Fast and Accurate Prediction via Evidence-Specific MRF Structure

Veselin Stoyanov VES@CS.JHU.EDU
Human Language Technology Center of Excellence, Johns Hopkins University, Baltimore, MD

Jason Eisner JASON@CS.JHU.EDU
Department of Computer Science, Johns Hopkins University, Baltimore, MD

Abstract

We are interested in speeding up approximate inference in Markov Random Fields (MRFs). We present a new method that uses *gates*—binary random variables that determine which factors of the MRF to use. Which gates are open depends on the observed evidence; when many gates are closed, the MRF takes on a sparser and faster structure that omits "unnecessary" factors. We train parameters that control the gates, jointly with the ordinary MRF parameters, in order to locally minimize an objective that combines loss and runtime.

1. Introduction

Markov Random Fields (MRFs) and Conditional Random Fields (CRFs) have been used successfully for structured prediction (Lafferty et al., 2001; Sha & Pereira, 2003; Peng & McCallum, 2006; Stoyanov & Eisner, 2012). Such graphical models support predictive inference by defining conditional distributions $P(Y|X)$, where X is a set of observed "input" variables, and Y is a set of unobserved "output" variables.

MRFs are traditionally used to model a joint distribution over all variables, so that X and Y may vary from example to example. In a CRF, the set X is traditionally fixed, and one trains discriminatively to maximize the conditional likelihood $p(Y|X)$ or to minimize the loss of predicted Y values. Fixing X determines *which* factors are computationally efficient to include, and *how* to train them. In this paper, we aim to combine the flexibility of the MRF approach with the improved tractability and predictive accuracy of the CRF approach, by dynamically choosing which factors to use for *each* given X, and training all factors accordingly.

The runtime of exact inference in an MRF or CRF is exponential in the tree-width of the factor graph on the unobserved variables. Including a variety of factors to capture known or suspected dependencies among the variables may increase this tree-width, necessitating *approximate* inference. Previous work has shown that this can be a worthwhile tradeoff (Sutton & McCallum, 2005; Finley & Joachims, 2008; Stoyanov & Eisner, 2012). However, even approximate inference can be slow when the factor graph is dense. For example, if one includes a binary factor between every pair of random variables (Stoyanov & Eisner, 2012), an approximate inference method such as loopy belief propagation (BP) will take $O(n^2)$ time per iteration.

We propose to speed up approximate inference by approximating further. We can learn to make do with only a subset of the factors—but a different subset on each example. In our present study, this subset will be kept fixed across all iterations on a given example. So each iteration updates all BP messages, and these updates are computed exactly and at full granularity.

We will train the model parameters (the factors) to work as well as possible with this approximation. As we have previously argued, when a model may have incorrect structure or may be used with approximate inference or decoding at test time, then its parameters should be robustly chosen by minimizing the loss that will be achieved *in the presence of all approximations*. Most straightforwardly, one can minimize the average loss on training data under the same approximations as those under which the system will be tested. We refer to this method as ERMA—"Empirical Risk Minimization under Approximations."

In previous work (Stoyanov et al., 2011), we trained CRF parameters in this way for the standard BP approximate inference method. We locally minimized the output loss or cross-entropy on training data, computing its gradient by back-propagation through time. This not only led to more accurate predictions on test

Presented at the International Conference on Machine Learning (ICML) workshop on *Inferning: Interactions between Inference and Learning*, Edinburgh, Scotland, UK, 2012. Copyright 2012 by the author(s)/owner(s).

data, but also permitted speedups by running fewer iterations of BP. The performance of our ERMA-trained models remained robust to this further approximation (and to misspecified model structure), whereas traditionally trained models degraded more rapidly.

Subsequently, we proposed to include runtime as an *explicit* term in the training objective, so that we learn parameters that balance speed and accuracy (Stoyanov & Eisner, 2011).[1] We learned *sparse* graphical models, in which fewer messages have to be passed per iteration. We also attempted to reduce the number of iterations, by learning when to terminate inference based on dynamic properties of the computation state. Training all parameters jointly, we learned models that contained an order of magnitude fewer edges (thus, speeding up inference by an order of magnitude), while suffering only modest decrease in accuracy. In our experiments, a dynamic termination condition proved unnecessary because setting a hard limit to only three iterations of BP resulted in accuracy similar to running BP to convergence (provided that the model parameters were trained for this condition).

The above papers assumed that each training or test example has the same sets of input (X) and output (Y) random variables. This may not always be the case—after all, a virtue of MRFs is that they define the distribution over any missing variables given any observed variables. Of particular interest is the case of relational data, where we want to model interdependencies between individuals, but we may observe different properties of each individual. For instance, in a collaboration network, we may know that Alice, Bob and Charlie have co-authored a paper and that Bob works for Johns Hopkins University. If we want to predict Alice's employer, the connection from Bob's employer is probably valuable. By contrast, using the connection from Charlie's employer would have greater cost for less benefit: computing an accurate distribution over that *unobserved* variable is costly, and if this distribution has high entropy or is redundant with the information from Bob's employer, it is unlikely to change our prediction for Alice. We may therefore learn not to use this connection on such a query.

The case for evidence-specific CRF structures is made by Chechetka and Guestrin (2010). They propose a two-stage method. The first stage uses the Chow-Liu (1968) algorithm to compute a suitable tree structure for each example, using models of mutual information between each pair of Y variables, conditioned on the evidence X. The second stage trains and tests the CRF parameters, using each example's tree structure. The CRF parameters are shared across trees and are separate from those of the mutual information model.

Inspired by Chechetka and Guestrin (2010), we extend our previous work to dynamically determine MRF structure based on which variables X are observed (and with what values), and which variables Y are being queried. Our approach uses *gates* (Minka & Winn, 2008)—additional random variables that can switch factors in the MRF on or off based on features of the evidence X. We train all weights jointly to optimize a linear combination of differentiable approximations to loss and runtime, as in (Stoyanov & Eisner, 2011). The training method uses back-propagation to compute the exact gradient of the training objective.

Unlike Chechetka and Guestrin, we

- allow heterogeneous patterns of missing data;
- seek to minimize loss on specified output variables, rather than maximize likelihood over all Y;
- perform approximate inference over an arbitrary (possibly very sparse) subgraph of the original factor graph, rather than exact inference over a tree;
- select the subgraph using fast approximate inference over gates;
- train structure selection (gate parameters) jointly with the model parameters, rather than serially.

In our current pilot experiments, the gates do not yet communicate with one another, so the evidence-specific structure is selected locally rather than globally. Even this simple method gives good results.

2. Background

Markov Random Field. A **Markov Random Field** (MRF) is defined as a triple $(\mathcal{V}, \mathcal{F}, \Psi)$. $\mathcal{V} = (V_1, \ldots, V_n)$ is a collection of n random variables; we write $\mathbf{v}$ to denote a joint assignment of values to these variables. $\mathcal{F}$ is a collection of subsets of $\{1, 2, \ldots, n\}$; for $\alpha \in \mathcal{F}$, we write $\mathbf{v}_\alpha$ for the restriction of $\mathbf{v}$ to the variables $\mathcal{V}_\alpha = \{V_j : j \in \alpha\}$. Finally, $\Psi = \{\psi_\alpha : \alpha \in \mathcal{F}\}$ is a set of **factors**, where each factor ψ_α is some function that maps a partial assignment $\mathbf{v}_\alpha$ to a **potential** value in $[0, \infty)$. This function depends implicitly on the global parameter vector θ. The probability of an assignment $\mathbf{v}$ is given by

$$p_\theta(\mathcal{V} = \mathbf{v}) = \frac{1}{Z} \prod_{\alpha \in \mathcal{F}} \psi_\alpha(\mathbf{v}_\alpha) \qquad (1)$$

where $Z = \sum_{\mathbf{v}} \prod_{\alpha \in \mathcal{F}} \psi_\alpha(\mathbf{v}_\alpha)$ serves to normalize the distribution. We write $Z_{\mathbf{x}}$ for the sum over just those assignments $\mathbf{v}$ that extend the partial assignment $\mathbf{x}$.

[1]Such an objective was also used by (Oliver & Horvitz, 2003; Eisner & Daumé III, 2011).

Each training input $\mathbf{x}^i$ is a partial assignment to some set of "input" variables $X^i \subseteq \mathcal{V}$. The corresponding training output $\mathbf{y}^i$ is a partial assignment to some set of "output" variables $Y^i \subseteq \mathcal{V}$ that must be predicted. $\mathcal{V}$ may include additional "hidden" variables.

Loopy Belief Propagation. Inference in general MRFs is intractable. In this paper we use a popular approximate inference technique, **loopy belief propagation** (BP) (Pearl, 1988). BP iteratively updates "messages" between the variables and factors. It terminates when the messages converge (i.e., only tiny updates remain) or a maximum number of iterations is reached. Approximate marginal probabilities called beliefs are computed from the final messages. Beliefs can be used to compute gradients of the log-likelihood for training, and to produce predictions during testing.

A desired conditional probability $p_\theta(Y^i = \mathbf{y}^i \mid X^i = \mathbf{x}^i)$ can be found as $Z_{\mathbf{x}^i,\mathbf{y}^i}/Z_{\mathbf{x}^i}$. Each of these $Z_{\dots}$ values can be approximated by running BP and combining the beliefs into an approximation of $-\log Z_{\dots}$, the Bethe free energy (Yedidia et al., 2000).

Log-Likelihood Maximization. An MRF or CRF is typically trained by maximizing the (conditional) log-likelihood of a given training set $\{(\mathbf{x}^i, \mathbf{y}^i)\}$. When BP is used to estimate this objective, the beliefs suffice to compute not only the Bethe free energy but also its gradient. Training CRFs using this approximation has been shown to work relatively well in practice (Vishwanathan et al., 2006; Sutton & McCallum, 2005).

Empirical Risk Minimization. Maximizing conditional log-likelihood is appropriate when it gives a good estimate of the true conditional distribution (e.g., the model structure and training data are adequate), and when that distribution will be used for exact inference and decoding (i.e., the system will actually make the minimum-risk decision given its input). Under other conditions, we have argued that one should *train* to minimize risk for the actual inference and decoding procedures that will be used at test time (Stoyanov et al., 2011). Suppose we have f_θ, a family of decision functions parameterized by θ. In our case, θ parametrically specifies the factor functions, and $f_\theta(\mathbf{x}^i)$ computes the result $\mathbf{y}^*$ of approximate BP inference followed by some decoding procedure that makes a prediction from the beliefs. We then use a given loss function $\ell(\mathbf{y}^*, \mathbf{y}^i)$ to evaluate whether the final output $\mathbf{y}^*$ was good. We should select θ to minimize the expected loss under the true data distribution over $(\mathbf{x}, \mathbf{y})$. In practice, we do not know the true data distribution, but we can do **empirical risk minimization**, taking the expectation over our sample of $(\mathbf{x}^i, \mathbf{y}^i)$ pairs.

To determine the gradient of $\ell(f_\theta(\mathbf{x}^i), \mathbf{y}^i)$ with respect to θ, we employ automatic differentiation in the reverse mode (Griewank & Corliss, 1991). The intuition is that our entire test-time system, regardless of how its constructed or the approximations that it uses, is nothing but a sequence of elementary differentiable operations. If intermediate results are recorded during the computation of the function (the forward pass), we can then compute the partial derivative of the loss with respect to each intermediate quantity, in the reverse order (the backward pass). At the end, we have accumulated the partials of the loss with respect to each parameter θ. A detailed explanation of our back-propagation algorithm as well as complete equations can be found in (Stoyanov et al., 2011).

Gradient information from the above procedure can be used in a local optimization method to minimize training loss. We will use stochastic meta descent (SMD) (Schraudolph, 1999), a second-order online method.

Gates. Gates are a notation for representing context-sensitive dependencies in directed graphical models (Minka & Winn, 2008). A *gate* is a hidden random variable that controls one or more factors of the graphical model, switching them off or on according to the value of the gate.

Gates are naturally used to specify certain models such as mixture models. We adapt them to instead improve the speed and/or accuracy of approximate inference. Turning off factors improves speed. It can also in principle improve accuracy—e.g., by breaking short cycles among unobserved variables in the factor graph (which can harm the BP approximation). The point is that although having many factors may make the model more expressive, it may degrade approximate inference.

If the MRF factor α is controlled by a gate whose value is $g_\alpha \in \{0, 1\}$, then we raise α to the power g_α. If the gate value is unobserved, Minka and Winn would marginalize it out (trying both values 0 and 1, which is expensive). In our architecture, however, we choose to use a "soft gate" by setting the exponent to our estimated probability that the gate is open.. Thus, equation (1) for the probability of an MRF configuration becomes:

$$p_\theta(\mathcal{V} = \mathbf{v} \mid X^i = \mathbf{x}^i) = \frac{1}{Z} \prod_{\alpha \in \mathcal{F}} (\psi_\alpha(\mathbf{v}_\alpha))^{p(g_\alpha = 1 \mid \mathbf{x}^i)} \quad (2)$$

Of course, we train for this "soft gate" architecture. Where does $p(g_\alpha = 1 \mid \mathbf{x}^i)$ come from? We will define a simple CRF that predicts the gates from the evidence $\mathbf{x}^i$, allowing us to first compute exact or approximate marginal beliefs over the gates. In a second step, our gated MRF (2) uses these gate marginals as fixed exponents, and we run BP over that MRF (with

X^i specialized to $\mathbf{x}^i$ as usual).[2] We will jointly train these two models' parameters, collectively denoted θ.

3. Speed-Aware Training

To incorporate runtime into our training objective, we will now seek θ to minimize $\sum_i \ell(f_\theta(\mathbf{x}^i), \mathbf{y}^i) + \lambda \cdot t_\theta(\mathbf{x}^i)$, where $t_\theta(\mathbf{x}^i)$ is the time that the system f_θ needs to make its prediction. Here $\lambda \geq 0$ defines the tradeoff between speed and accuracy, making this a multiobjective optimization. By varying λ, we can sweep out the Pareto frontier of this tradeoff.

With or without soft gates, each factor's runtime cost remains proportional to its size (times the number of BP iterations, which is constant at 3 in our current experiments). To achieve speedups, we must partially return to "hard gates." We *ignore* factors α for which $p(g_\alpha = 1 \mid \mathbf{x}^i) < \tau_\alpha$. We currently set the training threshold τ_α to 0.5. We observed that in practice we benefit from using a slightly lower test-time threshold as many of the gates have values close to our threshold. In our experiments we use a test-time threshold of .3.[3]). Our runtime on example i, $t_\theta(\mathbf{x}^i)$, is improved when the system learns to ignore factors—similarly to the group lasso objective (Schmidt et al., 2008).

Soft training. The previous paragraph says to multiply both the exponent and the runtime of the factor ϕ_α by a step function $\sigma_\alpha(p)$, which is 0 when $p < \tau_\alpha$ (the factor is ignored) and 1 when $p > \tau_\alpha$ (the factor is softly gated). Unfortunately, this hard threshold makes both speed and accuracy nondifferentiable. At training time we therefore use a differentiable surrogate, $\sigma_\alpha(p) = \frac{((1-\tau_\alpha)p)^\beta}{((1-\tau_\alpha)p)^\beta + (\tau_\alpha(1-p))^\beta} = 1/(1+\exp(-\beta \log \frac{(1-\tau_\alpha)p}{\tau_\alpha(1-p)}))$, which is a sigmoid function that maps [0,1] to [0,1] with a soft threshold at τ_α. The parameter $\beta > 0$ controls the steepness of the threshold. As an outer loop of training, we "anneal" by gradually decreasing the "temperature" $1/\beta$ toward 0, which makes σ_α approach the step function we will use at test time.

Back-propagation. To train, we perform ERMA on the full model including MRF parameters and the parameters for the gate variables. We follow the procedure from (Stoyanov et al., 2011), which relies on back-propagation to efficiently compute the gradient of the objective for use by stochastic meta-descent (Schraudolph, 1999). First, we back-propagate gradients through the MRF model to obtain partial derivatives with respect to the MRF parameters and the output values of the gate variables. We then back-propagate through the gate model to obtain partial derivatives with respect to the gate-CRF parameters.

[2]In principle, one could iterate these two steps, allowing inference on the MRF to feed back and affect the gates.

[3]We speculate that this disparity between train-time and test-time condition may be aliviated by the use of better annealing schedules.

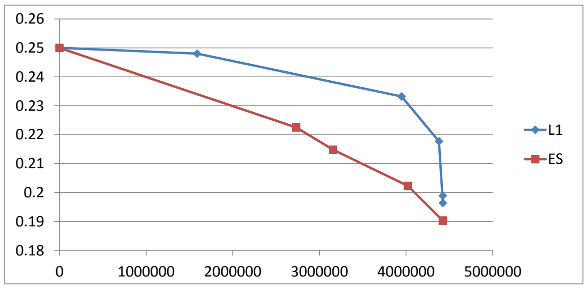

Figure 1. Loss (MSE) vs. runtime (total number of BP messages sent). Speed-up is achieved by increasing the strength of the L1-regularizer for the baseline method and by increasing the value of λ for our method.

4. Experiments

Synthetic data. We generate a random MRF with 50 binary random variables (RVs) and 200 factors each of which includes interactions between *four* random variables. We randomly sample factor weights from a normal distribution and then exponentiate them. We use Gibbs sampling to generate 1000 training and 1000 test data points. For each example, we randomly designate each random variable as input, output, or hidden with equal probability. Our loss function is mean squared error (MSE) on the output marginals.

We pretend that the true model structure is unknown, so training uses a complete graph of (only) binary factors as a reasonable and fairly expressive guess.

Gate-CRF topology. In this paper, each factor α has a dedicated binary gate g_α.[4] Each gate will be conditioned on the input evidence $\mathbf{x}^i$. Our gate-CRF could also model the dependencies between gate variables, so gates could cooperate (to ensure the presence of useful paths to Y^i from X^i) and compete (to avoid including redundant paths). The runtime $t_\theta(\mathbf{x}^i)$ would include the runtime of approximate inference in the gate-CRF. To avoid such complications, in our present experiments we will assume no connections among the gate random variables. In other words, each gate variable independently decides whether (or how much) its factor will be used. Fortunately, these independent

[4]One could achieve further speedup by letting a single gate control a group of related factors.

classifiers are jointly trained to interact well on typical patterns $\mathbf{x}$ of observed variables (if any).

Gate-CRF features. A training example $\mathbf{x}^i$ will specify the status of each variable V_k as 0, 1, unobserved output, or unobserved hidden. If $\alpha = \{5, 9\}$, then the gate g_α will make its prediction based on a single feature, namely the conjunction of the statuses of V_5 and V_9. However, we do not need features of the form $(V_5 = 1, V_9 = 0)$, since a factor α has no effect on inference when all its variables are observed, so we automatically ignore it in this case (in both our method and the baseline method). Thus we have $4 \cdot 4 - 2 \cdot 2 = 12$ features per gate.

Training. Our true objective is highly non-convex, so we first find good initial parameters by optimizing smoother, related functions. Like Stoyanov et al. (2011), we increase the BP approximation to conditional log-likelihood for three passes through the training data, before switching to our true objective. For annealing during this second stage, we use a simple, two-step schedule with temperatures $(1/\beta)$ of 1 and 0.5.

Results. Figure 1 shows the speed vs. accuracy curve for our method as compared to models that are still trained through ERMA, but using L1-regularization to induce sparsity. In the former case we achieve different levels of sparsity (speed) by varying the parameter λ, which controls how much weight is given to the speed term in the loss function. In the case of L1-regularization, different levels of sparsity are achieved by varying the strength of the regularizer.

Our method achieves better accuracy at all levels of speed. We note that our results are preliminary—this is just one setting, without statistical significance tests. We also have not experimented with hyperparameters such as τ_α and the annealing schedule.

5. Conclusions and Future Work

We showed a new way to induce evidence-specific structure in MRFs through the use of gate random variables. We presented an approach for jointly learning parameters of the MRF and parameters controlling the gates, based on ERMA and back-propagation. Preliminary experiments showed that compared to using L1-regularization to induce non-evidence-specific sparsity, our method achieves better accuracy at all levels of speed.

We plan to also explore learning sparse models by gradually adding edges, as in (Lee et al., 2006) but using our speed-augmented objective. We also intend to devise methods for learning prioritization and pruning heuristics for message updates.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant No. 0964681 to the second author.

References

Chechetka, A. and Guestrin, C. Evidence-specific structures for rich tractable CRFs. In *Advances in Neural Information Processing Systems*, 2010.

Chow, C. and Liu, C. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

Eisner, J. and Daumé III, H. Learning speed-accuracy tradeoffs in nondeterministic inference algorithms. In *COST: NIPS 2011 Workshop on Computational Trade-offs in Statistical Learning*, Sierra Nevada, Spain, December 2011.

Finley, T. and Joachims, T. Training structural SVMs when exact inference is intractable. In *Proceedings of ICML*, pp. 304–311, 2008.

Griewank, A. and Corliss, G. *Automatic differentiation of algorithms: theory, implementation, and application.* Society for Industrial and Applied Mathematics, 1991. ISBN 089871284X.

Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pp. 282–289, 2001.

Lee, Su-In, Ganapathi, Varun, and Koller, Daphne. Efficient structure learning of Markov networks using L_1-regularization. In *Proceedings of NIPS*, pp. 817–824, 2006.

Minka, T. and Winn, J. Gates: A graphical notation for mixture models. In *Proceedings of NIPS*, pp. 1073–1080, 2008.

Oliver, N. and Horvitz, E. Selective perception policies for guiding sensing and computation in multimodal systems: A comparative analysis. In *Proceedings of the Fifth International Conference on Multimodal Interfaces*, Vancouver, November 2003. ACM Press.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988. ISBN 1558604790.

Peng, F. and McCallum, A. Information extraction from research papers using conditional random fields. *Information Processing & Management*, 42 (4):963–979, 2006. ISSN 0306-4573.

Schmidt, M., Murphy, K., Fung, G., and Rosales, R. Structure learning in random fields for heart motion abnormality detection. *CVPR. IEEE Computer Society*, 2008.

Schraudolph, N.N. Local gain adaptation in stochastic gradient descent. In *Proceedings of ANN*, pp. 569–574, 1999.

Sha, F. and Pereira, F. Shallow parsing with conditional random fields. In *Proceedings of ACL/HLT*, pp. 134–141, 2003.

Stoyanov, V. and Eisner, J. Learning cost-aware, loss-aware approximate inference policies for probabilistic graphical models. In *COST: NIPS 2011 Workshop on Computational Trade-offs in Statistical Learning*, Sierra Nevada, Spain, December 2011.

Stoyanov, V. and Eisner, J. Minimum-risk training of approximate CRF-based NLP systems. In *Proceedings of NAACL*, 2012.

Stoyanov, V., Ropson, A., and Eisner, J. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, 2011.

Sutton, C. and McCallum, A. Piecewise training of undirected models. In *Proceedings of UAI*, pp. 568–575, 2005.

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

Yedidia, Jonathan S., Freeman, William T., and Weiss, Yair. Bethe free energy, Kikuchi approximations and belief propagation algorithms. Technical Report TR2001-16, Mitsubishi Electric Research Laboratories, 2000.