DivMCuts: Faster Training of Structural SVMs with Diverse M-Best Cutting-Planes

Abner Guzman-Rivera
University of Illinois

Pushmeet Kohli
Microsoft Research Cambridge

Dhruv Batra
Virginia Tech

Abstract

Training of Structural SVMs involves solving a large Quadratic Program (QP). One popular method for solving this QP is a cutting-plane approach, where the most violated constraint is iteratively added to a working-set of constraints. Unfortunately, training models with a large number of parameters remains a time consuming process. This paper shows that significant computational savings can be achieved by adding multiple *diverse* and highly violated constraints at every iteration of the cutting-plane algorithm. We show that generation of such diverse cutting-planes involves extracting diverse M-Best solutions from the loss-augmented score of the training instances. To find these diverse M-Best solutions, we employ a recently proposed algorithm [4]. Our experiments on image segmentation and protein side-chain prediction show that the proposed approach can lead to significant computational savings, *e.g.*, $\sim$28% reduction in training time.

1 Introduction

A number of problems in Computer Vision, Natural Language Processing and Computational Biology involve making predictions over complex but structured interdependent outputs – *e.g.*, the space of all possible segmentations of an image or all possible English translations of a Chinese sentence. Formulations like Max-Margin Markov Networks (M^{3}N) [23] and Structural Support Vector Machines (SSVMs) [24] have provided principled techniques for learning such structured-output models.

In all these settings, the learning algorithm has access to n training (input-output) pairs: $\{(\mathbf{x}_i, \mathbf{y}_i) \mid \mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}\}$ and the goal is to learn a mapping $f : \mathcal{X} \to \mathcal{Y}$ from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$, such that it minimizes a (regularized) task-dependent loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, where $\ell(\mathbf{y}_i, \bar{\mathbf{y}}_i)$ denotes the cost of predicting output $\bar{\mathbf{y}}_i$ when the correct prediction is $\mathbf{y}_i$.

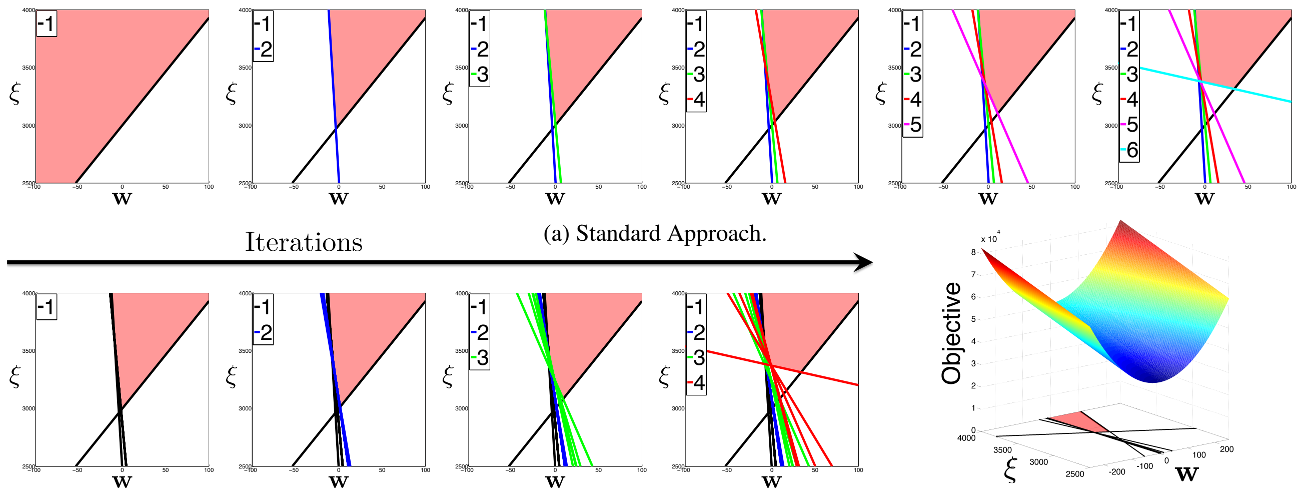
Cutting-Plane Training. This learning problem is generally formulated as a constrained Quadratic Program (QP) [10, 24] with exponentially many constraints. For instance, 1-slack SSVMs [10] involve $|\mathcal{Y}|^n$ constraints, one for each possible n-tuple of labels $(\bar{\mathbf{y}}_1, \ldots, \bar{\mathbf{y}}_n) \in \mathcal{Y}^n$. If the most violated constraint can be identified efficiently, a cutting-plane (CP) approach [11] may be used to solve this QP. A CP algorithm maintains a small working-set of constraints and alternates between: 1) solving for the optimum solution under the current working-set, and 2) adding the most violated constraint to the working-set by calling the max-violation-oracle. It can be shown [10, 24] that such a procedure converges in $O(\frac{1}{\epsilon})$ steps, where ϵ is the desired precision. Finding the most violated constraint involves maximizing the *loss-augmented score* [10] for each training instance.

Unfortunately, models for many real world problems have a large number of parameters and require many iterations of the above procedure. At every iteration, inference must be performed on the entire dataset and a large QP must be solved. Thus, training such models becomes a time consuming process.

Contribution. This paper shows that significant computational savings can be achieved in training SSVMs by generating and adding a diverse set of highly violated constraints (cutting-planes) at every training iteration. Fig. 1 illustrates the idea. One key observation of our work is that for multiple constraints to be useful and speed up convergence, they should satisfy the following desiderata:

1. **Marginal Relevance.** Each constraint should be informative w.r.t. the current approximation (*i.e.*, be highly violated) and also have *marginal* relevance w.r.t. the constraints added at the current iteration (*i.e.*, we need a *diverse* set of constraints).
2. **Efficiently Computable.** Finding a set of violated constraints should be fast enough so as to not offset

Appearing in Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS) 2013, Scottsdale, AZ, USA. Volume 31 of JMLR: W&CP 31. Copyright 2013 by the authors.



(a) Standard Approach.

(b) Proposed Method.

(c)

Figure 1: Illustration of the proposed approach with the feasibility region approximation at the current iteration in pink. (a) When adding a single cutting-plane at every iteration, 6 iterations are necessary to approximate the feasibility region to the desired precision. (b) In contrast, by adding 5 cutting-planes at every iteration only 4 iterations are necessary. (c) Depiction of the learning QP and approximated feasibility region.

the savings resulting from the reduction in the number of training iterations.

We show that generating sets of constraints satisfying the above desiderata involves extracting diverse M-Best solutions from the loss-augmented score – for this, we leverage algorithm DivMBest of Batra *et al.* [4] for producing diverse maximum a posteriori (MAP) solutions in structured-output models. We name our method DivMCuts and evaluate its performance on two applications of SSVMs: image segmentation, and protein side-chain prediction. Our results show that DivMCuts can lead to significant computational savings, *e.g.*, $\sim$28% reduction in training time. Finally, our results suggest that even greater savings are possible for more expressive models involving a larger number of parameters.

The outline for the rest of this paper follows: Sec. 2 provides notation and revisits CP training of SSVMs; Sec. 3 presents the proposed generalization identifying key challenges; Sec. 4 presents experiments; and Sec. 5 concludes with a discussion.

2 Preliminaries: Training SSVMs

This section establishes the notation used in the paper and revisits CP training of Structural SVMs.

Notation. Let $[n]$ be the shorthand for the set $\{1, 2, \dots, n\}$. We use $\mathbf{y}$ to denote a structured-output, and $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_{|\mathbf{Y}|})$ for a tuple of structured-outputs.

Given a training dataset of input-output pairs (examples) $S = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n) \,|\, \mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}\}$, we are interested in learning a mapping $f : \mathcal{X} \to \mathcal{Y}$ from an input space $\mathcal{X}$ to a structured-output space $\mathcal{Y}$, where $|\mathcal{Y}|$ is finite but typically exponentially large (*e.g.*, the set of all segmentations of an image, or all English translations of a Chinese sentence).

Structural Support Vector Machines (SSVMs). In an SSVM setting, the mapping is defined as $f(\mathbf{x}) = \text{argmax}_{\mathbf{y}\in\mathcal{Y}}\, \mathbf{w}^T \boldsymbol{\Psi}(\mathbf{x}, \mathbf{y})$, where $\boldsymbol{\Psi}(\mathbf{x}, \mathbf{y})$ is a joint feature map: $\boldsymbol{\Psi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$. The quality of the prediction $\hat{\mathbf{y}}_i = f(\mathbf{x}_i)$ is measured by a task-specific loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, where $\ell(\mathbf{y}_i, \hat{\mathbf{y}}_i)$ denotes the cost of predicting $\hat{\mathbf{y}}_i$ when the correct label is $\mathbf{y}_i$. Since the task-loss ℓ is non-continuous and non-convex in $\mathbf{w}$, typically a convex surrogate loss that upper bounds ℓ is optimized instead (*e.g.*, the hinge upper-bound [24]).

Optimization Problem 1 (OP1). The regularized hinge-loss SSVM learning problem can be formulated as a QP with exponentially many constraints. In this paper, we work with the margin-rescaling variant of the 1-slack formulation of Joachims *et al.* [10]:

$$\min_{\mathbf{w},\xi\geq 0} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \tag{1a}$$

$$s.t. \quad \frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n} \Big[\boldsymbol{\Psi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\Psi}(\mathbf{x}_i, \bar{\mathbf{y}}_i)\Big] \geq \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \xi \qquad \forall \bar{\mathbf{Y}} \in \mathcal{Y}^n \tag{1b}$$

Note that 1-slack SSVMs involve $|\mathcal{Y}|^n$ constraints, one for each possible n-tuple of labels $\bar{\mathbf{Y}} = (\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_n) \in \mathcal{Y}^n$, but there is a single slack variable ξ, shared across all constraints. The number of constraints is thus *exponentially* larger than in n-slack SSVMs (which involve $n|\mathcal{Y}|$ constraints). However, Joachims *et al.* [10] showed that: 1) the two formulations are equivalent, and, more importantly, 2) 1-slack leads to faster convergence, in theory and practice.

Algorithm 1 Cutting-Plane Training of Structural SVMs (margin-rescaling) via the 1-Slack Formulation OP1.

```
1: Input: $S = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}, C, \epsilon$
2: $\mathcal{W} \leftarrow \emptyset$
3: repeat
4:     $(\mathbf{w}, \xi) \leftarrow \text{argmin}_{\mathbf{w}, \xi \geq 0} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$
       $s.t. \quad \frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n} \left[ \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{\Psi}(\mathbf{x}_i, \bar{\mathbf{y}}_i) \right]$
           $\geq \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \xi_i \quad \forall \bar{\mathbf{Y}} \in \mathcal{W}$
5:     for $i = 1, \dots, n$ do
6:         $\hat{\mathbf{y}}_i \leftarrow \text{argmax}_{\mathbf{y}} \left\{ \ell(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) \right\}$
7:     end for
8:     $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_n)\}$
9: until $\frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{y}_i, \hat{\mathbf{y}}_i)$
       $-\frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n} \left[ \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{\Psi}(\mathbf{x}_i, \hat{\mathbf{y}}_i) \right] \leq \xi + \epsilon$
10: return $(\mathbf{w}, \xi)$
```

Cutting-Plane Training of SSVMs. Algorithm 1 provides a CP approach to solving OP1. At every iteration, the algorithm computes the solution over the current working-set $\mathcal{W}$ (Line 4) and then finds the most violated constraint (Lines 5-7) to add to $\mathcal{W}$ (Line 8). The algorithm stops when the most violated constraint is violated less than a desired precision ϵ (Line 9). Unlike the n-slack setting, Algorithm 1 adds a single constraint at every iteration – a linear combination of features coming from all examples.

Joachims *et al.* [10] showed that the number of iterations to convergence for Algorithm 1 does not depend on the number of training instances and grows as $O(\frac{1}{\epsilon})$. Specifically:

Theorem 1. *Iteration Complexity of Algorithm 1. For any* $0 < C$, $0 < \epsilon < 4R^2C$ *and any training sample* $S = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$, *Algorithm 1 terminates after at most*

$$\left\lceil \log_2\left(\frac{\ell}{4R^2C}\right) \right\rceil + 2\left\lceil \frac{8R^2C}{\epsilon} \right\rceil$$

iterations, where $R^2 = \max_{i,\bar{\mathbf{y}}} \|\mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{\Psi}(\mathbf{x}_i, \bar{\mathbf{y}})\|^2$, $\ell = \max_{i,\bar{\mathbf{y}}} \ell(\mathbf{y}_i, \bar{\mathbf{y}})$ *and* $\lceil . \rceil$ *is the integer ceiling function.*

Proof. See proof of Theorem 5 in [10]. □

3 Proposed Approach

Algorithm 1 incrementally builds an approximation to the constraint-set by adding a single linear inequality in each iteration. An intuitive approach to speed up this process is to instead add multiple constraints in each iteration. We will show that significant computational savings are possible if the additional constraints are highly violated and diverse. Finding the most violated constraint is equivalent to performing MAP inference on a loss-augmented score for each training instance. Similarly, finding multiple violated constraints involves generating multiple diverse solutions to the loss-augmented score of the training instances.

Techniques for producing multiple solutions in probabilistic models can be broadly characterized into two groups: M-Best MAP algorithms [7, 16, 17, 26] that find the top M most probable solutions and sampling-based algorithms [2, 18, 25]. Both of these groups fall short for our task. M-Best MAP algorithms do not place any emphasis on diversity and tend to produce solutions that are minor perturbations of each other. Thus, the resulting cutting-planes are unlikely to be of much value in tightening the constraint-set approximation and speeding up convergence. Sampling-based approaches typically exhibit long wait-times to transition from one mode to another, which is required for obtaining diversity.

3.1 Generating Diverse M-Best Solutions on Loss-Augmented Score

To explicitly enforce diversity, we leverage algorithm DivMBest of Batra *et al.* [4], which computes a set of diverse M-Best solutions in discrete probabilistic models. We briefly describe their approach here.

The approach is applicable to general structured-output models but for the sake of illustration let us consider a discrete Markov Random Field (MRF). Specifically, let $\mathbf{y} = \{y_1, \dots, y_p\} \in \mathcal{Y}$ be a set of discrete random variables, each taking value in a finite label set, *i.e.*, $y_u \in \mathcal{Y}_u$. Let $G = (\mathcal{V}, \mathcal{E})$ be a graph defined over the output variables, *i.e.*, $\mathcal{V} = [p]$, $\mathcal{E} \subseteq \binom{\mathcal{V}}{2}$, and let y_{uv} be shorthand for the tuple (y_u, y_v). It is known that for decomposable loss functions, the loss-augmented score for any configuration $\mathbf{y}$ can be expressed as a sum of terms that decompose along the graph-structure. Thus, loss-augmented inference corresponds to a MAP inference problem:

$$\max_{\mathbf{y} \in \mathcal{Y}} S(\mathbf{y}) = \max_{\mathbf{y} \in \mathcal{Y}} \sum_{u \in \mathcal{V}} \theta_u(y_u) + \sum_{(u,v) \in \mathcal{E}} \theta_{uv}(y_{uv}). \quad (2)$$

We assume availability of a function $\Delta(\tilde{\mathbf{y}}, \tilde{\mathbf{y}}')$ quantifying dissimilarity between solutions $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{y}}'$. Let $\tilde{\mathbf{y}}^{(m)}$ denote the mth-best solution. Thus, $\tilde{\mathbf{y}}^{(1)}$ is the MAP, $\tilde{\mathbf{y}}^{(2)}$ is the second DivMBest solution and so on. [4] proposed the following formulation for finding the mth solution:

$$\tilde{\mathbf{y}}^{(m)} = \underset{\mathbf{y} \in \mathcal{Y}}{\text{argmax}} \sum_{u \in \mathcal{V}} \theta_u(y_u) + \sum_{(u,v) \in \mathcal{E}} \theta_{uv}(y_{uv}) \quad (3a)$$

$$s.t. \quad \Delta(\mathbf{y}, \tilde{\mathbf{y}}^{(m')}) \geq k_{m'} \quad \forall m' \in [m-1] \quad (3b)$$

In order to solve this problem, [4] use the Lagrangian relaxation of (3), formed by *dualizing* the dissimilarity constraints $\Delta(\mathbf{y}, \tilde{\mathbf{y}}^{(m')}) \geq k_{m'}$:

$$f(\boldsymbol{\lambda}) = \max_{\mathbf{y} \in \mathcal{Y}} S_\Delta(\mathbf{y}) = \max_{\mathbf{y} \in \mathcal{Y}} \sum_{A \in \mathcal{V} \cup \mathcal{E}} \theta_A(y_A) + \sum_{m'=1}^{m-1} \lambda_{m'} \left(\Delta(\mathbf{y}, \tilde{\mathbf{y}}^{(m')}) - k_{m'} \right) \quad (4)$$

Here $\boldsymbol{\lambda} = \{\lambda_{m'} \mid m' \in [m-1]\}$ is the set of Lagrange multipliers, which determine the weight of the penalty imposed for violating the constraints. Intuitively, we see that the Lagrangian relaxation maximizes a Δ-augmented score, *i.e.*, a linear combination of the MRF score and the dissimilarity w.r.t. the previous solutions, with the weighting given by the Lagrange multipliers. For some classes of Δ-functions, we can solve the Δ-augmented score maximization problem using the *same algorithms* used for finding the MAP. An illustrative example follows.

Hamming Dissimilarity. $\Delta(\mathbf{y}, \tilde{\mathbf{y}}') = \sum_{u \in \mathcal{V}} [\![y_u \neq \tilde{y}'_u]\!]$. This function counts the number of nodes labeled differently between two solutions. For this dissimilarity function, the Δ-augmented scoring function can be written as:

$$S_\Delta(\mathbf{y}) = \sum_{u \in \mathcal{V}} \underbrace{\left(\boldsymbol{\theta}_u(y_u) + \sum_{m'=1}^{m-1} \lambda_{m'} [\![y_u \neq \tilde{y}_u^{(m')}]\!] \right)}_{\text{Perturbed Unary Score}} + \sum_{(u,v) \in \mathcal{E}} \boldsymbol{\theta}_{uv}(y_{uv}). \tag{5}$$

Thus (5) can be maximized by feeding a perturbed unary term to the MAP inference algorithm.

Practical Remarks. The computation of additional solutions can and should be warm-started by using dynamic inference techniques such as [12, 22]. The benefit of such warm-start typically decreases as magnitude of the perturbations $\lambda_{m'}$ increases. For certain models/tasks the perturbations could be limited to certain regions of the model so that dynamic inference is particularly effective.

3.2 Generating Diverse M-Best Cutting-Planes

Our proposed approach, DivMCuts, is summarized in Algorithm 2. It is parametrized by M, the number of constraints to add to the working-set at every iteration – note that Algorithm 1 is a special case of Algorithm 2 with $M{=}1$. Algorithm 2 finds M diverse loss-augmented solutions for each example (Line 6) and uses these solutions to generate M diverse cutting-planes to be added to the working-set (Lines 9-10). In order to fully specify the algorithm, we need to describe two procedures: 1) $\text{Update}\boldsymbol{\lambda}$ (Line 8) which controls the amount of diversity in the loss-augmented solutions; and 2) Combine (Line 9) which produces a set of M cutting-planes given the M loss-augmented solutions from all n examples.

Diversity Requirements ($\text{Update}\boldsymbol{\lambda}$)**.** The amount of diversity in the loss-augmented solutions is controlled by parameters k_m in (3b) and the corresponding Lagrangian multipliers λ_m in (4). However, the amount of diversity appropriate for faster convergence will typically be problem dependent and not known a priori. This is further complicated by the fact that in practice the Lagrangian relaxation may not be tight [4].

Algorithm 2 DivMCuts: Generalization of Algorithm 1 adding M constraints at every iteration.

```
1: Input: $S = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_y, \mathbf{y}_n)\}, C, \epsilon, M, \mathbf{K}, \boldsymbol{\lambda}_0$
2: $\mathcal{W} \leftarrow \emptyset$; $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}_0$
3: repeat
4:     $(\mathbf{w}, \xi) \leftarrow \text{argmin}_{\mathbf{w}, \xi \geq 0} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$
       $s.t. \quad \frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n} \left[ \boldsymbol{\Psi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\Psi}(\mathbf{x}_i, \bar{\mathbf{y}}_i) \right]$
       $\geq \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \xi_i \quad \forall \bar{\mathbf{Y}} \in \mathcal{W}$
5:     for $i = 1, \ldots, n$ do
6:         $\tilde{\mathbf{Y}}_i = (\tilde{\mathbf{y}}_i^{(1)}, \ldots, \tilde{\mathbf{y}}_i^{(M)}) \leftarrow$
             $\text{DivMBest}(\ell(\mathbf{y}_i, \cdot) + \mathbf{w}^T \Psi(\mathbf{x}_i, \cdot), M, \boldsymbol{\lambda})$
7:     end for
8:     $\boldsymbol{\lambda} \leftarrow \text{Update}\boldsymbol{\lambda}(M, \mathbf{K}, \boldsymbol{\lambda}, \tilde{\mathbf{Y}}_1, \ldots, \tilde{\mathbf{Y}}_n)$
9:     $(\hat{\mathbf{Y}}^{(1)}, \ldots, \hat{\mathbf{Y}}^{(M)}) \leftarrow \text{Combine}(M, \tilde{\mathbf{Y}}_1, \ldots, \tilde{\mathbf{Y}}_n)$
10:    $\mathcal{W} \leftarrow \mathcal{W} \cup \left\{ \hat{\mathbf{Y}}^{(1)}, \ldots, \hat{\mathbf{Y}}^{(M)} \right\}$
11: until $\frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{y}_i, \hat{\mathbf{y}}_i^{(1)})$
       $- \frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n} \left[ \boldsymbol{\Psi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\Psi}(\mathbf{x}_i, \hat{\mathbf{y}}_i^{(1)}) \right] \leq \xi + \epsilon$
12: return $(\mathbf{w}, \xi)$
```

To address both issues, Algorithm 2 uses a *feedback loop* to control the amount of diversity in the solutions. Let $\bar{K}_j$ be the observed *dataset-wide diversity*,

$$\bar{K}_j = \frac{\sum_{i=1}^{n} \Delta\left(\tilde{\mathbf{y}}_i^{(j)}, \tilde{\mathbf{y}}_i^{(j+1)} \right)}{\sum_{i=1}^{n} \max_{\mathbf{y}} \Delta\left(\tilde{\mathbf{y}}_i^{(j)}, \mathbf{y} \right)} \tag{6}$$

where Δ is the dissimilarity function used by DivMBest. For instance, in the case of Hamming distance, $\bar{K}_j$ corresponds to the fraction of nodes labeled differently in all $(j{+}1)^{th}$ solutions w.r.t. all j^{th} solutions. Let also $\mathbf{K} = (K_j \mid j \in [M{-}1])$ be a vector of diversity *setpoints*. These parameters specify the desired amount of dataset-wide diversity – this is preferred to specifying per-example diversity as it leads to a better compromise between *perturbation minimization* and *amount of 1-slack diversity*.

Then, at each iteration $\text{Update}\boldsymbol{\lambda}$ compares the observed dataset-wide diversity $\bar{K}_j$ with setpoint K_j, and updates λ_j to increase or decrease diversity at the next iteration. We obtained good results with the following update rule:

$$\lambda_j \leftarrow \lambda_j \left(1 + \frac{1}{2} \frac{K_j - \bar{K}_j}{\max(K_j, \bar{K}_j)} \right). \tag{7}$$

In Sec. 4 we will see that dataset-wide diversity of solutions has a direct impact on the convergence rate of the algorithm and will describe how to set parameter $\mathbf{K}$.

Combining Solutions into Constraints (Combine)**.** Given $(\tilde{\mathbf{Y}}_1, \ldots, \tilde{\mathbf{Y}}_n)$, the set of predictions computed in

Line 6 of Algorithm 2, we must construct M 1-slack constraints – each of them a linear combination of features corresponding to solutions from all n training examples (thus, there are M^n possibilities). Here, it is important to consider the diversity of the resulting cutting-planes w.r.t. each other – features appearing together in a constraint must be such that their "diversities" do not cancel out.

DivMCuts ensures that $\hat{\mathbf{Y}}^{(1)}$ corresponds to the standard most violated constraint, *i.e.*, the combination of MAP solutions. This is sufficient to preserve the correctness and convergence properties of the original algorithm. For the the remaining $M-1$ additional cutting-planes, we explore the following choices:

1. $\mathrm{DivMBest-Ordering}$: $\hat{\mathbf{Y}}^{(j)} \leftarrow (\tilde{\mathbf{y}}_1^{(j)}, \dots, \tilde{\mathbf{y}}_n^{(j)})$ for $j \in [M]$. That is, we combine all mth solutions together to obtain the mth constraint.
2. $\mathrm{DOP1-Heuristic}$: Informed by insight offered in the proof of Theorem 1, this strategy involves an optimization procedure that seeks to maximize the attainable increase (given the new constraints) in the objective of the Dual of OP1. The optimization procedure is a binary Integer Quadratic Program (IQP) on "flag" variables which select one solution from each example for each constraint. This IQP is, however, too slow for our purposes so we resort to, i) an approach based on relaxing the IQP, and ii) an approach based on a simplification of the IQP (by dropping quadratic terms) which is efficiently solvable via binary Integer Linear Programming (ILP). For lack of space, we relegate details to the supplementary materials.

We compare the effectiveness of the above strategies in the experiments section.

4 Experiments

Setup. We tested DivMCuts (Algorithm 2) on two problems: 1) foreground-background segmentation in image collections and 2) protein side-chain prediction.

For both problems we tuned parameter C on validation data. For the $\mathbf{K}$ and $\boldsymbol{\lambda}_0$ vectors we use the same value (scalars K and λ_0) for each of the $M-1$ elements. We performed grid-search on K and found the algorithm to be fairly robust to the choice of λ_0.

Our experiments show that the number of cutting-plane iterations can be reduced substantially, *i.e.*, up to $\sim 62\%$ in the case of foreground-background segmentation. However, a reduction in the number of iterations will not necessarily translate into a comparable reduction in training time since the time taken for computing additional constraints increases with M and K. It is crucial to employ warm-starting (*e.g.*, dynamic) techniques for inference and to compute feature vectors incrementally (*i.e.*, $\boldsymbol{\Psi}(\mathbf{x}_i, \mathbf{y}_i^{(2)}) = \boldsymbol{\Psi}(\mathbf{x}_i, \mathbf{y}_i^{(1)}) + \boldsymbol{\delta\Psi}(\mathbf{x}_i, \mathbf{y}_i^{(1)}, \mathbf{y}_i^{(2)})$). The greatest (time) speedup, $\sim 28\%$, was obtained under a more modest reduction in the number of iterations, $\sim 34\%$.

Practical Consideration. All QP solvers we used were slowed down by the additional constraints. As observed by [10], constraints that become inactive as optimization proceeds may be removed without affecting the theoretical guarantees. Thus, we discard constraints that have not been active in the last 50 QP solutions. For some problem instances, this strategy made a significant difference. We report results obtained with solver QPC.[1]

Baselines. We compare against three baselines obtained by replacing the oracle call (line 6) in Algorithm 2 as follows:

1. MAP inference. Since we obtain single solutions this becomes the special case of Algorithm 1.
2. $\mathrm{MBest\ MAP}$ inference. In the case of foreground-background segmentation we implemented BMMF [26] using Dynamic Graph-Cuts [12].
3. Rand: Nodes for relabeling are chosen at random so that the resulting diversity is as specified by parameter $\mathbf{K}$. To relabel a node the procedure computes a multinomial distribution from the unary potential (excluding the current label) and samples a new label from this distribution.

Caching Constraints. While techniques like caching (and warm-starting) are useful for speeding up CP training, they are orthogonal to the contributions of this paper. Both Algorithms 1 and 2 may benefit from such techniques. We performed experiments with caching and confirmed that the results in this paper apply directly to all iterations not constructing constraints from the cache. The effect of caching has large variance across applications: major savings on segmentation but negligible on side-chain prediction (also, *e.g.*, in [10], Fig. 3 and 6, somewhat different trends on the effect of caching are reported on three applications).

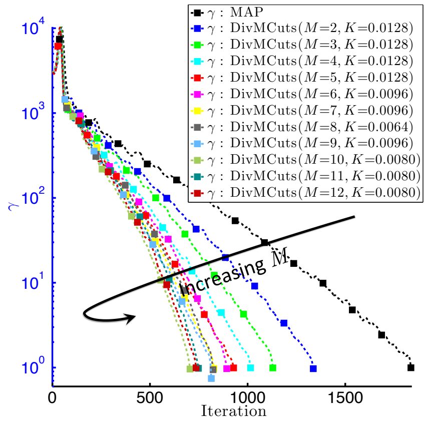
Finally, DivMCuts can be combined with caching to provide additional benefits: 1) If one were to cache the state of the separation-oracle, one could apply our method on iterations constructing constraints from the cache. 2) Caching the additional constraints produced with our method should enable constructing constraints from the cache more often. We leave such extensions for future work.

4.1 Foreground-Background Segmentation

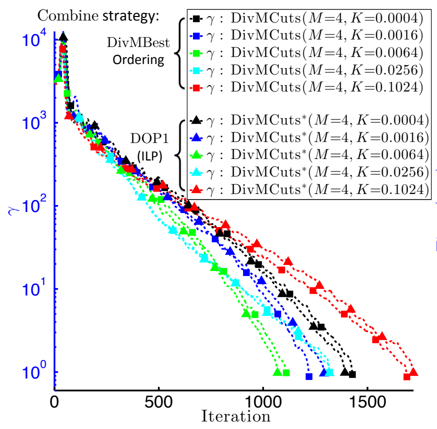
Dataset. We used the co-segmentation dataset, iCoseg, of Batra *et al.* [3]. iCoseg consists of 37 groups of related images mimicking typical consumer photograph collections. Each group may be thought of as an "event" (*e.g.*, images from a baseball game, a safari, *etc.*). The dataset provides pixel-level ground-truth foreground-background (f-b) segmentation for each image.

Model and Features. The segmentation task is modeled as

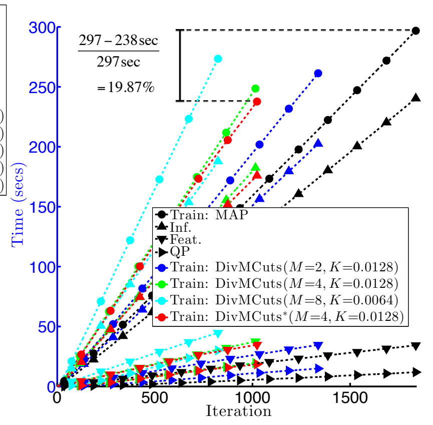
[1] http://sigpromu.org/quadprog/



(a) γ vs iters. @ multiple M.

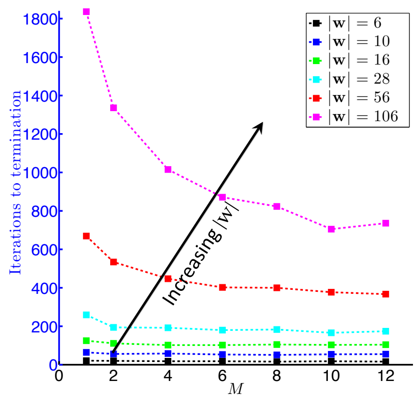


(b) γ vs iters. @ multiple K.

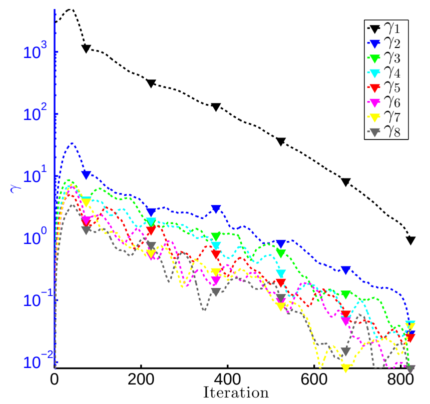


(c) Time vs iters. @ multiple M.

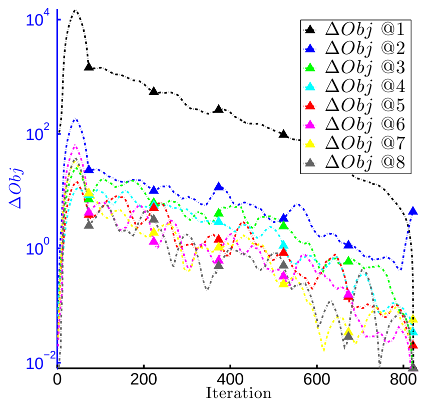
Figure 2: Violation of most violated constraint, γ, and execution times vs iterations to convergence for Alg. 2 on f-b segmentation.



(a) Effect of M @ multiple $|\mathbf{w}|$.



(b) Seq. γ vs iters.



(c) Seq. Δ-obj. vs iters.

Figure 3: (a) Effect of M on number of iterations to termination for different problem dimensionalities; (b) Marginal relevance of additional constraints; and (c) objective improvement when additional constraints (M=8) are added sequentially.

a binary pairwise MRF where each node corresponds to a superpixel [1] in the image. We extracted up to 51 features at each superpixel. Edge features were computed for each pair of adjacent superpixels. These correspond to a standard Potts model and a contrast-sensitive Potts model. The weights at each edge were constrained to be positive so that the resulting supermodular potentials could be maximized via Graph-Cuts [5, 14].

Effect of M on convergence. Let γ be the amount by which the most-violated-constraint is violated at the current iteration. The algorithm stops when $\gamma \leq \epsilon$. Fig. 2a plots γ vs iterations for a range of values of M. We observe that the greatest reduction in the number of iterations, $\frac{1836-705}{1836} \approx 61.60\%$, was achieved for M=10 and that a further increase in M resulted in a slight reversal of gains.

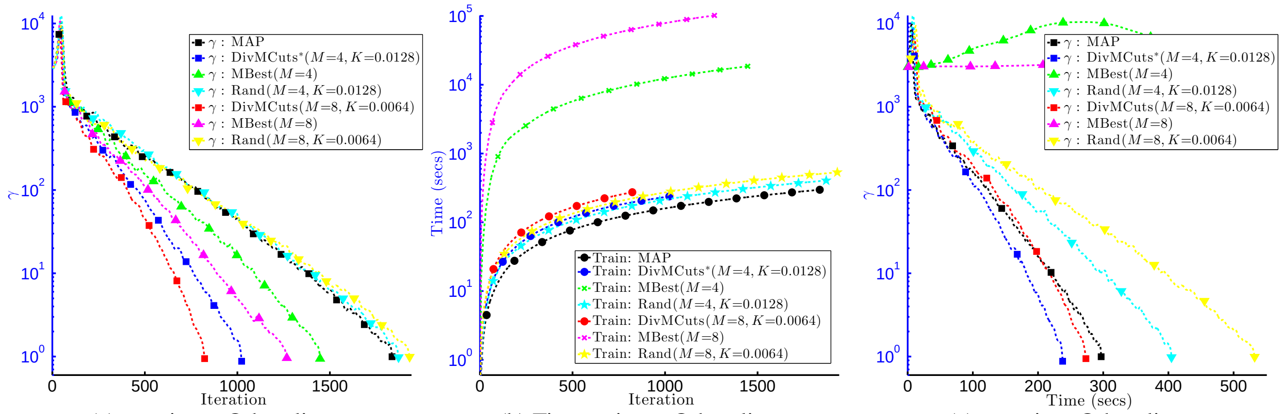
Effect of K on convergence. Fig. 2b shows the effect of the desired diversity parameter K on convergence. We observe that a mid-range value leads to fastest convergence.

This plot also compares two of the feature combination strategies suggested before. We note that the simple $\mathrm{DivMBest-Ordering}$ strategy is not significantly outperformed by the more complex $\mathrm{DOP1-ILP}$ strategy (detailed in the supplementary materials).

A close look at Fig. 2b also reveals that higher levels of diversity are beneficial in earlier iterations of the learning algorithm but actually hurt convergence in later iterations. This suggests it would be beneficial to "anneal" the desired diversity as the algorithm progresses.

Convergence time. Fig. 2c compares execution times for different values of M. We plot total train time as well as the time contributions of inference (*e.g.*, DivMBest), feature computation, and QP optimization. An annealed execution, which lowers K and M during execution, is also included in the plot (starred curve). This curve obtained the greatest speedup with an iteration reduction of $\frac{1836-1023}{1836} \approx 44.28\%$ and a running time reduction of $\frac{297-238\ secs}{297\ secs} \approx 19.87\%$.

Effect of Problem Dimensionality $|\mathbf{w}|$. We investigate the behavior of the proposed approach as the number of features in the learning problem is varied. The unary features we used are: mean RGB; mean HSV; 5 bin Hue histogram and histogram entropy; 3 bin Saturation histogram and histogram entropy; 10 bin HOG histogram and 25 bin SIFT histogram. The edge features are as detailed earlier. The



(a) γ vs iters. @ baselines. (b) Time vs iters. @ baselines. (c) γ vs time @ baselines.

Figure 4: (a,b) Convergence and execution time vs iterations; and (c) Convergence vs time against baselines on f-b segmentation.

	No cache M=1	No cache M=4	Cache M=1	Cache M=4
Total iterations	1836	1015	4048	3687
Iterations from cache	0	0	3974	3622
Time (secs)	297	245	104	**97**

Table 1: DivMCuts with caching on f-b segmentation.

model has a parameter for each feature-label combination, *i.e.*, 2 parameters for each unary feature and 4 parameters for each edge feature.

We dropped different subsets of the features and re-tested the effect of M on convergence. The plots in Fig. 3a show that as the dimensionality $|\mathbf{w}|$ of the problem increases, higher values of M lead to greater (percental) iteration reductions. This suggest we may expect greater computational savings on problems of higher dimensionality.

Value of additional constraints. In Fig. 3b, 3c we confirm that the cutting-planes generated by DivMCuts do posses marginal relevance throughout the training process. Specifically, we add each of the M constraints *sequentially* to the working-set. For every addition, we solve the intermediate QP and show: 1) Fig. 3b, the violation of each constraint just before it is added to the working-set; and 2) Fig. 3c, the improvement in the objective after the QP is solved. We see that the M constraints are in fact, i) violated, and ii) continue to improve the QP objective even in the presence of the previous constraints. This behavior precisely explains the observed reductions in training iterations.

Comparison against Baselines. In Fig. 4a we observe that the Rand baseline produced an increase in the number of iterations to convergence (w.r.t. MAP). MBest obtained about half the decrease in the number of iterations obtained by DivMCuts, but as shown in Fig. 4b, MBest is close to *three orders of magnitude* slower than the other algorithms. Fig. 4c compares the algorithms' convergence vs time performance.

Caching Constraints. We experimented with the caching methodology implemented in SVM-Struct [10] using default parameters (*e.g.*, cache size). When combining caching with DivMCuts, multiple constraints are added only when constructing a constraint from the cache fails. We cache only most violated constraints to be consistent with SVM-Struct. Table 1 shows results for a few combinations of caching and M. For this application, caching works very well and reduces the speedup due to our method. However, caching has negligible effect on the experiments in the next section.

4.2 Protein Side-Chain Prediction

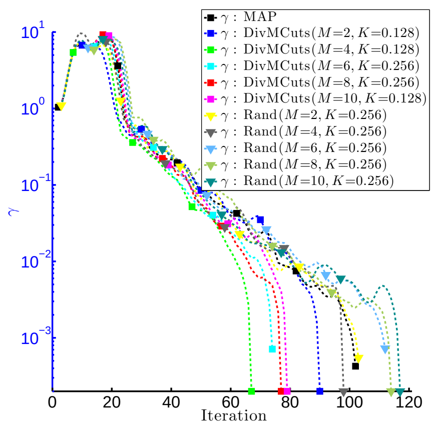
Model and Dataset. Given a protein backbone structure, the task here is to predict the amino acid side-chain configurations. This problem has been traditionally formulated as a pairwise MRF with node labels corresponding to (discretized) side-chain configurations (*rotamers*). These models include pairwise interactions between nearby side-chains, and between side-chains and backbone. We use the dataset of [6] which consists of 276 proteins (up to 700 residues long).[2] The energy function is defined as a weighted sum of eight known energy terms where the weights are to be learned.

To speedup inference and feature computation we carried both tasks in parallel (4 workers). We continue to report total train time below – times reported are wall-clock times and not CPU times. Note that parallelization has no effect in the number of cutting-plane iterations and minimal effect on training time ratios (due to non-zero overhead). For inference we used TRW-S [13].

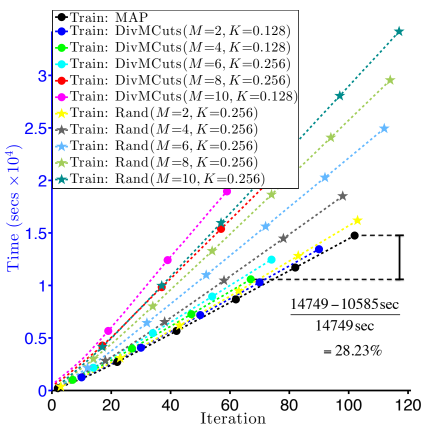
Effect of M on convergence. In Fig. 5a, 5b and 5c we observe that the greatest speedup was obtained for M=4 with an iteration reduction of $\frac{102-67}{102} \approx 34.31\%$ and a running time reduction of $\frac{14749-10585\,secs}{14749\,secs} \approx 28.23\%$.

Interestingly, DivMCuts achieved greater time savings for

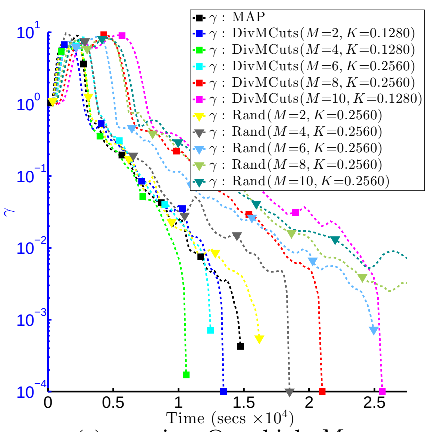
[2]Dataset available from: http://cyanover.fhcrc.org/recomb-2007/



(a) γ vs iters. @ multiple M.



(b) Time vs iters. @ multiple M.



(c) γ vs time @ multiple M.

Figure 5: (a,b) Convergence and execution time vs iterations; and (c) Convergence vs time on protein side-chain prediction (Rand baseline also plotted).

	No cache $M=1$	No cache $M=4$	Cache $M=1$	Cache $M=4$
Total iterations	102	67	120	76
Iterations from cache	0	0	4	2
Time (secs)	16299	**11468**	18399	12668

Table 2: DivMCuts with caching on protein side-chain prediction.

this application than for the previous one – especially considering a smaller reduction in the number of iterations. The reason for this is that inference and feature computation are much more expensive here. This translates into greater savings arising from dynamic inference and incremental feature computation.

Comparison against Baselines. Here, we do not report performance of the MBest baseline as it is prohibitively expensive. The Rand baseline is at best useless and often detrimental while DivMCuts always leads to iteration and time reductions, Fig. 5a, 5b and 5c.

Caching Constraints. We ran caching experiments analogous to those in the previous section. In this case, as shown in Table 2, caching has negligible effect on training iterations (*i.e.*, very few constraints are constructed from the cache). Hence, DivMCuts is able to provide essentially the same benefit as when caching is not used. [3]

5 Discussion and Conclusions

We investigated the effect of adding multiple highly violated constraints in the context of cutting-plane training of Structural SVMs. We noted that significant improvements in the convergence of the training algorithm are possible if the added constraints are: 1) highly violated, 2) diverse, and 3) efficiently computable. We presented an an efficient algorithm, DivMCuts, for generating such constraints and showed experimentally that our method leads to significant savings: $>60\%$ reduction in the number of iterations and $\sim$28% reduction in training time. Moreover, our results suggest that greater speedups are possible in applications with higher feature dimensionality.

The idea of adding multiple violated inequalities in CP optimization of SSVMs was mentioned in passing in [21]. In the operations research and mathematical programming literature, there is a line of work [8, 9, 27] analyzing the iteration complexity when the oracle returns multiple violated inequalities. However, this line of work assumes a general multiple-violation oracle which may return the same constraint M times. Thus, the resulting bounds on run-time are pessimistic and often *increasing* with M. Our setting is different in that we are able to guarantee the constraints returned are both highly violated and diverse. To our knowledge, no bounds specific to this setting exist and providing such bounds remains an interesting future direction.

Stochastic Subgradient (SSG) methods [19, 20] are popular because they achieve the same convergence rate of CP training while requiring a single call to the separation-oracle at every iteration. However, as noted in [10], CP methods have some benefits over SSG: 1) SSG is very sensitive to the choice of step-size rule. 2) While for CP the theory provides a practically effective stopping criterion based on duality gap, it is less clear when to stop primal SSG methods. Moreover, there are regimes (e.g., low regularization) where CP outperforms SSG (*e.g.*, Fig. 1 in [15]). Recently, [15] proposed a method that combines the strengths of SSG and CP methods (i.e., same convergence rate; single separation-oracle call per iteration; no step-size selection; and duality gap guarantee). We believe the ideas in this paper can be applied to this new online algorithm and leave this as a direction for future work.

Acknowledgments: AGR was supported by the C2S2 Focus Center (under SRC's Focus Center Research Program).

[3]The time results for "No cache" in Table 2 differ from those in Fig. 5b, 5c due to machine loading variations at the times the experiments were conducted.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *PAMI*, 34(11):2274–2281, 2012. 6

[2] A. Barbu and S.-C. Zhu. Generalizing Swendsen-Wang to Sampling Arbitrary Posterior Probabilities. *PAMI*, 27:1239–1253, August 2005. 3

[3] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. iCoseg: Interactive Co-segmentation with Intelligent Scribble Guidance. In *CVPR*, 2010. 5

[4] D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich. Diverse M-Best Solutions in Markov Random Fields. In *ECCV*, 2012. 1, 2, 3, 4

[5] Y. Boykov, O. Veksler, and R. Zabih. Efficient Approximate Energy Minimization via Graph Cuts. *PAMI*, 20(12):1222–1239, 2001. 6

[6] O. S.-F. Chen Yanover and Y. Weiss. Minimizing and Learning Energy Functions for Side-Chain Prediction. *Journal of Computational Biology*, 15(7):899–911, 2008. 7

[7] M. Fromer and A. Globerson. An LP View of the M-best MAP problem. In *NIPS*, 2009. 3

[8] J.-L. Goffin and J.-P. Vial. Multiple Cuts in the Analytic Center Cutting Plane Method. *SIAM J. on Optimization*, 11(1):266–288, Jan. 2000. 8

[9] J.-L. Goffin and J.-P. Vial. Convex Nondifferentiable Optimization: A Survey Focussed On The Analytic Center Cutting Plane Method. *Optimization Methods and Software*, 17(5):805–867, 2002. 8

[10] T. Joachims, T. Finley, and C.-N. Yu. Cutting-Plane Training of Structural SVMs. *Machine Learning*, 77(1):27–59, 2009. 1, 2, 3, 5, 7, 8

[11] J. E. Kelley Jr. The Cutting-Plane Method for Solving Convex Programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):pp. 703–712, 1960. 1

[12] P. Kohli and P. H. S. Torr. Effciently Solving Dynamic Markov Random Fields Using Graph Cuts. In *ICCV*, pages 922–929, 2005. 4, 5

[13] V. Kolmogorov. Convergent Tree-Reweighted Message Passing for Energy Minimization. *PAMI*, 28(10):1568–1583, 2006. 7

[14] V. Kolmogorov and R. Zabih. What Energy Functions can be Minimized via Graph Cuts? *PAMI*, 26(2):147–159, 2004. 6

[15] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. In *ICML*, 2013. 8

[16] E. R. Natalia Flerova and R. Dechter. Bucket and mini-bucket Schemes for M Best Solutions over Graphical Models. In *IJCAI Workshop on Graph Structures for KRR*, 2011. 3

[17] D. Nilsson. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing*, 8:159–173, 1998. 3

[18] J. Porway and S.-C. Zhu. C^4: Exploring Multiple Solutions in Graphical Models by Cluster Sampling. *PAMI*, 33(9):1713–1727, 2011. 3

[19] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. (Online) Subgradient Methods for Structured Prediction. In *AISTATS*, 2007. 8

[20] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. *Mathematical Programming*, 127(1):3–30, 2011. 8

[21] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs Using Graph Cuts. In *ECCV*, 2008. 8

[22] D. Tarlow, D. Batra, P. Kohli, and V. Kolmogorov. Dynamic Tree Block Coordinate Ascent. In *ICML*, 2011. 4

[23] B. Taskar, C. Guestrin, and D. Koller. Max-Margin Markov Networks. In *NIPS*, 2003. 1

[24] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *JMLR*, 6:1453–1484, 2005. 1, 2

[25] Z. Tu and S.-C. Zhu. Image Segmentation by Data-Driven Markov Chain Monte Carlo. *PAMI*, 24:657–673, May 2002. 3

[26] C. Yanover and Y. Weiss. Finding the M Most Probable Configurations Using Loopy Belief Propagation. In *NIPS*, 2003. 3, 5

[27] Y. Ye. Complexity analysis of the analytic center cutting plane method that uses multiple cuts. *Math. Program.*, 78(1):85–104, July 1997. 8