Dynamic Conditional Random Fields for Jointly Labeling Multiple Sequences

Andrew McCallum, Khashayar Rohanimanesh, and Charles Sutton
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{mccallum,khash,casutton}@cs.umass.edu

Abstract

Conditional random fields (CRFs) for sequence modeling have several advantages over joint models such as HMMs, including the ability to relax strong independence assumptions made in those models, and the ability to incorporate arbitrary overlapping features. Previous work has focused on linear-chain CRFs, which correspond to finite-state machines, and have efficient exact inference algorithms. Often, however, we wish to label sequence data in multiple interacting ways—for example, performing part-of-speech tagging and noun phrase segmentation simultaneously, increasing joint accuracy by sharing information between them. We present *dynamic conditional random fields (DCRFs)*, which are CRFs in which each time slice has a set of state variables and edges—a distributed state representation as in dynamic Bayesian networks—and parameters are tied across slices. (They could also be called conditionally-trained *Dynamic Markov Networks*.) Since exact inference can be intractable in these models, we perform approximate inference using the tree-based reparameterization framework (TRP). We also present empirical results comparing DCRFs with linear-chain CRFs on natural-language data.

1 Introduction

The problem of labeling and segmenting sequences of observations arises in many different areas, including bioinformatics, music modeling, computational linguistics, speech recognition, and information extraction. Probabilistic finite state automata, such as hidden Markov models (HMMs), have been popular for such sequence labeling tasks. Finite-state Conditional Random Fields (CRFs) [4] are another sequence model that offers several advantages over HMMs, relaxing the strong dependence assumptions made in those models and allowing rich sets of overlapping features.

Many sequence-processing problems are solved by chaining errorful subtasks. The traditional language understanding task, for example, is often broken into parsing, semantic understanding, and contextual and discourse analysis. In information extraction, one often performs part-of-speech tagging and then shallow parsing as pre-processing steps before the main extraction task. In such an approach, however, errors early in processing nearly

always cascade through the chain, causing errors in the final output.

In this paper, we address this problem by representing the multiple label sequences in a single graphical model, explicitly modeling limited depedencies between them. We introduce *Dynamic CRFs*, which are CRFs that repeat structure and parameters over a sequence. For example, the factorial structure in Figure 3 models dependencies between cotemporal labels, allowing information to flow between the subtasks in both directions.

DCRFs are named after *Dynamic Bayesian Networks (DBNs)* [2], directed sequence models for which there is a large body of literature addressing representation, learning, and inference (see [7]). Particular classes of DBNs, such as factorial HMMs, have also been extensively studied [12, 8, 3]. Previous work with CRFs has used the linear-chain structure, depicted in Figure 1, in which a first-order Markov assumption is made among labels. DCRFs combine the modeling advantages of the distributed hidden state in DBNs with the rich feature sets allowed in conditional models.

First, we briefly describe the general framework of CRFs. Then, we describe DCRFs, including how to do approximate inference and parameter estimation. Finally, we compare DCRFs to combinations of linear-chain CRFs on a task that involves both part-of-speech tagging and noun-phrase segmentation.

2 CRFs

Conditional Random Fields (CRFs) [4] are undirected graphical models that encode a conditional probability distribution using a given set of features. CRFs are defined as follows. Let $\mathcal{G}$ be an undirected model over sets of random variables $\mathbf{y}$and $\mathbf{x}$. Typically, $\mathbf{y} = \{y_t\}$ and $\mathbf{x} = \{x_t\}$ for $t = 1, \ldots, T$, so that $\mathbf{y}$ is a labeling of an observed sequence $\mathbf{x}$.[1] If $C = \{\{\mathbf{y}_c, \mathbf{x}_c\}\}$ is the set of cliques in $\mathcal{G}$, then CRFs define the conditional probability of a state sequence given the observed sequence as:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C} \Phi(\mathbf{y}_c, \mathbf{x}_c), \tag{1}$$

where Φ is a potential function and $Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{c \in C} \Phi(\mathbf{y}_c, \mathbf{x}_c)$ is normalization factor over all state sequences of length T. We assume the potentials factorize according to a set of features $\{f_k\}$, which are given and fixed, so that

$$\Phi(\mathbf{y}_c, \mathbf{x}_c) = \exp\left(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(\mathbf{y}_c, \mathbf{x}_c, t)\right) \tag{2}$$

The model parameters are a set of real weights $\theta = \{\lambda_k\}$, one weight for each feature.

Previous applications have used the *linear-chain CRF*, in which a first-order Markov assumption is made on the hidden variables. The graphical model for this is shown in Figure 1. In this case, the cliques of the conditional model are the nodes and edges, so that there are feature functions $f_k(y_{t-1}, y_t, \mathbf{x}, t)$ for each label transition and $g_k(y_t, \mathbf{x}, t)$ for each label. Feature functions can be arbitrary. For example, a feature function $f_k(y_{t-1}, y_t, \mathbf{x}, t)$ on a pair of variables (y_{t-1}, y_t) could be a binary test that has value 1 if and only if y_{t-1} has the label "*adjective*", y_t has the label "*proper noun*", and x_t begins with a capital letter.

Linear-chain CRFs correspond to finite state machines, and can be roughly understood as conditionally-trained hidden Markov models (HMMs). This class of CRFs is also a

[1] Note that in general, the set of labels may be different from the set of states of the FSM, in that multiple states can correspond to the same label. In practice, however, it is usually assumed that the set of states and labels are the same, or given the sequence of the labels, the set of states are unambiguously known.

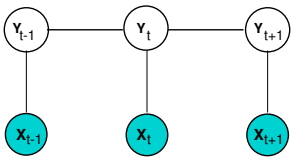


Figure 1: Graphical representation of linear-chain CRFs.

globally-normalized extension to *Maximum Entropy Markov Models* [6] that avoids the label bias problem [4].

3 Dynamic CRFs

3.1 Model Representation

A dynamic CRF is one that has repetitive structure and parameters over time. More specifically, we define a *2-CRF* to be a two-timeslice undirected graph, with set of feature functions $\{f_k\}$ and corresponding weights $\{\lambda_k\}$. Given an instance $\mathbf{x}$, we unroll the 2-CRF to get a full undirected model, in the same way as DBNs. The same set of features and weights is used at each time slices, so that the parameters are tied across the network. Then the conditional probability of a label sequence $\mathbf{y}$ is given by:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_t \sum_k \lambda_k f_k(\mathbf{y}_{(k,t)}, \mathbf{x}, t)\right) \tag{3}$$

DCRFs generalize not only linear-chain CRFs, but more complicated structures as well. In this paper, we use a *factorial DCRF*, which has two linear chains of labels, with labels at the same time step joined. Figure 3) is an example of an unrolled factorial DCRF.

3.2 Inference in DCRFs

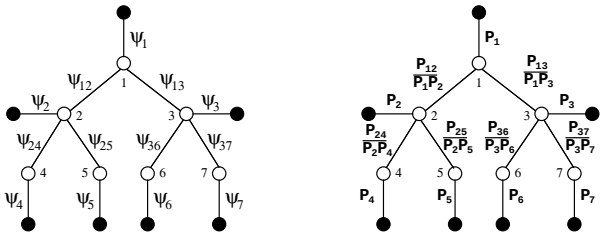
Inference in an unrolled DCRF can be done using any inference algorithm for undirected models. Because exact inference can be expensive in complex DCRFs, we use approximate methods. Here we describe approximate inference using tree-reparameterization (TRP) [11]. TRP is based on the fact that any exact algorithm for optimal inference on trees actually computes marginal distributions for pairs of neighboring nodes. For an undirected graphical model over variables $\mathbf{x}$, this results in an alternative parameterization of the distribution (Figure 2(a)) as:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{s \in V} \psi_s(x_s) \prod_{(s,t) \in E} \psi_{st}(x_s, x_t) \quad \Rightarrow \quad p(\mathbf{x}) = \prod_{s \in V} P_s(x_s) \prod_{(s,t) \in E} \frac{P_{st}(x_s, x_t)}{P_s(x_s) P_t(x_t)} \tag{4}$$

Figure 2(b) shows the reparameterized tree[2].

Here we summarize the TRP algorithm as a sequence of updates $\mathbf{T}^n \rightarrow \mathbf{T}^{n+1}$ on the graph $\mathbf{U}$ with the edge set $\mathcal{E}$, where $\mathbf{T}$ represents the set of marginal probabilities maintained by TRP consisting of single-node marginals $\mathbf{T}_u^{n+1}(x_u)$ and pairwise joint distribution $\mathbf{T}_{uv}^{n+1}(x_u, x_v)$; and n denotes the iteration number:

[2]This figure is adopted from [11].



(a) (b)

Figure 2: (a) A simple tree-structured graphical model and its original parameterization; (b) Alternative parameterization in terms of marginal distributions.

1. **Initialization:** for every node u and every pair of nodes (u, v), initialize $\mathbf{T}^0$ by $T_u^0 = \kappa\psi_u$ and $T_{uv}^0 = \kappa\psi_{uv}$, with κ being a normalization factor. (Other initializations are also possible.)
2. **TRP updates:** for $i = 1, 2, \ldots,$ do:
   - Select some spanning tree $\mathcal{T}^i \in \Upsilon$ with edge set $\mathcal{E}^i$, where $\Upsilon = \{\mathcal{T}\}$ is a set of spanning tress.
   - Use any exact algorithm, such as belief propagation, to compute exact marginals $p^i(x)$ on $\mathcal{T}^i$. For all $(u, v) \in \mathcal{E}^i$, set

$$\mathbf{T}_u^{i+1}(x_u) = p^i(x_u).$$

$$\mathbf{T}_{uv}^{i+1}(x_u, x_v) = \frac{p^i(x_u, x_v)}{p^i(x_u)p^i(x_v)}.$$

   - Set $\mathbf{T}_{uv}^{i+1} = \mathbf{T}_{uv}^i$ for all $(u, v) \in \mathcal{E}/\mathcal{E}^i$ (i.e., all the edges not included in the spanning tree $\mathcal{T}^i$).

When selecting spanning trees $\Upsilon = \{\mathcal{T}\}$, the only constraint is that the trees in Υ cover the edge set of the original graph $\mathbf{U}$.

3.3 Parameter Estimation in DCRFs

The parameter estimation problem is to find a set of parameters $\theta = \{\lambda_k\}$ given training data $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^N$. This is accomplished using standard convex optimization techniques, similar to other maximum-entropy models [4, 1]. More specifically, we optimize the conditional log-likelihood

$$\mathcal{O}(\theta) = \sum_i \log p_\theta(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}). \tag{5}$$

The derivative of this is

$$\frac{\partial \mathcal{O}}{\partial \lambda_k} = \sum_i \sum_t f_k(\mathbf{y}_t^{(i)}, \mathbf{x}^{(i)} t) - \sum_i \sum_t \sum_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x}_t^{(i)}) f_k(\mathbf{y}_t, \mathbf{x}_t^{(i)}) \tag{6}$$

Although this seems to require summing over all possible label sequences, if we observe that each feature function depends only on a single clique, we get

$$\frac{\partial \mathcal{O}}{\partial \lambda_k} = \sum_i \sum_t f_k(\mathbf{y}_t^{(i)}, \mathbf{x}^{(i)} t) - \sum_i \sum_t \sum_{c \in C} \sum_{\mathbf{y}_c} p_\theta(\mathbf{y}_c \mid \mathbf{x}_t^{(i)}) f_k(\mathbf{y_c}, \mathbf{x}_t^{(i)}), \tag{7}$$

	Confidence	in	the	pound	is	widely	expected	...
POS	NN	IN	DT	NN	VBZ	RB	VBN	
collapsed	NOUN	OTHER	OTHER	NOUN	VERB	RBP	VERB	
Phrases	B-NP	B-PP	B-NP	I-NP	B-VP	I-VP	I-VP	
NP	B	O	B	I	O	O	O	

Table 1: Example document with POS and NP labels, before and after collapsing the labels.

where $\mathbf{y_c}$ ranges over assignments to the clique c.

This loss function is convex, and can be optimized by any number of techniques. In the results below, we use L-BFGS, which has previously outperformed other optimization algorithms for linear-chain CRFs [9, 5].

Note that this optimization requires computing marginal probabilities for every training instance at every iteration of the optimizer. In the experiments reported here, it was typical to need to compute marginals in 32 000 different models. This intensifies the need for efficient inference.

4 Experiments

We used factorial DCRFs to do both part-of-speech tagging and noun-phrase segmentation on data from the CoNLL 2002 shared task data set[3]. Table 1 shows example data. We considered each sentence to be a training instance, with single words as tokens. The training data $\mathcal{D}_1$ contained 209 sentences. Table 2 shows some of the features we used.

There were three NP labels: begin-phrase, inside-phrase, and other. The original data contained 45 different POS labels. To reduce the inference time, we collapsed the POS labels from 45 to 5 as follows:

- Collapse all different types of nouns into one label NOUN.
- Collapse all different types of verbs into one label VERB.
- Collapse all different types of adjectives into one label JADJ.
- Collapse all different types of adverbs into one label RBP.
- Collapse the remaining POS labels into one label OTHER.

We present two experiments: one comparing factorial DCRFs with linear-chain models, and one comparing different inference algorithms in factorial CRFs.

4.1 Comparison to linear-chain CRFs

We compared three models: a factorial DCRF, a cascaded CRF, and a linear-chain CRF. The factorial DCRF used the graph structure in Figure 3, with the upper chain modeling part-of-speech (POS) process and the lower chain modeling noun-phrase (NP) process. The vertical edges are added to capture the dependencies between POS and NP labels.

We used L-BFGS to learn the parameters of the DCRF. Computing the gradient requires computing the marginals over vertices and edges of the unrolled DCRF at different portions in time. We used the TRP approximation to compute these marginals.

Each TRP iteration selects a random spanning tree from the graphical model unrolled over the current training instance. To ensure that all the edges of the graph will be covered by the

[3] See `http://lcg-www.uia.ac.be/~erikt/research/np-chunking.html`.

word (collapsed: years, year-spans, fractions, numbers, ...)
contains-dash "-"
contains-dash-based "-based"
capitalized
all-caps
single-capital-letter
mixed-capitalization
contains-digits (and other symbols)

Table 2: Some of the features used in these experiments.

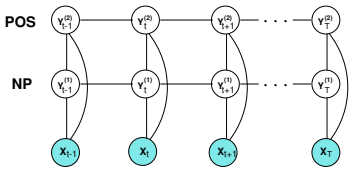


Figure 3: Factorial DCRF model for POS and NP tagging problem; (Top process): Part-of-Speech (POS); (Bottom process): Noun-Phrase (NP).

set of spanning trees used in TRP updates, we included eight hand-designed trees among the random spanning trees. o

Next, we trained two cascaded linear-chain CRFs, where one CRF predicted the POS labels, and then the other CRF predicted the NP labels, using the POS predictions as input features. More specifically, we trained a POS-tagger (which we call CRF$_{pos}$) using a training set $\mathcal{D}_2$ that had 86 instances labeled by their POS tags. We then used the learned model and substituted the POS labels of the original training set $\mathcal{D}_1$ by the labels predicted by the learned model (i.e., CRF$_{pos}$) over the data in $\mathcal{D}_1$, and generated the new training set $\mathcal{D}_3$. Note that $\mathcal{D}_3$ has exactly the same data (features) of $\mathcal{D}_1$, and the same NP labels, but could have have different POS labels. Using $\mathcal{D}_3$, we trained a new CRF model (which we call CRF$^{+}_{np}$) for predicting the NP labels, using the POS labels as a feature.

Finally, we trained a best-case linear-chain CRF (which we call CRF$^{*}_{np}$) for predicting NP labels using the true POS labels along with the base features from Table 2. Of course, it is unrealistic to assume that the true POS labels are provided, however, this model is intended to give an upper bound on how much POS knowledge can help noun-phrase segmentation.

In the cascaded model CRF$^{+}_{np}$ and the best case model CRF$^{*}_{np}$, we used POS labels as features, however CRF$^{+}_{np}$ uses the POS labels predicted by CRF_{pos} whereas CRF$^{*}_{np}$ uses the correct POS labels as originally provided with the training set $\mathcal{D}_1$.

Table 3 compares the performance of these models. We measured accuracy on POS labels, on NP labels, and also joint accuracy on (POS, NP) pairs. To compute the joint accuracy for CRF$^{+}_{np}$ on the test set, we used the predicted POS tags from CRF$_{pos}$ and the predicted NP tags using CRF$^{+}_{np}$. To compute the joint accuracy for CRF$^{*}_{np}$ we used the true POS tags of the test set together with the predicted NP tags on the test set using CRF$^{*}_{np}$.

The factorial DCRF outperform the cascaded CRF$^{+}_{pos}$ in joint accuracy and POS accuracy, however, but had lower NP accuracy. We conjecture that because there are more POS labels than NP labels, L-BFGS is forced to minimize the error across POS with more weight. The best-case model CRF$^{*}_{np}$ outperforms the other models in every category. The performance of 100% for POS labels is because this model was provided with true POS labels.

	CRF^{+}_{np}	**DCRF**	CRF^{*}_{np}
NP Accuracy	0.9084	0.8611	0.9249
POS Accuracy	0.7722	0.8203	1.0
Joint Accuracy	0.7197	0.7728	0.9249

Table 3: Comparison of performance of CRFs and DCRFs

Algorithm	Overall F1	Training time (hr)	LBFGS iterations
TRP	0.6740	5.342	87
Loopy	0.6756	14.728	81
Junction Tree	0.6675	8.614	83

Table 4: Comparison of inference algorithms for 2-chain factorial CRF on CoNLL 2002 data set. Overall F1 is the average of the F1 measure over all types of NP and POS labels. LBFGS iterations gives the number of iterations of the LBFGS gradient descent.

4.2 Comparison of Inference Algorithms

Because DCRFs can have rich graphical structure, and require many marginal computations during training, effective inference is critical to efficient training with many labels and large data sets. We compared the performance and running time of three different propagation algorithms: TRP, loopy belief propagation, and junction tree.

We ran TRP with random spanning trees, stopping after 25 iterations whether the algorithm had converged or not. Loopy belief propagation was run until all marginal probabilities had converged to within 10^{-4}, which usually took between 10 and 15 iterations of synchronous updates. Exact inference using junction tree was feasible because we used collapsed tags and only two chains. In this experiment, we used 410 training instances, a superset of the training set of the previous section. POS tags were collapsed as before. All experiments were run on an Intel Xeon 2.8 GHz machine with 3 GB RAM. We measured performance on a test set and total training time. The training times include a few non-inference tasks such as computing the gradient; however, the running time is dominated by the time used by inference.

The results are shown in Table 4. In overall F1 on a test set, the inference algorithms perform very similarly. For some reason, junction tree has a slightly lower F-measure on this test set; in other experiments, exact inference has had slightly higher F-measure. However, TRP trains much faster, using only 62% of the time needed by junction tree. Synchronous loopy belief propagation performed very slowly on this data set. Using a less strict stopping criterion might allow it to run faster without sacrificing performance on the tagging task.

Although these results need to be replicated in other data sets, they suggest that TRP is a a good choice for training this kind of model.

5 Conclusions

Dynamic CRFs are conditionally-trained sequence models with repetitive graphical structure and tied parameters. Inference in DCRFs can be done efficiently using approximate methods, and training can be done within the maximum-entropy framework. Because of their factorized state, we can use DCRFs do several labeling tasks at once, sharing information between themx. On a joint noun-phrase segmentation / part-of-speech tagging task, a factorial DCRF does better on joint accuracy than linear-chain CRFs, but apparently this

happens at the expense of NP accuracy. More work is needed for training DCRFs where accuracy on certain labels is more important than others.

References

[1] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[2] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[3] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, volume 8, pages 472–478. MIT Press, 1995.

[4] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields. *ICML*, 2001.

[5] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation, 2002.

[6] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th International Conf. on Machine Learning*, pages 591–598. Morgan Kaufmann, San Francisco, CA, 2000.

[7] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, July 2002.

[8] L. Saul and M. Jordan. Boltzmann chains and Hidden Markov Models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 435–442. The MIT Press, 1995.

[9] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL*, Edmonton, Canada, 2003.

[10] P. Smyth, D. Heckerman, and M. Jordan. Probabilistic independence networks for hidden markov probability models. Technical Report AIM-1565, 1996.

[11] M. Wainwright, T. Jaakkola, and A. Willsky. Tree-based reparameterization for approximate estimation on graphs with cycles. *Neural Information Processing Systems (NIPS)*, 2001.

[12] C. Williams and G. Hinton. Mean field networks that can learn to discriminate temporally distorted strings. *In D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton(Eds.) Connectionist Models: Proceedings of the 1990 Connectionist Summer School*, 1990.