Parameter Learning for Loopy Markov Random Fields with Structural Support Vector Machines

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

Discriminative learners for functions with structured outputs have been successfully applied to sequence prediction, parsing, sequence alignment, and other problems where exact inference is tractable. However, these learners face challenges when inference procedures must use approximations, e.g., for multi-label classification, clustering, image segmentation, or loopy graphical models.

In this paper, we explore methods for training structural SVMs on problems where exact inference is intractable. In particular, we consider pairwise fully connected Markov random fields, using multi-label classification as an example application. We show how to adapt loopy belief propagation, greedy search, and linear programming relaxations as (approximate) separation oracles in the structural SVM cutting-plane training algorithm. In addition to theoretical characterizations, we empirically evaluate and analyze the resulting algorithms on six multi-label classification datasets.

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 problems (e.g., clustering (Finley & Joachims, 2005), multi-label classification, image segmentation, general graphical models) exact inference and the separation oracle cannot be implemented efficiently. One may use approximate inference in this setting. Unfortunately, many of the theoretical guarantees of these methods no longer hold in this case, and relatively little is known about the behavior of structural SVMs using approximations.

This paper explores training structural SVMs on problems where exact inference is intractable. A pairwise fully connected Markov random field 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 relaxations to this problem. In addition to theoretical characterizations, the algorithms are empirically evaluated on multi-label classification datasets.

We find substantial differences between the different approximate training and inference algorithms. Most importantly, LBP is not robust as a separation oracle, often performing worse than a simple greedy method. The linear programming relaxation leads to a modified (approximate) SVM training problem that can be solved exactly in a relaxed output space, which appears to train more robust models than other approaches.

Appearing in the *ICML '2007 Workshop on Constrained Optimization and Structured Output Spaces*, Corvallis, OR, 2007.

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 (M^{3}Ns) (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 parameterizations of the hypothesis. We focus on structural SVMs in this paper.

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 $h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x}, \mathbf{y})$ with discriminant function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$. $\mathbf{\Psi}$ is a combined feature vector function relating inputs and outputs, and $\mathbf{w}$ contains the parameters that are 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 $R^{\Delta}_{\mathcal{S}}(h) = \frac{1}{n}\sum_{i=1}^{n} \Delta(\mathbf{y}_i, h(\mathbf{x}_i))$, SSVMs solve the following quadratic program (QP) for the case of the margin-rescaling hinge loss (Tsochantaridis et al., 2004):

Optimization Problem 1. (Struct. SVM QP)

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

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

Introducing a constraint for every wrong output is often impractical, so Algorithm 1 solves OP 1 by a cutting plane algorithm. It iteratively constructs a sufficient subset $\bigcup_i S_i$ of constraints and only solves the QP 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., 2004; Tsochantaridis et al., 2005):

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

Algorithm 1 Cutting plane algorithm to solve OP 1.

```
 1: Input: $(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_n,\mathbf{y}_n)$, $C$, $\epsilon$
 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 $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y})$
 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
```

Correctness: Algorithm 1 returns a solution of OP 1 accurate to the 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.

A special case of structural SVMs is M^{3}N (Taskar et al., 2003), where, first, $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is constructed from a Markov random field (MRF)

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

with graph structure $G = (V, E)$ and, second, the loss function is restricted to be linearly decomposable in the cliques, i.e., $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k\in\text{cliques}(G)} \delta_k(y_{\{k\}}, \hat{\mathbf{y}}_{\{k\}})$. 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 $h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x}, \mathbf{y})$ 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 differing 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 $\mathbf{y}_u \in \mathbb{B} = \{0, 1\}$), and we allow only $\phi_u(1)$ and $\phi_{uv}(1, 1)$ potentials to be non-zero. This may seem onerously restrictive, but in reality, for any pairwise binary MRF with non-zero $\phi_u(0), \phi_{uv}(0,0), \phi_{uv}(0,1), \phi_{uv}(1,0)$ there is an equivalent MRF where these potentials are zero.

2.2. Approximate Inference in Structural SVMs

To use Algorithm 1 for MRF prediction and training, one must solve two problems for each input example i:

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

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

Prediction is equivalent to MAP inference, and the separation oracle can be reduced to a MAP inference problem over a modified MRF: taking the MRF we would use for solving $\operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} \mathbf{w}^T\mathbf{\Psi}(\mathbf{x},\mathbf{y})$, we include $\Delta(\mathbf{y}_i,\mathbf{y})$ in the argmax by incrementing the node potential $\phi_u(y_u)$ by $1/|V|$ for each wrong value $y_u \neq y_{i,u}$ of variable u, since this wrong assignment increases $\Delta(\mathbf{y}_i,\mathbf{y})$ by $1/|V|$. Thus, we can also express the separation oracle as MAP inference.

Unfortunately, MAP inference is $\#P$-complete for general MRFs. Fortunately, we may use any of a variety of approximate inference methods instead. For prediction and the separation oracle, we explore these approximate inference methods.

Greedy iteratively makes single variable value assignments depending on what assignment most increases network potential.

LBP is loopy belief propagation (Pearl, 1988).

Combine runs both greedy and LBP, and picks the assignment with the highest network potential.

Relaxed relaxes an ILP encoding of MRF inference. Optimization variables within a labeling $\mathbf{y}$ include both existing $y_u \in \mathbb{B}$ values, and pairwise $y_{uv} \in \mathbb{B}$ values indicating if both $y_u = y_v = 1$.

$$\max_{\mathbf{y}} \sum_{u\in\{1..|V|\}} y_u\phi_u + \sum_{u,v\in\{1..|V|\}} y_{uv}\phi_{uv} \quad (4)$$

$$\text{s.t.} \quad \forall u,v. \quad y_u \geq y_{uv}, y_v \geq y_{uv}, y_{uv} \in \mathbb{B} \quad (5)$$

$$y_u + y_v \leq 1 + y_{uv} \qquad \forall u.\ y_u \in \mathbb{B} \quad (6)$$

In implementation, we relax "$\in \mathbb{B}$" to "$\in [0,1]$," which admits fractional solutions.

2.3. Implications of Approximations in SSVMs

How does approximate inference change the performance and the theoretical guarantees of the structural SVM? The trouble is, the theory of SSVMs is based on the existence of an *exact* classifier and separation oracle. When we rely on approximations to perform these tasks it is unclear which properties still hold.

The approximate inference methods listed above can be grouped into two families: undergenerating and overgenerating methods.

Undergenerating methods approximate $\operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}}$ by $\operatorname{argmax}_{\mathbf{y}\in\underline{\mathcal{Y}}}$, where $\underline{\mathcal{Y}} \subseteq \mathcal{Y}$. Greedy, LBP, and Combine methods are all undergenerating. The danger with these is some OP 1 constraints may not be found. What properties still hold for undergenerating approximations? Polynomial time termination still holds, because the proof does not depend upon the quality of the separation oracle. However, neither correctness and empirical risk bound hold, because Algorithm 1 may not find violated constraints present in OP 1.

Overgenerating approximations approximate $\operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}}$ by $\operatorname{argmax}_{\mathbf{y}\in\overline{\mathcal{Y}}}$, where $\overline{\mathcal{Y}} \supseteq \mathcal{Y}$. Relaxed is an overgenerating approximation: it finds an exact argmax, but its search space $\overline{\mathcal{Y}}$ admits fractional solutions. The danger with these is we may introduce constraints not present in OP 1, leading to overconstrained problems. What properties still hold for overgenerating approximations? Polynomial time termination holds, assuming that $\max\|\mathbf{\Psi}\|$ and $\max\|\Delta\|$ remain bounded over the new $\overline{\mathcal{Y}}$. Correctness no longer holds since the *true* solution OP 1 may be infeasible in this overconstrained problem, but the solution will be feasible in OP 1 unlike for undergenerating approximations. The empirical risk bound holds as all constraints in OP 1 are respected.

3. Experiment Setup

Our goal for the following experiments is to evaluate the performance of the different approximate MRF inference methods in SSVM learning and classification. We focus on multi-label classification using pairwise fully connected MRFs, since these problems are small enough to still allow exact inference via exhaustive search as a basis for comparison.

3.1. Multi-label Classification

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

Table 1. Basic statistics for the datasets, including number of labels, training and test set sizes, number of features, and parameter vector **w** size.

Dataset	Labels	Train	Test	Feats.	**w** Size
Scene	6	1211	1196	294	1779
Yeast	14	1500	917	103	1533
Mediamill	10	29415	12168	120	1245
Reuters	10	2916	2914	47236	472405
Synth1	6	1000	10000	6000	36015
Synth2	10	1000	10000	40	445

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 y_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 $\mathbf{w}$.

3.2. Datasets

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 **Mediamill** (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 only if label $i - 1$ is on, and then only with 50% probability, and label 1 is on with 50% probability. 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. With enough training data, this hypothesis can be learned without edge potentials, but exploiting the 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.

3.3. Model Training Details

We want to learn model parameters for each combination of separation oracle and dataset. Structural SVMs have a regularization hyperparameter C, chosen from a population of 14 possible values $\{1 \cdot 10^{-2}, 3 \cdot 10^{-2}, 1 \cdot 10^{-1}, \ldots, 3 \cdot 10^{4}\}$ by 10-fold cross validation on the training set. The "best" C was then used to train a model on all training data. The next section discusses the test-set performance of these models.

4. 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 two methods for doing exact inference in addition to the four approximate methods, as a point of comparison:

Edgeless involves an MRF with no edges, making exact inference trivial at the cost of having no label dependencies. Consider this baseline.

Exact exhaustively searches all labelings. For comparative purposes it is useful to know how we would do if we actually solved OP 1. Note that in order to enable comparisons on the Reuters and Mediamill datasets, we pruned these datasets so only the 10 most frequent labels were present.

The loss of the "edgeless" model is displayed following the dataset name, in turn followed by the loss of a "default" classifier (which always predicts the single labeling that performs best on the training set).

4.1. Edgy vs. Edgeless Models

In all datasets, an edged model always exceeds the performance of the edgeless model. However, do not take this comparison too seriously on Mediamill and Reuters: selecting only the 10 most frequent labels on these sets will tend to degrade "edgy" performance (by robbing it of dependency relationships) and increase "edgeless" performance (since remaining labels are those with the most data to learn node potentials).

4.2. The Sorry State of LBP

Notice that in Yeast, Reuters, Synth1, and Synth2, models trained with LBP have terrible performance

Table 2. Multi-labeling loss on various datasets. Results are grouped by dataset. Rows indicate separation oracle method. Columns indicate classification inference method. The two quantities in the dataset name row are "edgeless" inference (baseline) performance, and "default" performance (i.e., best constant model performance).

	Greedy	LBP	Combine	Exact	Relaxed	Greedy	LBP	Combine	Exact	Relaxed
	Scene Dataset			11.43±0.29	18.10	Mediamill Dataset			18.60±0.14	25.37
Greedy	10.67±0.28	10.74±0.28	10.67±0.28	10.67±0.28	10.67±0.28	23.39±0.16	25.66±0.17	24.32±0.17	24.92±0.17	27.05±0.18
LBP	10.45±0.27	10.54±0.27	10.45±0.27	10.42±0.27	10.49±0.27	22.83±0.16	22.83±0.16	22.83±0.16	22.83±0.16	22.83±0.16
Combine	10.72±0.28	11.78±0.30	10.72±0.28	10.77±0.28	11.20±0.29	19.56±0.14	20.12±0.15	19.72±0.14	19.82±0.14	20.23±0.15
Exact	10.08±0.26	10.33±0.27	10.08±0.26	10.06±0.26	10.20±0.26	19.07±0.14	27.23±0.18	19.08±0.14	18.75±0.14	36.83±0.21
Relaxed	10.55±0.27	10.49±0.27	10.49±0.27	10.49±0.27	10.49±0.27	18.50±0.14	18.26±0.14	18.26±0.14	18.21±0.14	18.29±0.14
	Yeast Dataset			20.91±0.55	25.09	Synth1 Dataset			8.99±0.08	16.34
Greedy	21.62±0.56	21.77±0.56	21.58±0.56	21.62±0.56	24.42±0.61	8.86±0.08	8.86±0.08	8.86±0.08	8.86±0.08	8.86±0.08
LBP	24.32±0.61	24.32±0.61	24.32±0.61	24.32±0.61	24.32±0.61	13.94±0.12	13.94±0.12	13.94±0.12	13.94±0.12	13.94±0.12
Combine	22.33±0.57	37.24±0.77	22.32±0.57	21.82±0.56	42.72±0.81	8.86±0.08	8.86±0.08	8.86±0.08	8.86±0.08	8.86±0.08
Exact	23.38±0.59	21.99±0.57	21.06±0.55	20.23±0.53	45.90±0.82	6.89±0.06	6.86±0.06	6.86±0.06	6.86±0.06	6.86±0.06
Relaxed	20.47±0.54	20.45±0.54	20.47±0.54	20.48±0.54	20.49±0.54	8.94±0.08	8.94±0.08	8.94±0.08	8.94±0.08	8.94±0.08
	Reuters Dataset			4.96±0.09	15.80	Synth2 Dataset			9.80±0.09	10.00
Greedy	5.32±0.09	13.38±0.21	5.06±0.09	5.42±0.09	16.98±0.26	7.27±0.07	27.92±0.20	7.27±0.07	7.28±0.07	19.03±0.15
LBP	15.80±0.25	15.80±0.25	15.80±0.25	15.80±0.25	15.80±0.25	10.00±0.09	10.00±0.09	10.00±0.09	10.00±0.09	10.00±0.09
Combine	4.90±0.09	4.57±0.08	4.53±0.08	4.49±0.08	4.55±0.08	7.90±0.07	26.39±0.19	7.90±0.07	7.90±0.07	18.11±0.15
Exact	6.36±0.11	5.54±0.10	5.67±0.10	5.59±0.10	5.62±0.10	7.04±0.07	25.71±0.19	7.04±0.07	7.04±0.07	17.80±0.15
Relaxed	6.73±0.12	6.41±0.11	6.38±0.11	6.38±0.11	6.38±0.11	5.83±0.05	6.63±0.06	5.83±0.05	5.83±0.05	6.29±0.06

Table 3. Known approximations table, showing performance change as we use increasingly inferior approximations.

Approx.	Scene		Yeast		Reuters		Mediamill		Synth1		Synth2	
Factor	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
0.000	0.0497	0.1006	0.1891	0.2023	0.0430	0.0559	0.1765	0.1875	0.0000	0.0686	0.0457	0.0704
0.010	0.0436	0.1087	0.1935	0.2106	0.0401	0.0539	0.1719	0.1813	0.0000	0.0861	0.0520	0.0736
0.025	0.0395	0.1145	0.1927	0.2056	0.0355	0.0499	0.1768	0.1840	0.0364	0.1272	0.0443	0.0676
0.050	0.0906	0.1072	0.1990	0.2098	0.0397	0.0568	0.1809	0.1966	0.0032	0.0664	0.0535	0.0790
0.100	0.0396	0.1074	0.1872	0.2014	0.0390	0.0551	0.1710	0.1784	0.0255	0.1319	0.0621	0.0884
0.150	0.0567	0.1132	0.2004	0.2135	0.0388	0.0521	0.1815	0.1997	0.0145	0.0908	0.0674	0.0857
0.200	0.0515	0.1059	0.1937	0.2104	0.0493	0.0641	0.1925	0.2086	0.0272	0.1409	0.0883	0.1102
0.300	0.0632	0.1108	0.2424	0.2626	0.0522	0.0628	0.2924	0.3001	0.0060	0.0869	0.0956	0.1157
0.400	0.1901	0.2000	0.1900	0.2080	0.0444	0.0544	0.1957	0.2026	0.0421	0.1523	0.1290	0.1548
0.500	0.1083	0.1228	0.2109	0.2231	0.0465	0.0569	0.2989	0.3042	0.0407	0.1092	0.1185	0.1368
1.000	0.7180	0.7100	0.4578	0.4536	0.5848	0.5865	0.3300	0.3475	0.3662	0.3684	0.4938	0.5001

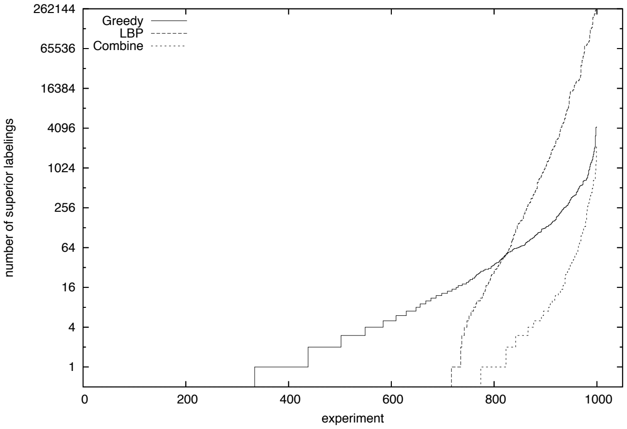

Figure 1. Inference on 1000 random 18 label problems.

even compared to a method as trivial as greedy, and on all datasets often yield the least performance as a classifier. Why might the LBP separation oracle and classifier perform so poorly?

In one small experiment, we generated 1000 random MRF problems, and ran Greedy, LBP, and Combine inference on them. Then, for each result, we exhaustively counted how many labelings with higher network potential exist. Figure 1 shows the resulting curve. The lower the curve, the better the inference method. Though LBP finds "perfect" labelings more often than Greedy, it also tends to fall into horrible local minima which skew the learner and classifier.

4.3. Relaxation in Learning and Prediction

First, compare the relaxed training and inference to using the exact method for training and inference. Both methods differ only in the presence of the fractional constraints. The relaxed training appears competitive with exact training: exact outperforms relaxed training on four datasets, but loses on two datasets. Furthermore, the performance difference between relaxed and exact is never very large.

Interestingly, however, relaxation used *in prediction* with models trained with non-relaxed methods (i.e., models that do not constrain fractional solutions) often performs quite poorly. The most ludicrous examples appear in Yeast, Reuters, Mediamill, and Synth 2.

On the other hand, models trained with the relaxed separation oracle have relatively consistent and robust performance, irrespective of the classification inference procedure. Most surprisingly, when using relaxed training, LBP never shows catastropic failures as it does even when using exact training (e.g. Mediamill, Synth2). The reason for this effect is uncertain and subject to future works. Perhaps the parameterization resulting from fractional constraints somehow "smoothes" the search space, but this is speculation.

4.4. Known Approximations

How robust is SSVM training to increasingly poor approximate separation oracle methods? To evaluate this, we built an "artificial" approximate separation oracle with a known approximation factor: given an approximation factor α, for example $\mathbf{x}$ we can find the optimal $\mathbf{y}^* = \text{argmax}_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y})$ with the exact classifier, and then run another exhaustive search of possible labelings to find the labeling $\hat{\mathbf{y}}$ such that $f(\mathbf{x},\hat{\mathbf{y}}) \approx (1-\alpha)f(\mathbf{x},\mathbf{y}^*)$. In this way, we build an approximate undergenerating MRF inference method with known quality.

Table 3 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 remaining reasonable even for approximation factors as high as 0.1.

5. Conclusion

We explored the performance of approximate inference and separation oracle methods in structural SVMs for learning parameters in completely connected Markov random fields. Methods were based on greedy search, loopy belief propagation, and a linear programming relaxation. In addition to a theoretical comparison, we also empirically compared performance on a multi-label classification problem. The relaxation approximation distinguished itself as preserving key theoretical properties of structural SVMs, as well as learning robust predictive models.

Acknowledgments

This work was supported under NSF Award IIS-0412894 and through a gift from Yahoo! Inc.

References

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

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

Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Hierarchical classification: combining Bayes with SVM. *ICML* (pp. 177–184). New York, NY, USA: ACM Press.

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* (pp. 1–8). Morristown, NJ, USA: Association for Computational Linguistics.

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* (pp. 217–224). New York, NY, USA: ACM Press.

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

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML* (pp. 282–289). Morgan Kaufmann, San Francisco, CA.

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.

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* (pp. 421–430). New York, NY, USA: ACM Press.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. In *NIPS 16*. Cambridge, MA: MIT Press.

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.

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