Efficient Second-Order Gradient Boosting for Conditional Random Fields

Tianqi Chen **Sameer Singh** **Ben Taskar** **Carlos Guestrin**
Computer Science and Engineering, University of Washington, Seattle WA
{tqchen,sameer,taskar,guestrin}@cs.washington.edu

Abstract

Conditional random fields (CRFs) are an important class of models for accurate structured prediction, but effective design of the feature functions is a major challenge when applying CRF models to real world data. Gradient boosting, which is used to automatically induce and select feature functions, is a natural candidate solution to the problem. However, it is non-trivial to derive gradient boosting algorithms for CRFs due to the dense Hessian matrices introduced by variable dependencies. Existing approaches thus use only first-order information when optimizing likelihood, and hence face convergence issues. We incorporate second-order information by deriving a Markov Chain mixing rate bound to quantify the dependencies, and introduce a gradient boosting algorithm that iteratively optimizes an adaptive upper bound of the objective function. The resulting algorithm induces and selects features for CRFs via functional space optimization, with provable convergence guarantees. Experimental results on three real world datasets demonstrate that the mixing rate based upper bound is effective for learning CRFs with non-linear potentials.

1 INTRODUCTION

Many problems in machine learning involve structured prediction, i.e., predicting a group of outputs that depend on each other. Conditional random fields (CRFs) [Lafferty et al., 2001] are among the most successful solutions to this problem. Variants of tree-shaped conditional random fields have been proposed and widely applied to structured prediction problems in domains such as natural language processing [Lafferty et al., 2001, Sha and Pereira, 2003], computer vision [He et al., 2004, Quattoni et al.], and bio-informatics [Vinson et al., 2007]. As opposed to classification models that assume independent output variables, CRFs model the dependencies between various output variables via potential functions. These potential functions are usually defined using a linear combination of carefully engineered features of the input and the output variables. Since these feature functions are crucial for learning accurate models, it is important to ask whether we can automatically induce arbitrary potential functions (via functional space optimization), instead of manually crafting them and/or restricting them to linear combinations.

Gradient boosting [Friedman, 2001], which performs additive training in functional spaces, is a natural candidate for this problem. Effective gradient boosting algorithms such as LogitBoost and its variants [Friedman et al., 1998, Li, 2010, Sun et al., 2012] have been proposed for inducing feature functions for (independent) multi-class classification problems. The key ingredient in these methods is the effective use of second order information via diagonal approximation of Hessian matrices. Unfortunately, it is non-trivial to develop such boosting methods for CRFs, since variable interdependencies introduce dense Hessian matrices that make gradient boosting infeasible due to the computational complexity. Instead, existing boosting approaches either optimize approximate objectives [Torralba et al., Liao et al., 2007] or only take first order information into account when optimizing exact likelihood [Dietterich et al., 2008]. Unfortunately, convergence of the latter method is guaranteed only with small step sizes.

In this paper, we present a novel gradient boosting algorithm inducing non-linear feature functions for tree-shaped CRFs. The CRF training is performed via iteratively optimizing adaptive upper bounds of the loss function, to address the challenge of dense Hessians. The adaptive bounds, which are derived using

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38.

Markov Chain mixing rates, measure the dependency between variables, and accordingly control the conservativeness of the updates. The resulting gradient boosting algorithm, which can be viewed as generalization of LogitBoost to structure prediction problems, optimizes the CRF objective with provable convergence guarantees. Experimental results on three real world datasets demonstrate the effectiveness and efficiency of the proposed algorithm.

2 OVERVIEW OF METHOD

Model Formalization: Given an input $\mathbf{x} \in \mathcal{X}$, a Conditional Random Field (CRF) [Lafferty et al., 2001] defines the distribution over the outputs $\mathbf{y} \in \mathcal{Y}$ as

$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp(\Phi(\mathbf{y}, \mathbf{x}))}{\sum_{\mathbf{y}' \in \mathcal{Y}} \exp(\Phi(\mathbf{y}', \mathbf{x}))} \tag{1}$$

where $\mathcal{Y}$ is the set of possible output combinations, and $\Phi(\mathbf{y}, \mathbf{x})$ captures the dependency between the input and output variables. The model Φ usually factorizes as a sum of unary and pairwise (edge) *potential functions* $\phi_i : \mathcal{X} \to \mathcal{R}$ between individual output variables, expressed as follows:

$$\begin{gathered}\Phi(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{m} \phi_i(\mathbf{x})\mu_i(\mathbf{y}),\ \ \phi_i \in \mathcal{F}, \mu_i(\mathbf{y}) \in N \cup \mathcal{E} \\ \text{subject to } \phi_i = \phi_j \text{ for } (i,j) \in \mathcal{C},\end{gathered} \tag{2}$$

where $N = \{\mathbb{1}(y_t = k)\}, \mathcal{E} = \{\mathbb{1}(y_s = k_1, y_t = k_2)\}$ are the sets of indicator functions for each node and edge state. Each μ_i corresponds to an event $y_t = k$ (for unary potentials) or $y_s = k_1, y_t = k_2$ (for edge potentials). In other words, the potential $\phi_i(\mathbf{x})$ defines the weight for the event $\mu_i(\mathbf{y}) = 1$ in $\Phi(\mathbf{y}, \mathbf{x})$. We use μ as short hand for $\mu(\mathbf{y})$ and view them as a vector of random variables. $\mathcal{F} = \mathcal{F}_N \cup \mathcal{F}_\mathcal{E}$ is the family of all possible node and edge potential functions, which may be infinite in size. The set of constraints in $\mathcal{C}$ represent the equivalence classes in different parts of the model, and is used to define the parameter sharing between potentials that is common to most applications of CRFs. In standard linear-chain CRFs with linear potentials, $\mathcal{F}$ contains linear functions of $\mathbf{x}$, and $\mathcal{C}$ is used to share parameters between the potential functions at different positions. LogitBoost considers arbitrary $\mathcal{F}$, but constrains the model to contain only node potentials (there are no edge potentials). In this paper, we are interested in arbitrary function families $\mathcal{F}$, and focus on tree-shaped $\mathcal{E}$ that allow exact inference of marginals.

Training Objective: Treating functions ϕ in Eq. (2) as the model parameters allows us to view model training as automatic induction of potential functions through functional space optimization. In particular, we generalize the standard CRF objective over training data $\mathcal{D} = \{(\mathbf{y}, \mathbf{x})\}$ to minimization over ϕ of the following:

$$\begin{aligned}L_\mathcal{D}(\phi) &= \sum_{\mathbf{y},\mathbf{x} \in \mathcal{D}} l(\mathbf{y}, \mathbf{x}, \phi) + \sum_c \Omega(\phi_c) \\ &= -\sum_{\mathbf{y},\mathbf{x} \in D} \ln P(\mathbf{y}|\mathbf{x}) + \sum_c \Omega(\phi_c).\end{aligned} \tag{3}$$

Here l is the negative log-likelihood function over each data point. $\sum_c \Omega(\phi_c)$ is a regularization term that measures the complexity of the learned function, and is defined as a sum over the equivalence classes in $\mathcal{C}$. In standard CRFs, for example, Ω is often the square of the L_2 norm of the parameter vector. This generalized objective function encourages us to select *predictive* (i.e., optimizes l) and *simple* (i.e., optimizes Ω) functions as potentials of a CRF.

Challenges for Function Learning: Since the model parameters in this formulation are functions, Eq. (3) cannot be directly optimized using traditional optimization techniques. Instead, we train the model additively: at each iteration t, our proposed algorithm first searches over the functional space $\mathcal{F}$ to find functions $\delta = [\delta_1, \delta_2, \cdots, \delta_m]$ that optimize the objective function $L_\mathcal{D}(\phi^{(t)} + \delta)$, and then adds them to the current model $\phi^{(t+1)} \leftarrow \phi^{(t)} + \delta$. Note that each δ_i is a scalar function of $\mathbf{x}$.

Directly performing such a brute-force search, however, requires a large amount of computation due to the complex nature of the objective function, and is thus infeasible. In the same spirit as LogitBoost for multi-class prediction [Friedman et al., 1998, Li, 2010], we consider the second order Taylor expansion of the negative log-likelihood $l(\mathbf{y}, \mathbf{x}, \phi)$:

$$l(\mathbf{y}, \mathbf{x}, \phi + \delta) \simeq l(\mathbf{y}, \mathbf{x}, \phi) + \delta^T \mathbf{G}(\mathbf{y}, \mathbf{x}) + \frac{1}{2}\delta^T \mathbf{H}(\mathbf{y}, \mathbf{x})\delta. \tag{4}$$

The gradient $\mathbf{G}$ and Hessian $\mathbf{H}$ in Eq.(4) are given by:

$$\mathbf{G}_i = \mu_i(\mathbf{y}) - p_i, \quad \mathbf{H}_{ij} = p_{ij} - p_i p_j. \tag{5}$$

where p_i and p_{ij} are short hand notations for $p_i \triangleq P(\mu_i = 1|\mathbf{x})$, $p_{ij} \triangleq P(\mu_i \mu_j = 1|\mathbf{x})$, i.e., the marginals that can be computed efficiently using dynamic programming for acyclic models. The derivation of $\mathbf{G}$ and $\mathbf{H}$ in Eq. (5), which is the same as used for standard CRFs, is provided in Appendix A for reference. Note that Eq.(5) holds for all i, j pairs, including two special cases: (1) $\mathbf{H}_{ii} = p_i(1 - p_i)$ when $i = j$, and (2) $\mathbf{H}_{ij} = -p_i p_j$ when μ_i and μ_j are mutually exclusive events. Intuitively, $\mathbf{H}_{ij}$ measures the correlation between two events, and is *nonzero* due to the dependencies in the CRF model. Unfortunately, these

dense elements of the Hessian make direct optimization of Eq. (4) still extremely expensive. An existing approach to functional optimization for CRFs, presented in Dietterich et al. [2008], avoids this concern by resorting to the first order approximation of the loss, and only guarantees convergence when the step size is small. An alternative is to iteratively update one δ_i for $i \in \{1, \cdots, m\}$ at a time. This approach would require m inference steps per iteration, and further, is simply not applicable when parameters are shared (i.e. when constraints $\mathcal{C}$ exist).

Our Approach: In this paper, we instead consider an upper bound of Eq. (4). The intuition behind this approach, which will be formalized in the following sections, is as follows: each variable in the CRF depends weakly on variables that are "far" from it. This motivates the use of a diagonal upper bound of the Hessian to construct loss functions, given by the following Lemma.

Lemma 2.1. *Let $\mathcal{U}$ be a index set of potential functions we want to update, and let γ be a vector function that satisfies the following inequality*

$$\gamma_i(\mathbf{y}, \mathbf{x})\mathbf{H}_{ii}(\mathbf{y}, \mathbf{x}) \geq \sum_{j \in \mathcal{U}} |\mathbf{H}_{ij}(\mathbf{y}, \mathbf{x})| \tag{6}$$

Then for $\delta \in \{[\delta_1, \delta_2, \cdots, \delta_m] \mid \delta_i = 0 \text{ for } i \notin \mathcal{U}\}$, the following inequality holds,

$$\begin{aligned} l(\mathbf{y}, \mathbf{x}, \phi + \delta) \leq & l(\mathbf{y}, \mathbf{x}, \phi) + \sum_{i \in \mathcal{U}} \delta_i \mathbf{G}_i(\mathbf{y}, \mathbf{x}) \\ & + \sum_{i \in \mathcal{U}} \frac{1}{2} \gamma_i(\mathbf{y}, \mathbf{x}) \mathbf{H}_{ii}(\mathbf{y}, \mathbf{x}) \delta_i^2(\mathbf{y}, \mathbf{x}) \\ & + o(\delta^2(\mathbf{y}, \mathbf{x})). \end{aligned} \tag{7}$$

We provide the detailed proof of the lemma in Appendix B. For any γ that satisfies the condition, we iteratively optimize $\tilde{L}_{\mathcal{D}}(\phi, \delta)$, which is an upper bound of $L_{\mathcal{D}}(\phi, \delta)$.

$$\begin{aligned} \tilde{L}_{\mathcal{D}}(\phi, \delta) = & L_{\mathcal{D}}(\phi) + \sum_{i \in \mathcal{U}} \sum_{\mathbf{x}, \mathbf{y} \in \mathcal{D}} \mathbf{G}_i(\mathbf{y}, \mathbf{x}) \delta_i(\mathbf{y}, \mathbf{x}) \\ & + \sum_{i \in \mathcal{U}} \frac{1}{2} \sum_{\mathbf{x}, \mathbf{y} \in \mathcal{D}} \gamma_i(\mathbf{y}, \mathbf{x}) \mathbf{H}_{ii}(\mathbf{y}, \mathbf{x}) \delta_i^2(\mathbf{y}, \mathbf{x}) \\ & + \sum_{i \in \mathcal{U}} \big(\Omega(\phi_i + \delta_i) - \Omega(\phi_i) \big). \end{aligned} \tag{8}$$

$\tilde{L}_{\mathcal{D}}(\phi, \delta)$ is composed of $|\mathcal{U}|$ independent loss functions with a regularization term, and can be used to guide the common function search (such as regression tree learning). Iteratively optimizing $\tilde{L}_{\mathcal{D}}$ will result in a gradient boosting algorithm that ensures the convergence of $L_{\mathcal{D}}$ (Proof in Section 4). Furthermore, the form of $\tilde{L}_{\mathcal{D}}$ allows the search of δ_i for $i \in \mathcal{U}$ to be done *in parallel* for each equivalence class defined by $\mathcal{C}$, which gives us further computational benefits. In the next two sections, we will discuss how we can efficiently estimate γ when $\mathcal{U}$ is the index set of all node potentials, and when it is the index set of all edge potentials, using the mixing rate of Markov chain.

3 UPPER BOUND DERIVATION USING A MARKOV CHAIN MIXING RATE

In this section, we will discuss how we can estimate γ when $\mathcal{U}$ is the index set of all node potentials, and the index set of all edge potentials. Conceptually, the choice of γ should be related to the dependencies between the variables in the current model. When the variables in the model are independent from each other γ should be small, and when the variables in the model have strong dependencies, γ should be large. We want to *quantitatively* measure the dependencies in the CRF. Specifically, we describe the connection of the level of dependency between variables to the mixing rate of a Markov chain defined by the conditional distribution $P(\mathbf{y}|\mathbf{x})$. To begin with, we re-express the right side of Eq. (6) using total variation distance, defined by $\|P - Q\|_{tv} = \frac{1}{2} \sum_x |P(x) - Q(x)|$.

Lemma 3.1. *Let $\mathcal{U}$ correspond to the set of all node potentials $\mathcal{U} = \{j | \phi_j \in N\}$, assuming index i corresponds to the event $y_t = k$, i.e., $\mu_i = \mathbb{1}(y_t = k)$, then*

$$\sum_{j \in \mathcal{U}} |\mathbf{H}_{ij}| = 2 p_i \sum_s \|P(y_s|\mathbf{x}, \mu_i = 1) - P(y_s|\mathbf{x})\|_{tv}. \tag{9}$$

Lemma 3.2. *Let $\mathcal{U}$ correspond to the set of all edge potentials $\mathcal{U} = \{j | \phi_j \in \mathcal{E}\}$, then*

$$\sum_{j \in \mathcal{U}} |\mathbf{H}_{ij}| = 2 p_i \sum_{(s,v) \in \mathcal{E}} \|P(y_s, y_v|\mathbf{x}, \mu_i = 1) - P(y_s, y_v|\mathbf{x})\|_{tv}. \tag{10}$$

Note that we abuse the notation slightly here, by using $\mathcal{E}$ to indicate the index set of edges in CRF.

The proof is a re-arrangement of terms, and is provided in Appendix C. Intuitively, the total variation terms in Lemma 3.1 and 3.2 measure how dependent y_s is on the event $y_t = k$. When y_s depends only weakly on y_t, the distance will be small. The complexity of calculating Eq. (9) for all i is quadratic in the number of nodes, which is too expensive to be computed directly for most applications. We instead need an algorithm that scales linearly in the number of nodes.

Intuitively, we expect the dependencies between y_s and y_t to become smaller as we change s to get away from t. We formally state this in the following theorem:

Theorem 3.1. *Mixing rate bound for Markov chain. Assume y_t, y_s and y_v form a Markov chain $y_t \to y_s \to y_v$, conditioned on $\mathbf{x}$, i.e., $P(y_v|y_s, y_t, \mathbf{x}) = P(y_v|y_s, \mathbf{x})$ holds. Define $d(s,t,k) \triangleq \|P(y_s|\mathbf{x}, y_t = k) - P(y_s|\mathbf{x})\|_{tv}$, and $\alpha_{s,v} \triangleq [1 - \sum_j \min_i P(y_v = j|y_s = i, \mathbf{x})]$ Then, the total variation $d(v,t,k)$ can be bounded by*

$$d(v,t,k) \leq \alpha_{s,v} d(s,t,k). \tag{11}$$

Proof. Define notation: $M_{ij} \triangleq P(y_v = j|y_s = i, \mathbf{x})$ and $Q_j \triangleq \min_i M_{ij}$, then

$$\begin{aligned}
&2d(v,t,k) \\
&= \sum_j |P(y_v = j|y_t = k, \mathbf{x}) - P(y_v = j|\mathbf{x})| \\
&= \sum_j |\sum_i M_{ij} P(y_s = i|y_t = k, \mathbf{x}) - \sum_i M_{ij} P(y_s = i|\mathbf{x})| \\
&= \sum_j |\sum_i (M_{ij} - Q_j)[P(y_s = i|y_t = k, \mathbf{x}) - P(y_s = i|\mathbf{x})]| \\
&\leq \sum_j \sum_i (M_{ij} - Q_j)|P(y_s = i|y_t = k, \mathbf{x}) - P(y_s = i|\mathbf{x})| \\
&= \sum_i (1 - \sum_j Q_j)|P(y_s = i|y_t = k, \mathbf{x}) - P(y_s = i|\mathbf{x})| \\
&= 2\alpha_{s,v} d(s,t,k)
\end{aligned}$$

□

The derivation of Theorem 3.1 is inspired, in spirit, by the mixing rate bounds of time homogeneous Markov Chains [Levin et al., 2008][1]. Intuitively, Theorem 3.1 shows that the dependency decays exponentially as s moves away from t. The following corollary holds as a direct consequence of the theorem.

Corollary 3.1. *Let $q = [q(1), q(2), \cdots q(n)]$ be the path in $\mathcal{E}$ from t to s (i.e., $q(1) = t, q(n) = s$) then we can bound $d(t,s,k)$ using $d(t,t,k)$ times the decay ratio α along the path,*

$$d(s,t,k) \leq \prod_i^{n-1} \alpha_{q(i),q(i+1)} d(t,t,k). \tag{12}$$

In the case when $\mathcal{E}$ is a chain, Corollary 3.1 simplifies to $d(s,t,k) \leq \prod_{h=t}^{s-1} \alpha_{h,h+1} d(t,t,k)$ when $s > t$, and $d(s,t,k) \leq \prod_{h=s+1}^{t} \alpha_{h,h-1} d(t,t,k)$ when $s < t$. An important property of Theorem 3.1 is that the position specific rate $\alpha_{s,v}$ can be computed efficiently (complexity analysis in Sec. 4). We still need to calculate $d(t,t,k)$, which is given by the following lemma.

Lemma 3.3. *Let M correspond to the index set of μ_i such that: 1) μ_i, μ_j are mutually exclusive (i.e., $\mu_i \mu_j = 0$ for $i \neq j, i, j \in M$); 2) $\sum_{j \in M} P(\mu_i = j|\mathbf{x}) = 1$. Then the following identity holds*

$$\begin{aligned}
&\frac{1}{2} \sum_{j \in M} |P(\mu_j = 1|\mu_i = 1, \mathbf{x}) - P(\mu_j = 1|\mathbf{x})| \\
&\qquad = 1 - P(\mu_i = 1|\mathbf{x})
\end{aligned} \tag{13}$$

The proof is given in Appendix D. From Lemma 3.3, it follows that $d(t,t,k) = 1 - p_i$. We will use Lemma 3.3 and Corollary 3.1 to efficiently estimate γ in next section.

4 GRADIENT BOOSTING FOR CONDITIONAL RANDOM FIELDS

In this section, we will present our gradient boosting algorithm. We will give estimation of γ for $\mathcal{U}$ (the index set of all node potentials or edge potentials), given by the following two theorems.

Theorem 4.1. *Let $\mathcal{U}$ be the index set of all node potentials, assume $\mu_i = \mathbb{1}(y_t = k)$ and define $\mathcal{Q}(s,t)$ to be the set of all edges in the path from s to t, then*

$$\gamma_i^{(n)}(\mathbf{y}, \mathbf{x}) \triangleq 2(1 + \sum_{s \neq t, s \in N} \prod_{(a,b) \in \mathcal{Q}(s,t)} \alpha_{b,a}) \tag{14}$$

satisfies Eq.(6). Here α is defined in Theorem 3.1.

Theorem 4.2. *Let $\mathcal{U}$ be the index the set of all edge potentials, assume $\mu_i = \mathbb{1}(y_t = k_1, y_{t'} = k_2)$. For $(a,b) \in \mathcal{E}$, define $\mathcal{V}((a,b);a)$ to be the set of nodes that are closer to a than b,*

$$\mathcal{V}((a,b);a) \triangleq \{s \in N | (b,a) \notin \mathcal{Q}(s,a), s \neq a, s \neq b\}$$

then

$$\begin{aligned}
\gamma_i^{(e)}(\mathbf{y}, \mathbf{x}) \triangleq 2\Big[3 &+ \sum_{s \in \mathcal{V}((t,t');t)} \prod_{(a,b) \in \mathcal{Q}(s,t)} \alpha_{b,a} \\
&+ \sum_{s \in \mathcal{V}((t',t);t')} \prod_{(a,b) \in \mathcal{Q}(s,t')} \alpha_{b,a}\Big]
\end{aligned} \tag{15}$$

satisfies Eq.(6), with the α as in Theorem 4.1.

Both theorems can be proved by using Corollary 3.1 and Lemma 3.3 to bound the total variation distance. Intuitively, we bound each of the total variation distances between two nodes' states by recursively applying the mixing rate bound along the path between the two nodes, resulting in the bounds given by the theorems. We include the proof of Theorem 4.1 here, and leave the proof of Theorem 4.2 for Appendix E.

Proof. **Proof for Theorem 4.1** Basically, we want to bound the total variation distance given by Eq. (9) in

[1] Our proof is actually for time inhomogeneous Markov Chains.

Lemma 3.1,

$$\begin{aligned}
&2p_i \sum_s \|P(y_s|\mathbf{x}, y_s = k) - P(y_s|\mathbf{x})\|_{tv} \\
&= 2p_i[d(t,t,k) + \sum_{s\neq t} d(s,t,k)] \\
&\leq 2p_i[d(t,t,k) + \sum_{s\neq t} d(t,t,k) \prod_{(a,b)\in\mathcal{Q}(s,t)} \alpha_{b,a}] \\
&= 2p_i(1-p_i)[1 + \sum_{s\neq t} \prod_{(a,b)\in\mathcal{Q}(s,t)} \alpha_{b,a}]
\end{aligned}$$

Here the inequality is given by Corollary 3.1 ($d(s,t,k) \leq d(t,t,k)\prod_{(a,b)\in\mathcal{Q}(s,t)} \alpha_{b,a}$), and the last equality is given by Lemma 3.3 ($d(t,t,k) = 1-p_i$). Recall that $\mathbf{H}_{ii} = p_i(1-p_i)$, we have proved Theorem 4.1. □

The calculation of γ can be performed using dynamic programming. To explain the algorithm clearly, let us define an auxiliary message variable

$$\beta_{s\to t} \triangleq \alpha_{t,s} \sum_{h\in\mathcal{V}((s,t);s)} \prod_{(a,b)\in\mathcal{Q}(h,s)} (\alpha_{b,a} + 1) \qquad (16)$$

We can rearrange $\gamma_i^{(n)}$ in Theorem 4.1 in terms of β, where the index i satisfies $\mu_i = \mathbb{1}(y_t = k)$, as

$$\gamma_i^{(n)}(\mathbf{y}, \mathbf{x}) = 2\Big(1 + \sum_{s:(s,t)\in\mathcal{E}} \beta_{s\to t}\Big).$$

Similarly for $\gamma_i^{(e)}$ in Theorem 4.2, where the index i satisfies $\mu_i = \mathbb{1}(y_t = k_1, y_{t'} = k_2)$, we have

$$\gamma_i^{(e)}(\mathbf{y}, \mathbf{x}) = 2\Big(3 + \sum_{s:(s,t)\in\mathcal{E}, s\neq t'} \beta_{s\to t} + \sum_{s:(s,t')\in\mathcal{E}, s\neq t} \beta_{s\to t'}\Big).$$

The calculation of β can be done efficiently using a message passing algorithm with the following update.

$$\beta_{s\to t} \leftarrow \alpha_{t,s}\Big(1 + \sum_{h:(h,s)\in\mathcal{E}, h\neq t} \beta_{h\to s}\Big) \qquad (17)$$

In the case of linear chain CRF, our problem is reduced to the calculation of $\beta_{t+1\to t} \triangleq \sum_{s=t}^{n} \prod_{i=t}^{s-1} \alpha_{i,i+1}$ and $\beta_{t-1\to t} \triangleq \sum_{s=1}^{t} \prod_{i=s+1}^{t} \alpha_{i,i-1}$, and the message updates in Eq. 17 correspond to a forward-backward algorithm using the following recursion formula:

$$\begin{aligned}
\beta_{t+1\to t} &= \alpha_{t,t+1}\big(1 + \beta_{t+2\to t+1}\big) \\
\beta_{t-1\to t} &= \alpha_{t,t-1}\big(1 + \beta_{t-1\to t-2}\big).
\end{aligned}$$

A direct consequence of Theorem 4.1 is that we can bound the loss based on the number of nodes in CRF. However, this bound is usually much worse than the bound using the mixing rate.

Algorithm 1 Gradient Boosting for CRF

repeat
  for $\mathcal{U} \in \{N, \mathcal{E}\}$ **do**
    for $\mathbf{y}, \mathbf{x} \in \mathcal{D}$ in parallel **do**
      {inference of p_i, γ_i are done using dynamic programming}
      Infer $\mathbf{G}_i(\mathbf{y}, \mathbf{x}) \leftarrow \mu_i(\mathbf{y}) - p_i$,
        $\mathbf{H}_{ii}(\mathbf{y}, \mathbf{x}) \leftarrow p_i(1-p_i)$ for each $i \in \mathcal{U}$
      Infer $\gamma_i(\mathbf{y}, \mathbf{x})$ using dynamic programming for each $i \in \mathcal{U}$
    end for
    for $[c] \subset \mathcal{U}$ in parallel **do**
      {We use $[c]$ to enumerate over set of equivalent index defined by $\mathcal{C}$ in $\mathcal{U}$}
      $\delta_c \leftarrow \operatorname{argmin}_{\delta\in\mathcal{F}_N} \Omega(\phi_i + \delta)$
        $+ \sum_{i\in[c]} \sum_{\mathbf{y},\mathbf{x}\in\mathcal{D}} \Big[\mathbf{G}_i(\mathbf{y}, \mathbf{x})\delta(\mathbf{y}, \mathbf{x})$
        $+\gamma_i(\mathbf{y}, \mathbf{x})\mathbf{H}_{ii}(\mathbf{y}, \mathbf{x})\delta^2(\mathbf{y}, \mathbf{x})\Big]$
      $\phi_c \leftarrow \phi_c + \epsilon\delta_c$
    end for
  end for
until convergence

Corollary 4.1. *When $\mathcal{U}$ is the index set of node potentials, $\gamma_i = 2n$ satisfies Eq. (6), where n is the number of nodes in the CRF.*

Based on Theorem 4.1 and 4.2, we can get an efficient gradient boosting algorithm for CRF (GBCRF), which is presented in Algorithm 1. Here ϵ is a shrinkage term used to avoid overfitting. Our algorithm adaptively estimates γ via the mixing rate calculation at each iteration. At the beginning, when each variable is nearly independent from each other, we will have a γ that is close to 2 (and thus the updates are aggressive). γ increases as the variables become dependent on each other (resulting in more conservative updates).

Relation to LogitBoost: Our algorithm can be viewed as a generalization of multi-class classification using LogitBoost [Friedman et al., 1998]. When $\mathcal{E} = \emptyset$ in Eq. (2), our model degenerates to the LogiBoost model. In this case, the variables in each position are independent, the estimation of γ is 2, and Algorithm 1 is exactly equivalent to LogitBoost. When the variables are dependent on each other, which is common in structured prediction, our model estimates the dependency level via the Markov Chain mixing rate to guide the boosting objective in each iteration.

Time Complexity: The time complexity for the gradient boosting statistics collection in Algorithm 1 is $O(|\mathcal{D}|nK^2)$, where K is the number of states in each node and n is the average number of nodes (e.g. length of sequence) in each instance. This is due to the fact that estimation of γ can be done in $O(|\mathcal{D}|nK^2)$ time,

by using a dynamic programming algorithm. This complexity is *same as* the complexity for traditional training methods for linear CRF. The time complexity of the entire algorithm is $O(|\mathcal{D}|nK^2 + g(|\mathcal{D}|, n))$, where $g(|\mathcal{D}|, n)$ is the cost of function learning given the statistics. For learning trees, the complexity of function learning is usually $O\left(|\mathcal{D}|n\log(|\mathcal{D}|n)\right)$. Thus our approach extends CRFs to non-linearity with only an additional log factor. Further, note that the time complexity is *same as* Dietterich et al. [2008]. We also observed that both methods ran in comparable times in our experiments. This gives our approach the benefit of second-order information without any extra computational penalty.

Convergence Analysis: In this section, we analyze the convergence of our algorithm. One advantage of our method is that it makes use of second order information, and guarantees convergence.

Theorem 4.3. *$L_{\mathcal{D}}(\phi)$ converges with the procedure described by Algorithm 1 for $\epsilon \leq 1$.*

Proof. During each iteration, assume δ^* is the function that optimizes $\tilde{L}_{\mathcal{D}}(\phi, \delta)$ defined in Eq. (8),

$$L_{\mathcal{D}}(\phi + \epsilon\delta^*) \leq \tilde{L}_{\mathcal{D}}(\phi, \epsilon\delta^*) \leq \tilde{L}_{\mathcal{D}}(\phi, \mathbf{0}) = L_{\mathcal{D}}(\phi) \quad (18)$$

Then the loss function $L_{\mathcal{D}}$ decreases after each boosting step, and the algorithm converges to a minima (possibly local minima when $\mathcal{F}$ is nonlinear) of $L_{\mathcal{D}}$. □

5 RELATED WORK

Conditional random fields [Lafferty et al., 2001] are among the most successful solutions to structured prediction problems. Variants of conditional random fields have been proposed and widely applied to structured prediction in domains such as natural language processing [Lafferty et al., 2001, Sha and Pereira, 2003], computer vision [He et al., 2004, Quattoni et al.] and bio-informatics [Vinson et al., 2007]. Most popular instantiations assume linear potential functions and improve the performance by carefully engineering features. Our work focuses on learning probabilistic models for tree-shaped CRFs with nonlinear potential functions. When there are loops in the CRF and inference is intractable, relaxation of the objective can be done to use approximate inference and learning [Hazan and Urtasun, 2012, Meshi et al., 2010, Domke, 2013]. A similar dependency based term is also used in the approximate inference [Meshi et al., 2010, Domke, 2013], but is usually set to be a constant value across all instances and training iterations. As a future work, it would be interesting to explore whether our adaptive Markov Chain mixing rate bound can be applied to this more general setting. Gradient boosting [Friedman, 2001], which performs additive optimization in the functional space, has been successfully applied to classification problems that assume independent outputs conditioned on the input [Friedman et al., 1998]. Most existing attempts to "boost" CRF models optimize approximate objectives [Torralba et al., Liao et al., 2007]. TreeCRF algorithm [Dietterich et al., 2008] is similar to our approach in that it directly optimizes the log-likelihood function defined using non-linear potential functions. However they only take first order information into account during optimization, requiring a decreasing step size. On the other hand, our method makes use of second order information, and guarantees convergence with fixed step size. Our method can also be viewed as a generalization of LogitBoost [Friedman et al., 1998] for CRF. It is worth noting that the recent improvements of LogitBoost, which uses adaptive base function [Li, 2010, Sun et al., 2012], can potentially be combined with our method to obtain further improvements.

6 EXPERIMENTS

In this section, we evaluate various training approaches on three applications of CRFs: named entity recognition, hand written character recognition, and protein secondary structure prediction. We compare the following methods: (1) **GBCRF** is the proposed method in this paper. We set $\mathcal{F}_N$ to be a set of regression trees, and $\mathcal{F}_{\mathcal{E}}$ to be linear functions of basic transition features between states; (2) **LogitBoost** is a gradient boosting method for multi-class classification [Friedman et al., 1998] that does not support the dependencies between outputs; (3) **TreeCRF** is a gradient boosting method that only takes *first-order* information [Dietterich et al., 2008]. We use the same family of edge and node potentials as GBCRF; (4) **Linear CRF** is the standard CRF model with linear edge and node potentials [Lafferty et al., 2001]. To control the complexity of the model, we limit maximum depth of the trees, and use L_2 regularization for the linear models. For all the methods, training hyper-parameters (including the maximum depths of the trees and the L_2 regularization) are selected using a validation set or cross validation, depending on the specific setup of each dataset.

6.1 Named Entity Recognition

We first test our methods on the natural language task of named entity recognition (NER) using the CoNLL-2003 shared task benchmark dataset [Tjong Kim Sang and De Meulder, 2003]. The dataset contains around 20K sentences, and defines a standard split into 14K as training set, 3.3K as validation set (also called development set), and 3.5K sentences as test set. Traditional approaches for NER involve time-consuming feature engineering that requires domain expertise, and build

Table 1: F1 Measure of Name Entity Recognition on CoNLL-2003 Dataset. We use subscript *val* to denote validation set, and subscript *test* to denote test set.

	Embedding		Features+Embd	
Method	$F1_{val}$	$F1_{test}$	$F1_{val}$	$F1_{test}$
Linear CRF	84.52	79.43	89.52	84.75
LogitBoost	85.32	78.87	87.17	81.97
TreeCRF	86.30	80.60	88.46	83.99
GBCRF	88.01	82.69	90.15	86.35

Table 2: Cross Validation Error on Handwritten Character Recognition Dataset.

Method	Error
Linear CRF	0.1292 ± 0.0080
LogitBoost	0.0967 ± 0.0049
TreeCRF	0.0699 ± 0.0040
GBCRF	0.0464 ± 0.0027
NeuroCRF [Do and Artieres, 2010]	0.0444

a Linear CRF over these features. Instead, in our experiments, we explore whether it is possible to perform minimal feature engineering, and use a representation learned from data for prediction.

Specifically, we use the word embedding vectors from Mikolov et al. [2013], which is learned from the Google news corpus, and train the models on this representation. In this setting, each word is represented by a 300 dimensional vector that captures the *semantics* of the word. For each position in the sentence, we take the embedding vector of the previous, current, and the next word as input to the node potential function. We call this setting "Embedding". We further perform minimal feature engineering to *only* generate the unigram features (word, postag, and case pattern of current word). We use these basic features to train a weak linear model, then use additive training to boost the base model using the word embedding representation. We call this setting "Features+Embd".

The results of official evaluation measure for these models, which computes an F1 measure over complete span predictions, are shown in Table 1. From these results, we see that GBCRF works better than Linear CRF in both settings. The gap between LogitBoost and GBCRF indicates the importance of introducing edge potentials to this problem. We also find that taking second order information into account helps us obtain a more accurate model. Our result is also comparable to the result that uses a comprehensive set of features, i.e., the approach behind the standard Stanford NER model [Finkel et al., 2005] that achieves an F1 score of around 0.85 on the test set.

Table 3: Predictive Q8 Accuracy on Protein Secondary Structure Dataset.

Method	Accuracy
Linear CRF	0.614
LogitBoost	0.710
TreeCRF	0.718
GBCRF	0.722
SC-GSN [Zhou and Troyanskaya, 2014]	0.711
SC-GSN with "kick-start"	0.721

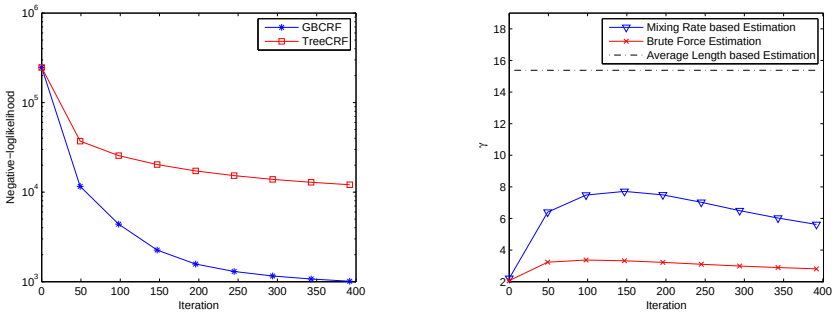
6.2 Handwriting Character Recognition

We also evaluate our method on a handwriting recognition dataset[2]. The dataset consists of 6877 words and corresponds to around 52 thousand handwritten characters [Kassel, 1995, Taskar et al., 2004], each represented by a binary pixel vector of 128 dimensions and belongs to one of 26 alphabets. The dataset is randomly split into ten folds for cross validation. We train the models on nine folds, test on the remaining fold, and use the cross validation error to compare the methods. The experimental results are shown in Table 2. Both our method and TreeCRF outperforms CRF with linear potential functions, indicating the advantage of introducing a non-linear potential functions into the CRF on this dataset. The gap between LogitBoost and CRF models suggests the importance of incorporating structure information of the outputs into the model. Our results are also comparable to NeuroCRF [Do and Artieres, 2010], which uses a deep neural network as a potential function whose weights are initialized using a Restricted Boltzmann Machine.

6.3 Protein Secondary Structure Prediction

We also conduct an experiment on protein secondary structure prediction. The task is to predict 8-state secondary structure labels for a given amino-acid sequence of a protein. We use the protein secondary structure data-set recently introduced by Zhou and Troyanskaya [2014], which is the largest publicly available protein secondary structure prediction dataset. The dataset contains 6128 proteins, with average sequence length around 208. We use exactly the same features and data split step as Zhou and Troyanskaya [2014]. The resulting data set contains 5600 sequences as training set, 256 sequences as validation set and 272 sequences as test set. Each position of the protein sequence contains 46 dimension features (22 for PSSM, 22 for sequence and 2 for terminals) for prediction. To train the models, we take the concatenation of feature vectors within 3 positions of the target position as input to the node potential, resulting in 322 input features in each position.

[2] http://www.seas.upenn.edu/~taskar/ocr/



(a) Evolution of Negative-loglikelihood (b) Evolution of average γ estimation

Figure 1: Convergence of GBCRF on hand written character dataset. (a) Convergence comparison between GBCRF and TreeCRF, with shrinkage rate of both algorithm set to 1, showing GBCRF converges faster than TreeCRF; (b) Evolution of different γ estimations on the CRF model in each round based on 500 sequences. Our mixing rate based estimation has the same trend as the exact brute force estimation, and provides a much tighter estimate than the length based one.

The performance of the model is measured by the accuracy of predictions on the test set (denoted as Q8). We train the models with parameters discovered using the validation set, and report the results in Table 3. From the table, we find that using trees as potential functions leads to better performance than restricting the model to using linear functions. Our results are comparable to the state-of-art result in this dataset, produced by Zhou and Troyanskaya [2014] (SC-GSN-3layer). The result is generated by a deep convolutional generative stochastic network model to perform secondary structure label prediction, optimized with a "kick-start" initialization.

6.4 Convergence of GBCRF

In this section we analyze the convergence of GBCRF on the handwritten character dataset. In Fig. 1(a), we show the convergence of the negative log-likelihood of GBCRF and TreeCRF. We find that GBCRF converges faster than TreeCRF, demonstrating that taking second order information into account not only gives a theoretical guarantee of convergence, but also helps the method converge faster in practice.

We also investigate the tightness of the γ estimates. Figure 1(b) gives the average of different γ estimations on models trained by GBCRF in each round. Mixing rate based estimation is the method proposed in this paper. We perform Brute Force estimation to compute γ exactly using Eq. (9); the complexity of this estimation is quadratic in the the number of nodes and outputs in the CRF, and hence cannot be used for most real-world sequences. Average length based estimation is a naive estimation that uses twice the number of nodes in the CRF, providing a valid estimate of γ since it upper bounds Eq. (14) as we show in Corollary 4.1. We restrict this evaluation to the shortest 500 sequences, due to the computation cost of brute force estimation. From the figure, we find that mixing rate based estimation exhibits the same trend as the brute force estimation, and is at most 2.3 times higher than the brute force estimation. Further, the mixing rate based bound is consistently lower than the fixed bound computed by the length based estimation. These results indicate that our mixing rate based estimation captures the changes in the dependencies in the model during training correctly. Hence our proposed mixing rate based approach is indeed useful to estimate γ efficiently.

7 CONCLUSIONS

In this paper, we present a novel gradient boosting algorithm for CRFs. It is challenging to design an effective gradient boosting for CRFs, primarily due to the dense Hessian matrices caused by variable interdependencies. To address this concern, we use a Markov Chain mixing rate to derive an efficiently computable adaptive upper bound of the loss function, and construct a gradient boosting algorithm that iteratively optimizes this bound. The resulting algorithm can be viewed as a generalization of LogitBoost to CRFs, thus introducing non-linearity in CRFs with only an additional log factor to the complexity. Experimental results demonstrate that our method is both efficient and effective. As future work, it will be important to investigate the generalization of this approach to loopy graphical models.

Acknowledgments

We would like to thank Marco Tulio Ribeiro, Ignacio Cano, Tianyi Zhou, and anonymous reviewers for their valuable feedback. This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

T. Dietterich, G. Hao, and A. Ashenfelter. Gradient tree boosting for training conditional random fields. *Journal of Machine Learning Research*, 9:2113–2139, 2008.

T. Do and T. Artieres. Neural conditional random fields. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS'10)*, 2010.

J. Domke. Structured learning via logistic regression. In *Advances in Neural Information Processing Systems 26 (NIPS'13)*, pages 647–655. 2013.

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

T. Hazan and R. Urtasun. Efficient learning of structured predictors in general graphical models. *arXiv*, 1210.2346, 2012.

X. He, R. S. Zemel, and M. A. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'04, pages 695–703, 2004.

R. Kassel. *A Comparison of Approaches to On-line Handwritten Character Recognition*. PhD thesis, Cambridge, MA, USA, 1995.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*, pages 282–289, 2001.

D. Levin, Y. Peres, and E. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2008.

P. Li. Robust Logitboost and adaptive base class (ABC) Logitboost. In *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI'10)*, pages 302–311, 2010.

L. Liao, T. Choudhury, D. Fox, and H. Kautz. Training conditional random fields using virtual evidence boosting. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, 2007.

O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, pages 783–790, 2010.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS'13)*, pages 3111–3119. 2013.

A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems 17 (NIPS'04)*, pages 1097–1104.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL '03)*, pages 134–141, 2003.

P. Sun, J. Zhou, and M. D. Reid. AOSO-LogitBoost: Adaptive one-vs-one Logitboost for multi-class problem. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*, pages 1087–1094, 2012.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16 (NIPS'04)*, pages 25–32, 2004.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using boosted random fields. In *Advances in Neural Information Processing Systems 17 (NIPS'04)*, pages 1401–1408.

J. Vinson, D. Decaprio, M. Pearson, S.Luoma, and J. Galagan. Comparative gene prediction using conditional random fields. In *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 1441–1448. 2007.

J. Zhou and O. Troyanskaya. Deep supervised and convolutional generative stochastic network for protein secondary structure prediction. In *Proceedings of the 30th International Conference on Machine Learning (ICML'14)*, volume 32, pages 745–753, 2014.

Supplementary Material

A Derivation of G and H

In this section, we derive the gradient and second order gradient given in Eq (5). The result is the same as standard CRF, we include the derivation here for completeness of the paper. We can write the negative log-likelihood

$$l(\mathbf{y}, \mathbf{x}, \phi) = -\sum_{i=1}^{m} \phi_i(\mathbf{x})\mu_i(\mathbf{y}) + \ln Z(\mathbf{x}) \tag{19}$$

Here $Z(\mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}} \exp(\sum_{i=1}^{m} \phi_i(\mathbf{x})\mu_i(\mathbf{y}'))$. The following equality holds for $Z(\mathbf{x})$

$$\begin{aligned}\partial_{\phi_k} Z(\mathbf{x}) &= \sum_{\mathbf{y}' \in \mathcal{Y}} \exp(\sum_{i=1}^{m} \phi_i(\mathbf{x})\mu_i(\mathbf{y}')) \\ &= Z(\mathbf{x}) \sum_{\mathbf{y}' \in \mathcal{Y}} \frac{\exp(\sum_{i=1}^{m} \phi_i(\mathbf{x})\mu_i(\mathbf{y}'))\mu_k(\mathbf{y}')}{Z(\mathbf{x})} \\ &= Z(\mathbf{x})E[\mu_k]\end{aligned} \tag{20}$$

In the calculation, ϕ is viewed as a vector, and partial derivative is defined by the derivative at $\phi(\mathbf{y}, \mathbf{x})$. Using this property, we can calculate the gradient as

$$\begin{aligned}\mathbf{G}_i(\mathbf{x}) &\triangleq \partial_{\phi_i} l(\mathbf{y}, \mathbf{x}, \phi) \\ &= -\mu_i(\mathbf{y}) + \frac{\partial_{\phi_i} Z(\mathbf{x})}{Z(\mathbf{x})} \\ &= -\mu_i(\mathbf{y}) + E[\mu_i] = p_i - \mu_i(\mathbf{y})\end{aligned} \tag{21}$$

Here the last equality holds because $\mu_i(\mathbf{y}) \in \{0, 1\}$. We can further calculate the second order gradient as

$$\begin{aligned}\mathbf{H}_{ij}(\mathbf{x}) &\triangleq \partial_{\phi_i \phi_j} l(\mathbf{y}, \mathbf{x}, \phi) \\ &= \partial_{\phi_j} \mathbf{G}_i(\mathbf{x}) \\ &= \sum_{\mathbf{y}' \in \mathcal{Y}} \frac{\exp(\sum_{i=1}^{m} \phi_i(\mathbf{x})\mu_i(\mathbf{y}'))\mu_i(\mathbf{y}')\mu_j(\mathbf{y}')}{Z(\mathbf{x})} \\ &\quad - \sum_{\mathbf{y}' \in \mathcal{Y}} \frac{\exp(\sum_{i=1}^{m} \phi_i(\mathbf{x})\mu_i(\mathbf{y}'))\mu_i(\mathbf{y}')}{Z^2(\mathbf{x})} \partial_{\phi_j} Z(\mathbf{x}) \\ &= E[\mu_i\mu_j] - E[\mu_i]E[\mu_j] = p_{ij} - p_i p_j\end{aligned} \tag{22}$$

The hessian $\mathbf{H}$ is also known as Fisher information matrix.

B Proof for Lemma 2.1

Proof. The following inequality holds for γ that satisfies the condition

$$\begin{aligned}\sum_{i \in \mathcal{U}} \gamma_i \mathbf{H}_{ii} \delta_i^2 &\geq \textstyle\sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{U}} |\mathbf{H}_{ij}| \delta_i^2 \\ &= \tfrac{1}{2} \textstyle\sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{U}} |\mathbf{H}_{ij}| (\delta_i^2 + \delta_j^2) \\ &\geq \textstyle\sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{U}} |\mathbf{H}_{ij}| \delta_i \delta_j\end{aligned}$$

Applying it to Talyor expansion in Eq (4), we have

$$\begin{aligned}l(\mathbf{y}, \mathbf{x}, \phi + \delta) &= l(\mathbf{y}, \mathbf{x}, \phi) + \sum_{i \in \mathcal{U}} \delta_i \mathbf{G}_i(\mathbf{y}, \mathbf{x}) \\ &\quad + \frac{1}{2} \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{U}} |\mathbf{H}_{ij}| \delta_i \delta_j + o(\delta^2) \\ &\leq l(\mathbf{y}, \mathbf{x}, \phi) + \sum_{i \in \mathcal{U}} \delta_i \mathbf{G}_i(\mathbf{y}, \mathbf{x}) \\ &\quad + \frac{1}{2} \sum_{i \in \mathcal{U}} \gamma_i \mathbf{H}_{ii} \delta_i^2 + o(\delta^2).\end{aligned}$$

□

C Proofs for Lemma 3.1 and 3.2

Proof. The proof is exactly the same for both node and potential case, we present the proof for $\mathcal{U}$ to be all node potentials here. Recall the definition of $\mathbf{H}$: $\mathbf{H}_{ij} = p_{ij}$. Note that p_i and p_{ij} are short hand notations for $p_i \triangleq P(\mu_i = 1|\mathbf{x})$, $p_{ij} \triangleq P(\mu_i\mu_j = 1|\mathbf{x})$, we have

$$\begin{aligned}\frac{1}{2p_i} \sum_{j \in \mathcal{U}} |\mathbf{H}_{ij}| &= \sum_j |p_{ij}/p_i - p_j| \\ &= \sum_{j \in \mathcal{U}} |P(\mu_j = 1|\mu_i = 1, \mathbf{x}) - P(\mu_j = 1|\mathbf{x})| \\ &= \sum_{s,k'} |P(y_s = k'|y_t = k, \mathbf{x}) - P(y_s = k'|\mathbf{x})| \\ &= \sum_s \|P(y_s|\mathbf{x}, y_t = k) - P(y_s|\mathbf{x})\|_{tv}\end{aligned}$$

□

D Proof for Lemma 3.3

Proof. Taking the fact that μ_i and μ_j are mutually exclusive for $j \neq i$, we have

$$\begin{aligned}&\sum_{j \in M} |P(\mu_j = 1|\mu_i = 1, \mathbf{x}) - P(\mu_j = 1|\mathbf{x})| \\ =&|P(\mu_i = 1|\mu_i = 1, \mathbf{x}) - P(\mu_i = 1|\mathbf{x})| \\ &+ \sum_{j \neq i} |P(\mu_j = 1|\mu_i = 1, \mathbf{x}) - P(\mu_j = 1|\mathbf{x})| \\ =&|1 - P(\mu_i = 1|\mathbf{x})| + \sum_{j \neq i} |0 - P(\mu_j = 1|\mathbf{x})| \\ =&(1 - P(\mu_i = 1|\mathbf{x})) + \sum_{j \neq i} P(\mu_j = 1|\mathbf{x}) \\ =&2(1 - P(\mu_i = 1|\mathbf{x}))\end{aligned}$$

□

E Proof for Theorem 4.2

Proof. In this proof, we will reduce the total variation distance between joint distribution of edge states into total variation distance of marginal distribution over nodes, as in Theorem 4.1. Assume the edge pairs are (y_t, y_{t+1}),

(y_s, y_{s+1}), and y_s is closer to y_{t+1} (without loss of generality), then

$$P(y_s, y_{s+1}|y_t, y_{t+1}, \mathbf{x}) = P(y_{s+1}|y_s, \mathbf{x})P(y_s|y_{t+1}, \mathbf{x})$$

We can convert total variation by

$$\begin{aligned}
&\|P(y_s, y_{s+1}|y_t, y_{t+1}, \mathbf{x}) - P(y_s, y_{s+1}|\mathbf{x})\|_{tv} \\
=&\sum_{y_s, y_{s+1}} |P(y_s, y_{s+1}|y_t, y_{t+1}, \mathbf{x}) - P(y_s, y_{s+1}|\mathbf{x})| \\
=&\sum_{y_s, y_{s+1}} P(y_{s+1}|y_s, \mathbf{x})|P(y_s|y_{t+1}, \mathbf{x}) - P(y_s|\mathbf{x})| \\
=&\sum_{y_s} |P(y_s|y_{t+1}, \mathbf{x}) - P(y_s|\mathbf{x})| \\
=&\|P(y_s|y_{t+1}, \mathbf{x}) - P(y_s|\mathbf{x})\|_{tv}
\end{aligned}$$

Now the case become same as node potential, we can make use of Corollary 3.1 bound the total variation.

$$\begin{aligned}
&\|P(y_s, y_{s+1}|y_t = k_t, y_{t+1} = k_{t+1}, \mathbf{x}) - P(y_s, y_{s+1}|\mathbf{x})\|_{tv} \\
=&\|P(y_s|y_{t+1} = k_{t+1}, \mathbf{x}) - P(y_s|\mathbf{x})\|_{tv} \\
\leq&\|P(y_{t+1}|y_{t+1} = k_{t+1}, \mathbf{x}) - P(y_{t+1}|\mathbf{x})\|_{tv} \prod_{(a,b)\in\mathcal{Q}(s,t+1)} \alpha_{b,a} \\
=&[1 - P(y_{t+1} = k_{t+1}|\mathbf{x})] \prod_{(a,b)\in\mathcal{Q}(s,t+1)} \alpha_{b,a} \\
\leq&[1 - P(y_t = k_t, y_{t+1} = k_{t+1}|\mathbf{x})] \prod_{(a,b)\in\mathcal{Q}(s,t+1)} \alpha_{b,a}
\end{aligned}$$

Here the first inequality is due to Corollary 3.1. Intuitively, this means that the total variational distance of between two edge states, can be bounded by recursively applying the mixing rate bound along the path between two edges. Summing the results of $(s, s+1)$ over all edges will give us Eq. (15). □