A comparison of algorithms for maximum entropy parameter estimation

Robert Malouf
Alfa-Informatica
Rijksuniversiteit Groningen
Postbus 716
9700AS Groningen
The Netherlands
malouf@let.rug.nl

Draft of May 15, 2002

Abstract

Conditional maximum entropy (ME) models provide a general purpose machine learning technique which has been successfully applied to fields as various as computer vision and econometrics, and which is used for a wide variety of classification problems in natural language processing. However, the flexibility of ME models is not without cost. While parameter estimation for ME models is conceptually straightforward, in practice ME models for typical natural language tasks are very large, and may well contain many thousands of free parameters. In this paper, we consider a number of algorithms for estimating the parameters of ME models, including iterative scaling, gradient ascent, conjugate gradient, and variable metric methods. Surprisingly, the standardly used iterative scaling algorithms perform quite poorly in comparison to the others, and for all of the test problems, a limited-memory variable metric algorithm outperformed the other choices.

1 Introduction

Maximum entropy (ME) models, variously known as log-linear, Gibbs, exponential, and multinomial logit models, provide a general purpose machine learning technique for classification and prediction which has been successfully applied to fields as various as computer vision and econometrics. In natural language processing, recent years have seen ME techniques used for sentence boundary detection, part of speech tagging, parse selection and ambiguity resolution, and stochastic attribute-value grammars, to name just a few applications (Abney, 1997; Berger et al., 1996; Ratnaparkhi, 1998; Johnson et al., 1999).

A leading advantage of ME models is their flexibility: they allow stochastic rule systems to be augmented with additional syntactic, semantic, and pragmatic features. However, the richness of the representations is not without cost. Even modest maximum entropy models can require considerable computational resources and very large quantities of annotated training data in order to accurately estimate the model's parameters. While parameter estimation for ME models is conceptually straightforward, in practice ME models for typical natural language tasks are usually quite large, and frequently contain hundreds of thousands of free parameters. Estimation of such large models is not only expensive, but also, due to sparsely distributed features, sensitive to round-off errors. Thus, highly efficient, accurate, scalable methods are required for estimating the parameters of practical models.

In this paper, we consider a number of algorithms for estimating the parameters of ME models, including *Generalized Iterative Scaling* and *Improved Iterative Scaling*, as well as general purposed optimization techniques such as *gradient ascent, conjugate gradient,* and *variable metric* methods. Surprisingly, the widely used iterative scaling algorithms perform quite poorly, and for all of the test problems, a limited memory variable metric algorithm outperformed the other choices.

2 Background

Suppose we have a probability distribution p over a set of events X which are characterized by a d dimensional feature vector function $f : X \rightarrow \mathbb{R}^d$. In the context of a stochastic context-free grammar (SCFG), for example, X might be the set of possible trees, and the feature vectors might represent the number of times each rule applied in the derivation of each tree. Our goal is to construct a model distribution q which satisfies the constraints imposed by the empirical distribution p, in the sense that:

$$\mathrm{E}_p[f] = \mathrm{E}_q[f] \tag{1}$$

where $E_p[f]$ is the expected value of the feature vector under the distribution p:

$$\mathrm{E}_p[f] = \sum_{x \in X} p(x) f(x)$$

In general, this problem is ill posed: a wide range of models will fit the constraints in (1). As a guide to selecting one that is most appropriate, we can call on Jaynes' (1957) *Principle of Maximum Entropy*: "In the absence of additional information, we should assume that all events have equal probability." In other words, we should assign the highest prior probability to distributions which maximize the entropy:

$$H(q) = -\sum_{x \in X} q(x) \log q(x) \tag{2}$$

This is effectively a problem in constrained optimization: we want to find a distribution q which maximizes (2) while satisfying the constraints imposed by (1). It can be straightforwardly shown (Jaynes, 1957; Good, 1963; Campbell, 1970) that the solution to this problem has the parametric form:

$$q_\theta(x) = \frac{\exp\left(\theta^T f(x)\right)}{\sum_{y \in X} \exp\left(\theta^T f(y)\right)} \tag{3}$$

where θ is a d-dimensional parameter vector and $\theta^T f(x)$ is the inner product of the parameter vector and a feature vector.

One complication which makes models of this form difficult to apply to problems in natural language processing is that the events space X is often very large or even infinite, making the denominator in (3) impossible to compute. One modification we can make to avoids this problem is to consider conditional probability distributions instead (Berger et al., 1996; Chi, 1998; Johnson et al., 1999). Suppose now that in addition to the event space X and the feature function f, we have also a set of contexts W and a function Y which partitions the members of X. In our SCFG example, W might be the set of possible strings of words, and $Y(w)$ the set of trees whose yield is $w \in W$. Computing the conditional probability $q_\theta(x|w)$ of an event x in context w as

$$q_\theta(x|w) = \frac{\exp\left(\theta^T f(x)\right)}{\sum_{y \in Y(w)} \exp\left(\theta^T f(y)\right)} \tag{4}$$

now involves evaluating a more much tractable sum in the denominator.

ESTIMATE(p)
1 $\theta^0 \leftarrow 0$
2 $k \leftarrow 0$
3 **repeat**
4 compute $q^{(k)}$ from $\theta^{(k)}$
5 compute update $\delta^{(k)}$
6 $\theta^{(k+1)} \leftarrow \theta^{(k)} + \delta^{(k)}$
7 $k \leftarrow k+1$
8 **until** converged
9 **return** $\theta^{(k)}$

Figure 1: General parameter estimation algorithm

3 Maximum likelihood estimation

Given the parametric form of an ME model in (4), fitting an ME model to a collection of training data entails finding values for the parameter vector θ which minimize the relative entropy between the model q_θ and the empirical distribution p, or, equivalently, which maximize the log likelihood:

$$L(\theta) = \sum_{x,y} p(x,y) \log q(y|x,\theta) \tag{5}$$

The gradient of the log likelihood function, or the vector of its first derivatives with respect to the parameter θ is:

$$G(\theta) = \sum_{x,y} p(x,y) f(y) - \sum_{x,y} p(x) q(y|x,\theta) f(y)$$

or, simply:

$$G(\theta) = \mathrm{E}_p[f] - \mathrm{E}_q[f] \tag{6}$$

Since the likelihood function (5) is concave over the parameter space, it has a global maximum where the gradient is zero. Unfortunately, simply setting $G(\theta) = 0$ and solving for θ does not yield a closed form solution, so we proceed iteratively, following the general algorithm in Figure 1. At each step, we adjust an estimate of the parameters $\theta^{(k)}$ to a new estimate $\theta^{(k+1)}$ based on the divergence between the estimated probability distribution $q^{(k)}$ and the empirical distribution p. We continue until successive improvements fail to yield a sufficiently large decrease in the divergence.

While all parameter estimation algorithms we will consider take the same general form, the method for computing the updates $\delta^{(k)}$ at search step differs substantially. As we shall see, this difference can have a dramatic impact on the number of updates required to reach convergence.

3.1 Iterative Scaling

One popular method for iteratively refining the model parameters is *Generalized Iterative Scaling* (GIS), due to Darroch and Ratcliff (1972). An extension of Iterative Proportional Fitting (Deming and Stephan, 1940), GIS scales the probability distribution $q^{(k)}$ by a factor proportional to the ratio of $\mathrm{E}_p[f]$ to $\mathrm{E}_{q^{(k)}}[f]$, with the restriction that $\sum_j f_j(x) = C$ for each event x in the training data (a condition which can be easily satisfied by the addition of a correction feature). We can adapt GIS to estimate the model parameters θ rather than the model probabilities q, yielding the update rule:

$$\delta^{(k)} = \log\left(\frac{\mathrm{E}_p[f]}{\mathrm{E}_{q^{(k)}}[f]}\right)^{\frac{1}{C}}$$

The step size, and thus the rate of convergence, depends on the constant C: the larger the value of C, the smaller the step size. In case not all rows of the training data sum to a constant, the addition of a correction feature effectively slows convergence to match the most difficult case. To avoid this slowed convergence and the need for a correction feature, Della Pietra et al. (1997) propose an *Improved Iterative Scaling* (IIS) algorithm, whose update rule is the solution to the equation:

$$\mathrm{E}_p[f] = \sum_{x,y} p(x) q^{(k)}(y|x) f(y) \exp(M(y)\delta^{(k)})$$

where $M(y)$ is the sum of the feature values for an event y in the training data. This is a polynomial in $\exp\left(\delta^{(k)}\right)$, and the solution can be found straightforwardly using, for example, the Newton-Raphson method.

3.2 First order methods

Iterative scaling algorithms have a long tradition in statistics and are still widely used for analysis of contingency tables. Their primary strength is that on each iteration they only require computation of the expected values $\mathrm{E}_{q^{(k)}}$. They do not depend on evaluation of the gradient of the log-likelihood function, which, depending on the distribution, could be prohibitively expensive. In the case of maximum entropy models, however, the vector of expected values required by iterative scaling essentially *is* the gradient G. Thus, it makes sense to consider methods which use the gradient directly.

The most obvious way of making explicit use of the gradient is by *Cauchy's method*, or the method

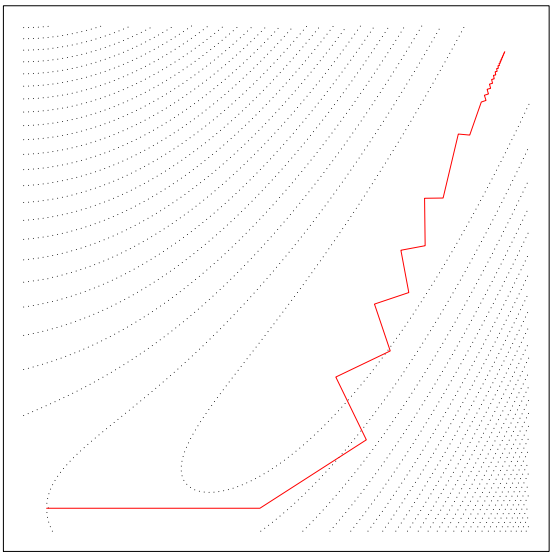

Figure 2: Steepest ascent in two dimensions

of *steepest ascent*. The gradient of a function is a vector which points in the direction in which the function's value increases most rapidly. Since our goal is to maximize the log-likelihood function, a natural strategy is to shift our current estimate of the parameters in the direction of the gradient via the update rule:

$$\delta^{(k)} = \alpha^{(k)} G(\theta^{(k)})$$

where the step size $\alpha^{(k)}$ is chosen to maximize $L(\theta^{(k)} + \delta^{(k)})$. Finding the optimal step size is itself an optimization problem, though only in one dimension and, in practice, only an approximate solution is required to guarantee global convergence.

Since the log-likelihood function is concave, the method of steepest ascent is guaranteed to find the global maximum. However, while the steps taken on each iteration are in a very narrow sense locally optimal, the global convergence rate of steepest ascent is very poor. As shown in Figure 2, each new search direction is orthogonal (or, if an approximate line search is used, nearly so) to the previous direction. This leads to a characteristic "zig-zag" ascent, with convergence slowing as the maximum is approached.

One way of looking at the problem with steepest ascent is that it considers the same search directions many times. We would prefer an algorithm which considered each possible search direction only once, in each iteration taking a step of ex-

actly the right length in a direction orthogonal to all previous search directions. This intuition underlies *conjugate gradient* methods (see, e.g., Shewchuk, 1994), which choose a search direction which is a linear combination of the steepest ascent direction and the previous search direction. For example, the *Fletcher-Reeves* algorithm uses the update rule:

$$\begin{aligned} \beta^{(k)} &= \frac{G(\theta^{(k)})^T G(\theta^{(k)})}{G(\theta^{(k-1)})^T G(\theta^{(k-1)})} \\ p^{(k)} &= G(\theta^{(k)}) + \beta^{(k)} p^{(k-1)} \\ \delta^{(k)} &= \alpha^{(k)} p^{(k)} \end{aligned}$$

where the step size $\alpha^{(k)}$ is selected by an approximate line search, as in the steepest ascent method. The scalar $\beta^{(k)}$ guarantees that the search direction $p^{(k)}$ is conjugate (i.e., orthogonal, in a particular sense) to the previous search direction. Other nonlinear conjugate gradient algorithms such as *Polak-Ribière* differ in the way $\beta^{(k)}$ is computed and thus show different numeric properties.

3.3 Second order methods

Another way of looking at the problem with steepest ascent is that while it takes into account the gradient of the log-likelihood function, it fails to take into account its curvature, or the gradient of the gradient. The usefulness of the curvature is made clear if we consider a second-order Taylor series approximation of $L(\theta+\delta)$:

$$L(\theta+\delta) \approx L(\theta) + \delta^T G(\theta) + \frac{1}{2}\delta^T H(\theta)\delta \qquad (7)$$

where H is *Hessian matrix* of the log-likelihood function, the $d \times d$ matrix of its second partial derivatives with respect to θ. If we set the derivative of (7) to zero and solve for δ, we get the update rule for *Newton's method*:

$$\delta^{(k)} = H^{-1}(\theta^{(k)}) G(\theta^{(k)})$$

Newton's method converges very quickly (for quadratic objective functions, in one step), but it requires the computation of the inverse of the Hessian matrix on each iteration.

While the log-likelihood function for ME models in (5) is twice differentiable, for large scale problems the evaluation of the Hessian matrix is computationally impractical, and Newton's method is not competitive with iterative scaling or first order methods. *Variable metric* or *quasi-Newton* methods avoid explicit evaluation of the Hessian by building up an approximation of it using successive evaluations of the gradient. Variable metric methods also show excellent convergence properties and can be much more efficient than using true Newton updates, but for large scale problems with hundreds of thousands of parameters, even storing the approximate Hessian is prohibitively expensive. For such cases, we can apply *limited memory variable metric* methods, which implicitly approximate the Hessian matrix in the vicinity of the current estimate of $\theta^{(k)}$ using the previous m search directions. Since in practical applications values of m between 3 and 10 suffice, this can offer a substantial savings in storage requirements over variable metric methods, while still giving reasonable convergence (see Figure 3).[1]

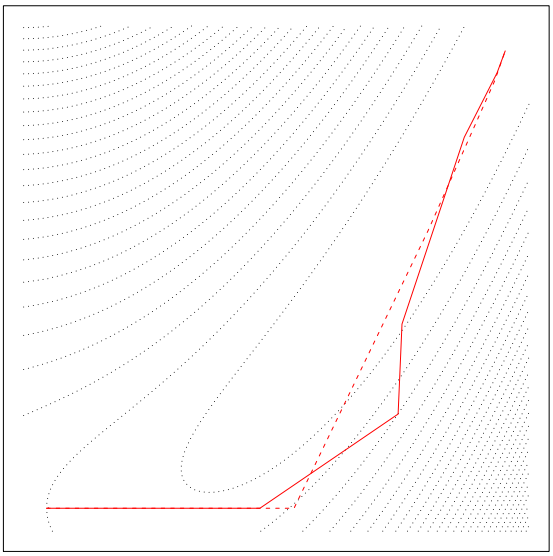

Figure 3: Limited memory variable metric method (dashed lines show Newton's method for comparison)

4 Comparing estimation techniques

The performance of optimization algorithms is highly dependent on the specific properties of the problem to be solved. Worst-case analysis typically does not reflect the actual behavior on actual problems. Therefore, in order to evaluate the performance of the optimization techniques sketched in

[1] For a detailed analysis and comparison of first and second order methods, see, e.g., Nocedal (1997) or Nocedal and Wright (1999).

previous section when applied to the problem of parameter estimation, we need to compare the performance of actual implementations on realistic data sets (Dolan and Moré, 2000; Benson et al., 2000; Dolan and Moré, 2002).

Minka (2001) is one earlier attempt to compare iterative scaling with other algorithms for parameter estimation in logistic regression, a problem similar to the one considered here. However, it is difficult to draw any conclusions from Minka's results for three reasons. First, he evaluates the algorithms with randomly generated training data. The performance and accuracy of optimization algorithms can be sensitive to the specific numerical properties of the function being optimized; results based on random data may or may not carry over to more realistic problems. Second, Minka measures performance in terms of the number of floating point operations required to achieve a particular precision. But, large-scale sparse problems are typically memory bandwidth-bound, not CPU bound. Therefore, the number of floating point operations is not very good indicator of the total time required to find a solution. And, finally, the test problems Minka considers are relatively small (100–500 dimensions). As we have seen, though, algorithms which perform well for small and medium scale problems may not always be applicable to problems with many thousands of dimensions.

4.1 Implementation

As a basis for the implementation, we have used PETSc (the "Portable, Extensible Toolkit for Scientific Computation"), a software library designed to ease development of programs which solve large systems of partial differential equations (Balay et al., 2001; Balay et al., 1997; Balay et al., 2002). PETSc offers data structures and routines for parallel and sequential storage, manipulation, and visualization of very large sparse matrices.

For any of the estimation techniques, the most expensive operation is computing the probability distribution q and the expectations $\mathrm{E}_q[f]$ for each iteration. In order to make use of the facilities provided by PETSc, we can store the training data as a (sparse) matrix F, with rows corresponding to events and columns to features. Then given a parameter vector θ, the unnormalized log probabilities $\dot{q}$ are the matrix-vector product:

$$\dot{q} = F\theta$$

and the feature expectations are the transposed matrix-vector product:

$$\mathrm{E}_q[f] = F^T q$$

By expressing these computations as matrix-vector products, we can take advantage of the high performance sparse matrix primitives of PETSc.

For the comparison, we implemented both Generalized and Improved Iterative Scaling in C++ using the primitives provided by PETSc. For the other optimization techniques, we used TAO (the "Toolkit for Advanced Optimization"), a library layered on top of the foundation of PETSc for solving nonlinear optimization problems (Benson et al., 2002). TAO offers the building blocks for writing optimization programs (such as line searches and convergence tests) as well as high-quality implementations of standard optimization algorithms (including conjugate gradient and variable metric methods).

Before turning to the results of the comparison, two additional points need to be made. First, in order to assure a consistent comparison, we need to use the same stopping rule for each algorithm. For these experiments, we judged that convergence was reached when the relative change in the log-likelihood between iterations fell below a predetermined threshold. That is, each run was stopped when:

$$\frac{|L(\theta^{(k)}) - L(\theta^{(k-1)})|}{L(\theta^{(k)})} < \varepsilon \qquad (8)$$

where the relative tolerance $\varepsilon = 10^{-7}$. For any particular application, this may or may not be an appropriate stopping rule, but is only used here for purposes of comparison.

Finally, it should be noted that in the current implementation, we have not applied any of the possible optimizations that appear in the literature (Lafferty and Suhm, 1996; Wu and Khudanpur, 2000; Lafferty et al., 2001) to speed up normalization of the probability distribution q. These improvements take advantage of a model's structure to simplify the evaluation of the denominator in (4). The particular test data sets examined here are unstructured, and such optimizations are unlikely to give any improvement. However, when these optimizations are appropriate, they will give a proportional speed-up to all of the algorithms. Thus, the use of such optimizations is independent of the choice of parameter estimation method.

dataset	classes	contexts	features	nz
rules	32,546	2,808	246	803,985
lex	46,769	2,808	135,182	4,324,576
summary	26,554	13,277	198,467	438,050
shallow	9,583,341	416,667	264,142	51,736,113

Table 1: Datasets used in experiments ('nz' is the number of non-zero feature values in the sparse training matrix F)

4.2 Experiments

To compare the algorithms described in §3, we applied the implementation outlined in the previous section to four training data sets (described in Table 1) drawn from the domain of natural language processing. The 'rules' and 'lex' datasets are examples of stochastic attribute value grammars, one with a small set of SCFG-like features, and with with a very large set of fine-grained lexical features (Bouma et al., 2001). The 'summary' dataset is part of a sentence extraction task (Osborne, to appear), and the 'shallow' dataset is drawn from a text chunking application (Osborne, 2002). These datasets vary widely in their size and composition, and are representative of the kinds of datasets typically encountered in applying ME models to NLP classification tasks.

The results of applying each of the parameter estimation algorithms to each of the datasets is summarized in Table 2. For each run, we report the relative entropy between the fitted model and the training data at convergence, the number of iterations required, the number of log-likelihood and gradient evaluations required (algorithms which use a line search may require several function evaluations per iteration), and the total elapsed time.[2]

There are a few things to observe about these results. First, while IIS converges in fewer steps the GIS, it takes substantially more time. At least for this implementation, the additional bookkeeping overhead required by IIS more than cancels any improvements in speed offered by accelerated convergence. This may be a misleading conclusion, however, since a more finely tuned implementation of IIS may well take much less time per iteration than the one used for these experiments. However, even if each iteration of IIS could be made as fast as an iteration of GIS (which seems unlikely), the benefits of IIS over GIS would in these cases be quite modest.

Second, note that for three of the four datasets, the relative entropy at convergence is roughly the same for all of the algorithms. For the 'summary' dataset, however, they differ by up to two orders of magnitude. This is an indication that the convergence test in (8) is sensitive to the rate of convergence and thus to the choice of algorithm. Any degree of precision desired could be reached by any of the algorithms, with the appropriate value of ε. However, GIS, say, would require many more iterations than reported in Table 2 to reach the precision achieved by the limited memory variable metric algorithm.

Finally, the most significant lesson to be drawn from these results is that, with the exception of steepest ascent, gradient-based methods outperform iterative scaling by a wide margin for almost all the datasets, as measured by both number of function evaluations and by the total elapsed time. And, in each case, the limited memory variable metric algorithm performs substantially better than any of the competing methods.

5 Conclusions

In this paper, we have described experiments comparing the performance of a number of different algorithms for estimating the parameters of a conditional ME model. The results show that variants of iterative scaling, the algorithms which are most widely used in the literature, perform quite poorly when compared to general function optimization algorithms such as conjugate gradient and variable metric methods. And, more specifically, for the NLP classification tasks considered, the limited memory variable metric algorithm of Benson and Moré (2001) outperforms the other choices by a substantial margin.

This conclusion has obvious consequences for the field. ME modeling is a commonly used machine

[2] The reported time does not include the time required to input the training data, which is difficult to reproduce and which is the same for all the algorithms being tested.

Dataset	Method	Div.	Iter	Evals	Time (secs)
rules	gis	5.19×10^{-2}	1201	1202	23.04
	iis	5.14×10^{-2}	923	924	42.48
	steepest ascent	5.13×10^{-2}	212	331	6.16
	conjugate gradient (fr)	5.07×10^{-2}	74	196	3.74
	conjugate gradient (prp)	5.08×10^{-2}	63	154	2.87
	limited memory variable metric	5.07×10^{-2}	70	76	1.44
lex	gis	1.61×10^{-3}	370	371	36.29
	iis	1.52×10^{-3}	241	242	102.18
	steepest ascent	3.47×10^{-3}	1041	1641	139.10
	conjugate gradient (fr)	1.39×10^{-3}	166	453	39.03
	conjugate gradient (prp)	1.62×10^{-3}	150	382	32.46
	limited memory variable metric	1.49×10^{-3}	136	143	17.25
summary	gis	1.83×10^{-3}	1446	1447	125.46
	iis	1.07×10^{-3}	626	627	208.22
	steepest ascent	2.64×10^{-3}	1163	3503	227.30
	conjugate gradient (fr)	1.01×10^{-4}	175	948	60.91
	conjugate gradient (prp)	7.30×10^{-4}	93	428	27.81
	limited memory variable metric	3.98×10^{-5}	81	89	10.38
shallow	gis	3.57×10^{-2}	3428	3429	27103.62
	iis	3.50×10^{-2}	3216	3217	71053.24
	steepest ascent †	—	—	—	—
	conjugate gradient (fr)	2.91×10^{-2}	1094	6056	46958.87
	conjugate gradient (prp)	4.13×10^{-2}	421	2170	16477.84
	limited memory variable metric	3.26×10^{-2}	429	444	3408.30

Table 2: Results. All tests were run using one CPU of a dual processor 1700MHz Pentium 4 with 2 gigabytes of main memory. († did not reach convergence within a twenty-four hour time limit)

learning technique, and the application of improved parameter estimation algorithms will it practical to construct larger, more complex models. And, since the parameters of individual models can be estimated quite quickly, this will further open up the possibility for more sophisticated model and feature selection techniques which compare large numbers of alternative model specifications.

In addition, there is a larger lesson to be drawn from these results. We typically think of computational linguistics as being primarily a symbolic discipline. However, statistical natural language processing involves non-trivial numeric computations that require a distinct set of skills and methods. As these results show, natural language processing can take great advantage of the algorithms and software libraries developed by and for more quantitatively oriented engineering and computational sciences.

6 Acknowledgements

The research of Dr. Malouf has been made possible by a fellowship of the Royal Netherlands Academy of Arts and Sciences and by the NWO PIONIER project *Algorithms for Linguistic Processing.* Thanks also to Miles Osborne and Gertjan van Noord for helpful discussions and test data.

References

Steven P. Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23:597–618.

Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. 1997. Efficienct management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Soft-*

ware Tools in Scientific Computing, pages 163–202. Birkhauser Press.

Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Lois Curfman McInnes, and Barry F. Smith. 2001. PETSc home page. http://www.mcs.anl.gov/petsc.

Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. 2002. PETSc users manual. Technical Report ANL-95/11–Revision 2.1.2, Argonne National Laboratory.

Steven J. Benson and Jorge J. Moré. 2001. A limited memory variable metric method for bound constrained minimization. Preprint ANL/ACS-P909-0901, Argonne National Laboratory. http://www-unix.mcs.anl.gov/~benson/blmvm/blmvm.ps.

Steven Benson, Lois Curfman McInnes, and Jorge Moré. 2000. GPCG: A case study in the performance and scalability of optimization algorithms. Technical Report ANL/MCS-P768-0799, Argonne National Laboratory. http://www.mcs.anl.gov/home/more/papers/gpcg.ps.gz.

Steven J. Benson, Lois Curfman McInnes, Jorge J. Moré, and Jason Sarich. 2002. TAO users manual. Technical Report ANL/MCS-TM-242–Revision 1.4, Argonne National Laboratory.

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: wide coverage computational analysis of Dutch. In W. Daelemans, K. Sima'an, J. Veenstra, and J. Zavrel, editors, *Computational Linguistics in the Netherlands 2000*, pages 45–59. Rodolpi, Amsterdam.

L.L. Campbell. 1970. Equivalence of Gauss's principle and minimum discrimination information estimation of probabilities. *Annals of Mathematical Statistics*, 41:1011–1015.

Zhiyi Chi. 1998. *Probability models for complex systems*. Ph.D. thesis, Brown University.

J. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *Ann. Math. Statistics*, 43:1470–1480.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393.

W.E. Deming and F.F. Stephan. 1940. On a least squares adjustment of a sampled frequency table when the expected marginals are known. *Annals of Mathematical Statistics*, 11:427–444.

Elizabeth D. Dolan and Jorge Moré. 2000. Benchmarking optimization software with COPS. Technical Report ANL/MCS-246, Argonne National Laboratory. http://www-unix.mcs.anl.gov/~more/cops/bcops.ps.gz.

Elizabeth D. Dolan and Jorge J. Moré. 2002. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213.

I.J. Good. 1963. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *Annals of Mathematical Statistics*, 34:911–934.

E.T. Jaynes. 1957. Information theory and statistical mechanics. *Physical Review*, 106,108:620–630.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 535–541, College Park, Maryland.

John Lafferty and Bernhard Suhm. 1996. Cluster expansions and iterative scaling for maximum entropy language models. In K. Hanson and R. Silver, editors, *Maximum Entropy and Bayesian Methods*. Kluwer.

John Lafferty, Fernando Pereira, and Andrew McCallum. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*.

Thomas P. Minka. 2001. Algorithms for maximum-likelihood logistic regression. Statistics Tech Report 758, CMU. http://www.stat.cmu.edu/tr/tr758/tr758.html.

Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer, New York.

Jorge Nocedal. 1997. Large scale unconstrained optimization. In A. Watson and I. Duff, editors, *The State of the Art in Numerical Analysis*, pages 311–338. Oxford University Press.

Miles Osborne. 2002. Shallow parsing using noisy and non-stationary training material. *Journal of Machine Learning Research*, 2:695–719.

Miles Osborne. to appear. Using maximum entropy for sentence extraction. In *Proceedings of the ACL 2002 Workshop on Automatic Summarization*, Philadelphia.

Adwait Ratnaparkhi. 1998. *Maximum entropy models for natural language ambiguity resolution*. Ph.D. thesis, University of Pennsylvania.

Jonathan Richard Shewchuk. 1994. An introduction to the conjugate gradient method without the agonizing pain. http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.ps.

Jun Wu and Sanjeev Khudanpur. 2000. Efficient training methods for methods maximum entropy language modelling. In *Proceedings of ICSLP2000*, volume 3, pages 114–117, Beijing.