Training Structural SVMs when Exact Inference is Intractable

Thomas Finley   TOMF@CS.CORNELL.EDU
Thorsten Joachims   TJ@CS.CORNELL.EDU
Cornell University, Department of Computer Science, Upson Hall, Ithaca, NY 14853 USA

Abstract

While discriminative training (e.g., CRF, structural SVM) holds much promise for machine translation, image segmentation, and clustering, the complex inference these applications require make exact training intractable. This leads to a need for approximate training methods. Unfortunately, knowledge about how to perform efficient and effective approximate training is limited. Focusing on structural SVMs, we provide and explore algorithms for two different classes of approximate training algorithms, which we call undergenerating (e.g., greedy) and overgenerating (e.g., relaxations) algorithms. We provide a theoretical and empirical analysis of both types of approximate trained structural SVMs, focusing on fully connected pairwise Markov random fields. We find that models trained with overgenerating methods have theoretic advantages over undergenerating methods, are empirically robust relative to their undergenerating brethren, and relaxed trained models favor non-fractional predictions from relaxed predictors.

1. Introduction

Discriminative training methods like conditional random fields (Lafferty et al., 2001), maximum-margin Markov networks (Taskar et al., 2003), and structural SVMs (Tsochantaridis et al., 2004) have substantially improved prediction performance on a variety of structured prediction problems, including part-of-speech tagging (Altun et al., 2003), natural language parsing (Tsochantaridis et al., 2005), sequence alignment (Yu et al., 2007), and classification under multivariate loss functions (Joachims, 2005). In the context

of structural SVMs, in all these problems, both the inference problem (i.e., computing a prediction) and the separation oracle required in the cutting-plane training algorithm can be solved exactly. This leads to theoretical guarantees of training procedure convergence and solution quality.

However, in many important problems (e.g., clustering (Culotta et al., 2007; Finley & Joachims, 2005), multi-label classification, image segmentation, machine translation) exact inference and the separation oracle are computationally intractable. Unfortunately, use of approximations in these settings abandons many of the existing theoretical guarantees of structural SVM training, and relatively little is known about discriminative training using approximations.

This paper explores training structural SVMs on problems where exact inference is intractable. A pairwise fully connected Markov random field (MRF) serves as a representative class of intractable models. This class includes natural formulations of models for multi-label classification, image segmentation, and clustering. We identify two classes of approximation algorithms for the separation oracle in the structural SVM cutting-plane training algorithm, namely undergenerating and overgenerating algorithms, and we adapt loopy belief propagation (LBP), greedy search, and linear-programming and graph-cut relaxations to this problem. We provide a theoretical and empirical analysis of using these algorithms with structural SVMs.

We find substantial differences between different approximate algorithms in training and inference. In particular, much of the existing theory holds for overgenerating though not undergenerating methods. In experimental results, intriguingly, our structural SVM formulations using the overgenerating linear-programming and graph-cut relaxations successfully learn models in which relaxed inference is "easy" (i.e., the relaxed solution is mostly integral), leading to robust and accurate models. We conclude that the relaxation formulations are preferable over the formulations involving LBP and greedy search.

Algorithm 1 Cutting plane algorithm to solve OP 1.
1: Input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, C, ϵ
2: $S_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
3: **repeat**
4:    **for** $i = 1, \ldots, n$ **do**
5:       $H(\mathbf{y}) \equiv \Delta(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) - \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i)$
6:       compute $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$
7:       compute $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$
8:       **if** $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$ **then**
9:          $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
10:        $\mathbf{w} \leftarrow$ optimize primal over $\bigcup_i S_i$
11:      **end if**
12:    **end for**
13: **until** no S_i has changed during iteration

---

2. Structured Output Prediction

Several discriminative structural learners were proposed in recent years, including conditional random fields (CRFs) (Lafferty et al., 2001), Perceptron HMMs (Collins, 2002), max-margin Markov networks (M³Ns) (Taskar et al., 2003), and structural SVMs (SSVMs) (Tsochantaridis et al., 2004). Notational differences aside, these methods all learn (kernelized) linear discriminant functions, but differ in how they choose model parameters.

2.1. Structural SVMs

Structural SVMs minimize a particular trade-off between model complexity and empirical risk. Based on a training set $\mathcal{S} = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n))$, an SSVM learns a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ to map inputs $\mathbf{x} \in \mathcal{X}$ to outputs $\mathbf{y} \in \mathcal{Y}$. Hypotheses take the form $h(\mathbf{x}) = \text{argmax}_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}, \mathbf{y})$ with discriminant function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$. The $\mathbf{\Psi}$ combined feature vector function relates inputs and outputs, and $\mathbf{w}$ contains parameters adjusted during training. We also define a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ indicating how bad output $h(\mathbf{x}_i)$ is w.r.t. true output $\mathbf{y}_i$. To find $\mathbf{w}$ balancing model complexity and empirical risk $R_{\mathcal{S}}^{\Delta}(h) = \frac{1}{n} \sum_{i=1}^{n} \Delta(\mathbf{y}_i, h(\mathbf{x}_i))$, SSVMs solve the following quadratic program (QP) for the case of the margin-rescaling hinge loss (Tsochantaridis et al., 2004):

Optimization Problem 1. (STRUCTURAL SVM)

$$\min_{\mathbf{w}, \xi \geq \mathbf{0}} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \qquad (1)$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \qquad (2)$$

Introducing a constraint for every wrong output is typically intractable. However, OP 1 can be solved by the cutting plane algorithm in Algorithm 1. This iteratively constructs a sufficient subset $\bigcup_i S_i$ of constraints and solves the QP only over this subset (line 10). The algorithm employs a separation oracle to find the next constraint to include (line 6). It finds the currently most violated constraint (or, a constraint that is violated by at least the desired precision ϵ). If such a separation oracle exists and can be computed in polynomial time, OP 1 and Algorithm 1 have three theoretical guarantees (Tsochantaridis et al., 2005):

Polynomial Time Termination: Algorithm 1 terminates in a polynomial number of iterations, and thus overall polynomial time.

Correctness: Algorithm 1 solves OP 1 accurate to a desired precision ϵ, since Algorithm 1 terminates only when all constraints in OP 1 are respected within ϵ (lines 8 and 13).

Empirical Risk Bound: Since each ξ_i upper bounds training loss $\Delta(\mathbf{y}_i, h(\mathbf{x}_i))$, $\frac{1}{n} \sum_{i=1}^{n} \xi_i$ upper bounds empirical risk.

Unfortunately, proofs of these properties rely on the separation oracle (line 6) being exactly solvable, and do not necessarily hold with approximations. We will later analyze which of the approximate training algorithms retain which properties.

2.2. Markov Random Fields in SSVMs

A special case of structural SVM that we will examine throughout this paper is M³N (Taskar et al., 2003). In this, $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is constructed from an MRF

$$f(\mathbf{x}, \mathbf{y}) = \sum_{k \in \text{cliques}(G)} \phi_k(y_{\{k\}}) \qquad (3)$$

with graph structure $G = (V, E)$ and the loss function is restricted to be linearly decomposable in the cliques, i.e., $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k \in \text{cliques}(G)} \delta_k(y_{\{k\}}, \hat{y}_{\{k\}})$. Here, $\mathbf{y}$ is the value assignment to variables, δ_k are sub-component local loss functions, and ϕ_k are potential functions representing the fitness of variable assignment $y_{\{k\}}$ to clique k. The network potential $f(\mathbf{x}, \mathbf{y})$ serves as a discriminant function representing the variable assignment $\mathbf{y}$ in the structural SVM, and $h(\mathbf{x}) = \text{argmax}_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}, \mathbf{y})$ serves as the maximum a posteriori (MAP) prediction.

OP 1 requires we express (3) in the form $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y})$. First express potentials as $\phi_k(y_{\{k\}}) = \mathbf{w}^T \psi(\mathbf{x}, y_{\{k\}})$. The feature vector functions ψ_k relate $\mathbf{x}$ and label assignments $y_{\{k\}}$. Then, $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ where $\mathbf{\Psi}(\mathbf{x}, \mathbf{y}) = \sum_{k \in \text{cliques}(G)} \psi_k(\mathbf{x}, y_{\{k\}})$.

In the following, we use a particular linearly decomposable loss function that simply counts the proportion of different labels in $\mathbf{y}$ and $\hat{\mathbf{y}}$, i.e., $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|_0/|V|$. Further, in our applications, labels are binary (i.e., each $y_u \in \mathbb{B} = \{0, 1\}$), and we allow only $\phi_u(1)$ and $\phi_{uv}(1, 1)$ potentials to be non-zero. This latter restriction may seem onerous, but any pairwise binary MRF with non-zero $\phi_u(0), \phi_{uv}(0, 0), \phi_{uv}(0, 1), \phi_{uv}(1, 0)$ has an equivalent MRF where these potentials are zero.

To use Algorithm 1 for MRF training and prediction, one must solve two argmax problems:

Prediction: $\mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$

Separation Oracle: $\mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y})$

The prediction problem is equivalent to MAP inference. Also, the separation oracle can be reduced to a MAP inference problem. Taking the MRF we would use to solve $\mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$, we include $\Delta(\mathbf{y}_i, \mathbf{y})$ in the argmax by incrementing the node potential $\phi_u(y)$ by $1/|V|$ for each wrong value y of u, since each wrong variable assignment increases loss by $1/|V|$. Thus, we may express the separation oracle as MAP inference.

3. Approximate Inference

Unfortunately, MAP inference is $\#P$-complete for general MRFs. Fortunately, a variety of approximate inference methods exist. For prediction and the separation oracle, we explore two general classes of approximate inference methods, which we call undergenerating and overgenerating approximations.

3.1. Undergenerating Approximations

Undergenerating methods approximate $\mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}$ by $\mathrm{argmax}_{\mathbf{y} \in \underline{\mathcal{Y}}}$, where $\underline{\mathcal{Y}} \subseteq \mathcal{Y}$. We consider the following undergenerating methods in the context of MRFs:

Greedy iteratively changes the single variable value y_u that would increase network potential most.

LBP is loopy belief propagation (Pearl, 1988).

Combine picks the assignment $\mathbf{y}$ with the highest network potential from both greedy and LBP.

We now attempt to theoretically characterize undergenerating learning and prediction. Due to space constraints, provided proofs are proof skeletons.

Since undergenerating approximations can be arbitrarily poor, we must restrict our consideration to a subclass of undergenerating approximations to make meaningful theoretical statements. This analysis focuses on ρ-approximation algorithms, with $\rho \in (0, 1]$. What is a ρ-approximation? In our case, for predic-

tive inference, if $\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is the true optimum and $\mathbf{y}'$ the ρ-approximation output, then

$$\rho \cdot \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}^*) \leq \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}') \qquad (4)$$

Similarly, for our separation oracle, for $\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y})$ as the true optimum, and if $\mathbf{y}'$ corresponds to the constraint found by our ρ-approximation, we know

$$\rho\big[\mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}^*) + \Delta(\mathbf{y}_i, \mathbf{y}^*)\big] \leq \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}') + \Delta(\mathbf{y}_i, \mathbf{y}') \quad (5)$$

For simplicity, this analysis supposes $\mathcal{S}$ contains exactly one training example $(\mathbf{x}_0, \mathbf{y}_0)$. To generalize these results, one may view n training examples as 1 example, where inference consists of n separate processes with combined outputs, etc. Fortunately, combined outputs from multiple ρ-approximations may be viewed as output from a single ρ-approximation.

Theorem 1. (POLYNOMIAL TIME TERMINATION) *If $\bar{R} = \max_{i, \mathbf{y} \in \mathcal{Y}} \|\mathbf{\Psi}(\mathbf{x}_i, \mathbf{y})\|$, $\bar{\Delta} = \max_{i, \mathbf{y} \in \mathcal{Y}} \|\Delta(\mathbf{y}_i, \mathbf{y})\|$ are finite, an undergenerating learner terminates after adding at most $\epsilon^{-2}(C\bar{\Delta}^2\bar{R}^2 + n\bar{\Delta})$ constraints.*

Proof. The original proof still holds as it does not depend upon separation oracle quality. $\square$

Lemma 1. *After line 6 in Algorithm 1, let $\mathbf{w}$ be the current model, $\hat{\mathbf{y}}$ the constraint found with the ρ-approximation separation oracle, and $\hat{\xi} = H(\hat{\mathbf{y}})$ the slack associated with $\hat{\mathbf{y}}$. Then, $\mathbf{w}$ and slack $\hat{\xi} + \frac{1-\rho}{\rho}\big[\mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_0, \hat{\mathbf{y}}) + \Delta(\mathbf{y}_0, \hat{\mathbf{y}})\big]$ is feasible in OP 1.*

Proof. If we knew the true most violated constraint $\mathbf{y}^*$, we would know the minimum ξ^* such that $\mathbf{w}, \xi^*$ was feasible in OP 1. The proof upper bounds ξ^*. $\square$

Theorem 2. *When iteration ceases with the result $\mathbf{w}, \xi$, if $\hat{\mathbf{y}}$ was the last found most violated constraint, we know that the optimum objective function value v^* for OP 1 lies in the interval*

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\xi \leq v^* \leq$$
$$\frac{1}{2}\|\mathbf{w}\|^2 + C\Big[\frac{1}{\rho}[\mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_0, \hat{\mathbf{y}}) + \Delta(\mathbf{y}_0, \hat{\mathbf{y}})] - \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_0, \mathbf{y}_0)\Big]$$

Proof. Lemma 1 applied to the last iteration. $\square$

The intuitive lesson of these proofs is that, even with ρ-approximate separation oracles, one may bound how far off a final solution is from solving OP 1. Sensibly, the better the approximation, i.e., as ρ approaches 1, the tighter the solution.

The last result concerns empirical risk. The SVM margin attempts to ensure that high-loss outputs have a low discriminant function value, and ρ-approximations produce outputs within a certain factor of optimum.

Theorem 3. (ρ-APPROXIMATE EMPIRICAL RISK) *For* $\mathbf{w}, \xi$ *feasible in OP 1 from training with single example* $(\mathbf{x}_0, \mathbf{y}_0)$, *the empirical risk using ρ-approximate prediction has upper bound* $(1 - \rho)\mathbf{w}^T \boldsymbol{\Psi}(\mathbf{x}_0, \mathbf{y}_0) + \xi$.

Proof. Take the $\mathbf{y}' = h(\mathbf{x}_0)$ associated constraint, then apply known bounds to its $\mathbf{w}^T \boldsymbol{\Psi}(\mathbf{x}_0, \mathbf{y}')$ term. $\qquad\square$

If also using undergenerating ρ-approximate training, one may employ Theorem 2 to get a feasible ξ.

3.2. Overgenerating Approximations

Overgenerating methods approximate $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}}$ by $\operatorname{argmax}_{\mathbf{y} \in \overline{\mathcal{Y}}}$, where $\overline{\mathcal{Y}} \supseteq \mathcal{Y}$. We consider the following overgenerating methods:

LProg is an expression of the inference problem as a relaxed integer linear program (Boros & Hammer, 2002). We first add $y_{uv} \in \mathbb{B}$ values indicating if $y_u = y_v = 1$ to linearize the program:

$$\max_{\mathbf{y}} \sum_{u \in \{1..|V|\}} y_u \phi_u(1) + \sum_{u,v \in \{1..|V|\}} y_{uv} \phi_{uv}(1,1) \quad (6)$$

$$\text{s.t.} \quad \forall u,v. \quad y_u \geq y_{uv} \qquad y_v \geq y_{uv} \qquad (7)$$

$$y_u + y_v \leq 1 + y_{uv} \qquad y_u, y_{uv} \in \mathbb{B} \, (8)$$

We then relax "$\in \mathbb{B}$" to "$\in [0,1]$" to admit fractional solutions. Importantly, there is always some optimal solution where all $y_u, y_{uv} \in \{0, \frac{1}{2}, 1\}$ (Hammer et al., 1984).

Cut is quadratic pseudo-Boolean optimization using a graph-cut (Kolmogorov & Rother, 2004). This is a different relaxation where, instead of $\mathbf{y} \in \mathbb{B}^{|V|}$, we have $\mathbf{y} \in \{0, 1, \varnothing\}^{|V|}$.

The LProg and Cut approximations share two important properties (Boros & Hammer, 2002; Hammer et al., 1984): *Equivalence* says that maximizing solutions of the Cut and LProg formulations are transmutable. One proof defines this transmutation procedure, where $\varnothing$ (in cuts optimization) and $\frac{1}{2}$ (in LP optimization) variable assignments are interchangeable (Boros & Hammer, 2002). The important practical implication of equivalence is both approximations return the same solutions. *Persistence* says unambiguous labels (i.e., not fractional or $\varnothing$) are optimal labels.

As a final detail, in the case of LProg, $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{|V|} \sum_{u \in \{1..|V|\}} |y_u - \hat{y}_u|$ and $\boldsymbol{\Psi}(\mathbf{x}, \mathbf{y}) = \sum_{u \in \{1..|V|\}} y_u \psi_u(1) + \sum_{u,v \in \{1..|V|\}} y_{uv} \psi_{uv}(1,1)$. Cut's functions have similar formulations.

Theorem 4. (POLYNOMIAL TIME TERMINATION) *If* $\bar{R} = \max_{i, \mathbf{y} \in \overline{\mathcal{Y}}} \|\boldsymbol{\Psi}(\mathbf{x}_i, \mathbf{y})\|$, $\bar{\Delta} = \max_{i, \mathbf{y} \in \overline{\mathcal{Y}}} \|\Delta(\mathbf{y}_i, \mathbf{y})\|$ *are finite ($\overline{\mathcal{Y}}$ replacing $\mathcal{Y}$ in the overgenerating case),*

an overgenerating learner terminates after adding at most $\epsilon^{-2}(C\bar{\Delta}^2 \bar{R}^2 + n\bar{\Delta})$ *constraints.*

Proof. The original proof holds as an overgenerating learner is a straightforward structural learning problem on a modified output range $\overline{\mathcal{Y}}$. $\qquad\square$

Theorem 5. (CORRECTNESS) *An overgenerating Algorithm 1 terminates with* $\mathbf{w}, \xi$ *feasible in OP 1.*

Proof. The learner considers a superset of outputs $\overline{\mathcal{Y}} \supseteq \mathcal{Y}$, so constraints in OP 1 are respected within ϵ. $\qquad\square$

With these "extra" constraints from overgenerating inference, Algorithm 1's solution may be suboptimal w.r.t. the original OP 1. Further, for undergenerating methods correctness does not hold, as Algorithm 1 may not find violated constraints present in OP 1.

Theorem 6. (EMPIRICAL RISK BOUND) *If prediction and the separation oracle use the same overgenerating algorithm, Algorithm 1 terminates with* $\frac{1}{n} \sum_i \xi_i$ *upper bounding empirical risk* $R_{\mathcal{S}}^{\Delta}(h)$.

Proof. Similar to the proof of Theorem 4.

3.3. Related Work

In prior work on discriminative training using approximate inference, structural SVMs have learned models for correlation clustering, utilizing both greedy and LP relaxed approximations (Finley & Joachims, 2005). For M³Ns, Anguelov et al. (Anguelov et al., 2005) proposed to directly fold a linear relaxation into OP 1. This leads to a very large QP, and is inapplicable to other inference methods like LBP or cuts. Furthermore, we will see below that the linear-program relaxation is the slowest method. With CRFs, likelihood training requires computing the partition function in addition to MAP inference. Therefore, the partition function is approximated (Culotta et al., 2007; He et al., 2004; Kumar & Hebert, 2003; Vishwanathan et al., 2006), or the model is simplified to make the partition function tractable (Sutton & McCallum, 2005), or CRF max-likelihood training is replaced with Perceptron training (Roth & Yih, 2005). Prior theoretical analysis has examined LBP and LP-relaxation approximations for MRF structural learning (Kulesza & Pereira, 2007). Our results differ in that this does not analyze a particular learning problem or method.

4. Experiments: Approximate Inference

Before we move into learning experiments, it helps to understand the runtime and quality performance char-
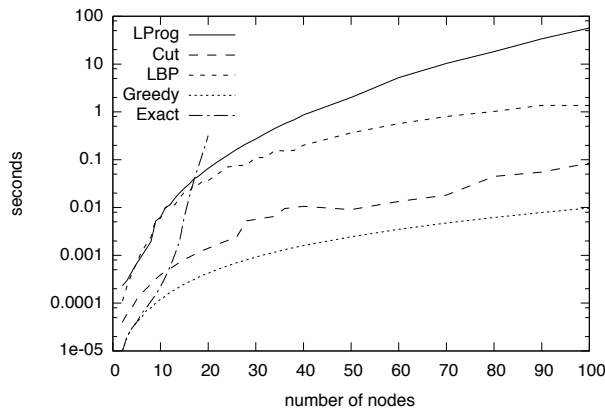
Figure 1. Runtime comparison. Average MAP inference time for different methods on random problems of different sizes.
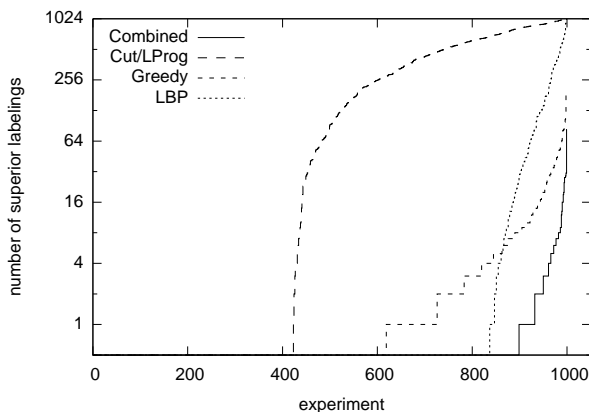

Figure 2. Quality comparison. Inference on 1000 random 18 label problems. Lower curves are better.

acteristics of our MAP inference algorithms.

For runtime, Figure 1 illustrates each approximate inference method's average time to solve a single pairwise fully connected MRF with random potentials as the number of nodes increases.[1] Note that cuts are substantially faster than LBP, and several orders of magnitude faster than the linear relaxation while maintaining equivalence.

For evaluating solution quality, we generate 1000 random problems, ran the inference methods, and exhaustively count how many labelings with higher discriminant value exist. The resulting curve for 10-node MRFs is shown in Figure 2. For cut, $\varnothing$ labels are randomly assigned to 0 or 1. The lower the curve,

---

[1]Implementation details: The methods were C-language Python extension modules. LProg was implemented in GLPK (see http://www.gnu.org/software/glpk/glpk.html). Cut was implemented with Maxflow software (Boykov & Kolmogorov, 2004). Other methods are home-spun. Experiments were run on a 2.6 GHz P4 Linux box.

---

the better the inference method. LBP finds "perfect" labelings more often than Greedy, but also tends to fall into horrible local maxima. Combined does much better than either alone; apparently the strengths of Greedy and LBP are complimentary.

Finally, note the apparent terrible performance of Cut, which is due to assigning many $\varnothing$ labels. At first glance, persistence is an attractive property since we *know* unambiguous labels are correct, but on the other hand, classifying only when it is certain leads it to leave many labels ambiguous.

5. Experiments: Approximate Learning

Our goal in the following experiments is to gain insight about how different approximate MRF inference methods perform in SSVM learning and classification. Our evaluation uses multi-label classification using pairwise fully connected MRFs as an example application.

Multi-label classification bears similarity to multi-class classification, except classes are not mutually exclusive, e.g., a news article may be about both "Iraq" and "oil." Often, incorporating inter-label dependencies into the model can improve performance (Cesa-Bianchi et al., 2006; Elisseeff & Weston, 2002).

How do we model this labeling procedure as an MRF? For each input $\mathbf{x}$, we construct a single MRF, with a vertex for each possible label, with possible values from $\mathbb{B} = \{0, 1\}$ (value 1 indicates $\mathbf{x}$ has the corresponding label), and an edge for each vertex pair (i.e., complete graph MRF).

What are our potential functions? In these problems, inputs $\mathbf{x} \in \mathbb{R}^m$ are feature vectors. Each of the ℓ possible labels u is associated with a weight vector $\mathbf{w}_u \in \mathbb{R}^m$. The resulting vertex potentials are $\phi_u(1) = \mathbf{w}_u^T \mathbf{x}$. Edge potentials $\phi_{uv}(1, 1)$ come from individual values in $\mathbf{w}$, one for each label pair. Thus, the overall parameter vector $\mathbf{w} \in \mathbb{R}^{\ell m + \binom{\ell}{2}}$ has ℓm weights for the ℓ different $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_\ell$ sub-component weight vectors, and $\binom{\ell}{2}$ parameters for edge potentials. In terms of ψ functions, $\psi_u(\mathbf{x}, 1)$ vectors contain an offset version of $\mathbf{x}$ to "select out" $\mathbf{w}_u$ from $\mathbf{w}$, and $\psi_{uv}(\mathbf{x}, 1, 1)$ vectors have a single 1 entry to "select" the appropriate element from the end of $\mathbf{w}$.

5.1. Datasets and Model Training Details

We use six multi-label datasets to evaluate performance. Table 1 contains statistics on these datasets. Four real datasets, **Scene** (Boutell et al., 2004), **Yeast** (Elisseeff & Weston, 2002), **Reuters** (the RCV1 subset 1 data set) (Lewis et al., 2004), and **Me-**

Table 1. Basic statistics for the datasets, including number of labels, training and test set sizes, number of features, and parameter vector **w** size, and performance on baseline trained methods and a default model.

DATASET	LABELS	TRAIN	TEST	FEATS.	**w** SIZE	BASELINE	DEFAULT
SCENE	6	1211	1196	294	1779	11.43±.29	18.10
YEAST	14	1500	917	103	1533	20.91±.55	25.09
REUTERS	10	2916	2914	47236	472405	4.96±.09	15.80
MEDIAMILL	10	29415	12168	120	1245	18.60±.14	25.37
SYNTH1	6	471	5045	6000	36015	8.99±.08	16.34
SYNTH2	10	1000	10000	40	445	9.80±.09	10.00

diamill (Snoek et al., 2006), came from the LIBSVM multi-label dataset collection (Chang & Lin, 2001). **Synth1** is a synthetic dataset of 6 labels. Labels follow a simple probabilistic pattern: label i is on half the time label $i-1$ is on and never otherwise, and label 1 is always on. Also, each label has 1000 related binary features (the learner does not know a priori which feature belong to each label): if i is on, a random 10 of its 1000 are set to 1. This hypothesis is learnable without edge potentials, but exploiting label dependency structure may result in better models. **Synth2** is a synthetic dataset of 10 labels. In this case, each example has exactly one label on. There are also 40 features. For an example, if label i is on, $4i$ randomly chosen features are set to 1. Only models with edge potentials can learn this concept.

We used 10-fold cross validation to choose C from a population of 14 possible values $\{1 \cdot 10^{-2}, 3 \cdot 10^{-2}, 1 \cdot 10^{-1}, \ldots, 3 \cdot 10^{4}\}$. This C was then used when training a model on all training data. A separate C was chosen for each dataset and separation oracle.

5.2. Results and Analysis

Table 2 reports loss on the test set followed by standard error. For each dataset, we present losses for each combination of separation oracle used in learning (the rows) and of predictive inference procedure used in classification (the columns). This lets us distinguish badly learned models from bad inference procedures as explanations for inferior performance.

We also employ three additional methods as a point of comparison. Our **Baseline** is an MRF with no edge potentials, and our **Default** classifier always predicts the best-performing single labeling; results for these appear in Table 1. The **Exact** classifier is one which exhaustively searches for the argmax; to enable comparisons on Reuters and Mediamill, we pruned these datasets to the 10 most frequent labels.

Cut is omitted from Table 2. Its equivalence to LProg means the two are interchangeable and always produce the same results, excepting Cut's superior speed.

Table 3. Percentage of "ambiguous" labels in relaxed inference. Columns represent different data sets. Rows represent different methods used as separation oracles in training.

	SCENE	YEAST	REUTERS	MEDIAMILL	SYNTH1	SYNTH2
GREEDY	0.43%	17.02%	31.28%	20.81%	0.00%	31.17%
LBP	0.31%	0.00%	0.00%	0.00%	0.00%	0.00%
COMBINE	2.90%	91.42%	0.44%	4.27%	0.00%	29.11%
EXACT	0.95%	84.30%	0.67%	65.58%	0.00%	27.92%
LPROG	0.00%	0.43%	0.32%	1.30%	0.00%	1.48%

In all datasets, some edged model always exceeds the performance of the edgeless model. On Mediamill and Reuters, selecting only the 10 most frequent labels robs the dataset of many dependency relationships, which may explain the relatively lackluster performance.

5.2.1. THE SORRY STATE OF LBP, BUT RELAX

Let's first examine the diagonal entries in Table 2. Models trained with LBP separation oracles yield generally poor performance. What causes this? LBP's tendency to fall into horrible local maxima (as seen in Section 4) misled Algorithm 1 to believe its most violated constraint was not violated, leading it to early termination, mirroring the result in (Kulesza & Pereira, 2007). The combined method remedies some of these problems; however, LProg still gives significantly better/worse performance on 3 vs. 1 datasets.

How does LProg training compare against exact training? Table 3 shows that both methods give similar performance. Exact-trained models significantly outperform relaxed-trained models on two datasets, but they also lose on two datasets.

5.2.2. RELAXATION IN LEARNING AND PREDICTION

Observe that relaxation used *in prediction* performs well when applied to models trained with relaxation. However, on models trained with non-relaxed methods (i.e., models that do not constrain fractional solutions), relaxed inference often performs quite poorly. The most ludicrous examples appear in Yeast, Reuters, Mediamill, and Synth2. Table 3 suggests an explanation for this effect. The table lists the percentage of ambiguous labels from the relaxed classifier (fractional in LProg, $\varnothing$ in Cut). Ignoring degenerate LBP-trained models, the relaxed predictor *always* has the fewest ambiguous judgments. Apparently, SSVMs with relaxed separation oracles produce models that disfavor non-integer solutions. In retrospect this is unsurprising: ambiguous labels always incur loss during training. Minimizing loss during training therefore not only reduces training error, but also encourages parameterizations that favor integral (i.e., exact) solutions. Undergenerating and exact training do not control for

Table 2. Multi-labeling loss on six datasets. Results are grouped by dataset. Rows indicate separation oracle method. Columns indicate classification inference method.

	Greedy	LBP	Combine	Exact	LProg	Greedy	LBP	Combine	Exact	LProg
	Scene Dataset					Mediamill Dataset				
Greedy	10.67±.28	10.74±.28	10.67±.28	10.67±.28	10.67±.28	23.39±.16	25.66±.17	24.32±.17	24.92±.17	27.05±.18
LBP	10.45±.27	10.54±.27	10.45±.27	10.42±.27	10.49±.27	22.83±.16	22.83±.16	22.83±.16	22.83±.16	22.83±.16
Combine	10.72±.28	11.78±.30	10.72±.28	10.77±.28	11.20±.29	19.56±.14	20.12±.15	19.72±.14	19.82±.14	20.23±.15
Exact	10.08±.26	10.33±.27	10.08±.26	10.06±.26	10.20±.26	19.07±.14	27.23±.18	19.08±.14	18.75±.14	36.83±.21
LProg	10.55±.27	10.49±.27	10.49±.27	10.49±.27	10.49±.27	18.50±.14	18.26±.14	18.26±.14	18.21±.14	18.29±.14
	Yeast Dataset					Synth1 Dataset				
Greedy	21.62±.56	21.77±.56	21.58±.56	21.62±.56	24.42±.61	8.86±.08	8.86±.08	8.86±.08	8.86±.08	8.86±.08
LBP	24.32±.61	24.32±.61	24.32±.61	24.32±.61	24.32±.61	13.94±.12	13.94±.12	13.94±.12	13.94±.12	13.94±.12
Combine	22.33±.57	37.24±.77	22.32±.57	21.82±.56	42.72±.81	8.86±.08	8.86±.08	8.86±.08	8.86±.08	8.86±.08
Exact	23.38±.59	21.99±.57	21.06±.55	20.23±.53	45.90±.82	6.89±.06	6.86±.06	6.86±.06	6.86±.06	6.86±.06
LProg	20.47±.54	20.45±.54	20.47±.54	20.48±.54	20.49±.54	8.94±.08	8.94±.08	8.94±.08	8.94±.08	8.94±.08
	Reuters Dataset					Synth2 Dataset				
Greedy	5.32±.09	13.38±.21	5.06±.09	5.42±.09	16.98±.26	7.27±.07	27.92±.20	7.27±.07	7.28±.07	19.03±.15
LBP	15.80±.25	15.80±.25	15.80±.25	15.80±.25	15.80±.25	10.00±.09	10.00±.09	10.00±.09	10.00±.09	10.00±.09
Combine	4.90±.09	4.57±.08	4.53±.08	4.49±.08	4.55±.08	7.90±.07	26.39±.19	7.90±.07	7.90±.07	18.11±.15
Exact	6.36±.11	5.54±.10	5.67±.10	5.59±.10	5.62±.10	7.04±.07	25.71±.19	7.04±.07	7.04±.07	17.80±.15
LProg	6.73±.12	6.41±.11	6.38±.11	6.38±.11	6.38±.11	5.83±.05	6.63±.06	5.83±.05	5.83±.05	6.29±.06

this, leading to relaxed inference yielding many ambiguous labelings.

On the other hand, observe that models trained with the relaxed separation oracle have relatively consistent performance, irrespective of the classification inference procedure; even LBP never shows the catastrophic failure it does with other training approximations and even exact training (e.g., Mediamill, Synth2). Why might this occur? Recall the persistence property from Section 3: unambiguous labels are optimal labels. In some respects this property is attractive, but Section 4 revealed its dark side: relaxation predictors are very conservative, delivering unambiguous labels only when they are *certain*. By making things "obvious" for the relaxed predictors (which are the most conservative w.r.t. what they label), it appears they simultaneously make things obvious for all predictors, explaining the consistent performance of relaxed-trained models regardless of prediction method.

SSVM's ability to train models to "adapt" to the weakness of overgenerating predictors is an interesting complement with Searn structural learning (Daumé III et al., 2006), which trains models to adapt to the weaknesses of undergenerating search based predictors.

5.2.3. Known Approximations

How robust is SSVM training to an increasingly poor approximate separation oracle? To evaluate this, we built an artificial ρ-approximation separation oracle: for example $(\mathbf{x}_i, \mathbf{y}_i)$ we exhaustively find the optimal $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} w^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y})$, but we return the labeling $\hat{\mathbf{y}}$ such that $f(\mathbf{x}, \hat{\mathbf{y}}) \approx \rho f(\mathbf{x}, \mathbf{y}^*)$. In this way, we build an approximate undergenerating MRF inference method with known quality.

Table 4 details these results. The first column indicates the approximation factor used in training each model for each dataset. The remaining columns show train and test performance using exact inference.

What is promising is that test performance does not drop precipitously as we use increasingly worse approximations. For most problems, the performance remains reasonable even for approximation factors as high as 0.1.

6. Conclusion

This paper theoretically and empirically analyzed two classes of methods for training structural SVMs on models where exact inference is intractable. Focusing on completely connected Markov random fields, we explored how greedy search, loopy belief propagation, a linear-programming relaxation, and graph-cuts can be used as approximate separation oracles in structural SVM training. In addition to a theoretical comparison of the resulting algorithms, we empirically compared performance on multi-label classification problems. Relaxation approximations distinguish themselves as preserving key theoretical properties of structural SVMs, as well as learning robust predictive models. Most significantly, structural SVMs appear to train models to avoid relaxed inference methods' tendency to yield fractional, ambiguous solutions.

References

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden Markov support vector machines. *ICML* (pp. 3–10).

Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., & Ng, A. (2005). Discriminative learning of Markov random fields for segmentation of 3D scan data. *CVPR*. IEEE Computer Society.

Table 4. Known ρ-approximations table, showing performance change as we use increasingly inferior separation oracles.

ρ APPROX. FACTOR	SCENE		YEAST		REUTERS		MEDIAMILL		SYNTH1		SYNTH2	
	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST
1.000	4.97	10.06	18.91	20.23	4.30	5.59	17.65	18.75	0.00	6.86	4.57	7.04
0.990	4.36	10.87	19.35	21.06	4.01	5.39	17.19	18.13	0.00	8.61	5.20	7.36
0.975	3.95	11.45	19.27	20.56	3.55	4.99	17.68	18.40	3.64	12.72	4.43	6.76
0.950	9.06	10.72	19.90	20.98	3.97	5.68	18.09	19.66	0.32	6.64	5.35	7.90
0.900	3.96	10.74	18.72	20.14	3.90	5.51	17.10	17.84	2.55	13.19	6.21	8.84
0.850	5.67	11.32	20.04	21.35	3.88	5.21	18.15	19.97	1.45	9.08	6.74	8.57
0.800	5.15	10.59	19.37	21.04	4.93	6.41	19.25	20.86	2.72	14.09	8.83	11.02
0.700	6.32	11.08	24.24	26.26	5.22	6.28	29.24	30.01	0.60	8.69	9.56	11.57
0.600	19.01	20.00	19.00	20.80	4.44	5.44	19.57	20.26	4.21	15.23	12.90	15.48
0.500	10.83	12.28	21.09	22.31	4.65	5.69	29.89	30.42	4.07	10.92	11.85	13.68
0.000	71.80	71.00	45.78	45.36	58.48	58.65	33.00	34.75	36.62	36.84	49.38	50.01

Boros, E., & Hammer, P. L. (2002). Pseudo-boolean optimization. *Discrete Appl. Math.*, *123*, 155–225.

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

Boykov, Y., & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, *26*, 1124–1137.

Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Hierarchical classification: combining Bayes with SVM. *ICML*. Pittsburgh, Pennsylvanias.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM : A library for support vector machines. Software at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. *ACL-EMNLP*.

Culotta, A., Wick, M., & McCallum, A. (2007). First-order probabilistic models for coreference resolution. *NAACL-HLT* (pp. 81–88).

Daumé III, H., Langford, J., & Marcu, D. (2006). Searn in practice. Tech Report.

Elisseeff, A., & Weston, J. (2002). A kernel method for multi-labelled classification. *NIPS*.

Finley, T., & Joachims, T. (2005). Supervised clustering with support vector machines. *ICML*. Bonn, Germany.

Hammer, P. L., Hansen, P., & Simeone, B. (1984). Roofduality, complementation, and persistency in quadratic 0–1 optimization. *Math. Program.*, *28*, 121–155.

He, X., Zemel, R. S., & Carreira-Perpinan, M. A. (2004). Multiscale conditional random fields for image labeling. *cvpr*, *02*, 695–702.

Joachims, T. (2005). A support vector method for multivariate performance measures. *ICML* (pp. 377–384). New York, NY, USA: ACM Press.

Kolmogorov, V., & Rother, C. (2004). Minimizing nonsubmodular functions with graph cuts – a review. *PAMI*, *26*, 147–159.

Kulesza, A., & Pereira, F. (2007). Structured learning with approximate inference. *NIPS*.

Kumar, S., & Hebert, M. (2003). Discriminative fields for modeling spatial dependencies in natural images. *NIPS*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, *5*, 361–397.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Roth, D., & Yih, W. (2005). Integer linear programming inference for conditional random fields. *Proc. of the International Conference on Machine Learning (ICML)*.

Snoek, C. G. M., Worring, M., van Gemert, J. C., Geusebroek, J.-M., & Smeulders, A. W. M. (2006). The challenge problem for automated detection of 101 semantic concepts in multimedia. *ACM-MULTIMEDIA*.

Sutton, C., & McCallum, A. (2005). *Fast, piecewise training for discriminative finite-state and parsing models* (Technical Report IR-403). Center for Intelligent Information Retrieval.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin Markov networks. In *NIPS 16*.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *ICML*.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *JMLR*, *6*, 1453–1484.

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., & Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. *ICML*.

Yu, C.-N., Joachims, T., Elber, R., & Pillardy, J. (2007). Support vector training of protein alignment models. *RECOMB*.