Structural Extension to Logistic Regression: Discriminative Parameter Learning of Belief Net Classifiers *

Russell Greiner (greiner@cs.ualberta.ca) †
Dept of Computing Science, University of Alberta, Edmonton, AB T6G 2H1 Canada

Xiaoyuan Su (xsu1@umsis.miami.edu)
Electrical & Computer Engineering, University of Miami, Coral Gables, FL 33124, USA

Bin Shen (bshen@cs.ualberta.ca)
Dept of Computing Science, University of Alberta, Edmonton, AB T6G 2H1 Canada

Wei Zhou (w2zhou@math.uwaterloo.ca)
School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada

Abstract. Bayesian belief nets (BNs) are often used for classification tasks — typically to return the most likely class label for each specified instance. Many BN-learners, however, attempt to find the BN that maximizes a different objective function — viz., likelihood, rather than classification accuracy — typically by first learning an appropriate graphical structure, then finding the parameters for that structure that maximize the likelihood of the data. As these parameters may not maximize the classification accuracy, "discriminative parameter learners" follow the alternative approach of seeking the parameters that maximize *conditional likelihood* (CL), over the distribution of instances the BN will have to classify. This paper first formally specifies this task, shows how it extends standard logistic regression, and analyzes its inherent sample and computational complexity. We then present a general algorithm for this task, ELR, that applies to arbitrary BN structures and that works effectively even when given incomplete training data. Unfortunately, ELR is not guaranteed to find the parameters that optimize conditional likelihood; moreover, even the optimal-CL parameters need not have minimal classification error. This paper therefore presents empirical evidence that ELR produces effective classifiers, often superior to the ones produced by the standard "generative" algorithms, especially in common situations where the given BN-structure is incorrect.

Keywords: (Bayesian) belief nets, Logistic regression, Classification, PAC-learning, Computational/sample complexity

1. Introduction

Many tasks — including fault diagnosis, pattern recognition and forecasting — can be viewed as *classification*, as each requires assigning the class ("label") to a given instance, which is specified by a set of attributes. An increasing number of projects are using "(Bayesian) belief nets" (*BN*) to represent the underlying distribution, and hence the stochastic mapping from evidence to response.

* This paper extends the earlier results that appear in [27] and [49].
† This e-mail address is available for all problems and questions.

© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

When this distribution is not known *a priori*, we can try to *learn* the model. Our goal is an *accurate* BN — *i.e.*, one that returns *the correct answer as often as possible*. While a perfect model of the distribution will perform optimally for any possible query, learners with limited training data are unlikely to produce such a model; moreover, optimality may be impossible for learners constrained to a restricted range of possible distributions that excludes the correct one (*e.g.*, when only considering parameterizations of a given *BN*-structure).

Here, it makes sense to find the parameters that do well with respect to the queries posed. This "discriminative learning" task differs from the "generative learning" that is used to learn an overall model of the distribution [47]. Following standard practice, our discriminative learner will seek the parameters that maximize the *log conditional likelihood* (LCL) over the data, rather than simple likehood — that is, given the data $S = \{\langle c_i, \mathbf{e}_i \rangle\}$ (each class label $C = c_i$ associated with evidence $\mathbf{E} = \mathbf{e}_i$), a discriminative learner will try to find parameters Θ that maximize

$$\widehat{\mathrm{LCL}}^{(S)}(\Theta) \quad = \quad \frac{1}{|S|} \sum_{\langle c_i, \mathbf{e}_i \rangle \in S} \log P_\Theta(c_i \,|\, \mathbf{e}_i) \tag{1}$$

rather than the ones that maximize $\sum_{\langle c_i, \mathbf{e}_i \rangle \in S} \log P_\Theta(c_i, \mathbf{e}_i)$ [47].

Optimizing the LCL of the root node (given the other attributes) of a *naïve-bayes* structure can be formulated as a standard logistic regression problem [39, 32]. General belief nets extend naïve-bayes-structures by permitting additional dependencies among the attributes. This paper provides a general discriminative learning tool ELR that can learn the parameters for an arbitrary structure, completing the analogy

$$\text{Naïve-bayes} \;:\; \text{General Belief Net} \;::\; \text{Logistic Regression} \;:\; \texttt{ELR}\,. \tag{2}$$

Moreover, while most algorithms for learning logistic regression functions require *complete* training data, the ELR algorithm can accept *incomplete* data. We also present empirical evidence, from a large number of datasets, to demonstrate that ELR works effectively.

Section 2 provides the foundations, overviewing belief nets then defining our task: discriminatively learning the parameters (for a fixed belief net structure, G) that maximize LCL. Section 3 formally analyses this task, providing both sample and computational complexity, and noting how these results compare with corresponding results for generative learning. Seeing that our task is NP-hard in general, Section 4 presents a gradient-descent discriminative parameter learning algorithm for general BNs, ELR. Section 5 reports empirical results that demonstrate that our ELR produces a classifier that is often superior to ones produced by standard learning algorithms (which maximize likelihood), over a variety of situations, involving both complete and

incomplete data. Section 6 provides a brief survey of the relevant literature. The webpage [25] provides the proofs of all of our theoretic claims (extending the proof sketch in the Appendix), as well as more information about the experiments shown in Section 5, and yet other experiments.

2. Framework

We assume there is a stationary underlying distribution $P(\cdot)$ over N (discrete) random variables $\mathcal{V} = \{V_1, \ldots, V_n\}$. For example, perhaps V_1 is the "Cancer" random variable, whose value ranges over $\{\texttt{true}, \texttt{false}\}$; V_2 is "Gender" $\in \{\texttt{male}, \texttt{female}\}$, V_3 is "Age" $\in \{0, .., 100\}$, etc. We refer to this joint distribution as the "underlying distribution" or the "event distribution".

We can encode this as a "(Bayesian) belief net" (BN) — a directed acyclic graph $B = \langle \mathcal{V}, A, \Theta \rangle$, whose nodes $\mathcal{V}$ represent variables, and whose arcs A represent dependencies. Each node $D_i \in \mathcal{V}$ also includes a conditional-probability-table (CPtable) $\theta_i \in \Theta$ that specifies how D_i's values depend (stochastically) on the values of its immediate parents. In particular, given a node $D \in \mathcal{V}$ with immediate parents $\mathbf{F} \subset \mathcal{V}$, the parameter $\theta_{d|\mathbf{f}}$ represents the network's term for $P(D = d \,|\, \mathbf{F} = \mathbf{f})$ [45].

The user interacts with the belief net by asking *queries*, each of the form "What is $P(C = c \,|\, \mathbf{E} = \mathbf{e})$?" — *e.g.*, What is $P(\texttt{Cancer = true} | \texttt{Gender=male}, \texttt{Smoke=true})$? — where $C \in \mathcal{V}$ is a single "query variable", $\mathbf{E} \subset \mathcal{V}$ is the subset of "evidence variables", and c (resp., $\mathbf{e}$) is a legal assignment to C (resp., $\mathbf{E}$). This paper focuses on the case where all queries involve the same variable; *e.g.*, all queries ask about Cancer. Moreover, we will follow standard practice by assuming the distribution of conditioning events matches the underlying distribution. This means there is a single distribution from which we can draw labeled instances, which each correspond to a "labeled query".[1] Note this corresponds to the data sample used by standard learning algorithms.

Given any unlabeled instance $\{\mathbf{E}_i = \mathbf{e}_i\}$, the belief net[2] Θ will produce a distribution over the values of the query variable; perhaps $P_\Theta(\, C = \texttt{true} \,|\, \mathbf{E} = \mathbf{e}\,) = 0.3$ and $P_\Theta(\, C = \texttt{false} \,|\, \mathbf{E} = \mathbf{e}\,) = 0.7$. In general, the associated H_Θ classifier system will then return the value $H_\Theta(\mathbf{e}) = \text{argmax}_c \{P_\Theta(\, C = c \,|\, \mathbf{E} = \mathbf{e}\,)\}$ with the largest posterior probability — here return $H_\Theta(\mathbf{E} = \mathbf{e}\,) = \texttt{false}$ as $P_\Theta(\, \texttt{Cancer = false} \,|\, \mathbf{E} = \mathbf{e}\,) > P_\Theta(\, \texttt{Cancer = true} \,|\, \mathbf{E} = \mathbf{e}\,)$.

A good belief net classifier is one that produces the appropriate answers to these unlabeled queries. We will use "classification error" (aka "0/1" loss)

[1] See [26] for an alternative position, and the challenges this requires solving.

[2] As we assume the structure $G = \langle \mathcal{V}, A \rangle$ of the belief net $B = \langle \mathcal{V}, A, \Theta \rangle$ is fixed, we will identify a belief net with its parameters Θ.

to evaluate the resulting Θ-based classifier H_Θ

$$\mathrm{err}(\Theta) \quad = \quad \sum_{\langle \mathbf{e}, c\rangle} P(\mathbf{e}, c) \times I(H_\Theta(\mathbf{e}) \neq c) \tag{3}$$

where $I(a \neq b) = 1$ if $a \neq b$, and $= 0$ otherwise.

Our goal is a belief net Θ^* that minimizes this score, with respect to the true distribution $P(\cdot)$. While we do not know this distribution *a priori*, we can use a sample drawn from this distribution, to help determine which belief net is optimal. This paper focuses on the task of learning the optimal CPtable Θ for a given BN-structure $G = \langle \mathcal{V}, A\rangle$.

Conditional Likelihood: In earlier work [53, Sections 3.1.3 and 3.2.2], we compared learners that optimized $\mathrm{err}(\Theta)$ versus ones that optimized the "log conditional likelihood" of a belief net Θ

$$\mathrm{LCL}_P(\Theta) \quad = \quad \sum_{\langle \mathbf{e}, c\rangle} P(\mathbf{e}, c) \times \log(P_\Theta(c \,|\, \mathbf{e})) \tag{4}$$

(as approximated by Equation 1), and found no significant difference in classification performance; this is consistent with [39, 24, 2]. Our work therefore focuses on learners that attempt to maximize $\mathrm{LCL}_P(\Theta)$.

While Equation 1's $\widehat{\mathrm{LCL}}^{(S)}(\Theta)$ formula closely resembles the (empirical) "log likelihood" function

$$\widehat{\mathrm{LL}}^{(S)}(\Theta) \quad = \quad \frac{1}{|S|} \sum_{\langle \mathbf{e}, c\rangle \in S} \log(P_\Theta(c, \mathbf{e})) \tag{5}$$

used by many BN-learning algorithms, there are some critical differences. Of course, $\widehat{\mathrm{LL}}^{(S)}(\Theta) = \frac{1}{|S|}\left[\sum_{\langle c, \mathbf{e}\rangle \in S} \log(P_\Theta(c \,|\, \mathbf{e})) + \sum_{\langle c, \mathbf{e}\rangle \in S} \log(P_\Theta(\mathbf{e}))\right]$, where the first term resembles our $\widehat{\mathrm{LCL}}(\cdot)$ score, which related to how well our network will answer the relevant queries, while the second term is irrelevant to our task [24]. This means a BN Θ_α that does poorly wrt the first "$\widehat{\mathrm{LCL}}(\cdot)$-like" term may be preferred to a Θ_β that does better — *i.e.*, it is possible that $\widehat{\mathrm{LL}}(\Theta_\alpha) > \widehat{\mathrm{LL}}(\Theta_\beta)$, while $\widehat{\mathrm{LCL}}(\Theta_\alpha) < \widehat{\mathrm{LCL}}(\Theta_\beta)$. (Section 5.3 provides other arguments explaining why our $\widehat{\mathrm{LCL}}(\cdot)$-based approach may work better than the $\widehat{\mathrm{LL}}(\cdot)$-based approaches; and Section 6 surveys other relevant literature.)

3. Theoretical Analysis

How many "labeled instances" are enough — *i.e.*, given any values $\varepsilon, \delta > 0$, how many labeled instances are needed to insure that, with probability at least $1-\delta$, an algorithm can produce a classifier that is within ε of optimal? While we believe there are comprehensive general bounds, our specific results require the relatively benign technical restriction that all CPtable entries must be bounded away from 0. That is, for any $\gamma > 0$, let

$$\mathcal{BN}_{\Theta \geq \gamma}(G) \quad = \quad \{\Theta \in [0,1]^K \mid \forall \theta_{d|\mathbf{f}} \in \Theta,\ \theta_{d|\mathbf{f}} \geq \gamma\} \tag{6}$$

be the subset of parameters (appropriate for the I-parameter belief net structure G) whose CPtable values are all at least γ.[3] We now restrict our attention to these parameters and in particular, let

$$\Theta^*_{G,\Theta\geq\gamma} \quad = \quad \text{argmax}\left\{\text{LCL}_P(\Theta) \,|\, \Theta \in \mathcal{BN}_{\Theta\geq\gamma}(G)\right\} \tag{7}$$

be a parameter setting with optimal score among $\mathcal{BN}_{\Theta\geq\gamma}(G)$ with respect to the true distribution $P(\cdot)$.

THEOREM 1 (Appendix A, and [25, #Proof]). *Let G be any belief net structure with K CPtable entries $\Theta = \{\theta_{d_i|\mathbf{f}_i}\}_{i=1..K}$, and let $\hat{\Theta} \in \mathcal{BN}_{\Theta\geq\gamma}(G)$ be the BN in $\mathcal{BN}_{\Theta\geq\gamma}(G)$ that has maximum* empirical *log conditional likelihood score (Equation 1) with respect to a sample of*

$$18\left(\frac{N\ln\gamma}{\varepsilon}\right)^2\left[\ln\frac{2}{\delta} + K\ln\frac{6K}{\gamma\varepsilon}\right] = O\left(\frac{N^2K}{\varepsilon^2}\ln(\frac{K}{\varepsilon\delta})\ \ln^3(\frac{1}{\gamma})\right) \tag{8}$$

labeled queries drawn from $P(\cdot)$. Then, with probability at least $1-\delta$, $\hat{\Theta}$ will be no more than ε worse than $\Theta^*_{G,\Theta\geq\gamma}$ — i.e.,

$$P(\mathit{LCL}_P(\hat{\Theta}) \leq \mathit{LCL}_P(\Theta^*_{G,\Theta\geq\gamma}) - \varepsilon) \leq \delta.$$

∎

A virtually identical proof shows that this same result holds when dealing with another approximation to the 0/1 error, $\text{err}(\Theta)$, *viz.*,

$$\mathit{MSE}(\Theta) = \sum_{\langle \mathbf{e}, c\rangle} P(\mathbf{e}, c) \times [P_\Theta(c\,|\,\mathbf{e}) - P(c\,|\,\mathbf{e})]^2 \tag{9}$$

rather than $\text{LCL}(\cdot)$.

For comparison, Dasgupta [17, Section 5] proves that

$$O\left(\frac{N^2K}{\varepsilon^2}\ln(\frac{K}{\varepsilon\delta})\ \ln^3(N)\ln^2(\frac{1}{\varepsilon})\right) \tag{10}$$

[3] This $\theta_{d|\mathbf{f}} \geq \gamma$ constraint is trivially satisfied by any parameter learner that uses Laplacian correction, or that produces the posterior distribution from uniform Dirichlet priors: These system can use $\gamma = 1/(m+2)$ where m is the number of training instances [30].

complete tuples[4] are sufficient to learn the parameters to a fixed structure that are with ε of the optimal *likelihood* (Equation 5). While comparing upper bounds is only suggestive, it is interesting to note that, ignoring the $\ln^{\ell}(\cdot)$ terms for $\ell > 1$, these bounds are asymptotically identical.

One asymmetry is that only our Equation 8 bound includes the γ term, which corresponds to the smallest CPtable entry allowed. While Dasgupta [17] (following [1]) can avoid this term by "tilting" the empirical distribution, this trick does not apply in our discriminative task: Our task inherently involves computing *conditional* likelihood, which requires *dividing* by some CPtable values, which is problematic when these values are near 0. This observation also means our proof is *not* an immediate application of the standard *PAC*-learning approaches. Of course, our sample complexity remains polynomial in the size (N, K) of the belief net even if this γ is exponentially small, $\gamma = O(1/2^N)$.

Note finally that the parameters that optimize (or nearly optimize) likelihood will not necessarily optimize our objective of *conditional* likelihood; this means Equation 10 describes the convergence to parameters that are typically inferior to the ones associated with Equation 1, especially when the structure is wrong; see Ng and Jordan [44].

The second question is computational: How hard is it to find these best parameters values, given this sufficiently large sample. Here, the news is mixed.

On the positive side, Wettig *et al.* [52] show this task corresponds to a convex optimization problem when the data is complete and the structure G satisfies certain specified properties; this implies a polynomial algorithm can find the values of the CPtables of G whose (empirical) conditional likelihood (Equation 1) is within ε of optimal [43, 5]. They show that these properties hold for Naïve-bayes and `TAN` structures (see Section 5).

Unfortunately...

THEOREM 2. *It is NP-hard to find the values for the CPtables of a fixed BN-structure that produce the largest (empirical) conditional likelihood (Equation 1) for a given* incomplete *sample.*[5] ∎

We do not know the complexity of this task for *arbitrary* structures, given complete data.

[4] We say a tuple is "complete" if it specifies a value for every attribute; hence "$E_1 = e_1, \ldots, E_n = e_n$" is complete (where $\{E_1, \ldots, E_n\}$ is the full set of evidence variables) but "$E_2 = e_2, E_7 = e_7$" is not.

[5] Proof sketch: Reduce from 3SAT, using the structure from Cooper [13], with queries that "specify" each clause, and one that requires that the conjunction of clauses has probability 1. Then the log conditional likelihood score is 0 iff there is a satisfying assignment to the 3SAT formula. Details are in [25, #Proof].

4. ELR Learning Algorithm

Given the intractability of computing the optimal CPtable entries in general, we defined a simple gradient-ascent algorithm, ELR, that attempts to improve the empirical score $\widehat{\mathrm{LCL}}(\Theta)$ by changing the values of each CPtable entry $\theta_{d|\mathbf{f}}$. (Of course, this will only find, at best, a *local* optimum.) To incorporate the constraints $\theta_{d|\mathbf{f}} \geq 0$ and $\sum_d \theta_{d|\mathbf{f}} = 1$, we used the different set of parameters, "$\beta_{d|\mathbf{f}}$", where each

$$\theta_{d|\mathbf{f}} \quad = \quad \frac{e^{\beta_{d|\mathbf{f}}}}{\sum_{d'} e^{\beta_{d'|\mathbf{f}}}} \,. \tag{11}$$

As the β_is sweep over the reals, the corresponding $\theta_{d_i|\mathbf{f}}$'s will satisfy the appropriate constraints. (In the naïve-bayes case, this corresponds to what many logistic regression algorithms would do, albeit with different parameters [32]: Find α, χ that optimize $P_{\alpha,\chi}(C = c \,|\, \mathbf{E} = \mathbf{e}) \;=\; e^{\alpha_c + \chi_c \cdot \mathbf{e}} / \sum_j e^{\alpha_j + \chi_j \cdot \mathbf{e}}$.[6] Recall that our goal is a more general algorithm — one that can deal with *arbitrary* structures; see Equation 2.)

Like all such algorithms, ELR is basically

$$\begin{array}{l} \text{Initialize } \beta^{(0)} \\ \text{For } k = 1..m \\ \qquad \beta^{(k+1)} \quad := \quad \beta^{(k)} \;+\; \alpha^{(k)} \times \mathbf{d}^{(k)} \end{array} \tag{12}$$

where $\beta^{(k)}$ represents the set of parameters at iteration k. In a simple gradient ascent approach, we would set $\mathbf{d}^{(k)}$ to be the total derivative with respect to the given set of labeled queries, $\nabla \widehat{LCL} \;=\; \langle \frac{\partial\, \widehat{\mathrm{LCL}}^{(S)}(\Theta)}{\partial\, \beta_{d|\mathbf{f}}} \rangle_{d,\mathbf{f}}$, where each element of the vector is the sum of the individual derivatives from each labeled training case $\langle \mathbf{e}; c \rangle$:

$$\frac{\partial\, \widehat{\mathrm{LCL}}^{(S)}(\Theta)}{\partial\, \beta_{d|\mathbf{f}}} \quad = \quad \sum_{\langle \mathbf{e},c \rangle \in S} \frac{\partial\, \widehat{\mathrm{LCL}}^{(\langle \mathbf{e},c \rangle)}(\Theta)}{\partial\, \beta_{d|\mathbf{f}}} \,.$$

PROPOSITION 3 ([26],[16],[25, #Proof]). *For the labeled training case* $\langle \mathbf{e}, c \rangle$ *and each "softmax" parameter* $\beta_{d|\mathbf{f}}$,

$$\frac{\partial\, \widehat{LCL}^{(\langle \mathbf{e},c \rangle)}(\Theta)}{\partial\, \beta_{d|\mathbf{f}}} \;=\; [P_\Theta(d, \mathbf{f} \,|\, \mathbf{e}, c) - P_\Theta(d, \mathbf{f} \,|\, \mathbf{e})] \;-\; \theta_{d|\mathbf{f}} [P_\Theta(\mathbf{f} \,|\, c, \mathbf{e}) - P_\Theta(\mathbf{f} \,|\, \mathbf{e})] \,.$$

[6] While the obvious tabular representation of the CPtables involves more parameters than appear in this logistic regression model, these extra BN-parameters are redundant.

Notice this expression is well-defined for any set of evidence variables assigned $\mathbf{E}$ — which can be all "non-query" variables (corresponding to a complete data tuple), or any subset, including the empty $\mathbf{E} = \{\}$.

To work effectively, ELR incorporates four instantiations/modifications to the basic gradient ascent idea, dealing with

1. the initial values $\beta^{(0)}$ ("plug-in parameters"),
2. the direction of the modification $\mathbf{d}^{(k)}$ (conjugate gradient),
3. the magnitude of the change $\alpha^{(k)}$ (line search) and
4. the stopping criteria m ("cross tuning").

Minka [40] has shown that the middle two ideas, conjugate gradient and line-search [46], are effective for the standard logistic regression task.

Begin with Plug-In Parameters: We must first initialize the parameters, $\beta^{(0)}$. One common approach is to set $\beta^{(0)}$ to small, randomly selected, values; another is to begin with the values specified by the generative approach — *i.e.*, using frequency estimates (OFE; Equation 13) in the complete data case, and a simple variant otherwise (see Section 5.1.3). Our empirical evidence shows that this second approach works better, especially for small samples. These easy-to-compute generative starting values are often used to initialize parameters for discriminative tasks, and called "plug-in parameters" [47].

Conjugate Gradient: Standard gradient ascent algorithms may require a great many iterations, especially for functions that have long, narrow valley structures. The *conjugate gradient method* addresses this problem by ascending along *conjugate directions*, rather than simply the local gradient. As this means that the directions that have already been optimized, stay optimized, this method often requires far fewer steps uphill to reach the local optimum [29].

In particular, ELR uses the *Polak-Rebiere* formula to update its search direction. The initial search direction is given by:

$$\mathbf{d}^{(0)} \quad := \quad \nabla\widehat{LCL}_{(0)}$$

On subsequent iterations,

$$\mathbf{d}^{(k)} \quad := \quad \nabla\widehat{LCL}_{(k)} \; - \; \frac{(\nabla\widehat{LCL}_{(k)} - \nabla\widehat{LCL}_{(k-1)}) \; \cdot \; \nabla\widehat{LCL}_{(k)}}{\nabla\widehat{LCL}_{(k-1)} \; \cdot \; \nabla\widehat{LCL}_{(k-1)}} \, \mathbf{d}^{(k-1)}$$

where the "$\cdot$" is the vector dot-product. Hence, the current update direction is formed by "subtracting off" the previous direction from the current derivative.

Line Search: ELR will ascend in the $\mathbf{d}^{(k)}$ direction; the next challenge is deciding how far to move — *i.e.*, in computing the $\alpha^{(k)} \in \Re^{+}$ within (12).

The good news is, as $\beta = \beta^{(k)}$ and $\mathbf{d} = \mathbf{d}^{(k)}$ are fixed, this is a one-dimensional search: find the α^{*} that maximizes $f(\alpha) = \widehat{\mathrm{LCL}}^{(S)}(\beta \; + \; \alpha \times \mathbf{d})$. ELR

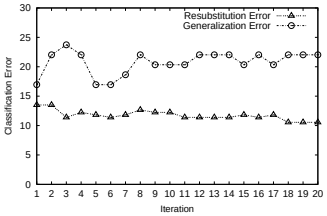

Figure 1. Comparing Training-Set Error with Test-Set Error, as function of Iteration

therefore uses a standard technique, *Brent*'s iterative line search procedure (a hybrid combination of the linear Golden Section search [29] and a quadratic interpolation), which has proven to be very effective at finding the optimal value for $\alpha^{(k)}$ [3]. In essense, this method first finds three α_i values, and computes the associated function values $\langle\ \langle \alpha_i,\ f(\alpha_i)\rangle\ \rangle_{i=1,2,3}$. It then assumes the $f(\cdot)$ function is (locally) quadratic, and so fits this trio of points to a second degree polynomial. It then finds the α^* value that minimizes this quadratic, replaces one of the three original α_i values with this $\langle \alpha^*, f(\alpha^*)\rangle$ pair, and iterates using the new trio of points. See [3] for details.

Cross tuning (stopping time): The final issue is deciding when to stop; *i.e.*, determining the value of m within (12). A naïve algorithm would just compute the training-set error (Equation 3) at each iteration, and stop when that error measure appears at a local optimum — *i.e.*, when the error appears to be getting larger. The graph in Figure 1 shows both training-set and test-set error on a particular dataset at each iteration. If we used this simple approach, we would stop on the third iteration; the generalization error shows that these parameters are very bad — in fact, they seem almost the worst values!

To avoid this, we use a variant of cross validation, which we call "cross tuning", to estimate the optimal number of iterations. Here, we divide the *training* data into $\ell = 5$ partitions, then for each fold, run the gradient ascent for remaining $4/5$ of the data, but evaluating the quality of the result (on each iteration) on the fold. (This produces a graph like the "Generalization Error" line in Figure 1.) We then determine, for each fold, when we should have stopped — here, it would be on $k = 5$, as that is the global optimum for this fold. We then set m to be the median values over these ℓ folds. When using all of the data to produce the final $\{\beta_{d|\mathbf{f}}\}$ values, we iterate exactly m times. The website [25, #CrossTuning] presents empirical evidence that cross-tuning is important, especially for complex models.

5. Empirical Studies

The ELR algorithm takes, as arguments, a BN-structure $G = \langle \mathcal{V}, A\rangle$ and a dataset of labeled queries (aka instances) $S = \{\langle \mathbf{e}_i, c_i\rangle\}_i$, and returns a value

for each CPtable parameter $\theta_{d|\mathbf{f}}$. To explore its effectiveness, we compared the $\mathrm{err}(\cdot)$ performance of the resulting Θ_{ELR} parameters with the results of other algorithms that similarly learn CPtable values for a given structure.

We say a data sample is "complete" if each instance is complete (see Footnote 4); otherwise it is incomplete. When the data is complete, we compare ELR to the standard "observed frequency estimate" (OFE) approach, which is known to produce the parameters that maximize likelihood (Equation 5) for a given structure [14]. For example, if 75 of the 100 $C = 1$ instances have $X_3 = 0$, then OFE sets

$$\theta_{X_3=0|C=1} \quad = \quad \frac{\#(X_3 = 0, C = 1)}{\#(C = 1)} \quad = \quad \frac{75}{100} \tag{13}$$

(Some versions use a Laplacian correction to avoid the problems caused by 0 probability events.) When the data is *incomplete*, we compare ELR to the standard Expectation Maximization algorithm EM [20, 37, 30] and to APN [2], which ascends to parameter values whose likelihood is locally optimal.[7]

Traditional wisdom holds that discriminative learning (ELR) is most relevant (*i.e.*, better than generative learning: OFE, APN, EM) when this underlying model $G = \langle \mathcal{V}, A \rangle$ is "wrong", that is, not an I-map of the true distribution T — which here means the graph structure does not include some essential arcs [45]. The situation, which we denote "$G < T$", is fairly common as many learners consider only structures as simple as naïve-bayes, or the class of TAN structures (defined below), which are typically much simpler than T.[8] Section 5.1 deals with this situation, considering both the complete and incomplete data cases.

Section 5.2 then considers another standard situation: where we employ a structure-learning algorithm to produce a structure that is similar to the truth; *i.e.*, where $G \approx T$. Here, we use the POWERCONSTRUCTOR system [9, 8] for the first step (to learn the structure), then compare the relative effectiveness of algorithms for finding parameters for this structure.

Finally, Section 5.3 summarizes all of these empirical results.

Notation: The notation "NB+ELR" will refer to the NB structure, whose parameters are learned using ELR; in general, we will use *x*+*y* to refer to the system produced when the *y* algorithm is used to produce the parameter for the *x* structure. Below we will compare various pairs of learners, in each case over a common dataset; we will therefore use a one-sided paired t-test [41]. When this result is significant at the $\rho < 0.05$ level, we will write $\alpha \Leftarrow_{(p<\rho)} \beta$

[7] While the original APN_θ [2] climbed in the space of parameters $\Theta = \{\theta_i\}$, we instead used a modified APN_β system that uses the $\beta = \{\beta_i\}$ values (Equation 11), as we found it produced better classifiers.

[8] The $G < T$ notation does *not* mean the arcs of G are a subset of T's, as G may also include arcs that are not in T.

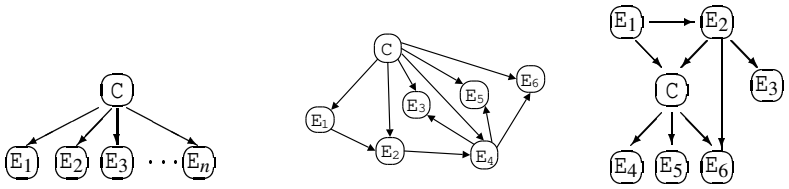

Figure 2. (a) Naïve-bayes structure (b) TAN structure [24, Fig 3] (c) Arbitrary GBN structure. (Note: all figures are numbered (a), (b), ... from left to right.)

— *e.g.*, we will soon see NB+ELR $\Leftarrow_{(p<0.005)}$ NB+OFE. For values larger than 0.05, we will use the notation $\alpha \leftarrow_{(p<\rho)} \beta$. (That is, we regard $p < 0.05$ as the cut-off for statistical significance.) Note the arrow points to the learner that appears to be better.[9]

While our main emphasis is comparing *x*+ELR to *x*+OFE (and to *x*+APN and *x*+EM) for various structures *x*, where relevant we will also compare across structure classes; *e.g.*, comparing *x*+ELR to *z*+ELR for different structures *x* and *z*.

5.1. Model is Simpler than the Truth ($G < T$)

Section 5.1.1 (resp., Section 5.1.2) compares algorithms for learning the parameters for a naïve-bayes model (resp., a TAN model) given *complete* data; Section 5.1.3 then considers learning these models given *incomplete* data.[10]

5.1.1. *Naïve-bayes — Complete, Real World Data*

Our first experiments deal with the simplest situation: learning the Naïve-bayes parameters from complete data. Recall that the Naïve-bayes structure requires that the attributes are independent given the class label; see Figure 2(a).

We compared the relative effectiveness of ELR with various other classifiers, over the same 25 datasets that Friedman *et al.* [24] used for their comparisons: 23 from UCIrvine repository [4], plus "MOFN-3-7-10" and "CORRAL", which were developed by John and Kohavi [34] to study feature selection; see [25, #25-datasets], which also specifies how we computed our accuracy values — based on 5-fold cross validation for small data, and holdout method for large data [33]. To deal with continuous variables, we implemented supervised entropy discretization [23]. The table in [25, #Results-Complete] succinctly summarizes the results.

We use the CHESS dataset (36 binary or ternary attributes) to illustrate the basic behaviour of the algorithms. Figure 3(a) shows the performance,

[9] This analysis makes the standard assumptions that the error are identical and independent, and each normally distributed; see Nadeau and Bengio [42].

[10] [25, #Study] uses a simple controlled study, on artificial data, to further investigate how ELR and OFE deal with increasingly more erroneous structures.

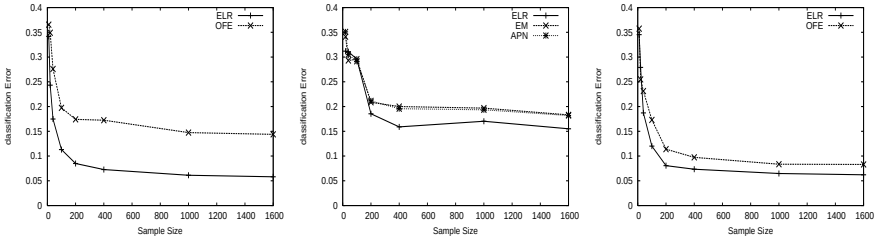

Figure 3. CHESS domain: (a) ELR vs OFE, complete data, structure is "incorrect" (naïve-bayes); (b) ELR vs EM, APN, incomplete data, structure is "incorrect" (naïve-bayes); (c) ELR vs OFE, complete data, structure is "≈correct" (POWERCONSTRUCTOR)

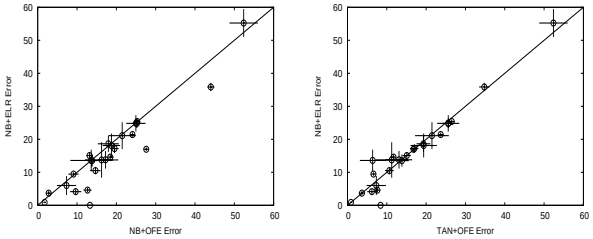

Figure 4. Comparing NB+ELR with (a) NB+OFE; (b) TAN+OFE

on this dataset, of our NB+ELR (a.k.a. "Naïve-bayes structure + ELR instantiation") system, versus the "standard" NB+OFE, which uses OFE to instantiate the parameters. We see that ELR is consistently more accurate than OFE, for any size training sample. We also see how quickly ELR converges to the best performance.

Figure 4(a) provides a more comprehensive comparison, across all 25 datasets. (Each point below the $x = y$ line is a dataset where NB+ELR was better than the other approach — here NB+OFE. The lines also express the 1 standard-deviation error bars in each dimension.[11]) As suggested by this plot, NB+ELR is significantly better than NB+OFE at the $p < 0.005$ level.

5.1.2. *TAN — Complete, Real World Data*

We next considered TAN ("tree augmented naïve-bayes") structures [24], which include a link from the classification node down to each attribute and, if we ignore those class-to-attribute links, the remaining links, connecting attributes to each other, form a tree; see Figure 2(b). (Hence this representation allows each attribute to have at most one "attribute parent", and so this class of structures strictly generalize Naïve-bayes.) Friedman *et al.* [24] provide an efficient algorithm for learning such TAN structures given complete data, based on Chow-Liu [12]: first compute the mutual information between each

[11] When using 5-fold cross-validation, we computed the standard deviation using the 5 computed accuracy values. When dealing with a single split of the data, we used the standard binomial formula, $\sqrt{p \times (1-p)/n}$, where p is the accuracy and n is the size of the test set.

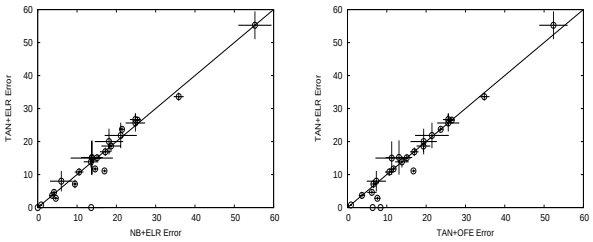

Figure 5. Complete data: Comparing TAN+ELR vs (a) NB+ELR; (b) TAN+OFE

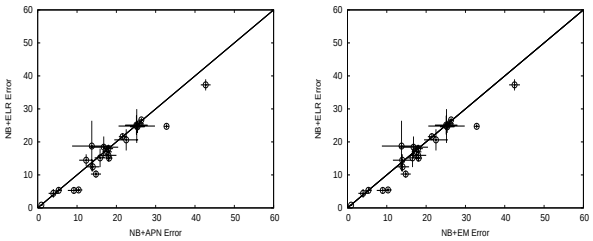

Figure 6. Incomplete data: Comparing NB+ELR vs (a) NB+APN; (b) NB+EM

pair of attributes, conditioned on the class variable, then find the minimum-weighted spanning tree within this complete graph of the attributes. (Each mutual information quantity is based on the empirical sample.) They prove that the resulting structure maximizes the likelihood of the data, over all possible TAN structures — *n.b.*, this is optimizing a generative measure.

Figure 4(b) compares NB+ELR to TAN+OFE. We see that ELR, even when handicapped with the simple NB structure, performs about as well as OFE on TAN structures. Of course, the limitations of the NB structure may explain the poor performance of NB+ELR on some data. For example, in the CORRAL dataset, as the class is a function of four interrelated attributes, one must connect these attributes to predict the class. As Naïve-bayes permits no such connection, Naïve-bayes-based classifiers performed poorly on this data. Of course, as TAN allows more expressive structures, it has a significant advantage here. It is interesting to note that our NB+ELR is still comparable to TAN+OFE, in general.

Would we do yet better by using ELR to instantiate TAN structures? While Figure 5(a) suggests that TAN+ELR is slightly better than NB+ELR, this is not significant. However, Figure 5(b) shows that TAN+ELR does consistently better than TAN+OFE — at a $p < 0.025$ level. We found that TAN+ELR did perfectly on the the CORRAL dataset, which NB+ELR found problematic.

5.1.3. NB, TAN — *Incomplete, Real World Data*

All of the above studies used *complete* data. We next explored how well ELR could instantiate the Naïve-bayes structure, using *incomplete* data.

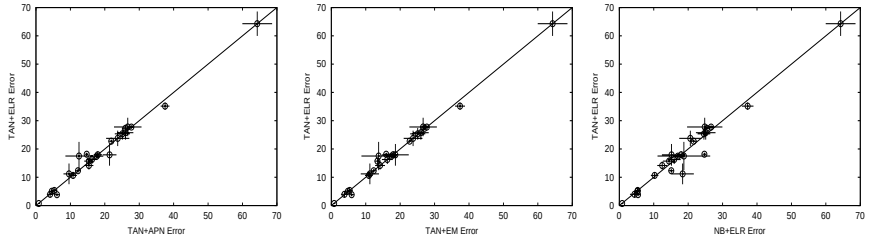

Figure 7. Incomplete data: Comparing TAN+ELR with (a) TAN+APN; (b) TAN+EM; (c) NB+ELR

Here, we used the datasets investigated above, but modified by randomly removing the value of each attribute, within each instance, with probability 0.25. (Hence, this data is "missing completely at random", MCAR [38].) We then compared ELR to the standard "missing-data" learning algorithms, APN and EM. In each case — for ELR, APN and EM — we initialize the parameters using the obvious variant of OFE that considers, for $\beta_{d|\mathbf{f}}$, only the records that include values for the relevant node and all of its parents $\{D\} \cup \mathbf{F}$.

Here, we first learned the parameters for the Naïve-bayes structure; Figure 3(b) shows the learning curve for the CHESS domain, comparing ELR to APN and EM. We see that ELR does better for essentially every sample size. We also compared these algorithms over the rest of the 25 datasets; see Figures 6(a) and 6(b) for ELR vs APN and ELR vs EM, respectively. As shown, ELR does consistently better — in each case, at the $p < 0.025$ level.

We next tried to learn the parameters for a TAN structure. Recall the standard TAN-learning algorithm uses the mutual information between each pair of attributes, conditioned on the class variable. This is straightforward to compute when given complete information. Here, given *incomplete* data, we approximate mutual information between attributes A_i and A_j by simply ignoring the records that do not have values for both of these attributes. Figures 7(a) and 7(b) compare TAN+ELR to TAN+APN and to TAN+EM. We see that these systems are roughly equivalent: while TAN+ELR appears slightly better than TAN+EM, this is not significant (only at $p < 0.25$); similarly there is no significant difference between TAN+ELR and TAN+APN. Finally, we compared NB+ELR to TAN+ELR (Figure 7(c)), but found no significant difference here either.

The table in [25, #Results-InComplete] presents all of our empirical results related to missing data. We also compared these parameter learners on 20 other UCIrvine datasets that are already missing datapoints. Our results here were consistent: NB+ELR was significantly better than NB+EM and NB+APN — NB+ELR $\Leftarrow_{(p<0.0056)}$ NB+EM, NB+ELR $\Leftarrow_{(p<0.026)}$ NB+APN — but there was no statistical separation difference between TAN+ELR and either TAN+EM or TAN+APN— TAN+ELR $\leftarrow_{(p<0.083)}$ TAN+EM, TAN+ELR $\leftarrow_{(p<0.078)}$ TAN+APN. We provide further details in [25, #MissingData].

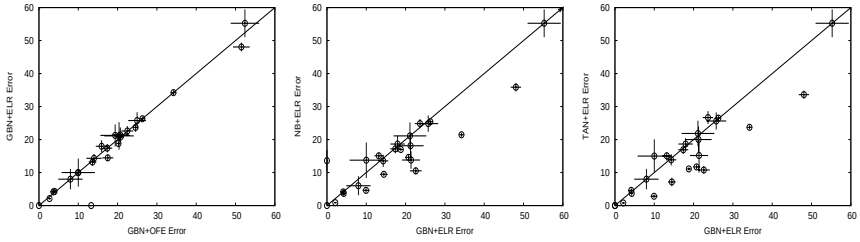

Figure 8. Comparing GBN+ELR vs (a) GBN+OFE; (b) NB+ELR; (c) TAN+ELR

5.2. MODEL APPROXIMATES THE TRUTH ($G \approx T$)

The previous section considered learners that were constrained to consider only some limited class of structures, such as NB or TAN. Other learners are allowed to first learn an arbitrary BN structure — seeking one that matches the underlying distribution — before learning the parameters of that structure, using ELR or OFE, etc. There are a number of algorithms for learning these BN structures, each of which will typically produce a structure that is close to correct. This paper considers the POWERCONSTRUCTOR system [9, 8], which uses mutual information tests to construct BN-structures from complete tuples. This algorithm is guaranteed to converge to the correct belief net structure, given enough data (and some other relatively benign assumptions). We will refer to the resulting POWERCONSTRUCTOR-produced structure as a "General Belief Net", or GBN; see Figure 2(c). This section explores the effectiveness of beginning with such learned structures.

For each of the 25 datasets, we first used POWERCONSTRUCTOR to produce a structure for the given dataset, given all available (non–hold-out) data; we then asked ELR (resp., OFE) to find best parameters for this (presumably near optimal) structure using the same non-hold-out data, and observed how well the resulting system performs on the held-out data.

Section 5.2.1 compares GBN+ELR to GBN+OFE; Section 5.2.2 compares GBN+ELR to simpler models instantiated using ELR; and Section 5.2.3 compares the OFE-instantiation of GBN to ELR-instantiations of simpler models. Section 5.2.4 investigates different algorithms for learning parameters (for these GBN structures) from *incomplete* data.

5.2.1. GBN+ELR *vs* GBN+OFE

Figure 8(a) shows that GBN+ELR is only insignificantly better than GBN+OFE: GBN+ELR $\leftarrow_{(p<0.2)}$ GBN+OFE. Hence, when considering structures that match the underlying distribution, there appears to be little difference between OFE and ELR.

5.2.2. GBN+ELR *vs* NB+ELR, TAN+ELR

The main purpose of our studies was to see how *x*+ELR compares to *x*+OFE (and *x*+EM/OFE), for various classes of commonly-used structures *x*. As a side issue, we also considered some cross-structure comparisons. In particular, given that POWERCONSTRUCTOR had no prior constraints on the structures it can produce, it has the potential of producing classifiers superior to the ones produced by the constrained NB or TAN systems. However, when we compared GBN+ELR to NB+ELR (Figure 8(b)), and to TAN+ELR (Figure 8(c)), we found that the simpler structures actually produced *better* classifiers than GBN did — NB+ELR $\Leftarrow_{(p<0.01)}$ GBN+ELR and TAN+ELR $\Leftarrow_{(p<0.008)}$ GBN+ELR.

There are several possible reasons for this. First, the optimization task for GBN (unlike the ones for NB and TAN) is not convex, meaning it could have local, non-global maxima [52]. The fact that GBN+ELR appears comparable to GBN+OFE suggests that this is not a major issue.

Another possibility is that the GBN structure might not be good for the classification task: POWERCONSTRUCTOR is seeking a good model of the underlying distribution, but might fail. (Recall its guarantee is asymptotic, and we have only a finite sample). Moreover, even if it obtained a good approximation to the underlying distribution, this might not produce a good classifier; see the arguments presented in Section 2.[12] (Of course, as our goal is to compare *x*+ELR to *x*+OFE over commonly used classes *x*, it does not really matter whether POWERCONSTRUCTOR provided a good structure *x* or not.)

5.2.3. GBN+OFE *vs* NB+ELR, TAN+ELR

One approach to learning a good belief net (classifier) is to first find a good structure, then instantiate this structure using the trivial OFE algorithm. The first step can be hard — *e.g.*, NP-hard if seeking the structure that maximizes the BIC score [10].[13] Another approach, suggested by our analysis, is to use a simple structure, such as NB or TAN, but then spend resources finding the best parameters, using ELR.

We therefore compared GBN+OFE to NB+ELR (Figure 9) and found NB+ELR to be significantly better: NB+ELR $\Leftarrow_{(p<0.03)}$ GBN+OFE. Moreover, TAN+ELR is yet stronger: TAN+ELR $\Leftarrow_{(p<0.008)}$ GBN+OFE.

Table I presents a succinct summary of the results on the UCI data, over all 25 datasets. (Note this repeats many of the results from the previous sections.)

[12] As another illustration, imagine the class node C depended on the variables, $\{E_i\}$. POWERCONSTRUCTOR would be happy returning a structure that appeared to match the distribution, even if that structure separated C from many of these relevant E_i's. This cannot happen with either NB or TAN, as these structure connect every variable to C.

[13] This BIC is a generative measure. We suspect finding the best "discriminative structure" would be as difficult. See also the iterative methods used in Grossman and Domingos [28] for this task.

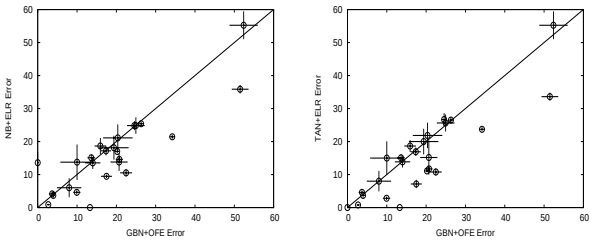

Figure 9. Comparing (a) GBN+OFE vs NB+ELR; (b) GBN+OFE vs TAN+ELR

Table I. Comparison of Different Parameter Learners — *Complete* (UCI) Data; all 25 datasets

	GBN+ELR	GBN+OFE	TAN+ELR	TAN+OFE	NB+ELR
GBN+OFE	$\uparrow$ (0.2)				
TAN+ELR	$\Leftarrow$ (0.008)	$\Leftarrow$ (0.008)			
TAN+OFE	$\Leftarrow$ (0.04)	$\Leftarrow$ (0.02)	$\uparrow$ (0.12)		
NB+ELR	$\Leftarrow$ (0.01)	$\Leftarrow$ (0.03)	$\uparrow$ (0.2)	$\uparrow$ (0.45)	
NB+OFE	$\uparrow$ (0.36)	$\uparrow$ (0.46)	$\Uparrow$ (0.006)	$\Uparrow$ (0.002)	$\Uparrow$ (0.005)

Legend: Each $\langle i, j \rangle$ entry consists of both an arrow that points to the superior learner (using a double arrow $\Uparrow$ or $\Leftarrow$ if this is significant, and a single arrow $\uparrow$ or $\leftarrow$ otherwise); and the associated p-value in parentheses.

The rows and columns are ordered based on our expectation that most of the entries would be $\Uparrow$'s or $\uparrow$'s — *i.e.*, the learners are arranged in the order of (anticipated) decreasing performance. We see some exceptions, but only when we cross structure classes; see above discussion.

5.2.4. GBN+*x vs other classifiers, with Incomplete data*

This section investigates the effectiveness of learning the parameters for GBN structures, from *incomplete* training data. As POWERCONSTRUCTOR is designed for complete data, we actually built each of the structures using complete data. We did this once, using all of the available data.[14]

To produce the data used for learning and evaluating the parameters, we then removed the values of each evidence attribute for each tuple, with probability 0.25 — so again we are dealing with MCAR data [38].

The overall results appear in Table II, where again we expected the majority of the entries to be $\Uparrow$'s or $\uparrow$'s. Most importantly, for each class x, we see that x+ELR is never significantly worse than either x+APN or x+EM, and sometimes it is significantly better. The fact that both TAN+y and NB+ELR

[14] As our goal was only to compare the effectiveness of the parameter-learners on reasonable structures, the source of these structures is irrelevant, and in particular, it does not matter that the structure was generated from all the data.

Table II. Comparison of Parameter Learners — *InComplete* (UCI) Data; all 25 datasets

	GBN ELR	GBN APN	GBN EM	TAN ELR	TAN APN	TAN EM	NB ELR	NB APN
GBN APN	⇑ (0.05)							
GBN EM	↑ (0.09)	↑ (0.25)						
TAN ELR	⇐ (0.02)	⇐ (0.007)	⇐ (0.012)					
TAN APN	⇐ (0.01)	⇐ (0.005)	⇐ (0.009)	↑ (0.32)				
TAN EM	⇐ (0.02)	⇐ (0.006)	⇐ (0.01)	↑ (0.25)	↑ (0.5)			
NB ELR	⇐ (0.02)	⇐ (0.01)	⇐ (0.02)	↑ (0.4)	↑ (0.32)	↑ (0.3)		
NB APN	↑ (0.08)	↑ (0.06)	⇑ (0.04)	↑ (0.07)	↑ (0.06)	⇑ (0.04)	⇑ (0.025)	
NB EM	↑ (0.06)	⇑ (0.05)	⇑ (0.04)	↑ (0.055)	⇑ (0.05)	⇑ (0.035)	⇑ (0.02)	↑ (0.2)

appear uniformly better than GBN+y, is consistent with the case for complete data; see Section 5.2.2.

5.3. DISCUSSION

This section has presented a number of empirical results, all in the context of producing a good belief-net based classifiers for a fixed structure. The main take-home messages are...

⋆ The discriminative learner ELR system works effectively in essentially every standard situation: Given complete data, it is at least as good as, and often superior to, OFE (Table I) and given incomplete data, it is at least as good as, and often superior to, APN and EM (Table II).[15]

⋆ While we typically found more expressive models produced better classifiers (*i.e.*, for each parameter-learner z, GBN+z was better than TAN+z, and TAN+z was better than NB+z), this was not universal; see discussion in Section 5.2.2.

Why ELR Works Well: We found that ELR worked effectively in many situations, and it was especially advantageous (*i.e.*, typically better than the alternative ways to instantiate parameters) when the BN-structure was *incorrect* — *i.e.*, when it is not an I-map of the underlying distribution by incorrectly claiming that two dependent variables are independent [45]. This is a very common situation, as many BN-learners will produce incorrect

[15] [25, #G>T] discusses another situation, where the model is *more* complex than the truth. It presents empirical data that the generative models often work better in this uncommon situation, and explains why.

structures, either because they are conservative in adding new arcs (to avoid overfitting the data [30, 51]), or because they are considering only a restricted class of structures (*e.g.*, Naïve-bayes [21], poly-tree [12, 45], TAN [24], etc.) that is not guaranteed to contain the correct structure.

To understand why a bad structure is problematic for OFE, note that when OFE is seeking the parameter $\theta_{d|\mathbf{f}}$, it is constrained to match the *local* empirical distribution, corresponding to $\#(D=d,\mathbf{F}=\mathbf{f})/\#(\mathbf{F}=\mathbf{f})$. Hence, if the given structure G is incorrect, the resulting instantiated belief net need not be a good model of the true tuple distribution, and so may return incorrect values for the queries. By contrast, the ELR algorithm is not as constrained by the specific structure, and so may be able to produce parameters that yield fairly accurate answers, even if the structure is sub-optimal. (See the standard comparison between discriminative versus generative training, overviewed in Section 6.)

Other Learners: We also compared ELR to various other learning algorithms, including SVMs, over these datasets. We found that ELR appeared comparable with several other standard learning algorithms, and superior to some. See [25, #OtherLearners] for details, which also repeats the results from Friedman *et al.* [24] and Grossman and Domingos [28]. That webpage also provides other information about the experiments; *e.g.*, presenting timing information, the $\widehat{\mathrm{LCL}}(\cdot)$ scores, etc.

6. Related Results

There are a number of other results related to learning belief nets. Much of this work focuses on learning the best *structure*, either for a general belief net, or within the context of some specific class of structures (*e.g.*, TAN-structures, or selective Naïve-bayes); see [30, 6] for extensive tutorials. By contrast, this paper suggests a way to learn the *parameters* for a given structure.

While most of those structure-learning systems also learn the parameters, essentially all use the OFE algorithm (Equation 13). This is well motivated in the *generative* situation, as these parameter values do optimize the likelihood of the data [14].

As noted earlier, however, our goal is different: as we are seeking the optimal *classifier — i.e., discriminative learning*. There have been many other systems that also considered discriminative learning of belief nets [35, 31, 24, 8, 28]. This research, like their generative counterparts, focused on learning structures; and usually used OFE to instantiate the resulting parameters.

As we saw above, when the model is wrong, the OFE-based parameters can produce inferior classifiers. Many researchers have employed tricks to improve the parameters; *e.g.*, tractable Bayesian model averaging of TAN [7], and exact model averaging of naïve-bayes [18]. Our approach is different, as

we explicitly seek the parameters of the BN model that maximize conditional likelihood.

Our results also relate closely to the work on *discriminant learning of Hidden Markov Models (HMMs)* [48, 11]. In particular, much of that work uses "Generalized Probabilistic Descent", which resembles our ELR system by descending along the derivative of the parameters, to maximize the conditional likelihood of the hypothesis (which typically correspond to specific words) given the observations — which they call "Maximimum Mutual Information" criterion. We differ by considering arbitrary structures, and evaluating based on classification error.

Edwards & Lauritzen [22] proposed the TM algorithm for maximizing conditional likelihood function, when the corresponding uncondition likelihood function is more easily maximized. They have found the algorithm is a useful tool in complex CG-regression models, which are the building blocks for graphical chain models, as well as in other situations [50]. Their TM algorithm is similar to the EM algorithm as it also alternates between maximization of a function related to the true likelihood function, but differs by being applied to the complete data case and by augmenting the parameters rather than the data.

This issue relates directly to the large literature on *discriminative learning* in general; see [15, 32, 47]. One standard model is Linear Discriminant Analysis (LDA), which typically assumes $P(\mathbf{E}\,|\,C=c)$ is multivariate normal — *i.e.*, $P(\mathbf{E}\,|\,C=c) \sim \mathcal{N}(\mu_c, \Sigma)$ where each μ_c mean can depend on the class $C=c$, but the covariance matrix Σ is the same for all classes. The LDA system then estimates the relevant $\{\mu_c,\ \Sigma,\ \hat{P}(C=c)\}$ parameters from a body of data, seeking the ones that maximize the likelihood of the data relevant to those parameters. Given these parameters, we can then use Bayes Rule to compute the conditional distribution of $P(C\,|\,\mathbf{E}=\mathbf{e}')$ given new evidence $\mathbf{E}=\mathbf{e}'$.

We can view LDA (like OFE/APN/EM) as being generative (aka "causal" or "class-conditional" [32], or "sampling" [19]), as it is attempting to fit parameters for the *entire* joint distribution, while our ELR is discriminative (aka "diagnostic", "predictive" [32]), as it focuses only on the *conditional* probabilities.

Our results echo the common wisdom obtained by the previous analyses of discriminative systems. In particular, (1) accuracy: discriminative training typically produces more accurate classifiers than generative training; (2) robustness: typically discriminative systems are more robust against incorrect models than generative ones; (3) efficiency: generative can be more efficient than discriminative (compare the efficient OFE with the iterative ELR).

The work reported in this paper has significant differences from those earlier analyses. First, we are dealing with a different underlying model, based on *discrete* variables (rather than continuous, say normally distributed, ones), in the context of a *specified belief net structure*, which corresponds to

a given set of independency claims. We also describe the inherent computational complexity of this task, produce algorithms specific to our task, and provide empirical studies to demonstrate that our algorithm works effectively, given either complete or incomplete training data. ([25, #CompareGGS97] compares the results in this paper with [26].)

7. Conclusions

Future Work: Section 3 notes that, in some situations (a specified class of structures, when given complete data), interior point methods can find the parameters that optimize conditional likelihood, in polynomial time. Of course, it is not clear whether these LCL$(\cdot)$-optimal parameters will optimize error (Equation 3), nor that the algorithm will necessarily be more efficient than ELR. We therefore plan to investigate these methods.

This paper explores the challenges of finding in the CPtables of a given BN-structure. While this is an important subtask, a general learner should be able to learn that structure as well — perhaps using *conditional likelihood* as the selection criterion; see [35, 31]. We plan to investigate ways to synthesize these approaches; see [28].

There are now several other classes of graphical models, such as Conditional Random Fields [36], that may be better adapted to optimizing conditional likelihood. (*E.g.*, as they use undirected arcs, they require only a single normalizing division, rather than one per CPtable row.) While this paper has focused on Belief Nets (as it is typically easier to acquire meaningful structures here, both because they allow users to express their prior knowledge, and because there are a number of algorithms for learning belief net structures), we plan to investigate these other models as well.

Contributions: This paper overviews the task of discriminative learning of belief net parameters for general BN-structures. We first describe this task, and discuss how it extends that standard logistic regression process by applying to arbitrary structures, not just naïve-bayes — see Equation 2. Our formal analyses show that, in general, discriminative learners can converge to a classifier optimizing conditional likelihood at essentially the same $O(\cdot)$ sample rate (ignoring polylog terms) as a generative classifier that is optimizing likelihood.[16] Moreover, it is well-known that discriminative learning can converge to a classifier superior to one learned generatively [44]. We also found that the *computational* complexities of these two tasks also appear fairly comparable; see Table III. (This is just the cost of finding good parameters, for a given structure. As this structure is not as critical in our discriminative case, we anticipate further savings in the overall task of learning both structure and parameters.)

[16] This differs from the Ng & Jordan [44] result, which compared only the simplest form, Naïve-bayes vs logistic regression, and dealt with error itself.

Table III. Summary of Known Complexity Results

	Complete Data	Incomplete Data
Likelihood	in P: (OFE)	Unknown (EM, APN)
Conditional Likelihood	in P for simple structures [52] Unknown in general	NP-hard (Theorem 2)

We next present an algorithm ELR for our task, and show that ELR works effectively over a variety of situations: when dealing with structures that range from trivial (NB), through less-trivial (TAN), to complex (ones learned by POWERCONSTRUCTOR). We also show that ELR works well when given *incomplete* training data. In particular, in essentially every standard situation, we see that ELR is at least as good, and often better, than the other contenders, which seek a good generative methods.

The idea of discriminative learning, especially in the context of belief nets, is only beginning to make in-roads into the general AI community. We hope this paper will help further introduce these ideas to this community, and demonstrate that these algorithms should be used here, as they can work very effectively.

ACKNOWLEDGEMENTS

We thank C. Cheng, T. Dietterich, A. Grove, P. Hooper, J. Lafferty, D. Schuurmans, and the anonymous reviewers for their many helpful suggestions. We also thank J. Cheng and T. Joachims for letting us use their POWERCONSTRUCTOR and SVM-Light systems. RG and WZ were partially funded by NSERC; RG and XS were also funded by the Alberta Ingenuity Centre for Machine Learning; and WZ, by Syncrude.

References

1. N. Abe, J. Takeuchi, and M. Warmuth. Polynomial learnability of probablistic concepts with respect to the Kullback-Leibler divergence. In *COLT*, 1991.
2. J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.
3. C. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1998.
4. C. Blake and C. Merz. UCI repository of machine learning databases, 2000. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
5. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge, 2004.
6. W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 1996.
7. J. Cerquides and R. L. de Mántaras. Tractable Bayesian learning of tree augmented naïve Bayes models. In *ICML-03*, pages 75–82, 2003.
8. J. Cheng and R. Greiner. Comparing Bayesian network classifiers. In *UAI*, 1999.
9. J. Cheng, R. Greiner, and J. Kelly. Learning Bayesian networks from data: an information-theory based approach. *Artificial Intelligence*, 137, 2002.

10. D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research, Nov. 1994.
11. W. Chou, B. Juang, and C. Lee. Segmental GPD training of HMM based speech recognizer. In *ICASSP*, volume 1, pages 473–476, 1992.
12. C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, pages 462–467, 1968.
13. G. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42, 1990.
14. G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning Journal*, 9:309–347, 1992.
15. D. R. Cox and E. J. Snell. *Analysis of Binary Data*. Chapman & Hall, London, 1989.
16. A. Darwiche. A differential approach to inference in Bayesian networks. In *UAI'00*, 2000.
17. S. Dasgupta. The sample complexity of learning fixed-structure Bayesian networks. *Machine Learning*, 29:165–180, 1997.
18. D. Dash and G. Cooper. Exact model averaging with naïve Bayesian classifiers. In *ICML-02*, pages 91–98, 2002.
19. A. P. Dawid. Properties of diagnostic data distributions. *Biometrics*, 32:647–658, 1976.
20. A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. (with discussion). *J. Royal Statistics Society, Series B*, 39, 1977.
21. R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
22. D. Edwards and S. Lauritzen. The TM algorithm for maximising a conditional likelihood function. *Biometrika*, 88:961–972, 2001.
23. U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, 1993.
24. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning Journal*, 29:131–163, 1997.
25. 2004. http://www.cs.ualberta.ca/∼greiner/ELR.
26. R. Greiner, A. Grove, and D. Schuurmans. Learning Bayesian nets that perform well. In *Uncertainty in Artificial Intelligence*, 1997.
27. R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminant parameter learning of belief net classifiers. In *AAAI*, 2002.
28. D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *ICML2004*, 2004.
29. M. Hagan, H. Demuth, and M. Beale. *Neural Network Design*. PWS Publishing, Boston, MA, 1996.
30. D. E. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*, 1998.
31. T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *NIPS2000*, 2000.
32. M. Jordan. Why the logistic function? a tutorial discussion on probabilities and neural networks, 1995.
33. R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, 1995.
34. R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2), 1997.
35. P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. On supervised selection of Bayesian networks. In *UAI99*, pages 334–342, 1999.
36. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML2001*, 2001.

37. S. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.
38. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley, New York, 1987.
39. P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman and Hall, 1989.
40. T. Minka. Algorithms for maximum-likelihood logistic regression. Technical report, CMU CALD, 2001. http://www.stat.cmu.edu/∼minka/papers/logreg/minka-logreg.pdf.
41. T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
42. C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52:239–281, 2003.
43. Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Methods in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.
44. A. Ng and M. Jordan. On discriminative versus generative classifiers: A comparison of logistic regression and naive Bayes. In *NIPS*, 2001.
45. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.
46. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge, 2002. http://www.nr.com/.
47. B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, UK, 1996.
48. R. Schlüter, W. Macherey, S. Kanthak, H. Ney, and L. Welling. Comparison of optimization methods for discriminative training criteria. In *Proc. EUROSPEECH'97*, 1997.
49. B. Shen, X. Su, R. Greiner, P. Musilek, and C. Cheng. Discriminative parameter learning of general Bayesian network classifiers. In *ICTAI-03*, 2003.
50. R. Sundberg. The convergence rate of the TM algorithm of Edwards and Lauritzen. *Biometrika*, 89:478–483, 2002.
51. T. Van Allen and R. Greiner. Model selection criteria for learning belief nets: An empirical comparison. In *ICML'00*, pages 1047–1054, 2000.
52. H. Wettig, P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri. When discriminative learning of Bayesian network parameters is easy. In *IJCAI2003*, pages 491–496, 2003.
53. W. Zhou. Discriminative learning of Bayesian net parameters. Master's thesis, Dept of Computing Science, University of Alberta, 2002.

A. Proof

Proof of Theorem 1 (sketch; see [25, #Proof]): Let $\Theta^{(1)} = \{\theta^{(1)}_{d_i|\mathbf{f}_i}\}_i$ and $\Theta^{(2)} = \{\theta^{(2)}_{d_i|\mathbf{f}_i}\}_i$ be two CPtables in $\mathcal{BN}_{\Theta\succeq\gamma}(G)$. We prove below

$$\begin{array}{ll} \text{if} & \forall i \ \left|\theta^{(1)}_{d_i|\mathbf{f}_i} - \theta^{(2)}_{d_i|\mathbf{f}_i}\right| \ \leq \ \frac{\gamma\varepsilon}{6K} \\ \text{then} & \forall c, \mathbf{e} \ \ |\ln(P_{\Theta^{(1)}}(c\,|\,\mathbf{e})) \ - \ \ln(P_{\Theta^{(2)}}(c\,|\,\mathbf{e}))| \ \leq \ \frac{\varepsilon}{6} \end{array} \tag{14}$$

implying $|\mathrm{LCL}_P(\Theta^{(1)}) - \mathrm{LCL}_P(\Theta^{(2)})| \leq \frac{\varepsilon}{6}$ and $|\widehat{\mathrm{LCL}}(\Theta^{(1)}) - \mathit{ECLs}\Theta^{(2)}| \leq \frac{\varepsilon}{6}$.

Now partition the $\mathcal{BN}_{\Theta\succeq\gamma}(G)$ space into $L = (\frac{6K}{\gamma\varepsilon})^K$ disjoint sets (where any two BNs from any partition will have similar CPtable values — *i.e.*,

satisfy condition of Equation 14), then define the set $R = \{\Theta_i\}_i$ to contain one representative from each partition. As $\Theta \in \mathcal{BN}_{\Theta \trianglerighteq \gamma}(G)$, $P_\Theta(c) \geq \gamma^N$ and so $\ln(P_\Theta(c\,|\,\mathbf{e})) \in [N \ln \gamma, 0]$. Hence Hoeffding's Inequality implies a sample S of size $M\left(\frac{\varepsilon}{6}, \frac{\delta}{L}\right) = 2\left(\frac{3N\log\gamma}{\varepsilon}\right)^2 \ln\frac{2L}{\delta}$ is sufficient for $P(|\widehat{\mathrm{LCL}}^{(S)}(\Theta_i) - \mathrm{LCL}(\Theta_i)| > \frac{\varepsilon}{6}) < \frac{\delta}{L}$.
As there are L representatives in R, the probability that *any* of the representative's scores are mis-estimated by more than $\varepsilon/6$ is at most $L\frac{\delta}{L} = \delta$.

This means we have, in effect, estimated the scores on *any* $\Theta \in \mathcal{BN}_{\Theta \trianglerighteq \gamma}(G)$ to within $\varepsilon/2$: For any $\Theta \in \mathcal{BN}_{\Theta \trianglerighteq \gamma}(G)$, let $\Theta' \in R$ be the representative in Θs partition. Observe $|\widehat{\mathrm{LCL}}(\Theta) - \mathrm{LCL}(\Theta)|$

$$\begin{array}{lccccccc} \leq & |\widehat{\mathrm{LCL}}(\Theta) - \widehat{\mathrm{LCL}}(\Theta')| & + & |\widehat{\mathrm{LCL}}(\Theta') - \widehat{\mathrm{LCL}}(\Theta')| & + & |\mathrm{LCL}(\Theta') - \mathrm{LCL}(\Theta)| & & \\ \leq & \varepsilon/6 & + & \varepsilon/6 & + & \varepsilon/6 & = & \varepsilon/2\,. \end{array}$$

As our estimates of both $\widehat{\Theta}$ and Θ^* are within $\varepsilon/2$, $\mathrm{LCL}(\widehat{\Theta}) - \mathrm{LCL}(\Theta^*)$

$$\begin{array}{lccccccc} \leq & |\mathrm{LCL}(\widehat{\Theta}) - \widehat{\mathrm{LCL}}(\widehat{\Theta})| & + & \widehat{\mathrm{LCL}}(\widehat{\Theta}) - \widehat{\mathrm{LCL}}(\Theta^*) & + & |\widehat{\mathrm{LCL}}(\Theta^*) - \mathrm{LCL}(\Theta^*)| & & \\ \leq & \varepsilon/2 & + & 0 & + & \varepsilon/2 & = & \varepsilon \end{array}$$

as desired.

To prove Equation 14: Consider the sequence $\Theta_0, \Theta_1, \ldots, \Theta_K$ where the first i of Θ_i come from $\Theta^{(1)}$, and the remaining from $\Theta^{(2)}$ — *i.e.*,

$$\Theta_i \quad \sim \quad \{\theta^{(1)}_{d_1|\mathbf{f}_1}, \ldots, \theta^{(1)}_{d_i|\mathbf{f}_i}, \theta^{(2)}_{d_{i+1}|\mathbf{f}_{i+1}}, \ldots, \theta^{(2)}_{d_K|\mathbf{f}_K}\}\,.$$

Now observe

$$|\ln(P_{\Theta^{(1)}}(c\,|\,\mathbf{e})) - \ln(P_{\Theta^{(2)}}(c\,|\,\mathbf{e}))| \;\leq\; \textstyle\sum_{i=1}^{K} |\ln(P_{\Theta_i}(c\,|\,\mathbf{e})) - \ln(P_{\Theta_{i-1}}(c\,|\,\mathbf{e}))|\,,$$

and each $|\ln(P_{\Theta_i}(c\,|\,\mathbf{e})) - \ln(P_{\Theta_{i-1}}(c\,|\,\mathbf{e}))|$ is based on changing a single CPtable entry. We therefore need only show $|\ln(P_{\Theta_i}(c\,|\,\mathbf{e})) - \ln(P_{\Theta_{i-1}}(c\,|\,\mathbf{e}))| \leq \frac{\varepsilon}{6K}$. For any value of $z = \theta_{d_i|\mathbf{f}_i}$, let $f(z) = \ln(P_{\Theta[z]}(c\,|\,\mathbf{e}))$, where $\Theta[z]$ be the BN whose first $i-1$ CPtable entries come from $\Theta^{(1)}$, whose final $K-i-1$ entries come from $\Theta^{(2)}$, and whose i^{th} CPtable entries is z; hence $f(\theta^{(1)}_{d_i|\mathbf{f}_i}) = \ln(P_{\Theta_i}(c\,|\,\mathbf{e}))$, and $f(\theta^{(2)}_{d_i|\mathbf{f}_i}) = \ln(P_{\Theta_{i+1}}(c\,|\,\mathbf{e}))$. As this function is continuous, $|f(a) - f(b)| = \frac{\partial f(z)}{\partial z}[b-a]$ for some $z \in [a,b]$. As $f(z) = \ln(P_{\Theta[z]}(c, \mathbf{e})) - \ln(P_{\Theta[z]}(\mathbf{e}))$,

$$\begin{array}{rcl} \frac{\partial f(z)}{\partial z} & = & \frac{1}{P_{\Theta[z]}(c,\mathbf{e})} P_{\Theta[z]}(c, \mathbf{e}\,|\,d_i, \mathbf{f}_i) \times P_{\Theta[z]}(\mathbf{f}_i) \;-\; \frac{1}{P_{\Theta[z]}(\mathbf{e})} P_{\Theta[z]}(\mathbf{e}\,|\,d_i, \mathbf{f}_i) \times P_{\Theta[z]}(\mathbf{f}_i) \\ & = & \frac{1}{z}[P_{\Theta[z]}(d_i, \mathbf{f}_i\,|\,c, \mathbf{e}) - P_{\Theta[z]}(d_i, \mathbf{f}_i\,|\,\mathbf{e})] \end{array}$$

which means that $|\frac{\partial f(z)}{\partial z}| \leq 1/z \leq 1/\gamma$. (Recall $\Theta \in \mathcal{BN}_{\Theta \trianglerighteq \gamma}(G)$.) Hence,

$$\begin{array}{rcl} |\ln(P_{\Theta_{i+1}}(c\,|\,\mathbf{e})) - \ln(P_{\Theta_i}(c\,|\,\mathbf{e}))| & = & |f(\theta^{(2)}_{d_i|\mathbf{f}_i}) - f(\theta^{(1)}_{d_i|\mathbf{f}_i})| \\ \leq \quad \frac{1}{\gamma} \times |\theta^{(2)}_{d_i|\mathbf{f}_i} - \theta^{(1)}_{d_i|\mathbf{f}_i}| & \leq & \frac{1}{\gamma} \times \frac{\gamma\varepsilon}{6K} \;=\; \frac{\varepsilon}{6K}\,. \end{array} \qquad \blacksquare$$