Computationally Efficient M-Estimation of Log-Linear Structure Models*

Noah A. Smith and **Douglas L. Vail** and **John D. Lafferty**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{nasmith,dvail2,lafferty}@cs.cmu.edu

Abstract

We describe a new loss function, due to Jeon and Lin (2006), for estimating structured log-linear models on arbitrary features. The loss function can be seen as a (generative) alternative to maximum likelihood estimation with an interesting information-theoretic interpretation, and it is statistically consistent. It is substantially faster than maximum (conditional) likelihood estimation of conditional random fields (Lafferty et al., 2001; an order of magnitude or more). We compare its performance and training time to an HMM, a CRF, an MEMM, and pseudolikelihood on a shallow parsing task. These experiments help tease apart the contributions of rich features and discriminative training, which are shown to be more than additive.

1 Introduction

Log-linear models are a very popular tool in natural language processing, and are often lauded for permitting the use of "arbitrary" and "correlated" features of the data by a model. Users of log-linear models know, however, that this claim requires some qualification: any feature is permitted in principle, but training log-linear models (and decoding under them) is tractable only when the model's independence assumptions permit efficient inference procedures. For example, in the original conditional random fields (Lafferty et al., 2001), features were confined to locally-factored indicators on label bigrams and label unigrams (with any of the observation).

Even in cases where inference in log-linear models is tractable, it requires the computation of a partition function. More formally, a log-linear model for random variables X and Y over $\mathcal{X}, \mathcal{Y}$ defines:

$$p_{\mathbf{w}}(x, y) = \frac{e^{\mathbf{w}^\top \mathbf{f}(x,y)}}{\sum_{x',y' \in \mathcal{X}\times\mathcal{Y}} e^{\mathbf{w}^\top \mathbf{f}(x',y')}} = \frac{e^{\mathbf{w}^\top \mathbf{f}(x,y)}}{Z(\mathbf{w})} \tag{1}$$

where $\mathbf{f} : \mathcal{X}\times\mathcal{Y} \rightarrow \mathbb{R}^m$ is the feature vector-function and $\mathbf{w} \in \mathbb{R}^m$ is a weight vector that parameterizes the model. In NLP, we rarely train this model by maximizing *likelihood*, because the partition function $Z(\mathbf{w})$ is expensive to compute exactly. $Z(\mathbf{w})$ can be approximated (e.g., using Gibbs sampling; Rosenfeld, 1997).

In this paper, we propose the use of a new loss function that is computationally **efficient** and statistically **consistent** (§2). Notably, repeated inference is not required during estimation. This loss function can be seen as a case of M-estimation[1] that was originally developed by Jeon and Lin (2006) for nonparametric density estimation. This paper gives an information-theoretic motivation that helps elucidate the objective function (§3), shows how to apply the new estimator to structured models used in NLP (§4), and compares it to a state-of-the-art noun phrase chunker (§5). We discuss implications and future directions in §6.

2 Loss Function

As before, let X be a random variable over a high-dimensional space $\mathcal{X}$, and similarly Y over $\mathcal{Y}$. $\mathcal{X}$

*This work was supported by NSF grant IIS-0427206 and the DARPA CALO project. The authors are grateful for feedback from David Smith and from three anonymous ACL reviewers, and helpful discussions with Charles Sutton.

[1] "M-estimation" is a generalization of MLE (van der Vaart, 1998); space does not permit a full discussion.

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 752–759,
Prague, Czech Republic, June 2007.

might be the set of all sentences in a language, and $\mathcal{Y}$ the set of all POS tag sequences or the set of all parse trees. Let q_0 be a "base" distribution that is our first approximation to the true distribution over $\mathcal{X} \times \mathcal{Y}$. HMMs and PCFGs, while less accurate as predictors than the rich-featured log-linear models we desire, might be used to define q_0.

The model we estimate will have the form

$$p_{\mathbf{w}}(x, y) \propto q_0(x, y) e^{\mathbf{w}^\top \mathbf{f}(x,y)} \tag{2}$$

Notice that $p_{\mathbf{w}}(x, y) = 0$ whenever $q_0(x, y) = 0$. It is therefore important for q_0 to be smooth, since the support of $p_{\mathbf{w}}$ is a subset of the support of q_0. Notice that we have not written the partition function explicitly in Eq. 2; it will never need to be computed during estimation or inference. The unnormalized distribution will suffice for all computation.

Suppose we have observations $\langle x_1, x_2, ..., x_n\rangle$ with annotations $\langle y_1, ..., y_n\rangle$. The (unregularized) loss function, due to Jeon and Lin (2006), is[2]

$$\begin{aligned}
\ell(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n} e^{-\mathbf{w}^\top \mathbf{f}(x_i,y_i)} \\
&\quad + \sum_{x,y} q_0(x, y) \left(\mathbf{w}^\top \mathbf{f}(x, y)\right) \qquad (3)\\
&= \frac{1}{n}\sum_{i=1}^{n} e^{-\mathbf{w}^\top \mathbf{f}(x_i,y_i)} + \mathbf{w}^\top \sum_{x,y} q_0(x, y)\mathbf{f}(x, y) \\
&= \frac{1}{n}\sum_{i=1}^{n} e^{-\mathbf{w}^\top \mathbf{f}(x_i,y_i)} + \mathbf{w}^\top \underbrace{\mathbf{E}_{q_0(X,Y)}[\mathbf{f}(X, Y)]}_{\text{constant}(\mathbf{w})}
\end{aligned}$$

Before explaining this objective, we point out some attractive computational properties. Notice that $\mathbf{f}(x_i, y_i)$ (for all i) and the expectations of the feature vectors under q_0 are *constant* with respect to $\mathbf{w}$. Computing the function in Eq. 3, then, requires no inference and no dynamic programming, only $O(nm)$ floating-point operations.

3 An Interpretation

Here we give an account of the loss function as a way of "cleaning up" a mediocre model (q_0). We show that this estimate aims to *model* a presumed perturbation that created q_0, by minimizing the KL divergence between q_0 and a perturbed version of the sample distribution $\tilde{p}$.

Consider Eq. 2. Given a training dataset, maximizing *likelihood* under this model means assuming that there is some $\mathbf{w}^*$ for which the true distribution $p^*(x, y) = p_{\mathbf{w}^*}(x, y)$. Carrying out MLE, however, would require computing the partition function $\sum_{x',y'} q_0(x', y')e^{\mathbf{w}^\top \mathbf{f}(x',y')}$, which is in general intractable. Rearranging Eq. 2 slightly, we have

$$q_0(x, y) \propto p^*(x, y)e^{-\mathbf{w}^\top \mathbf{f}(x,y)} \tag{4}$$

If q_0 is close to the true model, $e^{-\mathbf{w}^\top \mathbf{f}(x,y)}$ should be close to 1 and $\mathbf{w}$ close to zero. In the sequence model setting, for example, if q_0 is an HMM that explains the data well, then the additional features are not necessary (equivalently, their weights should be 0). If q_0 is imperfect, we might wish to make it more powerful by adding features (e.g., $\mathbf{f}$), but q_0 nonetheless provides a reasonable "starting point" for defining our model.

So instead of maximizing likelihood, we will *minimize* the KL divergence between the two sides of Eq. 4.[3]

$$\begin{aligned}
&D_{\mathrm{KL}}(q_0(x, y)\|p^*(x, y)e^{-\mathbf{w}^\top \mathbf{f}(x,y)}) \qquad (5)\\
&= \sum_{x,y} q_0(x, y) \log \frac{q_0(x, y)}{p^*(x, y)e^{-\mathbf{w}^\top \mathbf{f}(x,y)}} \qquad (6)\\
&\quad + \sum_{x,y} p^*(x, y)e^{-\mathbf{w}^\top \mathbf{f}(x,y)} - \sum_{x,y} q_0(x, y)\\
&= -H(q_0) + \sum_{x,y} p^*(x, y)e^{-\mathbf{w}^\top \mathbf{f}(x,y)} - 1\\
&\quad - \sum_{x,y} q_0(x, y) \log \left(p^*(x, y)e^{-\mathbf{w}^\top \mathbf{f}(x,y)}\right)\\
&= \text{constant}(\mathbf{w}) + \sum_{x,y} p^*(x, y)e^{-\mathbf{w}^\top \mathbf{f}(x,y)}\\
&\quad + \sum_{x,y} q_0(x, y) \left(\mathbf{w}^\top \mathbf{f}(x, y)\right)
\end{aligned}$$

[2]We give only the discrete version here, because it is most relevant for an ACL audience. Also, our linear function $\mathbf{w}^\top \mathbf{f}(x_i, y_i)$ is a simple case; another kernel (for example) could be used.

[3]The KL divergence here is generalized for unnormalized distributions, following O'Sullivan (1998):

$$D_{\mathrm{KL}}(\mathbf{u}\|\mathbf{v}) = \sum_j \left(u_j \log \frac{u_j}{v_j} - u_j + v_j\right)$$

where $\mathbf{u}$ and $\mathbf{v}$ are nonnegative vectors defining unnormalized distributions over the same event space. Note that when $\sum_j u_j = \sum_j v_j = 1$, this formula takes on the more familiar form, as $-\sum_j u_j$ and $\sum_j v_j$ cancel.

If we replace p^* with the empirical (sampled) distribution $\tilde{p}$, minimizing the above KL divergence is equivalent to minimizing $\ell(\mathbf{w})$ (Eq. 3). It may be helpful to think of $-\mathbf{w}$ as the parameters of a process that "damage" the true model p^*, producing q_0, and the estimation of $\mathbf{w}$ as learning to undo that damage.

In the remainder of the paper, we use the general term "M-estimation" to refer to the minimization of $\ell(\mathbf{w})$ as a way of training a log-linear model.

4 Algorithms for Models of Sequences and Trees

We discuss here some implementation aspects of the application of M-estimation to NLP models.

4.1 Expectations under q_0

The base distribution q_0 enters into implementation in two places: $\mathbf{E}_{q_0(X,Y)}[\mathbf{f}(X,Y)]$ must be computed for training, and $q_0(x,y)$ is a factor in the model used in *decoding*.

If q_0 is a familiar stochastic grammar, such as an HMM or a PCFG, or any generative model from which sampling is straightforward, it is possible to estimate the feature expectations by sampling from the model directly; for sample $\langle(\tilde{x}_i,\tilde{y}_i)\rangle_{i=1}^{s}$ let:

$$\mathbf{E}_{q_0(X,Y)}[f_j(X,Y)] \leftarrow \frac{1}{s}\sum_{i=1}^{s} f_j(\tilde{x}_i,\tilde{y}_i) \quad (7)$$

If the feature space is sparse under q_0 (likely in most settings), then smoothing may be required.

If q_0 is an HMM or a PCFG, the expectation vector can be computed exactly by solving a system of equations. We will see that for the common cases where features are local substructures, inference is straightforward. We briefly describe how this can be done for a bigram HMM and a PCFG.

4.1.1 Expectations under an HMM

Let $\mathcal{S}$ be the state space of a first-order HMM. If $\mathbf{s} = \langle s_1,...,s_k\rangle$ is a state sequence and $\mathbf{x} = \langle x_1,...,x_k\rangle$ is an observed sequence of emissions, then:

$$q_0(\mathbf{s},\mathbf{x}) = \left(\prod_{i=1}^{k} t_{s_{i-1}}(s_i)e_{s_i}(x_i)\right) t_{s_k}(\mathrm{stop}) \quad (8)$$

(Assume $s_0 = \mathrm{start}$ is the single, silent, initial state, and stop is the only stop state, also silent. We assume no other states are silent.)

The first step is to compute path-sums *into* and *out of* each state, under the HMM q_0. To do this, define i_s as the total weight of state-prefixes (beginning in start) ending in s and o_s as the total weight of state-suffixes beginning in s (and ending in stop):[4]

$$\begin{aligned} & i_{\mathrm{start}} = o_{\mathrm{stop}} = 1 \quad (9)\\ & \forall s \in \mathcal{S}\setminus\{\mathrm{start},\mathrm{stop}\}: \\ i_s &= \sum_{n=1}^{\infty}\sum_{\langle s_1,\ldots,s_n\rangle\in\mathcal{S}^n}\left(\prod_{i=1}^{n} t_{s_{i-1}}(s_i)\right) t_{s_n}(s) \\ &= \sum_{s'\in\mathcal{S}} i_{s'}t_{s'}(s) \quad (10)\\ o_s &= \sum_{n=1}^{\infty}\sum_{\langle s_1,\ldots,s_n\rangle\in\mathcal{S}^n} t_s(s_1)\left(\prod_{i=2}^{n} t_{s_{i-1}}(s_i)\right) \\ &= \sum_{s'\in\mathcal{S}} t_s(s')o_{s'} \quad (11) \end{aligned}$$

This amounts to two linear systems given the transition probabilities t, where the variables are $i_\bullet$ and $o_\bullet$, respectively. In each system there are $|\mathcal{S}|$ variables and $|\mathcal{S}|$ equations. Once solved, expected counts of transition and emission features under q_0 are straightforward:

$$\begin{aligned} \mathbf{E}_{q_0}[s \overset{\mathrm{transit}}{\rightarrow} s'] &= i_s t_s(s') o_{s'} \\ \mathbf{E}_{q_0}[s \overset{\mathrm{emit}}{\rightarrow} x] &= i_s e_s(x) o_s \end{aligned}$$

Given i and o, $\mathbf{E}_{q_0}$ can be computed for other features in the model in a similar way, provided they correspond to contiguous substructures. For example, a feature f_{627} that counts occurrences of "$S_i = s$ and $X_{i+3} = x$" has expected value $\mathbf{E}_{q_0}[f_{627}] =$

$$\sum_{s',s'',s'''\in\mathcal{S}} i_s t_s(s')t_{s'}(s'')t_{s''}(s''')e_{s'''}(x)o_{s'''} \quad (12)$$

Non-contiguous substructure features with "gaps" require summing over paths between any pair of states. This is straightforward (we omit it for space), but of course using such features (while interesting) would complicate inference in decoding.

[4] It may be helpful to think of i as forward probabilities, but for the observation set $\mathcal{Y}^*$ rather than a *particular* observation $\mathbf{y}$. o are like backward probabilities. Note that, because some counted prefixes are prefixes of others, i can be > 1; similarly for o.

4.1.2 Expectations under a PCFG

In general, the expectations for a PCFG require solving a quadratic system of equations. The analogy this time is to inside and outside probabilities. Let the PCFG have nonterminal set $\mathcal{N}$, start symbol $S \in \mathcal{N}$, terminal alphabet Σ, and rules of the form $A \rightarrow B\ C$ and $A \rightarrow x$. (We assume Chomsky normal form for clarity; the generalization is straightforward.) Let $r_A(B\ C)$ and $r_A(x)$ denote the probabilities of nonterminal A rewriting to child sequence $B\ C$ or x, respectively. Then $\forall A \in \mathcal{N}$:

$$\begin{aligned}
o_A &= \sum_{B \in \mathcal{N}} \sum_{C \in \mathcal{N}} o_B i_C [r_B(A\ C) + r_B(C\ A)] \\
&\quad + \begin{cases} 1 & \text{if } A = S \\ 0 & \text{otherwise} \end{cases} \\
i_A &= \sum_{B \in \mathcal{N}} \sum_{C \in \mathcal{N}} r_A(B\ C) i_B i_C + \sum_x r_A(x) i_x \\
o_x &= \sum_{A \in \mathcal{N}} o_A r_A(x), \forall x \in \Sigma \\
i_x &= 1, \forall x \in \Sigma
\end{aligned}$$

In most practical applications, the PCFG will be "tight" (Booth and Thompson, 1973; Chi and Geman, 1998). Informally, this means that the probability of a derivation rooted in S failing to terminate is zero. If that is the case, then $i_A = 1$ for all $A \in \mathcal{N}$, and the system becomes linear (see also Corazza and Satta, 2006).[5] If tightness is not guaranteed, iterative propagation of weights, following Stolcke (1995), works well in our experience for solving the quadratic system, and converges quickly.

As in the HMM case, expected counts of arbitrary contiguous tree substructures can be computed as products of probabilities of rules appearing within the structure, factoring in the o value of the structure's root and the i values of the structure's leaves.

4.2 Optimization

To carry out M-estimation, we minimize the function $\ell(\mathbf{w})$ in Eq. 3. To apply gradient descent or a quasi-Newton numerical optimization method,[6] it suffices to specify the fixed quantities $\mathbf{f}(x_i, y_i)$ (for all $i \in \{1, 2, ..., n\}$) and the vector $\mathbf{E}_{q_0(X,Y)}[\mathbf{f}(X, Y)]$. The gradient is:[7]

$$\frac{\partial \ell}{\partial w_j} = -\sum_{i=1}^{n} e^{-\mathbf{w}^\top \mathbf{f}(x_i, y_i)} f_j(x_i, y_i) + \mathbf{E}_{q_0}[f_j] \tag{13}$$

The Hessian (matrix of second derivatives) can also be computed with relative ease, though the space requirement could become prohibitive. For problems where m is relatively small, this would allow the use of second-order optimization methods that are likely to converge in fewer iterations.

It is easy to see that Eq. 3 is convex in $\mathbf{w}$. Therefore, convergence to a global optimum is guaranteed and does not depend on the initializing value of $\mathbf{w}$.

4.3 Regularization

Regularization is a technique from pattern recognition that aims to keep parameters (like $\mathbf{w}$) from overfitting the training data. It is crucial to the performance of most statistical learning algorithms, and our experiments show it has a major effect on the success of the M-estimator. Here we use a quadratic regularizer, minimizing $\ell(\mathbf{w}) + (\mathbf{w}^\top \mathbf{w})/2c$. Note that this is also convex and differentiable if $c > 0$. The value of c can be chosen using a tuning dataset. This regularizer aims to keep each coordinate of $\mathbf{w}$ close to zero.

In the M-estimator, regularization is particularly important when the expectation of some feature f_j, $\mathbf{E}_{q_0(X,Y)}[f_j(X, Y)]$ is equal to zero. This can happen either due to sampling error (f_j simply failed to appear with a positive value in the finite sample) or because q_0 assigns zero probability mass to any $x \in \mathcal{X}, y \in \mathcal{Y}$ where $f_j(x, y) \neq 0$. Without regularization, the weight w_j will tend toward $\pm\infty$, but the quadratic penalty term will prevent that undesirable tendency. Just as the addition of a quadratic regularizer to likelihood can be interpreted as a zero-mean Gaussian prior on $\mathbf{w}$ (Chen and Rosenfeld, 2000), it can be so-interpreted here. The regularized objective is analogous to maximum *a posteriori* estimation.

5 Shallow Parsing

We compared M-estimation to a hidden Markov model and other training methods on English noun

[5] The same is true for HMMs: if the probability of nontermination is zero, then for all $s \in \mathcal{S}$, $o_s = 1$.

[6] We use L-BFGS (Liu and Nocedal, 1989) as implemented in the R language's `optim` function.

[7] Taking the limit as $n \rightarrow \infty$ and setting equal to zero, we have the basis for a proof that $\ell(\mathbf{w})$ is statistically consistent.

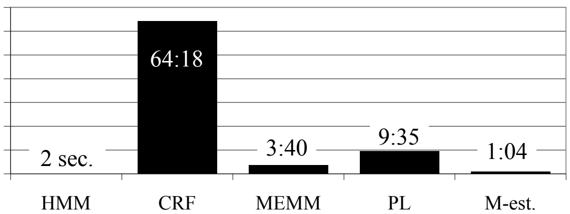

Figure 1: Wall time (hours:minutes) of training the HMM and 100 L-BFGS iterations for each of the extended-feature models on a 2.2 GHz Sun Opteron with 8GB RAM. See discussion in text for details.

phrase (NP) chunking. The dataset comes from the Conference on Natural Language Learning (CoNLL) 2000 shallow parsing shared task (Tjong Kim Sang and Buchholz, 2000); we apply the model to NP chunking only. About 900 sentences were reserved for tuning regularization parameters.

Baseline/q_0 In this experiment, the simple baseline is a second-order HMM. The states correspond to $\{\mathrm{B}, \mathrm{I}, \mathrm{O}\}$ labels, denoting the beginning, inside, and outside of noun phrases. Each state emits a tag and a word (independent of each other given the state). We replaced the first occurrence of every tag and of every word in the training data with an OOV symbol, giving a fixed tag vocabulary of 46 and a fixed word vocabulary of 9,014. Transition distributions were estimated using MLE, and tag- and word-emission distributions were estimated using add-1 smoothing. The HMM had 27,213 parameters. This HMM achieves 86.3% F_1-measure on the development dataset (slightly better than the lowest-scoring of the CoNLL-2000 systems). Heavier or weaker smoothing (an order of magnitude difference in add-λ) of the emission distributions had very little effect. Note that HMM training time is negligible (roughly 2 seconds); it requires counting events, smoothing the counts, and normalizing.

Extended Feature Set Sha and Pereira (2003) applied a conditional random field to the NP chunking task, achieving excellent results. To improve the performance of the HMM and test different estimation methods, we use Sha and Pereira's feature templates, which include subsequences of labels, tags, and words of different lengths and offsets. Here, we use only features observed to occur at least once in the training data, accounting (in addition to our OOV treatment) for the slight drop in performance compared to what Sha and Pereira report. There are 630,862 such features.

	prec.	recall	F_1
HMM features:			
HMM	85.60	88.68	87.11
CRF	90.40	89.56	89.98
PL	80.31	81.37	80.84
MEMM	86.03	88.62	87.31
M-est.	85.57	88.65	87.08
extended features:			
CRF	94.04	93.68	93.86
PL	91.88	91.79	91.83
MEMM	90.89	92.15	91.51
M-est.	88.88	90.42	89.64

Table 1: NP chunking accuracy on test data using different training methods. The effects of discriminative training (CRF) and extended feature sets (lower section) are more than additive.

Using the original HMM feature set and the extended feature set, we trained four models that can use arbitrary features: **conditional random fields** (a near-replication of Sha and Pereira, 2003), **maximum entropy Markov models** (MEMMs; McCallum et al., 2000), **pseudolikelihood** (Besag, 1975; see Toutanova et al., 2003, for a tagging application), and our M-estimator with the HMM as q_0. CRFs and MEMMs are discriminatively-trained to maximize conditional likelihood (the former is parameterized using a sequence-normalized log-linear model, the latter using a locally-normalized log-linear model). Pseudolikelihood is a consistent estimator for the *joint* likelihood, like our M-estimator; its objective function is a sum of log probabilities.

In each case, we trained seven models for each feature set with quadratic regularizers $c \in [10^{-1}, 10]$, spaced at equal intervals in the log-scale, plus an unregularized model ($c = \infty$). As discussed in §4.2, we trained using L-BFGS; training continued until relative improvement fell within machine precision or 100 iterations, whichever came first. After training, the value of c is chosen that maximizes F_1 accuracy on the tuning set.

Runtime Fig. 1 compares the wall time of carefully-timed training runs on a dedicated server. Note that Dyna, a high-level programming language, was used for dynamic programming (in the CRF) and summations (MEMM and pseudolikelihood). The runtime overhead incurred by using Dyna is estimated as a slow-down factor of 3–5 against a hand-tuned implementation (Eisner et al., 2005), though the slow-down factor is almost certainly less for the MEMM and pseudolikelihood. All training (except the HMM, of course) was done using the R language implementation of L-BFGS. In our implementation, the M-estimator trained substantially faster than the other methods. Of the 64 minutes required to train the M-estimator, 6 minutes were spent precomputing $\mathbf{E}_{q_0(X,Y)}[\mathbf{f}(X,Y)]$ (this need not be repeated if the regularization settings are altered).

Accuracy Tab. 1 shows how NP chunking accuracy compares among the models. With HMM features, the M-estimator is about the same as the HMM and MEMM (better than PL and worse than the CRF). With extended features, the M-estimator lags behind the slower methods, but performs about the same as the HMM-featured CRF (2.5–3 points over the HMM). The full-featured CRF improves performance by another 4 points. Performance as a function of training set size is plotted in Fig. 2; the different methods behave relatively similarly as the training data are reduced. Fig. 3 plots accuracy (on tuning data) against training time, for a variety of training dataset sizes and regularizaton settings, under different training methods. This illustrates the training-time/accuracy tradeoff: the M-estimator, when well-regularized, is considerably faster than the other methods, at the expense of accuracy. This experiment gives some insight into the relative importance of extended **features** versus **estimation methods**. The M-estimated model is, like the maximum likelihood-estimated HMM, a generative model. Unlike the HMM, it uses a much larger set of features–the same features that the discriminative models use. Our result supports the claim that good features are necessary for state-of-the-art performance, but so is good training.

5.1 Effect of the Base Distribution

We now turn to the question of the base distribution q_0: how accurate does it need to be? Given that the M-estimator is consistent, it should be clear that, in the limit and assuming that our model family p is correct, q_0 should not matter (except in its support).

q_0	selection	prec.	recall	F_1
HMM	F_1, prec.	88.88	90.42	89.64
l.u.	F_1	72.91	57.56	64.33
	prec.	84.40	37.68	52.10
emp.	F_1	84.38	89.43	86.83

Table 2: NP chunking accuracy on test data using different base models for the M-estimator. The "selection" column shows which accuracy measure was optimized when selecting the hyperparameter c.

In NLP, we deal with finite datasets and imperfect models, so q_0 may have practical importance.

We next consider an alternative q_0 that is far less powerful; in fact, it is uninformative about the variable to be predicted. Let $\mathbf{x}$ be a sequence of words, $\mathbf{t}$ be a sequence of part-of-speech tags, and $\mathbf{y}$ be a sequence of $\{\mathrm{B}, \mathrm{I}, \mathrm{O}\}$-labels. The model is:

$$q_0^{\text{l.u.}}(\mathbf{x}, \mathbf{t}, \mathbf{y}) \stackrel{\text{def}}{=} \left(\prod_{i=1}^{|\mathbf{x}|} p_{\text{uni}}(x_i) p_{\text{uni}}(t_i) \frac{1}{N_{y_{i-1}}} \right) \frac{1}{N_{y_{|\mathbf{x}|}}} \tag{14}$$

where N_y is the number of labels (including stop) that can follow y (3 for O and $y_0 = \text{start}$, 4 for B and I). p_{uni} are the tag and word unigram distributions, estimated using MLE with add-1 smoothing. This model ignores temporal effects. On its own, this model achieves 0% precision and recall, because it labels every word O (the most likely label sequence is $\mathrm{O}^{|\mathbf{x}|}$). We call this model **l.u.** ("locally uniform").

Tab. 2 shows that, while an M-estimate that uses $q_0^{\text{l.u.}}$ is not nearly as accurate as the one based on an HMM, the M-estimator *did* manage to improve considerably over $q_0^{\text{l.u.}}$. So the M-estimator is far better than nothing, and in this case, tuning c to maximize precision (rather than F_1) led to an M-estimated model with precision competitive with the HMM. We point this out because, in applications involving very large corpora, a model with good precision may be useful even if its coverage is mediocre.

Another question about q_0 is whether it should take into account all possible values of the input variables (here, $\mathbf{x}$ and $\mathbf{t}$), or only those seen in training. Consider the following model:

$$q_0^{\text{emp}}(\mathbf{x}, \mathbf{t}, \mathbf{y}) \stackrel{\text{def}}{=} q_0(\mathbf{y} \mid \mathbf{x}, \mathbf{t}) \tilde{p}(\mathbf{x}, \mathbf{t}) \tag{15}$$

Here we use the empirical distribution over tag/word

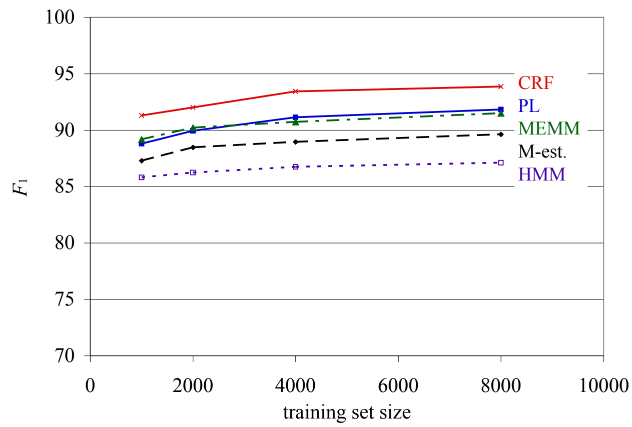

Figure 2: Learning curves for different estimators; all of these estimators except the HMM use the extended feature set.

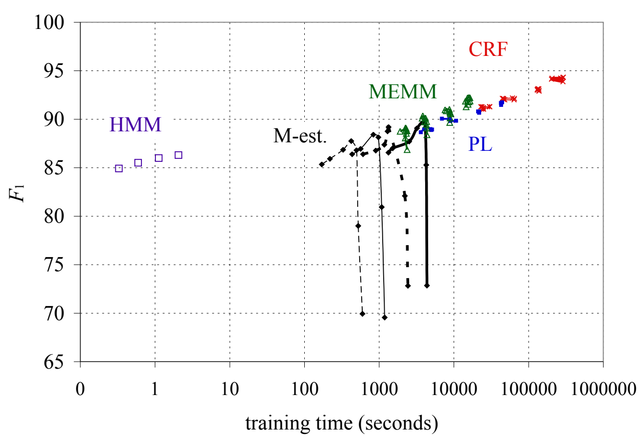

Figure 3: Accuracy (tuning data) vs. training time. The M-estimator trains notably faster. The points in a given curve correspond to different regularization strengths (c); M-estimation is more damaged by weak than strong regularization.

sequences, and the HMM to define the distribution over label sequences. The expectations $\mathbf{E}_{q_0^{\mathrm{emp}}(X)}[\mathbf{f}(X)]$ can be computed using dynamic programming over the training data (recall that this only needs to be done once, cf. the CRF). Strictly speaking, q_0^{emp} assigns probability zero to any sequence not seen in training, but we can ignore the $\tilde{p}$ marginal at decoding time. As shown in Tab. 2, this model slightly improves recall over the HMM, but damages precision; the gains of M-estimation seen with the HMM as q_0, are not reproduced. From these experiments, we conclude that the M-estimator might perform considerably better, given a better q_0.

5.2 Input-Only Features

We present briefly one negative result. Noting that the M-estimator is a modeling technique that estimates a distribution over both input and output variables (i.e., a generative model), we wanted a way to make the objective more discriminative while still maintaining the computational property that inference (of any kind) not be required during the inner loop of iterative training.

The idea is to reduce the predictive burden on the feature weights for $\mathbf{f}$. When designing a CRF, features that do not depend on the output variable (here, $\mathbf{y}$) are unnecessary. They cannot distinguish between competing labelings for an input, and so their weights will be set to zero during conditional estimation. The feature vector function in Sha and Pereira's chunking model does not include such features. In M-estimation, however, adding such "input-only" features might permit better modeling of the data and, more importantly, use the original features primarily for the discriminative task of modeling $\mathbf{y}$ *given* the input.

Adding unigram, bigram, and trigram features to $\mathbf{f}$ for M-estimation resulted in a very small decrease in performance: selecting for F_1, this model achieves 89.33 F_1 on test data.

6 Discussion

M-estimation fills a gap in the plethora of training techniques that are available for NLP models today: it permits **arbitrary features** (like so-called conditional "maximum entropy" models such as CRFs) but estimates a **generative** model (permitting, among other things, classification on input variables and meaningful combination with other models). It is similar in spirit to **pseudolikelihood** (Besag, 1975), to which it compares favorably on training runtime and unfavorably on accuracy.

Further, since no inference is required during training, *any* features really are permitted, so long as their expected values can be estimated under the base model q_0. Indeed, M-estimation is considerably easier to implement than conditional estimation. Both require feature counts from the training data; M-estimation replaces repeated calculation and differentiation of normalizing constants with inference or sampling (once) under a base model. So the M-estimator is much faster to train.

Generative and discriminative models have been compared and discussed a great deal (Ng and Jordan, 2002), including for NLP models (Johnson, 2001; Klein and Manning, 2002). Sutton and McCallum (2005) present approximate methods that keep a discriminative objective while avoiding full inference.

We see M-estimation as a particularly promising method in settings where performance depends on high-dimensional, highly-correlated feature spaces, where the desired features "large," making discriminative training too time-consuming—a compelling example is machine translation. Further, in some settings a locally-normalized conditional log-linear model (like an MEMM) may be difficult to design; our estimator avoids normalization altogether.[8] The M-estimator may also be useful as a tool in designing and selecting feature combinations, since more trials can be run in less time. After selecting a feature set under M-estimation, discriminative training can be applied on that set. The M-estimator might also serve as an *initializer* to discriminative models, perhaps reducing the number of times inference must be performed—this could be particularly useful in very-large data scenarios. In future work we hope to explore the use of the M-estimator within hidden variable learning, such as the Expectation-Maximization algorithm (Dempster et al., 1977).

7 Conclusions

We have presented a new loss function for generatively estimating the parameters of log-linear models. The M-estimator is fast to train, requiring no repeated, expensive calculation of normalization terms. It was shown to improve performance on a shallow parsing task over a baseline (generative) HMM, but it is not competitive with the state-of-the-art. Our sequence modeling experiments support the widely accepted claim that discriminative, rich-feature modeling works as well as it does *not just* because of rich features in the model, but also because of discriminative training. Our technique fills an important gap in the spectrum of learning methods for NLP models and shows promise for application when discriminative methods are too expensive.

[8]Note that MEMMs also require local partition functions—which may be expensive—to be computed at decoding time.

References

J. E. Besag. 1975. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195.

T. L. Booth and R. A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, 22(5):442–450.

S. Chen and R. Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Z. Chi and S. Geman. 1998. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305.

A. Corazza and G. Satta. 2006. Cross-entropy and estimation of probabilistic context-free grammars. In *Proc. of HLT-NAACL*.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

J. Eisner, E. Goldlust, and N. A. Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proc. of HLT-EMNLP*.

Y. Jeon and Y. Lin. 2006. An effective method for high-dimensional log-density ANOVA estimation, with application to nonparametric graphical model building. *Statistical Sinica*, 16:353–374.

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Proc. of ACL*.

D. Klein and C. D. Manning. 2002. Conditional structure vs. conditional estimation in NLP models. In *Proc. of EMNLP*.

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

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45:503–528.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. of ICML*.

A. Ng and M. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naïve Bayes. In *NIPS 14*.

J. A. O'Sullivan. 1998. Alternating minimization algorithms: from Blahut-Armijo to Expectation-Maximization. In A. Vardy, editor, *Codes, Curves, and Signals: Common Threads in Communications*, pages 173–192. Kluwer.

R. Rosenfeld. 1997. A whole sentence maximum entropy language model. In *Proc. of ASRU*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

C. Sutton and A. McCallum. 2005. Piecewise training of undirected models. In *Proc. of UAI*.

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. of CoNLL*.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of HLT-NAACL*.

A. W. van der Vaart. 1998. *Asymptotic Statistics*. Cambridge University Press.