Structure Compilation: Trading Structure for Features

Percy Liang PLIANG@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, CA, USA

Hal Daumé III ME@HAL3.NAME
School of Computing, University of Utah, Salt Lake City, UT, USA

Dan Klein KLEIN@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, CA, USA

Abstract

Structured models often achieve excellent performance but can be slow at test time. We investigate *structure compilation*, where we replace structure with features, which are often computationally simpler but unfortunately statistically more complex. We analyze this tradeoff theoretically and empirically on three natural language processing tasks. We also introduce a simple method to transfer predictive power from structure to features via unlabeled data, while incurring a minimal statistical penalty.

1. Introduction

Structured models have proven to be quite effective for tasks which require the prediction of complex outputs with many interdependencies, e.g., sequences, segmentations, trees, etc. For example, conditional random fields (CRFs) can be used to predict tag sequences where there are strong dependencies between adjacent tags (Lafferty et al., 2001). In part-of-speech tagging, for instance, a CRF can easily model the fact that adjectives tend to precede nouns in English. However, the excellent performance of structured models comes at a computational cost: inference in loopy graphs requires approximate inference, and even for sequences, there is a quadratic dependence on the number of tags.

In this paper, we ask a bold question: do we really need structure? Consider replacing the edges in a CRF with additional contextual features, i.e., having an independent logistic regression (ILR) at each position. A fundamental question is whether there is a gap between the expressive power of the ILR and that of the CRF. Punyakanok et al. (2005) investigated this question for margin-based models[1] and concluded that structure was not needed when the independent problems were "easy." They characterized difficulty in terms of classifier separability, which is a very rigid notion. In Section 3.1, we provide an information-theoretic analysis, decomposing the gap between the ILR and CRF into three terms, each one representing a shortcoming of the ILR. The impact of each is investigated empirically.

Even if the ILR were made as expressive as the CRF by adding additional features, an important remaining question is whether the ILR could generalize as well as the CRF given limited labeled data. Indeed, the ILR overfits more easily, and we provide generalization bounds in Section 3.2 to quantify this effect.

At this point, it seems as though we are forced to make a tradeoff between the computational simplicity of the ILR and the statistical simplicity of the CRF. However, we propose *structure compilation* as a way to have the best of both worlds. Our strategy is to label a plethora of unlabeled examples using the CRF and then train the ILR on these automatically labeled examples. If we label enough examples, the ILR will be less likely to overfit. Although training now takes longer, it is only a one-time cost, whereas prediction at test time should be made as fast as possible.

Many authors have used unlabeled data to transfer the predictive power of one model to another, for example, from high accuracy neural networks to more interpretable decision trees (Craven, 1996), or from high accuracy ensembles to faster and more compact neural networks (Bucilă et al., 2006). Our fo-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] In their case, the independent models are not endowed with extra features, but coherence of the predictions is enforced at test time.

cus is on structured classification tasks, specifically on studying the tradeoff between structure and features. We ran experiments on three tasks: part-of-speech tagging (POS), named-entity recognition (NER), and constituency parsing (Figure 1).

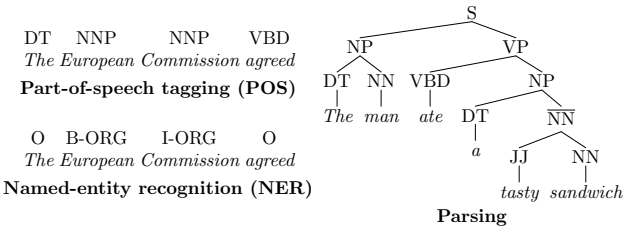


Figure 1. Examples of inputs and their outputs for the three tasks we experimented on. The input (what's seen at test time) is italicized.

2. From Structure to Features

In this section, we will walk through the process of replacing structure with features, using empirical results on POS[2] and NER[3] as running examples. Table 1 summarizes the notation we will use.

2.1. Conditional Random Fields (CRFs)

In structured classification, our goal is to learn to predict an output $\mathbf{y} \in \mathcal{Y}$ (e.g., a tag sequence, a segmentation, or a parse tree) given an input $\mathbf{x} \in \mathcal{X}$ (e.g., a sentence). In this paper, we consider *conditional exponential family* models, which have the following form:

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \exp\{\phi(\mathbf{x}, \mathbf{y})^\top \theta - A(\theta; \mathbf{x})\}, \quad (1)$$

where $\phi(\mathbf{x}, \mathbf{y})$ are the sufficient statistics (features), $\theta \in \mathbb{R}^d$ are the parameters, and $A(\theta; \mathbf{x}) = \log \sum_{\mathbf{y}} \exp\{\phi(\mathbf{x}, \mathbf{y})^\top \theta\}$ is the log-partition function.

One important type of conditional exponential family is a conditional random field (CRF) defined on a graph $G = (V, E)$. In this case, the output $\mathbf{y} = \{y_i\}_{i \in V}$ is a collection of labels, one for each node $i \in V$, with $y_i \in \{1, \ldots, K\}$. The features include functions over both nodes and edges:

$$\phi(\mathbf{x}, \mathbf{y}) = \sum_{i \in V} f(y_i, \mathbf{x}, i) + \sum_{(i,j) \in E} g(y_i, y_j).$$

In this work, we use a generic set of features for both POS and NER. The components of the node features $f(y_i, \mathbf{x}, i)$ are all indicator functions of the form $\mathbb{I}[y_i = a, s(x_{i+o}) = b]$, where a ranges over tags, $s(\cdot)$ ranges over functions on words,[4] b are values in the range of $s(\cdot)$, and $-L \le o \le L$ is an offset within a radius L window of the current position i (we used $L = 0$ for POS, $L = 1$ for NER). The components of the edge features $g(y_i, y_j)$ are of the form $\mathbb{I}[y_i = a, y_j = b]$. Let f_1 denote this *base feature set*.

Training Suppose we are given n labeled examples $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$, which for the purposes of our theoretical analysis are assumed to be drawn i.i.d. from some unknown true distribution p^*. We train the CRF using standard maximum likelihood:[5] $\max_\theta \mathbb{E}_{p^{\text{L}}(\mathbf{x},\mathbf{y})} \log p(\mathbf{y} \mid \mathbf{x}; \theta)$, where $p^{\text{L}}(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^n \mathbb{I}[\mathbf{x} = \mathbf{x}^{(i)}, \mathbf{y} = \mathbf{y}^{(i)}]$ denotes the empirical distribution of the labeled data. Later, we will consider other training regimes, so we need to establish some new notation. Let $p_{\text{C}} = \textsc{Tr}(\textsc{crf}, f_1, p^{\text{L}})$ denote the CRF trained with the base feature set f_1 on the labeled data.

CRFs achieve state-of-the-art performance on POS and NER. Using just our generic feature set, we obtain 96.9% tagging accuracy on POS (training with 30K examples) and 85.3% F_1 on NER (training with 10K examples). However, the performance does come at a computational cost, since inference scales quadratically with the number of tags K. This cost only increases with more complex models.

2.2. Independent Logistic Regression (ILR)

Let us try something drastic: remove the edges from the CRF to get an independent logistic regression (ILR), where now $\phi(\mathbf{x}, \mathbf{y}) = \sum_{i \in V} f(y_i, \mathbf{x}, i)$. For an ILR trained on the labeled data with our base feature set (denoted formally as $\textsc{Tr}(\textsc{ilr}, f_1, p^{\text{L}})$), inference can be done independently for each node. For POS, the ILR takes only 0.4ms to process one sentence whereas the CRF takes 2.7ms, which is a speedup of 5.8x, not including the time for precomputing features.[6] Unfortunately, the accuracy of POS drops from 96.9% to 93.7%. For NER, F_1 drops from 85.3% to 81.4%.

[2]We used the Wall Street Journal (WSJ) portion of the Penn Treebank, with sections 0–21 as the training set (38K sentences) and sections 22–24 as the test set (5.5K sentences).

[3]We used the English data from the 2003 CoNLL Shared Task, consisting of 14.8K training sentences and 3.5K test sentences (set A).

[4]We used 10 standard NLP functions which return the word, prefixes and suffixes (up to length 3) of the word, word signatures (e.g., *McPherson* maps to *AaAaaaaaa* and *AaAa*), and whether the word is capitalized.

[5]In our experiments, we ran stochastic gradient for 50 iterations with a 1/(iteration + 3) step-size.

[6]If we include the time for computing features, the speedup drops to 2.3x.

2.3. Adding New Features

Without edges, the ILR has less expressiveness compared to the CRF. We can compensate for this loss by expanding our base feature set. We will use f_2 to denote the *expanded feature set*.

We use a simple recipe to automatically construct f_2 from f_1, but in general, we could engineer the features more carefully for better performance (see the parsing experiments in Section 4, for example). Essentially, our recipe is to allow the ILR at node i to use the base features f_1 applied to the nodes in a local window around i. For the chain CRF, this amounts to simply increasing the window size from L to $L + r$ (Section 2.1), where we call r the *expansion radius*.

For POS, we used an expansion radius of $r = 1$; for NER, $r = 2$. By training with these new features ($\textsc{Tr}(\textsc{ilr}, f_2, p^{\text{L}})$), we get 96.8% on POS (compared to the 96.9% of the CRF), taking 0.8ms per example (compared to 2.7ms for the CRF). In this case, we have successfully traded structure for features with a negligible loss in performance and a 3.4x speedup. For NER, the story is quite different: adding features actually makes F_1 drop from 81.1% to 78.8%.

We believe there are two reasons for the poor performance on NER. First, since NER is a segmentation problem, the structure plays a more integral role and thus cannot be easily replaced with features. In other words, the *approximation error* of the ILR with respect to the CRF is higher for NER than POS. Section 3.1 provides a more formal treatment of this matter. Second, adding more features increases the risk of overfitting. In other words, the *estimation error* is larger when there are more features. Section 3.2 analyzes this error theoretically.

2.4. Using Unlabeled Data

There seems to be a tradeoff between approximation error and estimation error: More features can provide a better substitute for structure (decreasing the approximation error), but at the risk of overfitting the data (increasing the estimation error).

The algorithmic contribution of this paper is using unlabeled data to reduce the estimation error of the ILR via structure compilation. Suppose we have m unlabeled examples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$ (which we assume are also generated from p^*); let $p^{\text{U}}(\mathbf{x})$ denote the corresponding empirical distribution. We can use the CRF $p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ (which has been trained on p^{L}) to label this unlabeled data. Let $p^{\text{U}}_{\text{C}}(\mathbf{x}, \mathbf{y}) = p^{\text{U}}(\mathbf{x}) p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ denote this new plentiful source of labeled data. Instead of training the ILR on the limited amount of originally labeled data p^{L}, we instead train it on our automatically labeled data p^{U}_{C}.[7]

f_1	base feature set
f_2	expanded feature set
$p^*(\mathbf{x}, \mathbf{y})$	true data distribution
$p^{\text{L}}(\mathbf{x}, \mathbf{y})$	original labeled examples (few)
$p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$	$= \textsc{Tr}(\textsc{crf}, f_1, p^{\text{L}})$ [trained CRF]
$p_{\text{C}^*}(\mathbf{y} \mid \mathbf{x})$	$= \textsc{Tr}(\textsc{crf}, f_1, p^*)$ [limiting CRF]
$p^{\text{U}}(\mathbf{x})$	unlabeled examples (many)
$p^{\text{U}}_{\text{C}}(\mathbf{x}, \mathbf{y})$	$= p^{\text{U}}(\mathbf{x}) p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ [labeled with CRF]
$p^*_{\text{C}}(\mathbf{x}, \mathbf{y})$	$= p^*(\mathbf{x}) p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ [labeled with CRF]
$p_{\text{I}}(\mathbf{y} \mid \mathbf{x})$	$= \textsc{Tr}(\textsc{ilr}, f_2, p^{\text{U}}_{\text{C}})$ [trained ILR]
$p_{\text{I}^*}(\mathbf{y} \mid \mathbf{x})$	$= \textsc{Tr}(\textsc{ilr}, f_2, p^*_{\text{C}})$ [limiting ILR]

Table 1. Notation used in this paper. In general, superscripts denote marginal distributions over $\mathbf{x}$ and subscripts denote conditional distributions over $\mathbf{y}$ given $\mathbf{x}$.

Structure compilation is most useful when there are few original labeled examples. Figure 2 shows the performance of a ILR obtained using structure compilation when the CRF is trained on only 2K labeled examples. We see that using more unlabeled data reduces the performance gap between the CRF and ILR as the estimation error is reduced. For POS, the gap is closed entirely, whereas for NER, there is a remaining approximation error.

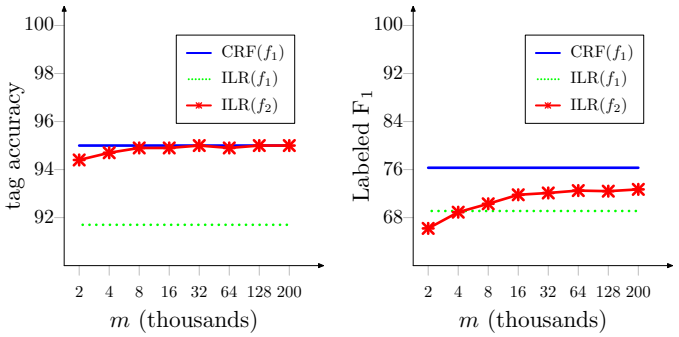


Figure 2. $\textsc{crf}(f_1)$ is the CRF trained using the base feature set on 2K examples. $\textsc{ilr}(f_1)$ is the ILR trained the same way; performance suffers. However, by using the expanded feature set and training on m examples (New York Times articles from Gigawords) which are labeled with the CRF, $\textsc{ilr}(f_2)$ can recover all of the performance for POS and more than half of the performance for NER.

[7] Strictly speaking, training on p^{U}_{C} would involve creating many examples weighted by $p_{\text{C}}(y_i \mid \mathbf{x})$ for each original $\mathbf{x}$. But since the posteriors are sharply peaked, using $\text{argmax}_{\mathbf{y}}\, p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ was faster and an adequate approximation for our applications.

3. Analysis

In this section, we try to get a better understanding of when structure compilation would be effective. For the theoretical analysis, we will measure performance in terms of log-loss (negative cross-entropy):

$$\epsilon(p_1, p_2) \stackrel{\text{def}}{=} \mathbb{E}_{p_1(\mathbf{x},\mathbf{y})}[-\log p_2(\mathbf{y} \mid \mathbf{x})], \tag{2}$$

where $p_1(\mathbf{x}, \mathbf{y})$ is the data generating distribution and $p_2(\mathbf{y} \mid \mathbf{x})$ is the model under evaluation. We will also use conditional KL-divergence to quantify approximation error:

$$\kappa(p_1, p_2) \stackrel{\text{def}}{=} \epsilon(p_1, p_2) - \epsilon(p_1, p_1). \tag{3}$$

We are interested in $\epsilon(p^*, p_{\text{I}})$, the loss of the final compiled ILR. As we will later show in (7), this loss can be expressed in terms of two parts: (1) $\epsilon(p^*, p_{\text{C}})$, the loss of the CRF; and (2) $\kappa(p_{\text{C}}, p_{\text{I}})$, the penalty due to structure compilation.

The CRF loss can be decomposed into approximation and estimation errors as follows, using telescoping sums (refer to Table 1 for notation):

$$\begin{aligned}\epsilon(p^*, p_{\text{C}}) \quad = \quad & \underbrace{\epsilon(p^*, p_{\text{C}}) - \epsilon(p^{\text{L}}, p_{\text{C}})}_{\text{CRF-EST-ERR}} + \\ & \underbrace{\epsilon(p^{\text{L}}, p_{\text{C}}) - \epsilon(p^{\text{L}}, p_{\text{C}^*})}_{\leq 0} + \\ & \underbrace{\epsilon(p^{\text{L}}, p_{\text{C}^*}) - \epsilon(p^*, p_{\text{C}^*})}_{\text{CRF-EST-ERR}} + \underbrace{\epsilon(p^*, p_{\text{C}^*})}_{\text{CRF-APX-ERR}}.\end{aligned} \tag{4}$$

The second term on the RHS is non-positive because because p_{C} is chosen to minimize log-loss on p^{L}. The first and third terms are the estimation errors resulting from using p^{L} instead of p^*; these will be uniformly bounded in Section 3.2. Finally, the last term is an approximation error reflecting the modeling limitations of the CRF.

The structure compilation penalty can be decomposed analogously:

$$\begin{aligned}\kappa(p^*_{\text{C}}, p_{\text{I}}) \quad = \quad & \underbrace{\kappa(p^*_{\text{C}}, p_{\text{I}}) - \kappa(p^{\text{U}}_{\text{C}}, p_{\text{I}})}_{\text{ILR-EST-ERR}} + \\ & \underbrace{\kappa(p^{\text{U}}_{\text{C}}, p_{\text{I}}) - \kappa(p^{\text{U}}_{\text{C}}, p_{\text{I}^*})}_{\leq 0} + \\ & \underbrace{\kappa(p^{\text{U}}_{\text{C}}, p_{\text{I}^*}) - \kappa(p^*_{\text{C}}, p_{\text{I}^*})}_{\text{ILR-EST-ERR}} + \underbrace{\kappa(p^*_{\text{C}}, p_{\text{I}^*})}_{\text{ILR-APX-ERR}}.\end{aligned} \tag{5}$$

We would now like to combine $\epsilon(p^*, p_{\text{C}})$ and $\kappa(p_{\text{C}}, p_{\text{I}})$ to get a handle on $\epsilon(p^*, p_{\text{I}})$, but unfortunately, KL-divergence does not satisfy a triangle inequality. We can, however, derive the following approximate triangle inequality, where we pay an extra multiplicative factor (see Appendix A.1 for the proof):

Theorem 1. *Consider a conditional exponential family $\mathcal{P}$ with features ϕ. Let Θ be a compact subset of parameters, and define $\mathcal{P}_\Theta = \{p_\theta : \theta \in \Theta\}$. For any $p_1, p_2 \in \mathcal{P}_\Theta$ and any distribution p_0 such that $p'_0 = \operatorname{argmax}_{p \in \mathcal{P}} \mathbb{E}_{p^*(\mathbf{x})p_0(\mathbf{y}|\mathbf{x})} \log p(\mathbf{y} \mid \mathbf{x}) \in \mathcal{P}_\Theta$,*

$$\begin{aligned}\mathbb{E}_{p^*(\mathbf{x})}\text{KL}\left(p_0(\mathbf{y} \mid \mathbf{x}) \,||\, p_2(\mathbf{y} \mid \mathbf{x})\right) \leq \\ \alpha\left[\mathbb{E}_{p^*(\mathbf{x})}\text{KL}\left(p_0(\mathbf{y} \mid \mathbf{x}) \,||\, p_1(\mathbf{y} \mid \mathbf{x})\right) + \right. \\ \left. \mathbb{E}_{p^*(\mathbf{x})}\text{KL}\left(p_1(\mathbf{y} \mid \mathbf{x}) \,||\, p_2(\mathbf{y} \mid \mathbf{x})\right)\right],\end{aligned} \tag{6}$$

where $\alpha = 2\frac{\sup_{\theta\in\Theta} \lambda_{\max}(\mathbb{E}\,\text{var}_\theta(\phi|\mathbf{x}))}{\inf_{\theta\in\Theta} \lambda^+_{\min}(\mathbb{E}\,\text{var}_\theta(\phi|\mathbf{x}))}$. Here, $\lambda_{\max}(\Sigma)$ and $\lambda^+_{\min}(\Sigma)$ are the largest and smallest nonzero eigenvalues of Σ, respectively.

Theorem 1 generalizes Lemma 3 of Crammar et al. (2007) to conditional distributions and the case where p_0 is not necessarily in an exponential family.

Let us apply Theorem 1 with p^*, p_{C}, p_{I}. We then add the conditional entropy $\mathbb{E}\,H(p^*(\mathbf{y} \mid \mathbf{x}))$ to the LHS of the resulting inequality and $\alpha\mathbb{E}\,H(p^*(\mathbf{y} \mid \mathbf{x}))$ to the RHS (note that $\alpha \geq 1$), thus obtaining a bound for the total loss of the final compiled ILR:

$$\begin{aligned}\epsilon(p^*, p_{\text{I}}) \quad \leq \quad & \alpha(\epsilon(p^*, p_{\text{C}}) + \kappa(p^*_{\text{C}}, p_{\text{I}})) \\ \leq \quad & \alpha(\textsc{crf-apx-err} + \textsc{ilr-apx-err}) + \\ & 2\alpha(\textsc{crf-est-err} + \textsc{ilr-est-err}).\end{aligned} \tag{7}$$

In the remaining sections, we analyze the various pieces of this bound.

3.1. Approximation Error

We start by analyzing $\textsc{ilr-apx-err} = \kappa(p^*_{\text{C}}, p_{\text{I}^*})$, which measures how well the ILR can approximate the CRF. Specifically, we show that $\kappa(p^*_{\text{C}}, p_{\text{I}^*})$ decomposes into three terms, each reflecting a limitation of the ILR: (1) I_{C}, the inability to produce a coherent output; (2) I_{N}, the inability to express nonlinearities; and (3) I_{G}, the inability to use information about the input outside a local window. The following theorem formalizes these concepts (see Appendix A.2 for the proof):

Theorem 2 (Decomposition of approximation error)**.** $\kappa(p^*_{\text{C}}, p_{\text{I}^*}) = I_{\text{C}} + I_{\text{N}} + I_{\text{G}}$*, where*

$$\begin{aligned}I_{\text{C}} \quad &= \quad \mathbb{E}_{p^*(\mathbf{x})}\text{KL}\left(p_{\text{C}}(\mathbf{y} \mid \mathbf{x}) \,||\, \prod_{i\in V} p_{\text{C}}(y_i \mid \mathbf{x})\right), \\ I_{\text{N}} \quad &= \quad \mathbb{E}_{p^*(\mathbf{x})} \sum_{i\in V} \text{KL}\left(p_{\text{C}}(y_i \mid \mathbf{x}) \,||\, p_{\text{A}^*}(y_i \mid \mathbf{x})\right), \\ I_{\text{G}} \quad &= \quad \mathbb{E}_{p^*(\mathbf{x})} \sum_{i\in V} \text{KL}\left(p_{\text{A}^*}(y_i \mid \mathbf{x}) \,||\, p_{\text{I}^*}(y_i \mid \mathbf{x})\right),\end{aligned}$$

with $p_{\text{A}^*} = \text{TR}(\text{ILR}, f_\infty, p_{\text{C}}^*)$*, where the node features* f_∞ *are constructed from* f_1 *with an expansion radius of* ∞ *(so the entire input sequence* $\mathbf{x}$ *is used).*

Coherence One advantage of structured models is their ability to predict the output jointly. This could be especially important for NER, where the output tag sequence actually codes for a segmentation of the input. I_{C} measures the information lost by using the independent marginals of the CRF rather than the joint.[8]

For a chain CRF defined on $\mathbf{y} = (y_1, \ldots, y_\ell)$, one can check that I_{C} is the sum of mutual information terms along the edges: $I_{\text{C}} = \mathbb{E} \sum_{i=1}^{\ell-1} I(y_i, y_{i+1} \mid \mathbf{x})$. We computed I_{C} empirically: for POS, $I_{\text{C}} = 0.003$ and for NER, $I_{\text{C}} = 0.009$ (normalized by sequence length). Also, when we predict using the CRF marginals, the performance drops from 76.3% to 76.0% for NER but stays at 95.0% for POS. From these results, we conclude that coherence is not a big concern for our applications, although it is slightly more serious for NER, as we would expect.

Nonlinearities Although we think of CRFs as linear models, their marginal predictions actually behave nonlinearly. I_{N} captures the importance of this nonlinearity by comparing $p_{\text{C}}(y_i \mid \mathbf{x})$ and $p_{\text{A}^*}(y_i \mid \mathbf{x})$. Both depend on $\mathbf{x}$ through the same sufficient statistics $f_1(\cdot, \mathbf{x}, \cdot)$, but p_{A^*} acts on these sufficient statistics in a linear way whereas p_{C} allows the information about $\mathbf{x}$ to propagate in a nonlinear way through the other hidden labels $\mathbf{y}_{-i} = \{y_j : j \neq i\}$ in a manner roughly similar to that of a neural network. However, one difference is that the parameters of $p_{\text{C}}(y_i \mid \mathbf{x})$ are learned with $\mathbf{y}_{-i}$ fixed at training time; they are not arbitrary hidden units in service of y_i. A neural network therefore offers more expressive power but could be more difficult to learn.

We would like to measure the effect of nonlinearity empirically, but p_{A^*} has too many parameters to learn effectively. Thus, instead of comparing p_{A^*} and p_{C}, we compare p_{I^*} and a *truncated CRF* p_{TC}, which we train as follows:[9] For each labeled example $(\mathbf{x}, \mathbf{y})$ (which are sequences of length ℓ), we create ℓ new examples: $(\mathbf{x}_{i-L-r..i+L+r}, \mathbf{y}_{i-r..i+r})$ for $i = 1, \ldots, \ell$, where r is the expansion radius (Section 2.3). Then we train a CRF with features f_1 on these new examples to get p_{TC}. To label node i, we use $p_{\text{TC}}(y_i \mid \mathbf{x}_{i-L-r..i+L+r})$, marginalizing out $\mathbf{y}_{i-r..i-1,i+1..i+r}$. By this setup, both $p_{\text{TC}}(y_i \mid \mathbf{x})$ and $p_{\text{I}^*}(y_i \mid \mathbf{x})$ depend on $\mathbf{x}$ through the same features. Table 2 compares the NER performance of p_{TC} and p_{I}. As we can see, the truncated CRF significantly outperforms the ILR, demonstrating the power of nonlinearities.

Expansion radius r	1	2	3	∞
compiled ILR	0.725	0.727	0.721	—
truncated CRF	0.748	0.760	0.762	0.760

Table 2. NER F_1 (2K originally labeled examples, 200K automatically labeled examples for structure compilation) showing the importance of nonlinearity. Both the ILR and truncated CRF depend on the input $\mathbf{x}$ in the same way, but only the latter permits nonlinearities.

Global information I_{G} compares p_{A^*} and p_{I^*}, both of which are independent linear classifiers. The difference is that p_{A^*} uses all features of $\mathbf{x}$, while p_{I^*} uses only features of $\mathbf{x}$ in a local window.

Instead of comparing p_{A^*} and p_{I^*}, we compare their nonlinear counterparts p_{C} and p_{TC}. From Table 2, we can see that the truncated CRF with just an expansion radius of 2 has the same performance as the original CRF (expansion radius ∞). Therefore, we suspect that the features on $\mathbf{x}$ outside a local window have little impact on performance, and that most of the approximation error is due to lacking nonlinearities.

3.2. Estimation Error

In this section, we quantify the estimation errors for the CRF and ILR. First, we establish a general result about the estimation error of log-loss for exponential families. Our strategy is to uniformly bound the difference between empirical and expected log-losses across all parameter values of the exponential family. Our proof uses covering numbers and is based on Collins (2001), which derived an analogous result for a 0-1 margin loss.

Assume our parameters and features are bounded: $\Theta = \{\theta : ||\theta||_2 \leq B\}$ and $R = \sup_{\mathbf{x},\mathbf{y}} ||\phi(\mathbf{x}, \mathbf{y})||_2$. The following theorem relates the difference between empirical and expected log-loss to the number of examples n (see Appendix for proof):

Theorem 3. *For* $\delta > 0$*, with probability* $\geq 1 - \delta$*, for* $|\epsilon(p^*, p_\theta) - \epsilon(p^{\text{L}}, p_\theta)| \leq \eta$ *to hold for all* $\theta \in \Theta$*, it suffices to use* $n = \Omega(B^4 R^4 \log^2 |\mathcal{Y}| \eta^{-4} \log(1/\delta))$ *examples, where we have suppressed logarithmic factors.*

The above result gives uniform convergence of $\epsilon(\cdot, \cdot)$, which allows us to bound CRF-EST-ERR. In order to bound ILR-EST-ERR, we need uniform convergence of

[8] On the other hand, if we evaluate predictions using Hamming distance, it could actually be better to use the marginals. In that case, coherence is irrelevant.

[9] Truncated CRFs are closely related to piecewise-trained CRFs (Sutton & McCallum, 2005).

$\kappa(\cdot,\cdot)$ (5). This requires the convergence of one additional term: $|\epsilon(p_{\mathrm{C}}^{\mathrm{U}}, p_{\mathrm{C}}) - \epsilon(p_{\mathrm{C}}^{*}, p_{\mathrm{C}})| \xrightarrow{P} 0$ (p_{C} is non-random in this context). The asymptotics for n in Theorem 3 therefore remain unchanged.

We now apply Theorem 3 to the CRF and ILR with the features described in Section 2.1. For both models, $\log|\mathcal{Y}| = K|V|$. Where they differ is on the norms of the parameters and features (B and R). Let d_1 be the total number of features in the base feature set; d_2, the expanded feature set. Let c_k be the number of nonzero entries in $f_k(y_i, \mathbf{x}, i)$. Natural language processing is typified by sparse binary feature vectors, so $d_k \gg c_k$. For the CRF, $||\phi(\mathbf{x}, \mathbf{y})||_2$ is bounded by $R \leq \sqrt{c_1|V|^2 + |E|^2} \leq \sqrt{c_1}|V| + |E|$. The ILR has no edge potentials so $R \leq \sqrt{c_2}|V|$. In general, R is small for NLP applications.

On the other hand, $B \sim \sqrt{d_1}$ for the CRF and $B \sim \sqrt{d_2}$ for the ILR if the magnitude of the individual parameters are comparable. Since d_2 is significantly larger than d_1, the generalization bound for the ILR is much worse than for the CRF. While comparing upper bounds is inconclusive, showing that one upper bound is larger than another via the same methodology is weak evidence that the actual quantities obey a similar inequality.

4. Parsing Experiments

We now apply structure compilation to parsing. In this case, our structured model is a log-linear parser (Petrov & Klein, 2008), which we would like to replace with independent logistic regressions. For simplicity, we consider unlabeled binary trees. We could always use another independent classifier to predict node labels.

Standard parsing algorithms require $O(|G|\ell^3)$ time to parse a sentence with ℓ words, where $|G|$ is the size of the grammar. The ILR-based parser we will describe only requires $O(\ell^3)$ time. An extension to the labeled case would require $O(\ell^3 + K\ell)$ time, where K is the number of labels. This is a significant gain, since the grammars used in real-world parsers are quite large ($|G| \gg \ell, K$).

4.1. Independent Model

An example parse tree is shown in Figure 1 (recall that we do not predict the labels). For each span (i, j) ($1 \leq i < j \leq \ell$), the independent model predicts whether a node in the parse tree dominates the span. For example, of the 14 non-trivial spans in the sentence in Figure 1, the positively labeled spans are $(1, 2)$, $(5, 6)$, $(4, 6)$, and $(3, 6)$. To parse a sentence at test time, we first use the independent model to assign each span a probability and then use dynamic programming to find the most likely tree.

The independent model uses the following features evaluated (a) within the span, (b) at the boundary of the span, and (c) within a window of 3 words on either side of the span: identity, parts-of-speech, prefixes/suffixes (length 1-3), and case patterns. Additional features include 3- and 4-grams of the words and POS tags that occur within the span; the entire POS sequence; the entire word sequence; the 3-character suffix sequence; the case sequence within the span; the length of the span; the position of the span relative to the entire sentence; the number of verbs, conjunctions and punctuation marks within the span; and whether the span centers on a conjunction and has symmetric POS tags to the left and right.

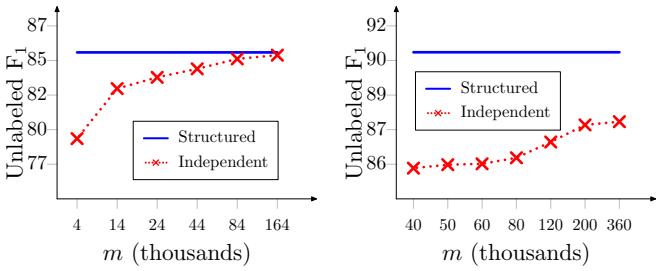
4.2. Results

In all of our experiments, we evaluate according to the F_1 score on unlabeled, binarized trees (using right-binarization). This scoring metric is slightly non-standard, but equally difficult: a parser that achieves a labeled F_1 (the usual metric) of 89.96% on the Treebank test data (section 23) achieves 90.29% under our metric; on 10% of the data, the two metrics are 82.84% and 85.16%, respectively.

To test our independent parser, we trained a structured parser on 10% of the WSJ portion of the Penn Treebank (4K sentences). We then used the structured parser to parse 160K unlabeled sentences,[10] which were then used, along with the original 4K sentences, to train the independent model. Figure 3(a) shows the F_1 scores of the various models. When only 4K is used, the independent parser achieves a score of 79.18%, whereas the structured parser gets 85.16%. With more automatically labeled examples, the performance of the independent parser approaches that of the structured parser (84.97% with 164K sentences).

On the other hand, if we trained the structured parser on 40K sentences, then the independent parser has a much harder time catching up, improving from 85.42% (40K sentences) to just 87.57% (360K sentences) compared to the structured parser's 90.78% (Figure 3(b)). Since parsing is a much more complex task compared to POS or NER, we believe that richer features would be needed to reduce this gap.

[10] These sentences are from the North American National Corpus, selected by test-set relativization.



(a) 4K sentences (b) 40K sentences

Figure 3. A comparison of the structured and independent parsers when the structured parser is trained on 4K (a) and 40K (b) sentences. m is the number of examples (original + automatically labeled) used to train the independent parser.

5. Conclusion

The importance of deploying fast classifiers at test time motivated our investigation into the feasibility of replacing structure with features. We presented a method to compile structure into features and conducted theoretical and empirical analyses of the estimation and approximation errors involved in structure compilation. We hope that a better understanding of the role structure plays will lead to more computationally efficient methods that can still reap the benefits of structure.

References

Bucilă, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. *International Conference on Knowledge Discovery and Data Mining (KDD)*.

Collins, M. (2001). Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. *International Workshop on Parsing Technologies*.

Crammar, K., Kearns, M., & Wortman, J. (2007). Learning from multiple sources. *Advances in Neural Information Processing Systems (NIPS)*.

Craven, M. W. (1996). *Extracting comprehensible models from trained neural networks*. Doctoral dissertation, University of Wisconsin at Madison.

Csiszár, I., & Shields, P. (2004). Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory, 1*, 417–528.

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

Petrov, S., & Klein, D. (2008). Discriminative log-linear grammars with latent variables. *Advances in Neural Information Processing Systems (NIPS)*.

Pollard, D. (1984). *Convergence of stochastic processes*. Springer-Verlag.

Punyakanok, V., Roth, D., Yih, W., & Zimak, D. (2005). Learning and inference over constrained output. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Sutton, C., & McCallum, A. (2005). Piecewise training of undirected models. *Uncertainty in Artificial Intelligence (UAI)*.

Zhang, T. (2002). Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research, 2*, 527–550.

A. Proofs

Lemma 1 gives conditions under which a conditional KL-divergence can be decomposed exactly. Lemma 2 specializes to the exponential family. These lemmas are variants of standard results from information geometry (see Csiszár and Shields (2004)). We will use them in the proofs of Theorems 1 and 2.

Lemma 1 (Conditional Pythagorean identity)**.** *Let* $d(p, p') = \mathbb{E}_{p^*(\mathbf{x})p(\mathbf{y}|\mathbf{x})} \log p'(\mathbf{y} \mid \mathbf{x})$ *be the negative conditional cross-entropy. For any three conditional distributions* p_0, p_1, p_2*, if* $d(p_0, p_1) = d(p_1, p_1)$ *and* $d(p_0, p_2) = d(p_1, p_2)$*, then*

$$\begin{aligned}\mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_0(\mathbf{y} \mid \mathbf{x}) \,||\, p_2(\mathbf{y} \mid \mathbf{x})\right) = \\ \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_0(\mathbf{y} \mid \mathbf{x}) \,||\, p_1(\mathbf{y} \mid \mathbf{x})\right) + \\ \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_1(\mathbf{y} \mid \mathbf{x}) \,||\, p_2(\mathbf{y} \mid \mathbf{x})\right).\end{aligned} \tag{8}$$

Proof. Use the fact that $\mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p \,||\, p'\right) = d(p, p) - d(p, p')$ and perform algebra. □

Lemma 2 (Conditional Pythagorean identity for exponential families)**.** *Let* $\mathcal{P}$ *be a conditional exponential family. If* $p_1 = \operatorname{argmax}_{p \in \mathcal{P}} \mathbb{E}_{p^*(\mathbf{x})p_0(\mathbf{y}|\mathbf{x})} \log p(\mathbf{y} \mid \mathbf{x})$ *and* $p_2 \in \mathcal{P}$*, (8) holds for* p_0, p_1, p_2*.*

Proof. Since p_1 is the maximum likelihood solution, $\mu \stackrel{\text{def}}{=} \mathbb{E}_{p^*(\mathbf{x})p_0(\mathbf{y}|\mathbf{x})}\phi(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{p^*(\mathbf{x})p_1(\mathbf{y}|\mathbf{x})}\phi(\mathbf{x}, \mathbf{y})$, where ϕ are the features of $\mathcal{P}$. Then for $p \in \mathcal{P}$ with parameters θ, $d(p_0, p) = \mu^\top \theta - \mathbb{E}\, A(\theta; \mathbf{x}) = d(p_1, p)$ (follows from (1)). Plug in $p = p_1, p_2$ and apply Lemma 1. □

A.1. Proof of Theorem 1

Proof. The first part of the proof is similar to that of Lemma 3 in Crammar et al. (2007). Denote $k(p, p') = \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p(\mathbf{y} \mid \mathbf{x}) \,||\, p'(\mathbf{y} \mid \mathbf{x})\right)$ and $B(\theta) = \mathbb{E}_{p^*(\mathbf{x})} A(\theta; \mathbf{x})$.

The key is to note that for $p, p' \in \mathcal{P}$, the conditional KL-divergence is the residual term in the first-order approximation of B: $k(p, p') = \nabla B(\theta)^\top (\theta - \theta') - (B(\theta) - B(\theta'))$, where θ, θ' are the parameters of p, p'. Thus, we can use Taylor's theorem to get that

$k(p,p') = \frac{1}{2}||\theta - \theta'||^2_{V_{p,p'}}$, where $V_{p,p'} = \nabla^2 B(\tilde{\theta}) = \mathbb{E}\,\mathrm{var}_{\tilde{\theta}}(\phi \mid \mathbf{x})$ for some $\tilde{\theta} \in \Theta$.

One can check that $||\theta_0' - \theta_2||^2_{V_{0,2}} \leq 2[||\theta_0' - \theta_1||^2_{V_{0,2}} + ||\theta_1 - \theta_2||^2_{V_{0,2}}]$, where $V_{0,2} = V_{p_0',p_2}$. By definition of α, $V_{0,2} \preceq \frac{\alpha}{2} V_{0,1}$ and $V_{0,2} \preceq \frac{\alpha}{2} V_{1,2}$,[11] so $||\theta_0' - \theta_2||^2_{V_{0,2}} \leq \alpha[||\theta_0' - \theta_1||^2_{V_{0,1}} + ||\theta_1 - \theta_2||^2_{V_{1,2}}]$. Rewritten another way: $k(p_0', p_2) \leq \alpha[k(p_0', p_1) + k(p_1, p_2)]$.

Applying Lemma 2 twice to p_0, p_0', p_1 and p_0, p_0', p_2 yields $k(p_0,p_2) - k(p_0,p_0') \leq \alpha[k(p_0,p_1) - k(p_0,p_0') + k(p_1,p_2)]$. Subtracting $k(p_0,p_0')$ from both sides and noting $\alpha \geq 1$ yields the theorem. □

A.2. Proof of Theorem 2

Proof. Define $p_{\mathrm{MC}}(\mathbf{y} \mid \mathbf{x}) = \prod_{i \in V} p_{\mathrm{C}}(y_i \mid \mathbf{x})$. Check that $d(p_{\mathrm{C}}(\mathbf{y} \mid \mathbf{x}), \prod_{i \in V} p(y_i \mid \mathbf{x})) = d(\prod_{i \in V} p_{\mathrm{C}}(y_i \mid \mathbf{x}), \prod_{i \in V} p(y_i \mid \mathbf{x}))$ for any p, in particular, p_{MC} and p_{I^*}. Thus we can apply Lemma 1 with $p_{\mathrm{C}}, p_{\mathrm{MC}}, p_{\mathrm{I}^*}$ to get $\kappa(p^*_{\mathrm{C}}, p_{\mathrm{I}^*}) = I_{\mathrm{C}} + \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_{\mathrm{MC}}(\mathbf{y} \mid \mathbf{x}) \,||\, p_{\mathrm{I}^*}(\mathbf{y} \mid \mathbf{x})\right)$.

Since f_∞ is a superset of f_2, both p_{I^*} and p_{A^*} are members of the f_∞-exponential family, with p_{A^*} being the maximum likelihood solution. Apply Lemma 2 with $p_{\mathrm{MC}}, p_{\mathrm{A}^*}, p_{\mathrm{I}^*}$ to get $\mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_{\mathrm{MC}}(\mathbf{y} \mid \mathbf{x}) \,||\, p_{\mathrm{I}^*}(\mathbf{y} \mid \mathbf{x})\right) = I_{\mathrm{N}} + I_{\mathrm{G}}$. □

A.3. Proof of Theorem 3

We use covering numbers to bound the complexity of the class of log-losses. Our proof is inspired by Collins (2001), who works with a 0-1 margin-based loss. Define the loss class:

$$\mathcal{M} \stackrel{\mathrm{def}}{=} \{(\mathbf{x},\mathbf{y}) \mapsto -\log p_\theta(\mathbf{y} \mid \mathbf{x}) : \theta \in \Theta\}. \tag{9}$$

We first show that the elements of $\mathcal{M}$ are bounded:

Lemma 3. *For each $f \in \mathcal{M}$ and $(\mathbf{x},\mathbf{y})$, $0 \leq f(\mathbf{x},\mathbf{y}) \leq L$, where $L \stackrel{\mathrm{def}}{=} BR(1 + \log|\mathcal{Y}|)$.*

Proof. The lower bound holds since probabilities are bounded above by 1. For the upper bound, consider the absolute value of the two terms in (1) separately. For the linear term, $|\phi(\mathbf{x},\mathbf{y})^\top \theta| \leq BR$ by the Cauchy-Schwartz inequality. The log-partition function can be bounded by $BR\log|\mathcal{Y}|$ by applying the linear term result to the exponent. Add the two bounds. □

Theorem 1 of Zhang (2002) (originally due to Pollard (1984)) applied to $\mathcal{M}$: With probability $\geq 1 - \delta$,

$$P\left(\sup_{\theta \in \Theta} |\epsilon(p^*, p_\theta) - \epsilon(p^{\mathrm{L}}, p_\theta)| > \eta\right) \leq 8\mathcal{N}_1(\mathcal{M}, \eta/8, n) \exp\left\{\frac{-n\eta^2}{128L^2}\right\}, \tag{10}$$

where $\mathcal{N}_p(\mathcal{F}, \epsilon, n)$, the covering number of function class $\mathcal{F}$, is the supremum over all points $\mathbf{z}_1, \ldots, \mathbf{z}_n$, of the size of the smallest cover $\{g_1, \ldots, g_k\}$ such that for all $f \in \mathcal{F}$, there exists a g_j in the cover with $(\frac{1}{n}\sum_{i=1}^n |f(\mathbf{z}_i) - g_j(\mathbf{z}_i)|^p)^{1/p} \leq \epsilon$.

We now upper bound $\mathcal{N}_\infty(\mathcal{M}, \epsilon/8, n)$, adapting the method used in Collins (2001). First define the set of linear functions:

$$\mathcal{L} \stackrel{\mathrm{def}}{=} \left\{\mathbf{v} \mapsto \theta^\top \mathbf{v} : \theta \in \Theta\right\}. \tag{11}$$

Theorem 4 of Zhang (2002) (with $p = q = 2$) allows us to bound the complexity of this class:

$$\log_2 \mathcal{N}_\infty(\mathcal{L}, \epsilon, n) \leq 36(BR/\epsilon)^2 \log_2(2\lceil 4BR/\epsilon + 2\rceil n + 1). \tag{12}$$

We now relate the covering numbers of $\mathcal{L}$ and $\mathcal{M}$:

Lemma 4. $\mathcal{N}_\infty(\mathcal{M}, \epsilon, n) \leq \mathcal{N}_\infty(\mathcal{L}, \epsilon/2, n|\mathcal{Y}|)$.

Proof. Let $S = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}$. We will construct a covering of $\mathcal{M}$ (with respect to S) by reducing to the problem of finding a covering of $\mathcal{L}$. Let $\mathbf{v}_{i\mathbf{y}} = \phi(\mathbf{x}^{(i)}, \mathbf{y})$ and $V = \{\mathbf{v}_{i\mathbf{y}} : i = 1, \ldots, n, \mathbf{y} \in \mathcal{Y}\}$. By (12), we can cover $\mathcal{L}$ with respect to V with a set $C_{\mathcal{L}}$. Consider the corresponding set $C_{\mathcal{M}} \subset \mathcal{M}$ (note the natural 3-way bijections between Θ, $\mathcal{L}$, and $\mathcal{M}$).

To prove the lemma, it suffices to show that $C_{\mathcal{M}}$ is a covering of $\mathcal{M}$. Fix some $g \in \mathcal{M}$, which is associated with some $f \in \mathcal{L}$ and $\theta \in \Theta$. There exists a $\tilde{f} \in C_{\mathcal{L}}$ (corresponding to a $\tilde{\theta} \in \Theta$ and a $\tilde{g} \in C_{\mathcal{M}}$) such that $|f(\mathbf{x},\mathbf{y}) - \tilde{f}(\mathbf{x},\mathbf{y})| = |\theta^\top \phi(\mathbf{x},\mathbf{y}) - \tilde{\theta}^\top \phi(\mathbf{x},\mathbf{y})| \leq \epsilon/2$ for all $(\mathbf{x},\mathbf{y}) \in S$ and $\mathbf{y} \in \mathcal{Y}$. We now argue that $\tilde{g}$ is close to g. For each $(\mathbf{x},\mathbf{y}) \in S$,

$$\begin{aligned} g(\mathbf{x},\mathbf{y}) &= -\log p_\theta(\mathbf{y} \mid \mathbf{x}) \\ &= -\theta^\top \phi(\mathbf{x},\mathbf{y}) + \log \sum_{\mathbf{y}' \in \mathcal{Y}} e^{\theta^\top \phi(\mathbf{x},\mathbf{y}')} \\ &\leq -(\tilde{\theta}^\top \phi(\mathbf{x},\mathbf{y}) - \epsilon/2) + \log \sum_{\mathbf{y}' \in \mathcal{Y}} e^{\tilde{\theta}^\top \phi(\mathbf{x},\mathbf{y}') + \epsilon/2} \\ &= \tilde{g}(\mathbf{x},\mathbf{y}) + \epsilon. \end{aligned}$$

Similarly, $g(\mathbf{x},\mathbf{y}) \geq \tilde{g}(\mathbf{x},\mathbf{y}) - \epsilon$. Therefore, $C_{\mathcal{M}}$ is a cover of $\mathcal{M}$. □

Substitute (12) into Lemma 4 to get an expression for $\mathcal{N}_\infty(\mathcal{M}, \epsilon, n)$; substitute that into (10), using the fact that $\mathcal{N}_1 \leq \mathcal{N}_\infty$. Solving for n yields the theorem.

[11] It suffices to consider nonzero eigenvalues because zero eigenvalues correspond to non-identifiable directions, which are the same for all parameters θ.