Learning to speed up MAP decoding with column generation

Abstract

In this paper, we show how the connections between max-product message passing for max-product and linear programming relaxations allow for a more efficient exact algorithm for the MAP problem. Our proposed algorithm uses *column generation* to pass messages only on a small subset of the possible assignments to each variable, while guaranteeing to find the exact solution. This algorithm is three times faster than Viterbi decoding for part-of-speech tagging on WSJ data and equivalently fast as beam search with a beam of size two while being exact.

The empirical performance of column generation depends on how quickly we can rule out entire sets of assignments to the edges of the chain, which is done by bounding the contribution of the pairwise factors to the score of the solution. This provides an opportunity at the intersection of inference and learning: at training time, we can regularize the model in a way that makes inference faster without changing its structure.

1. Introduction

Many basic tasks in natural language processing—part-of-speech tagging, named entity recognition, noun-phrase chunking, and others—are often approached with linear-chain conditional random fields (Lafferty et al., 2001). Maximum a posteriori (MAP) decoding in these graphical models is known to be $O(nk^2)$, where k is the number of labels for each token and n is the lenght of the sequence, which can be expensive especially when processing large amounts of data. Even though training is known to fail to converge when using approximate inference (Kulesza et al., 2007) and approximate search at test time can decrease accuracy, most actual implementations use beam search or other approximations instead of exact inference for performance reasons.

While there are known costs to these approximate approaches in general, most of the time exact solutions are actually recovered with approximate search, at a much smaller computational cost than standard exact inference. This is true because in most linear-chain models many tagging decisions can be safely made based on local information, and transition scores are only necessary to disambiguate between states that appear equally reasonable in the absence of contextual information. It is desirable to directly exploit this property by performing efficient local decoding and then expanding the transition factors only when necessary, while still returning exact results.

In this paper we show how an approach based on delayed column generation for the LP relaxation of the MAP decoding problem in linear chains leads to exactly this behavior, where most decisions are made locally and yet the inference process is provably exact. Moreover, as the performance of this search algorithm depends precisely on making local decisions, regularizing the model to minimize the magnitude of the transition scores leads to even faster inference, effectively learning to search faster.

2. Delayed column generation in linear programs

Column generation is a method for exactly solving linear programs with a large number of variables. It works by restricting the problem to a subset of its variables[1] (implicitly setting the others to zero) and alternating between solving this *restricted linear program* and selecting which variables should be added to it, based on whether they could potentially improve the objective. If no variables can improve the objective, one has a primal-dual pair which is optimal for the original, unrestricted, linear program.

The keys to column generation, then, are the method by which variables are added to the linear program and the test that no additional variable could potentially improve the objective. This criterion, based on the

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1] Here we refer to variables in the linear program, which are not the same as variables in the model.

notion of *reduced cost*, is the same criterion used for pivoting in the simplex algorithm (Bertsimas & Tsitsiklis, 1997; Lubbecke & Desrosiers, 2004). The difference between the algorithms is that simplex relies on the primal variables being enumerated explicitly, while column generation leaves them implicitly defined and "generates" them only as needed. This does not guarantee that column generation will scale better than simplex, however, since in the worst case all variables will be added to the restricted primal problem.

Consider the general LP:

$$\textbf{max.} \quad c^T x \quad \textbf{s.t.} \quad Ax \leq b, \quad x \geq 0 \tag{1}$$

With corresponding Lagrangian:

$$L(x, \lambda) = c^T x + \lambda^t (Ax - b) = \sum_i \left(c_i - A_i^T \lambda\right) x_i \tag{2}$$

For a given assignment to the dual variables λ, a variable x_i is a candidate for being added to the restricted problem if its *reduced cost* $r_i = c_i - A_i^T \lambda$, the scalar multiplying it in the Lagrangian, is positive. Another way to justify this decision rule is by considering the constraints in the LP dual:

$$\textbf{min.} \quad b^T \lambda \quad \textbf{s.t.} \quad A^T \lambda \geq c \quad \lambda \geq 0 \tag{3}$$

Here, the reduced cost of a primal variable equals the degree to which its dual constraint is violated, and thus column generation in the primal is equivalent to cutting planes in the dual (Lubbecke & Desrosiers, 2004). Note that if there is no variable of positive reduced cost then the current dual variables from the restricted problem are feasible in the unrestricted problem, and thus we have a primal-dual optimal pair, and can terminate column generation.

3. Linear programming for MAP inference in chains

For any Markov random field whose graph is a tree the MAP inference problem can be formulated as a linear programming problem, where one maximizes the sum of the scores assigned by the model's factors to all variables and adjacent pairs of variables, subject to the assignment being contained in the marginal polytope, which for tree graphs is equivalent to marginals over nodes summing to one and marginals over edges being consistent with the node marginals. Performing message-passing for max-product belief propagation on this graph can be shown to be equivalent to computing a set of optimal dual variables for this linear program. We exploit this connection to design an efficient algorithm for MAP inference based on column generation.

We focus on inference in models that are chain-structured, a special case of trees. A chain model over variables $V_1, \ldots, V_n$ can be expressed in terms of local factors $\theta_i(x_i)$, where x_i refers to a setting of variable i, and pairwise transition factors $\tau_i(x_i, x_{i+1})$. Following Wainwright & Jordan (2008), we write the MAP inference problem as the following LP:

$$\begin{array}{ll} \textbf{max.} & \sum_{i,x_i} \mu_i(x_i)\theta_i(x_i) \\ & + \sum_i \sum_{x_i, x_{i+1}} \mu_i(x_i, x_{i+1})\tau_i(x_i, x_{i+1}) \\ \textbf{s.t.} & \sum_{x_i} \mu_i(x_i) = 1 \\ & \sum_{x_i} \mu_i(x_i, x_{i+1}) = \mu_{i+1}(x_{i+1}) \\ & \sum_{x_{i+1}} \mu_i(x_i, x_{i+1}) = \mu_i(x_i) \end{array} \tag{4}$$

We refer to the first family of constraints as "normalization" constraints and the other two as "marginalization" constraints.

We can invoke the first marginalization constraint to combine the local and pairwise scores and replace the coefficient of $\mu_i(x_i, x_{i+1})$ in the objective with $(\tau_i(x_i, x_{i+1}) + \theta_{i+1}(x_{i+1}))$. Next, by relaxing the marginalization constraints (but keeping normalization as a hard constraint) and grouping all the terms according to the primal variables they multiply, we have the Lagrangian $L(\mu, \alpha, \beta)$, equal to

$$\begin{aligned} & \sum_1 \mu_1(x_1)\nu_1(x_1) \\ & + \sum_{i, x_i, x_{i+1}} \mu_i(x_i, x_{i+1}) \left(\nu_i(x_i, x_{i+1}) - \nu_{i+1}(x_{i+1})\right) \end{aligned} \tag{5}$$

This expression is a function of the max-marginals

$$\nu_i(x_i) = \alpha_i(x_i) + \beta_i(x_i) + \theta_i(x_i) \tag{6}$$

$$\begin{aligned} \nu_i(x_i, x_{i+1}) = {} & \alpha_i(x_i) + \theta_i(x_i) + \tau_i(x_i, x_{i+1}) \\ & + \theta_{i+1}(x_{i+1}) + \beta_{i+1}(x_{i+1}). \end{aligned} \tag{7}$$

Here, $\alpha_i(x_i)$ and $\beta_i(x_i)$ are dual variables corresponding to the marginalization constraints when variable i is set to x_i. Wainwright & Jordan (2008) show that the fixed point of the max-product message passing rules provides an assignment to the dual variables α and β that, together with decoding based on the max-marginals form a primal-dual optimal pair for this LP. The update equations for these messages are as follows:

$$\alpha_{i+1}(x_{i+1}) = \max_{x_i} \alpha_i(x_i) + \theta_i(x_i) + \tau(x_i, x_{i+1}) \tag{8}$$

$$\beta_{i-1}(x_{i-1}) = \max_{x_i} \tau(x_{i-1}, x_i) + \theta_i(x_i) + \beta_i(x_i). \tag{9}$$

Note that in the Lagrangian of equation (5) we made an arbitrary decision to single out the first node in

the chain. An equivalent decision can be made by choosing the last node, which will be important in the next section.

4. A column-generation algorithm for fast chain decoding

In this section we derive a column-generation algorithm for efficient MAP inference in chains by exploiting the connection between the max-product messages and the LP view of the inference problem.

In many applications of MAP inference in chains, such as in text sequence tagging, the graphical model variables have large domain sizes. Let D be the number of values that x_i can take on and n be the length of the chain. The LP in Equation (4) would have $O(nD^2)$ variables. Most of the time, however, we can be confident that many of these variables will not be used in the final solution, due to the strength of local factors $\theta_i(x_i)$ relatively to the pairwise factors $\tau_i(x_i, x_{i+1})$. We leverage this property by lazily adding variables to the LP using column generation.

4.1. Deriving the algorithm

We seek to solve the LP for MAP inference in chains with a column generation strategy. Such a strategy requires efficient components for choosing the initial set of variables in the restricted LP, solving the restricted LP, and finding variables of positive reduced cost.

To initialize the LP, we first define for each node in the graphical model a restricted domain consisting of only $x_i = \text{argmax}\,\theta_i(x_i)$. Note that any initialization strategy is equally valid, and one could, for example, also add the high-scoring transitions, or add the k best x_i instead of the single best. Next, we include in the initial restricted LP all the $\mu_i(x_i)$ and $\mu_i(x_i, x_{i+1})$ indicator variables corresponding to these size-one domains.

To solve the restricted LP, we use max-product message passing, but adapt the recursive definition of $\alpha_i(x_i)$ and $\beta_i(x_i)$ in equations (8) and (9) to only search over the restricted domain of a variable's neighbors in the chain. This can be derived by looking at the Lagrangian of the LP with all $\mu_i(x_i)$ variables but only a subset of the $\mu_i(x_i, x_{i+1})$ variables, but we omit this derivation due to space constraints.

For the column generation approach to work we need to compute the reduced costs of each pairwise marginal variable $\mu_i(x_i, x_{i+1})$ in terms of θ, τ, α, and β. By using the Lagrangian from equation (5) we get the following reduced cost for these pairwise marginals

$$\begin{aligned} R'_i(x_i, x_{i+1}) &= \nu_i(x_i, x_{i+1}) - \nu_{i+1}(x_{i+1}). \quad (10) \\ &= \tau(x_i, x_{i+1}) + \theta_i(x_i) \quad (11) \\ &\quad +\alpha_i(x_i) - \alpha_{i+1}(x_{i+1}). \end{aligned}$$

When we solved the restricted LP, we didn't allow $x_i \notin D_i$ to take on nonzero values. However, we still needed to enforce marginalization constraints that involve it. Therefore, $\alpha_i(x_i)$, the dual variable for this constraint, is defined as in equation 8, even for $x_i \notin D_i$.

Equation (10) has a simple interpretation. $\alpha_{i+1}(x_{i+1})$ is defined as a max over the current restricted domain D_i. For $x_i \notin D_i$, the first three terms of R'_i represent the value of $\alpha_{i+1}(x_{i+1})$, had this maximization used x_i. Therefore, $R'_i(x_i, x_{i+1})$ represents the gain in forward message $\alpha_{i+1}(x_{i+1})$ attainable if x_i is added to restricted domain D_i.

Because this reduced cost only involves α variables, it only considers one direction in the chain when judging the desirability of a pair of variables. As described in section 3, our Lagrangian has an inherent asymmetry where V_1 was chosen as the root of the chain. If we had chosen V_n as the root, our reduced cost would only use β information. Both of these Lagrangians come from the same original LP, and have the same optimum. Therefore, we can average them in order to obtain a Lagrangian for which the reduced cost contains global information from both directions. Doing so leads us to the expression for the reduced cost we'll use in the remainder of this paper,

$$\begin{aligned} 2R_i(x_i, x_{i+1}) &= 2\tau(x_i, x_{i+1}) + \theta_i(x_i) + \theta_{i+1}(x_{i+1}) \\ &\quad + (\alpha_i(x_i) - \alpha_{i+1}(x_{i+1})) \\ &\quad + (\beta_{i+1}(x_{i+1}) - \beta_i(x_i)) \quad (12) \end{aligned}$$

At every iteration of our column generation procedure, we find for each position i in the chain the setting of x_i and x_{i+1} that maximizes the reduced cost $R_i(x_i, x_{i+1})$. If that reduced cost is positive, that setting of the variables is added to our domain. If no setting with a positive reduced cost was found we can prove that we have an optimal solution. Algorithm 1 shows the pseudocode for this approach.

4.2. Finding a state with positive reduced cost

While the general column generation algorithm as presented in Algorithm 1 can be quite efficient (and it is in practice faster than max-product message passing but slower than viterbi decoding) it is possible to improve that by an order of magnitude by pruning the search space when looking for settings of variables with positive reduced cost.

Algorithm 1 Column Generation Chain MAP

for $i = 1 \rightarrow n$ **do** $D_i = \{\text{argmax}\, \theta_i(x_i)\}$
end for
while domains haven't converged **do**
  $(\alpha, \beta) \leftarrow GetMessages(D, \theta)$
  for $i = 1 \rightarrow n$ **do**
    $(x_i, x_{i+1}, rc) \leftarrow \text{argmax}\, R_i(x_i, x_{i+1})$
    if $rc > 0$ **then**
      $D_i \leftarrow D_i \cup x_i$
      $D_{i+1} \leftarrow D_{i+1} \cup x_{i+1}$
    end if
  end for
end while

Algorithm 2 Efficient search for a setting with positive reduced cost

$U_\tau(\cdot, x_{i+1}) \leftarrow \max_{x_i} \tau(x_i, x_{i+1})$
$U_\tau(x_i, \cdot) \leftarrow \max_{x_{i+1}} \tau(x_i, x_{i+1})$
$U_i \leftarrow \max_{x_i} N_i(x_i)$
$C'_i \leftarrow \{x_{i+1} | N'_i(x_{i+1}) + U_i + 2U_\tau(\cdot, x_{i+1}) > 0\}$
$U'_i \leftarrow max_{x_{i+i} \in C'_i} N'_i(x_{i+1})$
$C_i \leftarrow \{x_i | N_i(x_i) + U'_i + 2U_\tau(x_i, \cdot) > 0\}$
$(x^*_i, x^*_{i+1}) \leftarrow \text{argmax}_{x_i \in C_i, x_{i+1} \in C'_i} R(x_i, x_{i+1})$

Note that the terms in the reduced cost Equation (12) can be divided into three groups: those that depend only on x_i, $N_i(x_i) = \theta_i(x_i) + \alpha_i(x_i) - \beta_i(x_i)$, those that depend only on x_{i+1} (henceforth $N'_i(x_{i+1})$) and the transition scores $\tau_i(x_i, x_{i+1})$. Henceforth we assume the transition scores are the same across locations in the chain, though it is trivial to generalize to the case where they vary, and it is possible to obtain tighter bounds while doing that.

Given the decomposition above of the reduced cost we can develop an efficient searching strategy based on independent bounds as in Algorithm 2.

This strategy effectively reduces the common-case complexity of the search for settings with positive reduced cost from $O(k^2)$ to $O(k)$. Note that this strategy can be inefficient if the bounds $U_\tau(x_i, \cdot)$ and $U_\tau(\cdot, x_{i+1})$ on the rows and columns of the transition matrix are loose, which suggests that, during learning, we should try to keep the transition scores as small as possible.

This is the source of our connection between inference and learning: by learning to keep the transition scores small we can effectively learn to search faster. Therefore, we can obtain an accuracy-speed tradeoff of our decoding algorithm by applying different regularization strategies to the transition scores when training.

Table 1. Results of the timing experiments for the joint POS and NER task. The numbers displayed are averages of 20 trials for each run.

Algorithm	sentences/s	% exact
Viterbi	13.0	100
Column generation	160.6	100
Beam-1	252.8	66.6
Beam-5	228.0	99.5
Beam-15	171.1	99.7
Beam-20	151.7	99.9

5. Related Work

Column generation is a cutting plane algorithm applied to the dual problem. Cutting planes have been successfully applied to problems such as training structured SVMs and improving approximate MAP inference in loopy graphical models (Tsochantaridis et al., 2006; Sontag & Jaakkola, 2007). While column generation has enabled solutions to key operations research problems, such as those by Gilmore & Gomory (1961) and Desrochers & Soumis (1989), it is not widely known in machine learning.

Most related work in methods for decoding chains has focussed on methods for improving the accuracy of approximate methods, such as beam search. For example, Pal et al. (2006) selected an adaptive beam width at every variable by ensuring that the marginal distribution captured by the beam was sufficiently close to the true marginal distribution of the chain variable.

In NLP some prior work has also strived for exact fast Viterbi decoding of linear chains. Kaji et al. (2010) presents an efficient decoding strategy by dividing the set of values for each node in the chain into two groups, and computing the messages explicitly for values in one group while upper-bounding the messages for values in the other group. Their approach managed to successfully speed up inference for POS tagging, joint POS tagging and chunking, and CCG supertagging, achieving slightly higher relative speedups in POS tagging than our approach.

6. Experiments

We perform two sets of experiments. The first explores the tradeoff between computation time and exactness, comparing our column-generation approach to beam search and exact Viterbi inference. The second explores the relationship between regularizing the transition weights and inference efficiency, allowing one to trade accuracy against time in another dimension, by learning a model in which it is easier to search.

Table 2. Results for the timing experiments for the POS tagging task.

Algorithm	sentences/s	% exact
Viterbi	856	100
Column generation	2484	100
Beam-1	2533	97.3
Beam-2	2417	99.8
Beam-3	2321	99.9
Beam-4	2239	100

6.1. Exploring the exactness versus time tradeoff

In this section we compare the speed of our algorithm on a standard part-of-speech tagging task on WSJ data and on a joint part-of-speech tagging and named-entity recognition task on CoNLL 2003 data.

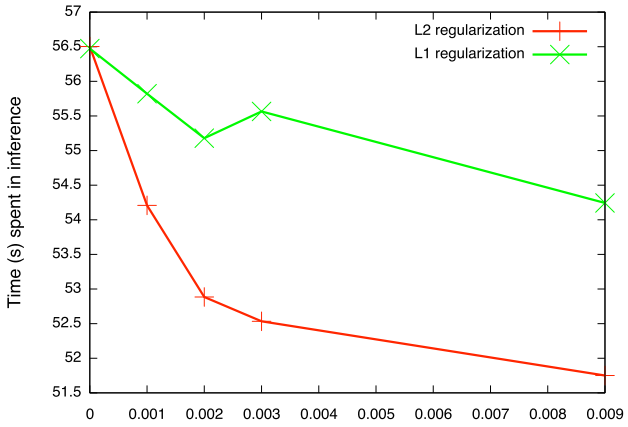
Table 1 shows the results for joint POS and NER. The model is a factorial conditional random field with one linear chain for NER and another for POS tagging and factors connecting both these chains to each other and to observed data. All inference was performed on the cluster graph for this model, in which each variable can take one out of 360 possible values (45 POS tags times 8 NER labels). Column generation is about 8 times faster than viterbi. It performed equivalently to beam search with a beam of size 20, which only returned the exact score for 99% of the sentences, while the column generation approach is exact. Note that even a beam of size 50 is not enough for inference to be exact in this task.

Table 2 shows the results for the POS tagging experiment. The model was trained with 50 iterations of perceptron on the training set of the Penn Treebank. Note that the column generation algorithm processed slightly more sentences per second than beam search with a beam of size two, while it took a beam of size four to obtain exact MAP scores on all sentences.

6.2. Exploring the regularization strength versus time tradeoff

In this section we perform a simple perceptron training experiment to ilustrate the effect of regularizing the transition weights on inference time. We use a standard part-of-speech tagging conditional random field on Wall Street Journal data (Marcus et al., 1993) trained with a variant of the structured perceptron algorithm (Collins, 2002) with ℓ_2 or ℓ_1 regularization on the transition weights. Note that while the perceptron is insensitive to uniform regularization of all the weight vectors, regularizing different subsets of the weights differently affects the loss function, so this could potentially lead to a trade-off between inference time and test-set accuracy, though this was not observed in these experiments; in fact stronger regularization sometimes led to better performance.

Figure 1. Results highlighting the tradeoff between regularization strength and inference time



The design is as follows. The first 5000 sentences in the training chapters of the Penn Treebank used for training. We perform 5 iterations of the perceptron algorithm, recording the time spent doing inference. The first two runs are discarded, as our experiments run on the Hotspot JVM and the first runs are slower due to the JIT compiler. Figure 1 shows the effect of ℓ_2 regularization on inference time.

7. Conclusions and future work

We present an efficient algorithm for MAP decoding in linear chains that uses the LP relaxation of the inference problem to adaptively prune the state-space necessary for exact inference. We also show how adapting the learning procedure can lead to further speed benefits of about 5% for this approach. It is easy to generalize this approach to trees, and in that case instead of averaging two Lagrangians it is more natural to average one Lagragian per leaf, which leads to a reduced cost equation where the messages from each "side" of an edge are weighted by how many leaves can be reached from there. This algorithm can also be applied as a subroutine in tree-reweighted max-product belief propagation or in tree block coordinate descent message-passing for MAP inference, leading to speed improvements without any change in the guarantees.

One limitation of this approach is that while Viterbi decoding and beam search can be easily adapted to produce the k best solutions rather than the single best

solution the algorithm presented here only returns the single best-scoring solution. This is due to the fundamental importance in our algorithm of the LP relaxation and its connection to message-passing. While there are known ways of extending LP relaxations for MAP inference to produce k-best lists (Fromer & Globerson, 2009) these are not known to lead to efficient message-passing algorithms, and general LP solvers are an order of magnitude slower for these problems than message-passing.

Another limitation is that this algorithm currently supports only local and transition factors. While it is trivial to generalize it to higher-order chains (with factors over trigrams, four-grams, etc) we only obtain a reduction in complexity from $O(k^d)$ to $O(k^{d-1})$, which, while significant, is still not practical for most purposes. However, perhaps by exploiting solutions to the bigram chain decoding model as initialization to higher-order models it might be possible to break this barrier, and effectively learn analogues to structured prediction cascades Weiss & Taskar (2010) where the final model is a single model on which exact decoding is performed, but the learning process is set up in a way that this decoding is fast without approximate pruning.

8. Acknowledgement

This work was supported in part by the Center for Intelligent Information Retrieval, in part by Army prime contract number W911NF-07-1-0216 and University of Pennsylvania subaward number 103-548106, and in part by UPenn NSF medium IIS-0803847. The University of Massachusetts also gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Bertsimas, Dimitris and Tsitsiklis, John. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. ISBN 1886529191.

Collins, M. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 1–8. Association for Computational Linguistics, 2002.

Desrochers, M. and Soumis, F. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13, 1989.

Fromer, M. and Globerson, A. An LP view of the m-best map problem. *Advances in Neural Information Processing Systems*, 22:567–575, 2009.

Gilmore, P.C. and Gomory, R.E. A linear programming approach to the cutting-stock problem. *Operations research*, pp. 849–859, 1961.

Kaji, N., Fujiwara, Y., Yoshinaga, N., and Kitsuregawa, M. Efficient staggered decoding for sequence labeling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 485–494. Association for Computational Linguistics, 2010.

Kulesza, A., Pereira, F., et al. Structured learning with approximate inference. *Advances in neural information processing systems*, 20:785–792, 2007.

Lafferty, J., McCallum, A., and Pereira, F.C.N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

Lubbecke, Marco and Desrosiers, Jacques. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2004.

Marcus, M.P., Marcinkiewicz, M.A., and Santorini, B. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

Pal, C., Sutton, C., and McCallum, A. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pp. V–V. IEEE, 2006.

Sontag, David and Jaakkola, Tommi. New outer bounds on the marginal polytope. In *In Advances in Neural Information Processing Systems*, 2007.

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

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

Weiss, D. and Taskar, B. Structured prediction cascades. In *Proc. AISTATS*, volume 1284, 2010.