Spanning Tree Approximations for Conditional Random Fields

Patrick Pletscher
Department of Computer Science
ETH Zurich, Switzerland
pletscher@inf.ethz.ch

Cheng Soon Ong
Department of Computer Science
ETH Zurich, Switzerland
chengsoon.ong@inf.ethz.ch

Joachim M. Buhmann
Department of Computer Science
ETH Zurich, Switzerland
jbuhmann@inf.ethz.ch

Abstract

In this work we show that one can train Conditional Random Fields of intractable graphs effectively and efficiently by considering a mixture of random spanning trees of the underlying graphical model. Furthermore, we show how a maximum-likelihood estimator of such a training objective can subsequently be used for prediction on the full graph. We present experimental results which improve on the state-of-the-art. Additionally, the training objective is less sensitive to the regularization than pseudo-likelihood based training approaches. We perform the experimental validation on two classes of data sets where structure is important: image denoising and multilabel classification.

1 Introduction

In many applications of machine learning one is interested in a segmentation of the data into its subparts. Two areas where this is of immense interest include computer vision and natural language processing (NLP). In computer vision one is interested in a full segmentation of the images into for example foreground and background or a more sophisticated labeling including semantic labels such as 'car' or 'person'. In NLP one may be interested in part-of-speech tagging. Usually in these applications an annotated and fully segmented set of training data is provided on which a classifier is trained. While it is possible to train a classifier for each atomic part independently, one expects improved accuracy by assuming an underlying structure arising from the application. This structure might correspond to a chain or parse tree for NLP, a grid for computer vision or a complete graph for multilabel classification.

While probabilistic inference for simple structures like a chain or a tree are computationally tractable, more complex graphs like grids or complete graphs, which include loops, are computationally intractable. A discriminative model that has gained large popularity recently and which allows for a principled integration of the structure, the data and the labels, is the Conditional Random Field (CRF) [Lafferty et al., 2001]. In this work we consider the training of CRFs for intractable graphs. We here follow the notation of [Sutton and McCallum, 2006] and assume a factor graph G where $\mathcal{C} = \{C_1, \dots, C_p, \dots, C_P\}$ are the set of factors that are parametrized in the same way; sometimes referred to as clique templates. Each of the clique templates C_p consists of a subset of factors of G. Given the observation x, the posterior of a complete segmentation y has the form:

$$P(y|x;\theta) = \frac{1}{Z(x;\theta)} \exp\Bigg(\sum_{C_p \in \mathcal{C}} \sum_{c \in C_p} \langle \theta_p, s_c(x, y_c) \rangle\Bigg). \tag{1}$$

θ summarizes all the parameters of the model. y_c is a subset of the segmentation that only involves the variables of factor c. The individual variables y_i take values from a finite set $\mathcal{Y}$. $s_c(x, y_c)$ denote the sufficient statistics for factor c and $Z(x, \theta)$ denotes the partition sum, which normalizes the distribution:

$$Z(x;\theta) = \sum_y \exp\Bigg(\sum_{C_p \in \mathcal{C}} \sum_{c \in C_p} \langle \theta_p, s_c(x, y_c) \rangle\Bigg).$$

Alternatively, the distribution can be written as an exponential family distribution.

$$P(y|x;\theta) = \exp(\langle \theta, s(x, y) \rangle - A(x;\theta)).$$

Here, the individual sufficient statistics of factors belonging to the same clique template are summed up

Appearing in Proceedings of the 12^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5.

and subsequently concatenated into one big vector $s(x, y)$ (in the corresponding order as in θ). $A(x; \theta)$ denotes the log partition sum.

In this present work we only consider CRFs of factors that consist of at most two label variables, i.e., a setting where a factor can also be represented as an edge in an undirected graphical model.

Training the CRF consists of finding the maximum-a-posteriori estimate for examples $\{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$ and a Gaussian prior:

$$\widehat{\theta} = \arg\max_{\theta} \prod_{n=1}^{N} P(y^{(n)}|x^{(n)}; \theta) \exp(-\lambda\|\theta\|_2^2).$$

Taking the negative logarithm we obtain a convex optimization problem, a property of the exponential family distribution. In theory we are thus guaranteed of finding a global minima by following the gradient. However, computing the data log-likelihood itself or the exact gradient of the objective w.r.t. the parameter θ are intractable for general graphs with loops, since both computations include computing the partition sum $Z(x; \theta)$. This makes the learning at least as hard as probabilistic inference, which is computationally intractable in general.

2 Related Work

An established approach is to train the model parameters for the pseudo-likelihood [Besag, 1975, Kumar and Hebert, 2006]. The pseudo-likelihood is a tractable approximation of the true likelihood. It models the unobserved variables independently conditioned on the true label of the Markov blanket. One problem often encountered with pseudo-likelihood is the overestimation of the edge parameters [Kumar and Hebert, 2006]. A heuristic used to relax this problem is by regularizing only the edge parameters, and not the node parameters, this is then called the penalized pseudo-likelihood. An alternative approximate objective is piecewise training [Sutton and McCallum, 2005]. In piecewise training the parameters are trained over each clique individually and in a second step these parameters are combined to give a global model. Piecewise-training has been shown to outperform pseudo-likelihood. However, we are unaware of any comparison to penalized pseudo-likelihood, which in our tests has performed much better than the standard pseudo-likelihood.

Another approach is to use an approximate inference algorithm, such as Loopy Belief Propagation [Yedidia et al., 2005] in combination with the Bethe free energy, to compute an approximation of the data log-likelihood (and possibly the gradient), see e.g. [Vishwanathan et al., 2006]. One problem here is that the learning might get trapped in poor local minima caused by the approximate inference. This behaviour has been studied in [Kulesza and Pereira, 2008], where it was shown that even for relatively good approximation guarantees of the inference algorithm, one might end up with poor parameters. Recently there was also a training proposed in which high-probability labelings are generated by an approximate MAP labeling algorithm. The objective of the training is then to find parameters that separate these high-probability labelings from the ground-truth labeling well [Szummer et al., 2008]. However, this approach optimizes a maximum margin objective rather than trying to find a maximum likelihood estimator. There also have been results about approximate training of structural SVMs in [Finley and Joachims, 2008], where it is shown that one can give approximation guarantees about the learning, given an approximation guarantee for the inference. The results presented there are about the max-margin objective and do not directly apply to the maximum-likelihood setting.

In this paper, we propose a novel approach that formulates an alternative training objective that is tractable, similar to the pseudo-likelihood. We use spanning trees to approximate the intractable graph. It is more promising since the training step considers globally normalized probability distributions on tractable subgraphs, instead of restricting to locally (per unobserved variable) normalized distributions as in pseudo-likelihood.

The use of tractable subgraphs for probabilistic inference in loopy graphs has been pioneered by Wainwright et al. [2003]. While this work gave important insights into approximations for probabilistic inference, it assumed that the model is given and no parameter learning is needed. Our work deals with the question of how to use tractable subgraphs in training CRFs. The rationale behind our training objective is that while we might constrain the model through ignoring certain dependencies in the data, at least we are not trapped in poor local minima created by approximations of the data log-likelihood during learning. In our approximate objective the effect of ignoring dependencies in the data should be mitigated by randomization and averaging over multiple trees.

Alternating iterative approaches where in one step the mixture weights over the different tractable subgraphs are learned (as in Wainwright's work) and in a second step the model parameters for the given mixture weights are learned, might be beneficial. However, as with other approximate probabilistic inference approaches for parameter estimation, one would have to come up with a strategy for identifying local minima

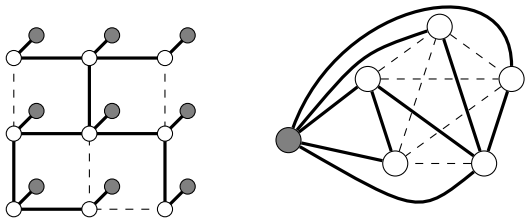

Figure 1: Examples of intractable graphs. Left: Grid graph often encountered in computer vision. Right: Complete graph in multilabel classification. For simplicity we do not show a factor graph, but an undirected graphical model, where the edges represent factors. The underlying graph is shown by dashed edges. We superimpose two examples of maximal loop-free coverings of the factors by a spanning tree. Note that loops "going through" observed nodes are allowed, as this does not pose any computational problems.

created by the approximation. Our work is meant to explore what can be achieved by a simple uniform mixture of tractable subgraphs.

3 Training a CRF by a mixture of spanning trees

Our approximate training objective considers random spanning trees of the intractable graph. For the spanning tree we can compute all the required quantities exactly. The underlying assumption being that a random spanning tree still captures some of the important dependencies in the data. We thus propose the alternative (regularized) training objective:

$$\widehat{\theta}^{SP} = \arg\max_{\theta} \left(\prod_{n=1}^{N} \sum_{t \in \mathrm{sp}(G^{(n)})} P(t, y^{(n)} | x^{(n)}; \theta) \right) \times \exp(-\lambda \|\theta\|_2^2). \quad (2)$$

with

$$P(t, y^{(n)} | x^{(n)}; \theta) = P(y^{(n)} | x^{(n)}, t; \theta) P(t).$$

Here $\mathrm{sp}(G)$ denotes the set of all maximal coverings of the factors of G, such that no loops exist between unobserved variables. For factors of at most two unobserved variables this corresponds to the set of all spanning trees (on the unobserved variables) in an undirected graphical model. $P(y|x,t;\theta)$ is the same distribution given in (1), restricted to only the factors of spanning tree t

$$P(y|x,t;\theta) = \frac{1}{Z(x,t;\theta)} \exp\left(\sum_{C_p \in \mathcal{C}} \sum_{c \in C_p^t} \langle \theta_p, s_c(x, y_c) \rangle \right),$$

where $c \in C_p^t$ is a shorthand for $c \in C_p \wedge c \in t$. The partition sum $Z(x,t;\theta)$ must also be adapted correspondingly. Again we can write this distribution in an exponential family form:

$$P(y|x,t;\theta) = \exp(\langle \theta, s(x,y,t) \rangle - A(x,t;\theta)).$$

Here $s(x,y,t)$ is the sufficient statistics that we get by only summing up factors that are included in the spanning tree t. The objective in (2) can be motivated as follows: We consider a generative model for the labels in which first a random spanning tree of the undirected model is drawn. In a second step the labels of the tree are sampled according to a Gibbs distribution. In this setting the chosen tree is treated as a hidden variable. There are two key points to stress: First, $P(y|x,t;\theta)$ for a given spanning tree t can be trained efficiently and exactly as probabilistic inference for a tree is tractable. Second, the estimate of $\widehat{\theta}^{SP}$ is not consistent for intractable graphs and will not converge to $\widehat{\theta}$ for such cases. The two training objectives consider two different problems and the sufficient statistics $s(x,y)$ and $s(x,y,t)$ differ to a large extent in densely connected graphs. Nevertheless, we can construct from $\widehat{\theta}^{SP}$ an estimate of $\widehat{\theta}$, this is described in subsection 3.1. This then allows us to perform prediction on the full graph with approximate probabilistic inference, such as Loopy Belief Propagation. Two examples of maximal loop-free coverings are given in Figure 1.

So far we have converted an intractable problem into another hard problem, since many interesting problems have exponentially many spanning trees over which we would need to marginalize. Nevertheless, in theory it is possible to sum over all spanning trees in running time cubic in the number of variables by the Matrix-Tree Theorem [Tutte, 1984]. For many practical applications a cubic running time is too slow and we thus decided to sample a small number of spanning trees $\mathcal{T} \subseteq \mathrm{sp}(G)$ uniformly at random by the algorithm described in [Wilson, 1996]. In our implementation we only sample the spanning trees once at the beginning of the training and then keep them fixed. We sample the spanning trees for each training example independently. However, alternative approaches where e.g., a spanning tree is sampled anew at each iteration of the optimization are also possible. Ignoring the sampling of the spanning trees (of which the expected running time is proportional to the mean hitting time), the running time of one optimization step is $\mathcal{O}(|\mathcal{T}|m|\mathcal{Y}|^2)$, where m denotes the number of variables.

To learn the parameters we use BFGS, a quasi-Newton method. For this we also need the derivative of the log-likelihood w.r.t. the parameters. For only one sample

(x, y) and without regularizer it is given by

$$\begin{aligned}\frac{\partial \ell(y|x,\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} \log\left(\frac{\sum_t P(t)\exp(\langle\theta, s(x,y,t)\rangle)}{\sum_{t,y'} P(t)\exp(\langle\theta, s(x,y',t)\rangle)}\right) \\ &= \sum_t P(t|y,x;\theta)s(x,y,t) \\ &\quad - \sum_{y',t} P(t,y'|x;\theta)s(x,y',t)\end{aligned}$$

The gradient has a similar form as in hidden CRFs [Quattoni et al., 2007] and can be reformulated as a combination of the features, weighted by the marginals. All of the quantities involved can be computed efficiently. The hidden variable in our framework is the spanning tree along which we are working. We are unable to prove convexity for (2), but we observed empirically that in the experimental results we always converged to the same solution for different initializations.

There are two advantages of the objective in (2) when compared to pseudo-likelihood. First, it can capture longer-range dependencies between variables than only between neighboring variables. Second, it is less sensitive to overestimation of the edge parameters, as the objective does indeed model a joint distribution of all the labels, whereas pseudo-likelihood models them independently conditioned on the true label of the neighboring variables. One disadvantage might arise from dropping certain dependencies in training, which can be prevented to a certain extent by dropping the dependencies at random.

3.1 MAP inference

Once we have trained the parameters, we also want to predict segmentations on unseen data. For this we usually look for the maximum-a-posterior (MAP) labeling, which is also computationally intractable for general graphs. Here we answer the question of how to use the parameters learned with the objective in (2) to obtain a MAP labeling. Computing the MAP for one spanning tree is easy, but generally we would want to consider the whole graph and not just one of its spanning trees. We experimented with two approaches for inference, each with different strengths and weaknesses.

Voting Compute MAP labelings for several random spanning trees $t \in \mathcal{T}$ and finally for each variable take a majority vote over the different MAP solutions we get. Formally we can express the label y_i^* for a node i as follows:

$$y_i^* = \arg\max_{y_i} \sum_{t\in\mathcal{T}} \delta(y_i^{\text{MAP}}(t) = y_i)$$

with

$$y^{\text{MAP}}(t) := \arg\max_y P(y|x,t;\widehat{\theta}^{SP}).$$

Rescaling Rescale the parameters $\widehat{\theta}^{SP}$ according to the connectivity in the spanning trees relative to the full graph. In other words we adjust the parameters $\widehat{\theta}^{SP}$ such that the inner product with the full sufficient statistics leads to similar values as observed in the training phase. We have

$$y^* = \arg\max_y P(y|x;\theta')$$

with

$$\theta_p^{'} = \frac{\sum_{t\in\mathcal{T}} P(t)|C_p^t|}{|C_p|}\widehat{\theta}_p^{SP},$$

and we get θ' by concatenation of the individual $\theta_p^{'}$. Here we assume that the underlying factor graphs for the individual samples are the same, otherwise we also need to average over the set sizes of the samples. To compute $\arg\max_y P(y|x;\theta')$ we can use approximate inference algorithms like Loopy Belief Propagation or graph-cut. Instead of computing a MAP, we can also use maximum-marginal inference with parameters θ'.

We also experimented with a voting strategy based on the marginals instead of the MAP, which however led to similar results. We expect the voting strategy to work well in settings where max-marginal inference works better than MAP, which we confirmed experimentally. A more sophisticated algorithm for the MAP inference could also be developed, by considering the MAP over a subsample of spanning trees as discussed in [Meila and Jordan, 2000].

3.2 Grid Graphs and Complete Graphs

In our experiments we will consider two instances of the general model given in (1). The first model is often used in computer vision and considers a grid graph where all the vertex and edge potentials are parametrized in the same way denoted by θ_v and θ_e, respectively:

$$\begin{aligned}P(y|x;\theta) = \frac{1}{Z(x;\theta)}\exp\Bigg(&\sum_{i\in V}\langle\theta_v, s_i(x,y_i)\rangle \\ &+ \sum_{\substack{(i,j)\in E \\ i<j}}\langle\theta_e, s_{ij}(x,y_i,y_j)\rangle\Bigg).\end{aligned}$$

The undirected graphical model is given as $G = (V, E)$ and the sufficient statistics in the model above are features extracted from possibly overlapping image patches. Another specific problem we will consider is multilabel classification, which can be modeled as

follows:

$$P(y|x;\theta) = \frac{1}{Z(x;\theta)} \exp\Bigg(\sum_{i\in V} \langle \theta_i, s_i(x,y_i)\rangle + \sum_{\substack{(i,j)\in E \\ i<j}} \langle \theta_{ij}, s_{ij}(x,y_i,y_j)\rangle\Bigg).$$

In this model each label-label co-occurrence is modelled by a parameter θ_{ij} and each class i is parametrized by θ_i. The graph is given by the complete graph and $y_i \in \{0,1\}$ describes whether a data point has label i. In the simplest case, s_{ij} ignores x and is 1 for all configurations of y_i, y_j.

3.3 Informative Spanning Trees

For the case when we have a grid graph with common parameters, the linearity of the dot product allows us to analyze the log likelihood $\ell(y|x;\theta)$ in more detail. For a particular tree $t \in \mathrm{sp}(G)$ the exact model and the spanning tree model have the same terms corresponding to the vertices. Hence, the difference in log likelihood is given by

$$\ell(y|x;\theta) - \ell(y|x,t;\theta) = \left\langle \theta_e, \sum_{\substack{(i,j)\in E\setminus t \\ i<j}} s_{ij}(x,y_i,y_j) \right\rangle - A(x;\theta) + A(x,t;\theta).$$

Observe that the sum in the expression above is over the edges which are *not* contained in the spanning tree.

This means that we do not lose any information by the spanning tree approximation when the feature vector $s_{ij}(x,y_i,y_j) = 0$ for $(i,j) \in E\setminus t$. In models with categorical observations and the edge features as the absolute difference (for example as in the Ising model), this corresponds to the case when neighboring observations are equal. Of course we do not expect this to be true in general, but this indicates possible further improvements in the spanning tree approximation by choosing spanning trees that pass through regions with high edge features.

4 Evaluation

We evaluate our algorithm on a number of synthetic and real world data sets to gain better insights into the spanning tree approximation. In some of the experiments we show the classification error for different regularization parameters λ. We point out that in the penalized pseudo-likelihood the regularization is different than in the other methods, as only edge parameters are regularized. In reality we have the regularizer $\lambda\|\beta \odot \theta\|_2^2$, where β denotes a 0/1 vector of the same dimension as θ, with 0 for all node parameters and 1 otherwise; and $\odot$ denotes the elementwise product. Hence the hyperparameter λ for which the smallest test error is obtained by penalized pseudo-likelihood does not in general correspond to the hyperparameter minimizing the test error for the spanning tree training.

4.1 Comparison to exact training on toy data

In this experiment we compare our training method to the gold-standard which is the exact training. For the exact training all computations are performed exactly (up to numerical errors) and absolutely no approximations are made. Obviously, exact training is feasible only for very small grids, in our case, we consider a 3×3 grid with a fixed labeling (lower right 2×2 corner label 2, otherwise label 1). As features we added Gaussian noise (for varying variance σ^2) to the labels. In the comparison we also include the results from training the penalized pseudo-likelihood. We evaluate the training approaches based on exact MAP prediction by exhaustive enumeration.

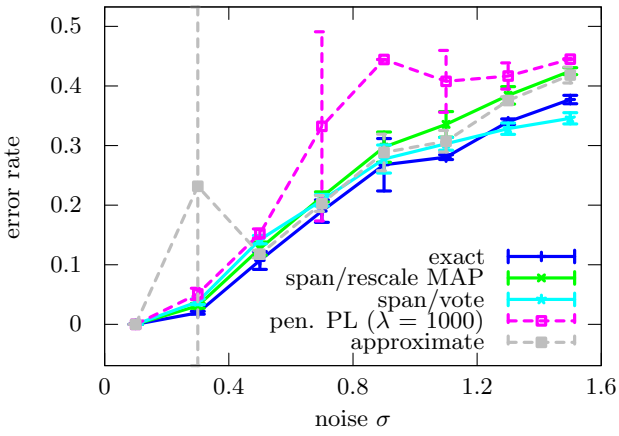

Figure 2: Test error for different training objectives on the 3×3 grid with fixed labeling.

In Figure 2, we observe as expected that exact training is superior when compared to penalized pseudo-likelihood. However, we had to heavily penalize the edge weights for the penalized pseudo-likelihood: with $\lambda = 1000$ on the edges and no regularization of the node parameters. Decreasing λ for the penalized pseudo-likelihood resulted in worse performance for moderate to high noise levels. Also, regularizing both, the edge and node parameters, i.e. a L_2 regularizer on θ, as done in standard pseudo-likelihood, proved detrimental. For exact training and the spanning tree training we used a regularizer of $\lambda = 0.1$, but the results were comparable over two orders of magnitude of

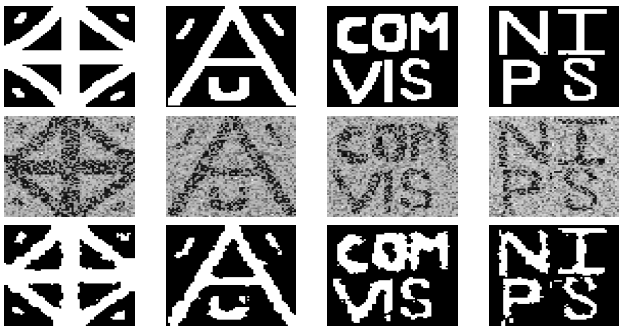

Figure 3: Overview of the binary image denoising data set. First row: original images, second row: images corrupted by bimodal noise, third row: labeling as inferred by our algorithm.

λ. Comparing the accuracy of exact training and the spanning tree approximation shows a difference of usually around 2-3%. However, for high values of noise, the spanning tree learning combined with voting inference seem to perform favorably. One possible explanation for this is that maximum marginal prediction often shows better results than MAP. Approximate training with LBP resulted in similar performance as the spanning tree approaches, in some runs the estimated parameters were however substantially inferior, leading to a large variance.

4.2 Binary image denoising task

We consider the binary image denoising dataset of Kumar and Hebert [2006] shown in Figure 3. It includes two observation sets that are perturbed by unimodal and bimodal Gaussian noise, respectively. We would like to point out that we use a slightly different model than Kumar and Hebert [2006], as they consider an objective where the node potentials are already the log class probabilities of a logistic regression classifier. In our model we do not normalize the node potentials.

Since the training set was fairly small, we could not sensibly define a validation set, and hence used the training set to select λ. Our results are summarized in Table 1. Observe that we perform slightly worse in the unimodal dataset but significantly better in the bimodal dataset.

We investigated the effect of the number of spanning trees sampled for each example. For training, we did not see any improvement by using more than one spanning tree. This can be explained by observing that for an image grid most of the configurations of labels/features are already present in a single spanning tree. However, for inference by voting, increasing the number of spanning trees improves the performance (Figure 4). This is as expected since the coverage of the edges increases with the number of spanning trees and hence we obtain a better estimate.

Table 1: Pixelwise classification error (%). Comparison to published work, where KH refers to the results published in [Kumar and Hebert, 2006, Kumar et al., 2005]. There for the unimodal dataset MAP inference was used, for the bimodal dataset maximum-marginal (MM) inference was used. We average over 5 different runs, where the training/test set is fixed and the spanning trees are resampled anew for each run.

		unimodal	bimodal
spanning	voting	2.63 ± 0.01	$\mathbf{5.23 \pm 0.01}$
	rescale/MAP	2.33 ± 0.01	5.69 ± 0.03
	rescale/MM	2.53 ± 0.01	5.26 ± 0.01
KH		**2.30**	5.48

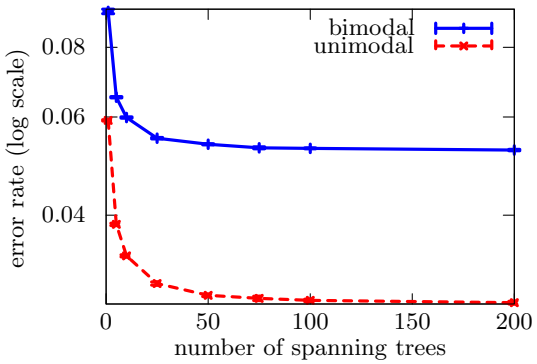

Figure 4: Test error on the binary image denoising datasets as a function of the number of spanning trees used for the inference by voting.

Similar to the toy problem, for varying noise level, the rescale and voting inference approaches show different merits. On the unimodal dataset the rescaling approach performs better, whereas on the bimodal data set the trend is inverted and voting performs better (Table 1). If we use the rescale approach together with max-marginal inference, we get similar test error as the voting approach.

Figure 5 shows the train and test error for different values of λ. We observe that the spanning tree objective is more insensitive to the regularizer and performs well over a wide range of regularizer parameters λ, which is not observed for penalized pseudo-likelihood training, where sensitivity to λ is high.

4.3 Multiclass image denoising

While the experiment in the previous subsection is important as a comparison to previously published results, the experiment does not show the full potential

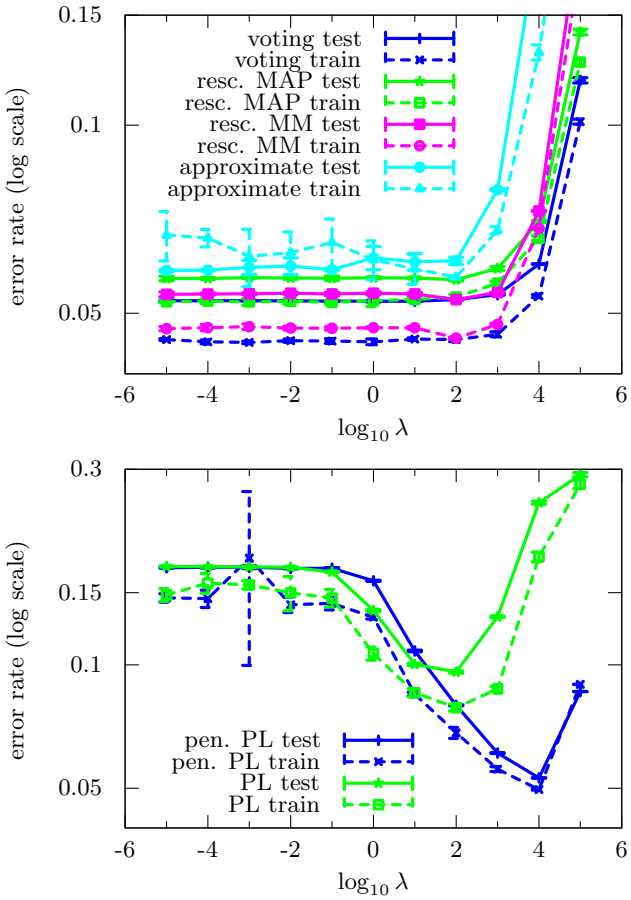

Figure 5: Influence of the hyperparamter on the bimodal image denoising dataset. Top: spanning tree and approximate training. Bottom: pseudo-likelihood training. The approximate training shows a similar behaviour w.r.t. the regularizer as the spanning tree objectives, the variance and error are however larger.

of our models. We are ultimately interested in more complex scenarios where the number of states per variable is bigger than two. It is also important to study a scenario where certain labels only interact with subsets of other labels. We created a novel data set similar in spirit to the one in [Kumar and Hebert, 2006] with the extensions mentioned above. We created images of chess boards with two different player figures, each player occupying half of his fields. The positions of the figures are sampled randomly. The RGB colors of the different labels involved (two fields, two players) are selected such that they are equidistant in the RGB space. We then add Gaussian noise to the colors for varying noise levels.

Our spanning tree based training approach successfully identifies the dependencies between the labels and does not assign a player's label to an opponent's

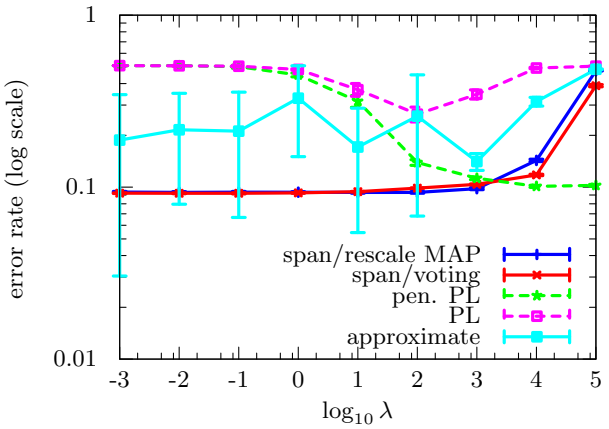

Figure 6: Test error on the chess board data set for a noise level of 1.

field even for high levels of noise. In the experiments we observed that penalized pseudo-likelihood performs slightly better for small noise levels, the difference is however not statistically significant. For higher noise, the spanning tree approximation performs better ($9.3\% \pm 0.1$ test error vs. $10.1\% \pm 0.1$), see Figure 6. We also included results of approximate training where the log-partition sum is approximated by the Bethe free energy computed by Loopy Belief Propagation, which showed inferior performance. Observe that the spanning tree training is again less sensitive to changes in hyperparameter λ.

4.4 Multilabel problem: Yeast dataset

Here we consider a different graph structure. We use the multilabel yeast data set containing 14 labels from Elisseeff and Weston [2001]. The underlying graphical model (a complete graph) as well as the prediction step ($\arg\max_y \langle \theta, s(x,y) \rangle$) of our model are the same as in [Finley and Joachims, 2008]. The key difference is that we train a CRF by maximum-likelihood whereas Finley and Joachims [2008] train a structural SVM by a maximum-margin objective.

We performed 5-fold cross-validation on the training data (where the model was subsequently trained on the full training data). We obtained a test error of $19.98\% \pm 0.06$, which is slightly superior (but not statistically significant) than the best published result of $20.23 \pm 0.53\%$ [Finley and Joachims, 2008] where exact training was used. For the penalized pseudo-likelihood we got an unsatisfactory test error of $21.27\% \pm 0.03$. We show in Figure 7 the performance for different settings of λ. We used 3 spanning trees per example in the training. However, we could again not find statistically significant evidence that training on more than one spanning tree per example helps. For a complete

graph of size K we have $K(K-1)/2$ edges of which $K-1$ are contained in one spanning tree. The probability of missing an edge in an independently sampled spanning tree t is thus given by

$$P((i,j) \notin t) = \left(1 - \frac{2}{K}\right).$$

The probability of missing an edge in N independent spanning trees decreases exponentially in N. For the yeast data set we have $K = 14$ labels and 1500 training examples. If we independently sample one spanning tree for each example, we have $N = 1500$ and the probability of missing an edge is less than 10^{-100}.

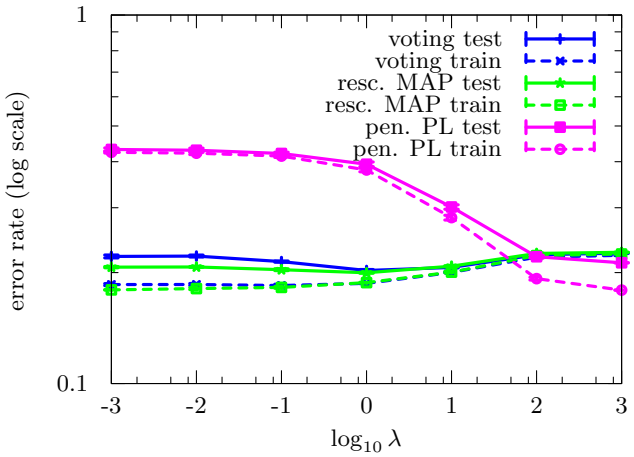

Figure 7: Test error on the yeast multilabel data set for different regularizers.

5 Conclusion

We presented an alternative approximate training objective for CRFs based on mixtures of spanning trees. Similar in spirit to pseudo-likelihood we can train this objective exactly. The key difference to approaches such as pseudo-likelihood and piecewise training is that the distribution is globally normalized over all variables, with some dependencies removed, instead of normalizing over a small set of variables (possibly conditioned on the Markov blanket). We have demonstrated that the spanning tree approximation can be trained efficiently and it shows state-of-the-art prediction performance. Furthermore, we have found it to be less sensitive to the hyperparameter, which is not the case for pseudo-likelihood and its penalized version.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback which helped improve the paper. We thank Yvonne Moh for proof-reading an early version of this work. This work was supported in parts by the Swiss National Science Foundation (SNF) under grant number 200021-117946.

References

J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:3:179–195, 1975.

A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *NIPS*, pages 681–687, 2001.

T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *ICML*, 2008.

A. Kulesza and F. Pereira. Structured learning with approximate inference. In *NIPS*, 2008.

S. Kumar and M. Hebert. Discriminative Random Fields. *IJCV*, 68(2):179–201, 2006.

S. Kumar, J. August, and M. Hebert. Exploiting inference for approximate parameter learning in discriminative fields: An empirical study. In *EMMCVPR*, 2005.

J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, 2001.

M. Meila and M. I. Jordan. Learning with mixtures of trees. *JMLR*, 1:1–48, 2000.

A. Quattoni, S. Wang, L. P. Morency, M. Collins, and T. J. Darrell. Hidden-State Conditional Random Fields. *IEEE PAMI*, 29(10):1848–1852, 2007.

C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*. MIT Press, 2006.

C. Sutton and A. McCallum. Piecewise training for undirected models. In *UAI*, 2005.

M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *ECCV*, 2008.

W. T. Tutte. *Graph Theory*. Addison-Wesley, 1984.

S. Vishwanathan, N. Schraudolph, M. Schmidt, and K. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML*, pages 969–976, 2006.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE TIT*, 49(5):1120–1146, 2003.

D. Wilson. Generating Random Spanning Trees More Quickly Than the Cover Time. In *STOC*, 1996.

J. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE TIT*, 51:2282–2312, 2005.