sets were generated randomly and labeled using w. Test set size was 1000 and train set size varied. Results (averaged over 10 repetitions) are shown in Figure 1.

We observe that the structured models do significantly better than the Empty model. Additionally, we see that the Full, Greedy, and CRANK algorithms are comparable in terms of prediction quality, with a slight advantage for CRANK over the others. We also notice that Project and MST do much worse than the other structured models.

[6]We modified the algorithm to handle the $\ell_1 + \ell_2$ case.
[7]The Project scheme thus trains a model consisting of the maximum-spanning-tree over the weights learned by Full, and can be viewed as a "tree-projection" of the full model.

	Hamming	0/1	F_1	Hamming	0/1	F_1
	Scene			Emotions		
CRANK	90.5 (2)	58.9 (2)	64.8 (2)	**79.2** (1)	29.2 (2)	60.5 (2)
Full	**90.7** (1)	**62.2** (1)	**67.8** (1)	79.0 (3)	**33.7** (1)	**62.5** (1)
Greedy	90.2 (3)	56.9 (3)	62.6 (3)	78.5 (4)	24.3 (4)	54.5 (4)
Project	89.5 (5)	52.1 (5)	59.2 (5)	77.6 (5)	20.8 (5)	49.8 (5)
MST	89.9 (4)	53.0 (4)	59.6 (4)	79.1 (2)	28.2 (3)	57.5 (3)
Empty	89.3 (6)	49.5 (6)	56.5 (6)	76.7 (6)	20.3 (6)	48.5 (6)
	Medical			Yeast		
CRANK	**96.9** (1)	74.0 (2)	78.2 (3)	80.1 (2)	17.6 (2)	60.4 (3)
Full	**96.9** (1)	**75.0** (1)	**78.3** (1)	**80.2** (1)	**19.0** (1)	**60.9** (1)
Greedy	**96.9** (1)	74.0 (2)	**78.3** (1)	NA	NA	NA
Project	96.7 (4)	72.7 (4)	77.0 (5)	80.1 (2)	16.4 (3)	60.7 (2)
MST	96.7 (4)	71.9 (5)	77.5 (4)	80.1 (2)	16.1 (4)	60.3 (4)
Empty	96.4 (6)	71.0 (6)	76.1 (6)	79.8 (5)	12.1 (5)	58.0 (5)

Table 1: Performance on test data for real-world multi-label datasets. The rank of each algorithm for each dataset and evaluation measure is shown in brackets. Greedy was too slow to run on Yeast.

Real Data: We next performed experiments on four real-world datasets.[8] In the **Scene** dataset the task is to classify a given image into several outdoor scene types (6 labels, 294 features). In the **Emotions** dataset we wish to assign emotional categories to musical tracks (6 labels, 72 features). In the **Medical** dataset the task is document classification (reduced to 10 labels, 1449 features). This is the experimental setting used by Zhang and Schneider [2012]. Finally, to test how training time scales with the number of labels, we also experiment with the **Yeast** dataset, where the goal is to predict which functional classes each gene belongs to (14 labels, 103 features). The results are summarized in Table 1.

We first observe that in this setting the Full model generally has the best performance and Empty the worst. As before, CRANK is typically close to Full and outperforms Greedy and the other baselines in the majority of the cases.

Runtime analysis: In Table 2 we report train and test run times, relative to the Full model, for the Yeast dataset.

Table 2: Running times for the Yeast dataset.

Time	CRANK	Full	Project	MST	Empty
Train	1.82	1.0	0.17	1.32	0.01
Test	0.02	1.0	0.06	0.02	0.02

It is important to note that the greedy algorithm is very slow to train, since it requires solving $O(n^3)$ training problems. It is thus impractical for large problems, and this is true even for the Yeast dataset which has only $n = 14$ labels (and hence does not appear in Table 2). The Full model on the other hand has much longer test times (compared to the tree-based models), since it must use an LP solver for prediction. CRANK has a training time that is reasonable (comparable to Full) and a much better test time. On the Yeast dataset prediction with CRANK is 50 times faster than with Full.

In conclusion, CRANK seems to strike the best balance in terms of accuracy, training time, and test time: it achieves an accuracy that is close to the best among the baselines, with a scalable training time, and very fast test time. This is particularly appealing for applications where we can afford to spend some more time on training while test time is critical (e.g., real-time systems).

6 Related Work

The problem of structure learning in graphical models has a long history, dating back to the celebrated algorithm by Chow and Liu [1968] for finding a maximum likelihood tree in a generative model. More recently, several elegant works have shown the utility of ℓ_1 regularization for structure learning in generative models [Friedman et al., 2008, Lee et al., 2007, Ravikumar et al., 2008, 2010]. These works involve various forms of ℓ_1 regularization on the model parameters, coupled with approximation of the likelihood function (e.g., pseudo-likelihood) when the underlying model is non-Gaussian. Some of these works also provide finite sample bounds on the number of of samples required to correctly reconstruct the model structure. Unlike ours, these works focus on generative models, a difference

[8]Taken from Mulan (`http://mulan.sourceforge.net`)

that can be quite fundamental. For example, as we proved in Section 3, learning trees is a problem that is NP-hard in the M^3N setting while it is *polynomial* in the number of variables in the generative one. We note that we are not aware of finite sample results in the discriminative setting, where a different MRF is considered for each input x.

In the discriminative setting, Zhu et al. [2009] define an ℓ_1 regularized M^3N objective and present algorithms for its optimization. However, this approach does not consider the structure of the underlying graphical model over outputs y. Torralba et al. [2004] propose a greedy procedure based on boosting to learn the structure of a CRF, while Schmidt et al. [2008] present a block-ℓ_1 regularized pseudo-likelihood approach for the same task. While these methods do consider the structure of the graphical model, they do not attempt to produce tractable predictors. Further, they do not aim to learn models using a max-margin objective.

Bradley and Guestrin [2010] address the problem of learning trees in the context of conditional random fields. They show that using a particular type of tractable edge scores together with a maximum-spanning-tree (MST) algorithm may fail to find the optimal tree structure. Our work differs from theirs in several aspects. First, we consider the max-margin setting for structured prediction rather than the CRF setting. Second, they assume that the feature function $\phi(x, y)$ has a particular form where y and x are of the same order and where the outputs depend only on local inputs. Finally, we do not restrict our attention to MST algorithms. Consequently, our hardness result is more general and the approximations we propose are quite different from theirs. Finally, Chechetka and Guestrin [2010] also consider the problem of learning tree CRFs. However, in contrast to our work, they allow the structure of the tree to depend on the input x, which is somewhat more involved as it requires learning an additional mapping from inputs to trees.

7 Conclusion

We tackled the challenge of learning tree structured prediction models. To the best of our knowledge, ours is the first work that addresses the problem of structure learning in a discriminative M^3N setting. Moreover, unlike common structured sparsity approaches that are used in the setting of generative and conditional random field models, we explicitly target tree structures due to their appealing properties at test time. Following a proof that the task is NP-hard in general, we proposed a novel approximation scheme that relies on a circuit rank regularization objective that penalizes non-tree models, and that can be optimized using the CCCP algorithm. We demonstrated the effectiveness of our CRANK algorithm in several real-life domains. Specifically, we showed that CRANK obtained accuracies very close to those achieved by a full-dependence model, but with a much faster test time.

Many intriguing questions arise from our work: Under which conditions can we learn the optimal model, or guarantee approximation quality? Can we extend our framework to the case where the tree depends on the input? Can we use a similar approach to learn other graphs, such as low treewidth or high girth graphs? Such settings introduce novel forms of *graph-structured* sparsity which would be interesting to explore.

Acknowledgments

This research is funded by the ISF Centers of Excellence grant 1789/11, and by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI). Ofer Meshi is a recipient of the Google Europe Fellowship in Machine Learning, and this research is supported in part by this Google Fellowship.

A Realizing the parameters

Here we complete the hardness proof in Section 3 by describing the training set that realizes the parameters of Eq. (8) and Eq. (9). The approach below makes use of two training samples to constrain each parameter, one to upper bound it and one to lower bound it. Appealingly, the training samples are constructed in such a way that other samples do not constrain the parameter value, allowing us to realize the needed value for each and every parameter in the model.

The construction is similar to Sontag et al. [2010]. For all parameters it consists of the following steps: (i) define an assignment $x(y)$; (ii) identify two y values that potentially maximize the score; and (iii) show that these two complete assignments to x and y force the desired parameter value.

Preliminaries: Recall that the parameter vector θ is defined in Eq. (6) via a product of the features $\phi_i(y_i, x)$ and $\phi_{ij}(y_i, y_j, x)$ and the weights w_i and w_{ij} which do not depend on x or y and are shared across the parameters. For the hardness reduction, it will be convenient to set the features to indicator functions and show that the implied weight values are realizable. Specifically, we set the features to:

$$\begin{aligned}\phi_{ij}^{\text{off-diag}}(y_i, y_j) =& \mathbb{1}\{y_i \neq y_j\} \; \forall i, j \\ \phi_{ij}^{\text{diag}}(y_i, y_j) =& \mathbb{1}\{y_i = y_j = i \text{ or } y_i = y_j = j\} \; \forall i, j \\ \phi_1^{\text{bound}}(y_1) =& \mathbb{1}\{y_1 = 0\}\end{aligned}$$

Recall that to realize the desired parameters θ, we need to introduce training samples such that for all i, j:

$$w_{ij}^{\text{off-diag}} = -n^2, \quad w_{ij}^{\text{diag}} = 1, \quad w_1^{\text{bound}} = D,$$

with the dimension of w equalling $2|E(G)| + 1$.

Finally, using N_i to denote the set of neighbors of node i in the graph G, we will use the following (large) value to force variables not to take some chosen states:

$$\gamma_i = \begin{cases} 1 + |N_i|(n^2+1) & i \neq 1 \\ 1 + |N_i|(n^2+1) + D & i = 1 \end{cases}$$

Realizing the weight w_1^{bound}: We define $x(y)$ as:

$$x_1(y_1) = \begin{cases} 0 & y_1 = 0 \\ -D & y_1 = 1 \\ -\gamma_1 & y_1 \geq 2 \end{cases}$$

$$x_i(y_i) = \begin{cases} 0 & y_i = 2 \\ -\gamma_i & y_i \neq 2 \end{cases} \qquad \text{for all } i \neq 1$$

Recalling the definition of γ above, this implies that the only assignments to y that can maximize the score of Eq. (6) are $(0, 2, 2, ..., 2)$ and $(1, 2, 2, ..., 2)$. In particular, we have:

$$S(0,2,2,...,2;x,w) = \sum_{k \in N_1} w_{1k}^{\text{off-diag}} + x_1(0) + \bar{S}$$
$$S(1,2,2,...,2;x,w) = w_1^{\text{bound}} + \sum_{k \in N_1} w_{1k}^{\text{off-diag}} + x_1(1) + \bar{S}$$

where $\bar{S}$ is the sum of all components that do not involve the first variable.

For the final step we recall that the weights w have to satisfy the constraints: $S(y^m; x^m, w) \geq S(y; x^m, w)$ for all m, y. Thus, we will define two instances (x^m, y^m) for which some y assignment will constrain the weight as needed (in both cases, x^m is defined as above). When $y^{(m)} = (0, 2, 2, \ldots, 2)$, the assignment $y = (1, 2, 2, \ldots, 2)$ yields $w_1^{\text{bound}} \leq D$ and all other assignments do no further constrain the weight. Similarly, for $y^{(m')} = (1, 2, 2, \ldots, 2)$, the assignment $y = (0, 2, 2, \ldots, 2)$ yields $w_1^{\text{bound}} \geq D$. Together, the two assignments constrain the weight parameter to $w_1^{\text{bound}} = D$, as desired.

Realizing the weights $w_{ij}^{\text{off-diag}}$: We define $x(y)$ as:

$$x_i(y_i) = \begin{cases} 0 & y_i = 0 \\ -\gamma_i & y_i \neq 0 \end{cases}, \quad x_j(y_j) = \begin{cases} 0 & y_j = 0 \\ n^2 & y_j = 1 \\ -\gamma_j & y_j \geq 2 \end{cases}$$

$$x_k(y_k) = \begin{cases} 0 & y_k = k \\ -\gamma_k & y_k \neq k \end{cases} \qquad \text{for all } k \neq i, j$$

This implies that except for i and j, all y_k's must take their corresponding assignment so that $y_k = k$. W.l.g., suppose that $i = 1$ and $j = 2$. The only assignments that can maximize the score are $(0, 0, 3, 4, 5, ..., n)$ and $(0, 1, 3, 4, 5, ..., n)$ with values:

$$S(0,0,3,4,5,...,n;x,w) =$$
$$\sum_{k \in N_i} w_{ik}^{\text{off-diag}} + \sum_{k' \in N_j} w_{jk'}^{\text{off-diag}} + x_i(0) + x_j(0) + \bar{S}$$
$$S(0,1,3,4,5,...,n;x,w) =$$
$$w_{ij}^{\text{off-diag}} + \sum_{k \in N_i} w_{ik}^{\text{off-diag}} + \sum_{k' \in N_j} w_{jk'}^{\text{off-diag}} + x_i(0) + x_j(1) + \bar{S}$$

As before, setting $y^{(m)} = (0, 0, 3, 4, 5, ..., n)$ and then $y^{(m')} = (0, 1, 3, 4, 5, ..., n)$ yields the constraint $w_{ij}^{\text{off-diag}} = -n^2$.

Realizing the weights w_{ij}^{diag}: We define $x(y)$ as:

$$x_i(y_i) = \begin{cases} 0 & y_i = i \\ -\gamma_i & y_i \neq i \end{cases}, \quad x_j(y_j) = \begin{cases} n^2 & y_j = 0 \\ -1 & y_j = i \\ -\gamma_j & y_j \notin \{0, i\} \end{cases}$$

$$x_k(y_k) = \begin{cases} 0 & y_k = k \\ -\gamma_k & y_k \neq k \end{cases} \qquad \text{for all } k \neq i, j$$

As before, for all $k \neq i, j$ the assignment is forced to $y_k = k$. The maximizing assignments are now $(1, 0, 3, 4, 5, ..., n)$ and $(1, 1, 3, 4, 5, ..., n)$ (assuming w.l.g., $i = 1, j = 2$) with score values:

$$S(1,0,3,4,5,...,n;x,w) =$$
$$w_{ij}^{\text{off-diag}} + \sum_{k \in N_i} w_{ik}^{\text{off-diag}} + \sum_{l \in N_j} w_{jl}^{\text{off-diag}} + x_i(i) + x_j(0) + \bar{S}$$
$$S(1,1,3,4,5,...,n;x,w) =$$
$$w_{ij}^{\text{diag}} + \sum_{k \in N_i} w_{ik}^{\text{off-diag}} + \sum_{l \in N_j} w_{jl}^{\text{off-diag}} + x_i(i) + x_j(i) + \bar{S}$$

Now, setting $y^{(m)} = (1, 0, 3, 4, 5, ..., n)$, the assignment $y = (1, 1, 3, 4, 5, ..., n)$ implies $w_{ij}^{\text{off-diag}} + n^2 \geq w_{ij}^{\text{diag}} - 1$. Since we already have that $w_{ij}^{\text{off-diag}} = -n^2$, we obtain $w_{ij}^{\text{diag}} \leq 1$. Similarly, adding $y^{(m')} = (1, 1, 3, 4, 5, ..., n)$ implies $w_{ij}^{\text{diag}} \geq 1$, so together we have $w_{ij}^{\text{diag}} = 1$, as required.

Importantly, our trainset realizes the same edge parameters for any possible tree, since edge weights are constrained independently of other edges.

References

H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation complexity of min-max (regret) versions of shortest path, spanning tree, and knapsack. In *Proceedings of the 13th annual European conference on Algorithms*, pages 862–873, Berlin, Heidelberg, 2005. Springer-Verlag.

F. Bach. Structured sparsity-inducing norms through submodular functions. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 118–126. 2010.

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. The MIT Press, 2007.

C. Berge. *The Theory of Graphs*. Dover Books on Mathematics Series. Dover, 1962.

J. K. Bradley and C. Guestrin. Learning tree conditional random fields. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 127–134. Omnipress, 2010.

A. Chechetka and C. Guestrin. Evidence-specific structures for rich tractable crfs. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 352–360. 2010.

C. I. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286, 2010.

T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *Proceedings of the 25th International Conference on Machine learning*, pages 304–311, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

E. Hazan and S. Kale. Beyond convexity: Online submodular minimization. In *Advances in Neural Information Processing Systems 22*, pages 700–708. 2009.

R. Horst and N. Thoai. Dc programming: Overview. *Journal of Optimization Theory and Applications*, 103 (1):1–43, 1999. ISSN 0022-3239. doi: 10.1023/A:1021765131316. URL `http://dx.doi.org/10.1023/A%3A1021765131316`.

J. B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Math. Society*, 7(1):48–50, 1956.

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proceedings of The 30th International Conference on Machine Learning*, pages 53–61, 2013.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th Int. Conf. on Machine Learning*, pages 282–289, 2001.

S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In *Advances in Neural Information Processing Systems (NIPS 2006)*, 2007.

O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *ICML*, pages 783–790, New York, NY, USA, 2010. ACM.

J. G. Oxley. *Matroid Theory*. Oxford University Press, 2006.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

P. Ravikumar, G. Raskutti, M. J. Wainwright, and B. Yu. Model selection in gaussian graphical models: High-dimensional consistency of l1-regularized MLE. In *Advances in Neural Info. Processing Systems 17*. 2008.

P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *Annals of Statistics*, 38(3):1287–1319, 2010.

M. Schmidt, K. Murphy, G. Fung, and R. Rosales. Structure learning in random fields for heart motion abnormality detection. In *CVPR*, pages 1 –8, 2008.

Y. Shimony. Finding the MAPs for belief networks is NP-hard. *Aritifical Intelligence*, 68(2):399–410, 1994.

D. Sontag, O. Meshi, T. Jaakkola, and A. Globerson. More data means less inference: A pseudo-max approach to structured learning. In *Advances in Neural Information Processing Systems 23*, pages 2181–2189. 2010.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using boosted random fields. In *Advances in Neural Information Processing Systems 17*, pages 1401–1408. 2004.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2006.

M. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA, 2008.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Comput.*, 15(4):915–936, Apr. 2003.

Y. Zhang and J. Schneider. Maximum margin output coding. In *ICML*, 2012.

J. Zhu, E. P. Xing, and B. Zhang. Primal sparse max-margin markov networks. In *Proceedings of the ACM SIGKDD international conf. on Knowledge discovery and data mining*, KDD '09, pages 1047–1056, 2009.

Chapter 3

Discussion and Conclusions

In this thesis we have proposed and analyzed efficient methods for inference and learning in structured output prediction problems. Since these tasks are usually hard, our focus has been on approximation algorithms which provide high accuracy solutions as well as efficient computation.

For the prediction (i.e., inference) task we have proposed a new algorithm based on optimizing the dual of the LP relaxation of the prediction task (Section 2.1). Our algorithm has simple updates and shows good empirical performance. In addition, we have analyzed the convergence rate of coordinate descent algorithms that optimize the smoothed dual LP relaxation (Section 2.2). We have shown that a greedy schedule for updating blocks has theoretical and empirical advantages over a stochastic schedule. We also propose a simple procedure to map dual variables to primal ones, and derive an upper bound on the convergence rate of the mapped primal solutions. This is very useful in practice since it provides a sound stopping criterion.

For the learning task we have shown that replacing the primal LP relaxation with its dual during training has some computational advantages (Section 2.3). Specifically, this results in a joint minimization problem over model weights and dual variables. This means that at the beginning of training, when the model weights are far from optimal, it is not necessary to solve the inference problem completely. Instead, a few improving steps on the dual variables are enough to move the weights in the right direction. Moreover, at the end of training, when model weights are close to optimal, they usually do not change by much between the iterations. In that case our formulation allows for warm-starting the dual variables from their previous values, which speeds up training.

After our paper has been published two other closely related works appeared. In the first work Hazan and Urtasun [2010] proposed a formulation very similar to ours, where the main difference is that they used the *smooth* dual LP (see Section 1.2.4) instead of the non-smooth dual. In the second work Komodakis [2011] used the same training objective as we did, but applied a pure subgradient descent optimization scheme. In contrast, in our paper we mixed subgradient descent on the weights with coordinate descent steps on the dual variables.

Instead of learning complex models and approximating intractable prediction problems at test time, an alternative approach is to learn only models which accommodate efficient test-time prediction. In Section 2.4 we followed this path by constraining the learned models to tree-structured graphs. We have shown that finding the optimal tree is intractable, and proposed an efficient approximation scheme. Our proposed CRANK algorithm yields tree-structured models which are almost as accurate as a fully-connected graph, with scalable training, and fast (linear-time) prediction.

3.1 Future directions

I next discuss a number of possible ways to extend our work.

Input-specific trees: When learning tractable models, as we did with trees in Section 2.4, it makes sense to let the model depend on the input x. Hence, instead of learning a single model that fits all inputs we aim at learning a *procedure* that maps inputs to models. In the context of tree-structured models, we would like to learn a mapping from input to tree $f : x \mapsto T$. Notice that this formulation is more broadly applicable than the single model approach as it does not require all problem instances to be of the same size. This new learning problem is not easier than learning a single tree model, but can be approximated efficiently via an alternating minimization scheme [see Weiss and Taskar, 2010, for a related approach]. Chechetka and Guestrin [2010] study this problem in the context of CRFs, and it would be interesting to see how this extends to the max-margin objective.

Richer models for structured prediction: Most previous works on structured prediction use decompositions (Eq. (1.2)) where factors are over small subsets of variables (e.g., singleton and pairwise factors). However, in light of the recent success of LP relaxation in learning and prediction for complex structured outputs [Kulesza and Pereira, 2008, Finley and Joachims, 2008, Martins et al., 2009b, Meshi et al., 2010, Hazan and Urtasun, 2010, Komodakis, 2011], it is reasonable to expect that this technique will yield accurate and efficient predictors for even more complex models. Indeed, several recent works introduce specific high-order factors (or losses) to the structured model and show that prediction and learning can still be handled efficiently in these more complex models [Joachims, 2005, Ranjbar et al., 2010, Tarlow and Zemel, 2012, Ranjbar et al., 2012]. We believe that other high-order models will be found useful for various applications in the future. An important goal of future research is to identify such cases and design efficient algorithms that can handle the resulting inference and learning tasks. For example, it is interesting to study the problem of learning to rank with structured models. Specifically, in this setting we need to output a ranking of elements that have additional complex dependencies. Interestingly, although the area-under-the-ROC-curve (AUC) is a natural measure of ranking quality, its use in the context of structured models has not been thoroughly studied. In this case one can define an AUC-loss that can be seen as a global factor which is added to an already intractable model. Learning and inference in such a complex model pose a serious computational challenge, even when using approximations.

One can think of even more expressive models that can be used to advance state-of-the-art predictors. The motivation for structured output prediction is that predicting multiple outputs together is better than predicting each variable alone. Interdependence between variables is taken into account in order to improve prediction quality. A similar motivation can justify the design of models that solve *multiple tasks* together. Information from one prediction task can then be used when solving another related task. Indeed, an increasing number of works propose methods for solving multiple related tasks simultaneously. For example, in NLP it was shown that solving POS tagging and parsing together improves prediction accuracy on both tasks [Rush et al., 2010, Lee et al., 2011]. In computer vision, Heitz et al. [2009] propose a "cascaded classification" formalism to combine several task-specific classifiers in a way that allows each model to benefit from the expertise of the others. The resulting joint model is used for holistic scene understanding [see Yao et al., 2012, for a related approach]. Designing such *holistic predictors* is an important step towards building large intelligent systems. Applying such rich models to specific domains and training them from data is a great challenge of contemporary research in machine learning.

An interesting application for such methods is video analysis. First, the number of video data sources is constantly growing (e.g., YouTube, or cameras installed on robots and vehicles [Geiger et al., 2012]). Second, the challenge here is designing scalable algorithms that can deal with the large number of variables and the vast amount of data. Indeed, very recently, some works started to address this great challenge [e.g., Hu et al., 2013]. Video analysis problems are natural to handle within the structured prediction framework since video streams are inherently structured. The structure emerges both temporally (e.g., dependence between pixels in consecutive frames) and spatially (e.g., dependence between adjacent pixels in a single frame). We believe that the methodology proposed in the previous sections can be very suitable for this domain.

Learning to tighten: In Section 1.2.4 we mentioned the standard LP relaxation for MAP inference. Unfortunately, the first-order LP relaxation of Eq. (1.8) is often not tight on real-world inference problems. Therefore, several recent works suggest to tighten the relaxation in various ways [Sontag et al., 2008, Werner, 2008, Komodakis and Paragios, 2008, Batra et al., 2011, Sontag et al., 2012]. However, all of these works focus solely on the prediction task and do not address the effect of tightening on learning. Thus, it would be interesting to study the problem of learning with tighter relaxations. As mentioned earlier (Section 1.3.4), learning with LP relaxation can be understood as optimizing over an inner bound on the space of separating weights w. Accordingly, tightening the relaxation at training has the effect of tightening this bound on weights. One of the interesting questions that arise in this context is that of "region pursuit", namely, which additional constraints should be used for tightening the bound. Similar to the input-specific trees problem, here too the problem is that of learning a procedure that should work well at test time. Finally, it seems that the resulting learning problem is not easier than the one which uses the basic relaxation, so there will be a need for efficient approximation algorithms to perform the tightening.

Understanding integrality of LP relaxation: As mentioned in Section 1.3.4, in surprisingly many cases, the LP relaxation at test time happens to be tight (i.e., an integral solution is found). Investigating and quantifying this phenomenon is an important question which can improve our theoretical understanding of the success of LP relaxations for structured prediction problems. Moreover, such inquiry also has the potential of leading us to new and better approximation methods. One promising step in this direction is to find the *integrality-dimension* of structured prediction problems. More specifically, we would like to bound the rate of tight solutions at test time as a function of the rate of tight solutions at train time and the number of training samples. This would imply that if many train instances are exact, then test instances are also very likely to be exact.

Semi-supervised highly structured learning: In this thesis we focused on the supervised learning setting. However, supervised training requires large amounts of labeled data, which is often expensive to obtain. On the other hand, in many domains *unlabeled* data is abundant. For instance, we currently have access to a virtually unlimited number of images and text on the Internet. This situation motivates the development of semi-supervised learning methods. In the semi-supervised setting one tries to use a large number of unlabeled examples on top of the small number of labeled examples in order to improve prediction accuracy. In recent years there has been some effort to handle structured output prediction in the semi-supervised setting [Altun et al., 2006, Brefeld and Scheffer, 2006, Belkin et al., 2005, Wang et al., 2009]. However, in previous works relatively simple models were assumed where prediction is tractable. An interesting question is how to extend this line of research to the regime of complex structured models where prediction is hard and approximations are necessary. This nicely complements the view of holistic prediction mentioned earlier, since to achieve background knowledge needed for such prediction, large amounts of unlabeled data must inevitably be used.

In conclusion, I believe that structured output prediction will continue to be a powerful and leading tool for solving complex tasks in machine learning. As we have seen, problems in this regime pose interesting and challenging questions in both machine learning and optimization. Developing new methods for structured output prediction that can handle large-scale datasets accurately and efficiently is therefore a prominent goal of contemporary research in this field.

Bibliography

A. Agarwal, P. Bartlett, P. Ravikumar, and M. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *Information Theory, IEEE Transactions on*, 58(5):3235–3249, 2012.

Y. Altun, D. McAllester, and M. Belkin. Maximum margin semi-supervised learning for structured variables. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 33–40. MIT Press, Cambridge, MA, 2006.

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data.* The MIT Press, 2007.

D. Batra, S. Nowozin, and P. Kohli. Tighter relaxations for map-mrf inference: A local primal-dual gap based separation algorithm. In *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2011.

M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In R. G. Cowell and Z. Ghahramani, editors, *AISTATS*, pages 17–24. Society for Artificial Intelligence and Statistics, 2005. (Available electronically at http://www.gatsby.ucl.ac.uk/aistats/).

D. Bertsekas, A. Nedić, and A. Ozdaglar. *Convex Analysis and Optimization.* Athena Scientific optimization and computation series. Athena Scientific, 2003. ISBN 9781886529458.

J. Besag. On the Statistical Analysis of Dirty Pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986.

S. Boyd, N. Parikh, and E. Chu. *Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers.* Now Publishers, 2011. ISBN 9781601984609. URL `http://books.google.co.il/books?id=8MjgLpJ0_4YC`.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

J. K. Bradley and C. Guestrin. Learning tree conditional random fields. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 127–134. Omnipress, 2010.

U. Brefeld and T. Scheffer. Semi-supervised learning for structured output variables. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 145–152, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143863. URL http://doi.acm.org/10.1145/1143844.1143863.

V. Chandrasekaran and M. I. Jordan. Computational and statistical tradeoffs via convex relaxation. *CoRR*, abs/1211.1073, 2012.

A. Chechetka and C. Guestrin. Evidence-specific structures for rich tractable crfs. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 352–360. 2010.

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P. L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *The Journal of Machine Learning Research*, 9:1775–1822, 2008.

W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS J. on Computing*, 11(2):138–148, 1999.

A. Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM*, 50(3): 280–305, 2003.

R. Dechter. *Constraint Processing.* The Morgan Kaufmann Series in Artificial Intelligence Series. Morgan Kaufmann Publishers, 2003.

R. Dechter and R. Mateescu. And/or search spaces for graphical models. *Artificial Intelligence*, 171 (2-3):73–106, 2007.

K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286, 2010.

M. M. Deza and M. Laurent. *Geometry of Cuts and Metrics.* Algorithms and Combinatorics. Springer, 1997.

Y. Dinitz. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Doklady Akademii nauk SSSR*, 11:1277–1280, 1970.

J. Edmonds. Maximum matching and a polyhedron with $0, 1$ vertices. *Journal of Research of the National Bureau of Standards*, 69 B:125–130, 1965.

P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32 (9):1627–1645, 2010.

T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *Proceedings of the 25th International Conference on Machine learning*, pages 304–311, 2008.

M. E. Fisher. On the dimer solution of planar ising models. *Journal of Mathematical Physics*, 7(10): 1776–1781, 1966.

B. Flach and D. Schlesinger. Best labeling search for a class of higher order gibbs models. *Pattern Recognition and Image Analysis*, 14:249–254, 2004.

L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

M. Frank and P. Wolfe. *An algorithm for quadratic programming*, volume 3, pages 95–110. 1956.

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. *Computers and Mathematics with Applications*, 2:17–40, 1976.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, June 2012.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. PAMI*, 6(6):721–741, 1984.

A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NIPS 20*. MIT Press, 2008.

R. Glowinski and A. Marrocco. Sur lapproximation, par elements finis dordre un, et la resolution, par penalisation-dualité, dune classe de problems de dirichlet non lineares. *Revue Française d'Automatique, Informatique, et Recherche Opérationelle*, 9:4176, 1975.

D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 51(2):pp. 271–279, 1989.

P. Hammer. Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 13:388–399, 1965.

T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*, 56(12):6294–6316, 2010.

T. Hazan and R. Urtasun. A primal-dual message-passing algorithm for approximated large scale structured prediction. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 838–846. 2010.

B. He and X. Yuan. On the \$o(1/n)\$ convergence rate of the douglas-rachford alternating direction method. *SIAM J. Numer. Anal.*, 50(2):700–709, Apr. 2012. ISSN 0036-1429.

G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classification models: Combining models for holistic scene understanding. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 641–648. 2009.

H. Hu, D. Munoz, J. A. Bagnell, and M. Hebert. Efficient 3-d scene analysis from streaming data. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

H. Ishikawa. Exact optimization for markov random fields with convex priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1333–1336, 2003.

T. Jebara. Perfect graphs and graphical modeling. In L. Bordeaux, Y. Hamadi, P. Kohli, and R. Mateescu, editors, *Tractability: Practical Approaches to Hard Problems*. Cambridge Press, 2013.

T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 377–384. ACM Press, 2005.

T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *Machine Learning*, 76(1), 2009.

J. Johnson. *Convex Relaxation Methods for Graphical Models: Lagrangian and Maximum Entropy Approaches*. PhD thesis, EECS, MIT, 2008.

V. Jojic, S. Gould, and D. Koller. Fast and smooth: Accelerated dual decomposition for MAP inference. In *Proceedings of International Conference on Machine Learning (ICML)*, 2010.

J. H. Kappes and C. Schnörr. Map-inference for highly-connected graphs with dc-programming. In G. Rigoll, editor, *DAGM-Symposium*, volume 5096 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2008.

J. H. Kappes, B. Savchynskyy, and C. Schnörr. A bundle approach to efficient map-inference by lagrangian relaxation. In *CVPR 2012*, 2012.

J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, J. Lellmann, N. Komodakis, and C. Rother. A comparative study of modern inference techniques for discrete energy minimization problems. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 2013.

P. W. Kasteleyn. Dimer statistics and phase transitions. *Journal of Mathematical Physics*, 4(2): 287–293, 1963.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220: 671–680, 1983.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

N. Komodakis. Efficient training for pairwise or higher order crfs via dual decomposition. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1841–1848, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-0394-2. doi: 10.1109/CVPR.2011.5995375. URL `http://dx.doi.org/10.1109/CVPR.2011.5995375`.

N. Komodakis and N. Paragios. Beyond loose lp-relaxations: Optimizing MRFs by repairing cycles. In *ECCV*, pages 806–820, 2008.

N. Komodakis and G. Tziritas. Approximate labeling via graph cuts based on linear programming. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(8):1436–1453, Aug 2007.

N. Komodakis, N. Paragios, and G. Tziritas. Mrf optimization via dual decomposition: Message-passing revisited, 2007.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *EMNLP*, 2010.

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 2001. to appear.

A. Kulesza and F. Pereira. Structured learning with approximate inference. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 785–792. MIT Press, Cambridge, MA, 2008.

A. Kumar and S. Zilberstein. MAP estimation for graphical models by likelihood maximization. In *Proceedings of the Twenty-Fourth Neural Information Processing Systems Conference*, pages 1180–1188, Vancouver, British Columbia, Canada, 2010.

A. Kumar, S. Zilberstein, and M. Toussaint. Message-passing algorithms for map estimation using dc programming. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, pages 656–664, La Palma, Canary Islands, 2012.

M. P. Kumar, V. Kolmogorov, and P. H. Torr. An analysis of convex relaxations for map estimation of discrete mrfs. 10:71–106, 2009.

M. P. Kumar, O. Veksler, and P. H. Torr. Improved moves for truncated convex models. *Journal of Machine Learning Research*, 12:31–67, Feb. 2011.

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proceedings of The 30th International Conference on Machine Learning*, pages 53–61, 2013.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th Int. Conf. on Machine Learning*, pages 282–289, 2001.

J. Lee, J. Naradowsky, and D. A. Smith. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 885–894, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. URL `http://dl.acm.org/citation.cfm?id=2002472.2002584`.

P. Liang, H. Daumé, III, and D. Klein. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 592–599, New York, NY, USA, 2008. ACM.

D. Lowd and P. Domingos. Learning arithmetic circuits. In *UAI*, pages 383–392. AUAI Press, 2008.

A. Martins, N. Smith, and E. P. Xing. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, 2009a.

A. Martins, N. Smith, and E. P. Xing. Polyhedral outer approximations with application to natural language parsing. In *Proceedings of the 26th International Conference on Machine Learning*, 2009b.

A. L. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. An augmented lagrangian approach to constrained map inference. In *ICML*, pages 169–176, 2011.

D. McAllester, T. Hazan, and J. Keshet. Direct loss minimization for structured prediction. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1594–1602. 2010.

O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *ICML*, pages 783–790, New York, NY, USA, 2010. ACM.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of Uncertainty in AI*, 1999.

K. Murty. *Linear programming.* Wiley, 1983. URL `http://books.google.com/books?id=67PuAAAAMAAJ`.

Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Prog.*, 103(1):127–152, May 2005. ISSN 0025-5610. doi: 10.1007/s10107-004-0552-5. URL `http://dx.doi.org/10.1007/s10107-004-0552-5`.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

D. Plummer and L. Lovász. *Matching Theory.* North-Holland Mathematics Studies. Elsevier Science, 1986.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *UAI*, pages 337–346, 2011.

M. Ranjbar, G. Mori, and Y. Wang. Optimizing complex loss functions in structured prediction. In *European Conference on Computer Vision*, 2010.

M. Ranjbar, A. Vahdat, and G. Mori. Complex loss optimization via dual decomposition. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

N. Ratliff, J. A. D. Bagnell, and M. Zinkevich. (Online) subgradient methods for structured prediction. In *AISTATS*, 2007.

P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 737–744, New York, NY, USA, 2006. ACM.

P. Ravikumar, A. Agarwal, and M. J. Wainwright. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *J. Mach. Learn. Res.*, 11:1043–1080, Mar. 2010. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1756006.1756040`.

C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary mrfs via extended roof duality. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

A. M. Rush, D. Sontag, M. Collins, and T. Jaakkola. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2010.

B. Savchynskyy, S. Schmidt, J. Kappes, and C. Schnorr. A study of Nesterov's scheme for lagrangian decomposition and map labeling. *CVPR*, 2011.

S. Schmidt, B. Savchynskyy, J. H. Kappes, and C. Schnörr. Evaluation of a first-order primal-dual algorithm for MRF energy minimization. In *EMMCVPR*. Springer, 2011.

A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Globally Convergent Dual MAP LP Relaxation Solvers using Fenchel-Young Margins. In *Proc. NIPS*, 2012.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM, 2007.

O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML*, 2013.

Y. Shimony. Finding the MAPs for belief networks is NP-hard. *Aritifical Intelligence*, 68(2):399–410, 1994.

N. Z. Shor, K. C. Kiwiel, and A. Ruszcayǹski. *Minimization methods for non-differentiable functions.* Springer-Verlag New York, Inc., New York, NY, USA, 1985.

D. Sontag and T. Jaakkola. Tree block coordinate descent for MAP in graphical models. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AI-STATS)*, volume 8, pages 544–551. JMLR: W&CP, 2009.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 503–510, Arlington, Virginia, 2008. AUAI Press.

D. Sontag, O. Meshi, T. Jaakkola, and A. Globerson. More data means less inference: A pseudo-max approach to structured learning. In *Advances in Neural Information Processing Systems 23*, pages 2181–2189. 2010.

D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*, pages 219–254. MIT Press, 2011.

D. Sontag, D. K. Choe, and Y. Li. Efficiently searching for frustrated cycles in MAP inference. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence (UAI-12)*, pages 795–804, Corvallis, Oregon, 2012. AUAI Press.

R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(6): 1068–1080, 2008.

D. Tarlow and R. Zemel. Structured output learning with high order loss functions. In *AISTATS*, 2012.

D. Tarlow, I. E. Givoni, R. S. Zemel, and B. J. Frey. Graph cuts is a max-product algorithm. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative Markov networks. In *Proc. ICML*. ACM Press, 2004a.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. D. Manning. Max-margin parsing. In *EMNLP*, 2004b.

B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction, dual extragradient and Bregman projections. *JMLR*, pages 1627–1653, 2006.

C. H. Teo, S. Vishwanathan, A. J. Smola, and Q. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2005.

M. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference.* Now Publishers Inc., Hanover, MA, USA, 2008.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49(5):1120–1146, 2003.

M. J. Wainwright. Estimating the ”wrong” graphical model: Benefits in the computation-limited setting. *Journal on Machince Learning Research*, 7:1829–1859, Dec. 2006.

M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.

H. Wang and A. Banerjee. Online alternating direction method. In *ICML*, 2012.

Y. Wang, G. Haffari, S. Wang, and G. Mori. A rate distortion approach for semi-supervised conditional random fields. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2008–2016. 2009.

D. Weiss and B. Taskar. Structured Prediction Cascades. In *AISTATS*, 2010.

Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 416–425, Arlington, Virginia, 2007. AUAI Press.

T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007.

T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In *CVPR*, 2008.

T. Werner. Revisiting the decomposition approach to inference in exponential families and graphical models. Technical Report CTU-CMP-2009-06, Czech Technical University, 2009.

C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, pages 702–709. IEEE, 2012.