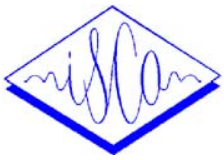
ISCA Archive
http://www.isca-speech.org/archive



INTERSPEECH 2004 - ICSLP
8th International Conference on Spoken Language Processing
ICC Jeju, Jeju Island, Korea
October 4-8, 2004

Fast parameter estimation for joint maximum entropy language models

Edward Schofield

ftw Telecommunications Research Center Vienna
Imperial College London
schofield@ftw.at

10.21437/Interspeech.2004-494

Abstract

This paper discusses efficient parameter estimation methods for joint (unconditional) maximum entropy language models such as whole-sentence models. Such models are a sound framework for formalizing arbitrary linguistic knowledge in a consistent manner. It has been shown that general-purpose gradient-based optimization methods are among the most efficient algorithms for estimating parameters of maximum entropy models for several domains in natural language processing. This paper applies gradient methods to whole-sentence language models and other domains whose sample spaces are infinite or practically innumerable and require simulation. It also presents Open Source software for easily fitting and testing joint maximum entropy models.

1. Introduction

The principle of maximum entropy was born in statistical mechanics in the 1930s, and later promoted by E. T. Jaynes in a series of articles (*e.g.* [1]) explaining the rationale for using maximum entropy methods and their relation to Bayesian inference. When only the marginals of a distribution are known, the principle instructs one to choose the distribution with the highest entropy as the most plausible. This corresponds well to the intuitive principle of modelling all that is known but assuming as little as possible about what is unknown. It is by now widely known (*e.g.* [2, 3, 4]) that, subject to a set of constraints of the form

$$\mathbb{E}_p f_i(X) = k_i, \tag{1}$$

where f_i are arbitrary real-valued functions on the sample space of X, the distribution $p(x)$ with the highest entropy has form $p(x) = \frac{1}{Z} \exp \lambda^T f(x)$, where λ_i are real-valued parameters and $Z = Z(\lambda)$ is a normalizing term independent of x. Furthermore, when the constants k_i represent empirical expectations of observations from a distribution p, the maximum entropy distribution is also the maximum likelihood distribution among all distributions of the following form:

$$p(s) = \frac{1}{Z(\lambda)} p_0(s) \exp\left(\sum_i \lambda_i f_i(s)\right). \tag{2}$$

In 1994 Rosenfeld submitted his PhD thesis [5] on the application of maximum entropy methods to language modelling. His thesis reports perplexities 32% lower than baseline trigram models, early evidence that such methods can improve language model fits in practice. This focus of this work was conditional models $p(x|w)$ of the next word x in the sentence following the words w. Such models have two disadvantages. First, they can be expensive to evaluate at run-time because the normalization term Z is dependent on the context w rather than a true constant. Second, and more importantly, they restrict our flexibility in selecting linguistic features for the model to those that are computable left-to-right.

More recently Rosenfeld and others [6] have proposed maximum entropy models of the joint probability of a whole sentence. Following this work we define a whole-sentence maximum entropy language model as one with the form of equation 2, where f_i are arbitrary computable features of the sentence s with associated parameters λ_i, where p_0 is an arbitrary "prior" distribution, and where Z is a normalization term defined as the sum $\sum_{s \in \Omega} \exp \lambda^T f(s)$ over the sample space Ω. In this paper we assume $p_0 \equiv 1$ for simplicity; the extension to the more general case should be straightforward.

For whole-sentence language modelling we would choose f_i to be features of a sentence that characterize the domain, and set k_i to be the expected values of these features in a corpus of representative text. The same model structure (2) could equally represent whole-paragraph or whole-document models in the case that s denotes a paragraph or document and Ω the sample space of all possible paragraphs or documents. We will refer to these generically as 'joint' maxent models.

Section 2 of this paper considers ways to find the parameters λ that satisfy the constraints in the case when the sample space Ω is countably infinite, as it is for sentences of arbitrary length, or finite but innumerable in practice. Section 3 describes an identity useful for the numerical implementation of models on innumerable sample spaces. Section 4 describes the implementation of these ideas in an open-source toolkit for fitting joint maxent models. Section 5 draws some conclusions.

2. Parameter estimation on innumerable sample spaces

We now consider how to find the parameters of the model $p(s)$ of exponential form (2) that has highest entropy $H(p)$ assuming that λ are set to satisfy the constraints (1). Standard results (*e.g.* [4]) imply that this choice of λ does in fact satisfy the constraints. We write $H(p)$ as:

$$H(p) = -\sum_s p(s)\left[\lambda_i f_i(s) - \log Z(\lambda)\right] \tag{3}$$

$$= -\sum_i \lambda_i k_i + \log Z(\lambda) \tag{4}$$

We wish to find the parameters λ that minimize this (dual) expression. It is well known ([2, 7]) that $\log Z(\lambda)$ is a convex function of the parameters $\{\lambda_i\}$. Provided we can compute $\log Z$ we can fit the maxent model with an iterative scaling algorithm or a standard global optimization routine such the direction set method implemented as `powell` in *Numerical Recipes* [8].

But Malouf [9] has shown that we can expect to reduce the number of iterations required for convergence by using both the objective function (4) and its gradient. We can use the following property of the normalization term Z:

$$\frac{\partial}{\partial \lambda_i} \log Z(\lambda) = \mathbb{E}_p f_i(S) \tag{5}$$

to express the partial derivatives of the entropy function (4) with respect to λ_i as:

$$\frac{\partial H}{\partial \lambda_i} = k_i - \mathbb{E}_p f_i(S) \tag{6}$$

So provided we can compute the expectation $\mathbb{E}_p f_i(S)$ we can fit the maxent distribution p with a gradient-based optimization algorithm like the conjugate gradient and variable metric routines described and compared in [9]; the `frprmn` routine from [8] is one example.

Now consider the case when the sample space Ω is innumerable, by which we mean infinite or too large to iterate over. The space of all possible sentences is an example in this class. In this case neither Z in (4) nor $\mathbb{E}_p f_i(S)$ in (5) can be computed. We can attempt to estimate them by simulation as follows. By an argument similar to the one in [6], for any instrumental distribution q we can express Z as:

$$Z(\lambda) = \mathbb{E}_q \frac{\exp \lambda^T f(S)}{q(S)}.$$

We can therefore approximate Z as

$$\widehat{Z}(\lambda) = \frac{1}{m} \sum_i \frac{\exp \lambda^T f(s_i)}{q(s_i)} \tag{7}$$

where $\{s_i\}$ is a sample of size m from the distribution q. By the Law of Large Numbers the variance of $\widehat{Z}$ tends to zero almost surely as the sample size m grows.

Using this estimator $\widehat{Z}$ in (4) yields an estimator for the entropy that allows one to fit the approximate maxent model using a standard optimization algorithm as before, even in the case of an innumerable sample space.

Now consider trying to reduce the number of iterations by using gradient information in conjunction with this entropy estimator. Suppose we cannot compute the expectations $\mathbb{E}_p f_i(S)$ exactly. If we try to estimate them by sampling, as Rosenfeld *et al*. did in [6], the following question raises itself:

> Should we use the gradient of an estimator of (4), or an estimator of the gradient (6)?

The choice may impact the convergence of an optimization algorithm that uses both objective and gradient functions. (Note that the GIS algorithm considered in [6] uses only the gradient.)

The following result shows that, if we estimate Z by $\widehat{Z}$ as above and estimate $\mathbb{E}_p f_i(S)$ by importance sampling with the same instrumental distribution q, these alternatives are in fact the same.

Lemma 1. *Suppose* $p(s_j) = \frac{1}{Z(\lambda)} \exp \lambda^T f(s_j)$ *and* $\widehat{Z} = \frac{1}{m} \sum_j \frac{\exp \lambda^T f(s_j)}{q(s_j)}$, *where* s_j *is a finite sample drawn from the instrumental distribution* q. *Then*

$$\frac{\partial}{\partial \lambda_i} \log \widehat{Z}(\lambda) = \frac{\sum_{j=1}^m \frac{p(s_j)}{q(s_j)} f_i(s_j)}{\sum_{j=1}^m \frac{p(s_j)}{q(s_j)}} \tag{8}$$

Proof. By the chain rule for partial differentiation:

$$\frac{\partial \log \widehat{Z}}{\partial \lambda_i} = \frac{\partial \widehat{Z}}{\partial \lambda_i} \cdot \frac{1}{\widehat{Z}}$$

and

$$\frac{\partial \widehat{Z}}{\partial \lambda_i} = \frac{1}{m} \sum_j f_i(s_j) \frac{\exp \lambda^T f(s_j)}{q(s_j)}.$$

Now, upon re-expressing $\frac{\partial \widehat{Z}}{\partial \lambda_i}$ and $\frac{1}{\widehat{Z}}$ in terms of p, the Z terms in the denominators cancel, giving the result. □

The right hand side of equation (8) is the importance sampling estimate of $\mathbb{E}_p f(S)$, giving a natural analogy with the exact case in equation (5). Note that this is a property of the importance sampler that renders it particularly suitable for exponential models.

Algorithms may not exist for which we can guarantee convergence to the maxent distribution using Monte Carlo estimators of Z and $\mathbb{E}_p f_i(S)$. But with a large enough sample size the implementation described in Section 4, which is derived from the estimator $\widehat{Z}$ in (7) and the right side of equation (8), converges reliably in practice.

3. The normalization term $Z(\lambda)$

Consider the normalization term $Z = \sum_{s\in\Omega} \exp \lambda^T f(s)$. We may potentially have many features active for any given s (especially if s is not a sentence but a paragraph or document). If the inner product $\lambda^T f(s)$ is much greater than approximately 700 for any one of the events s in the sample space the value of Z will overflow a standard 64-bit floating point type. We could potentially scale all features to within a certain range $[a,b]$ and then make modifications to our optimization algorithm to ensure that the λ_i are never large enough to overflow Z. But it is not clear how this would be done. Instead I suggest using the following trick, which is sometimes used in the literature on Turbo Coding.

Lemma 2.

$$\log(e^{a_1} + e^{a_2}) = \max\{a_1, a_2\} + \log\left(1 + e^{-|a_1 - a_2|}\right) \quad (9)$$

The proof is straightforward, treating the two cases $a_1 \geq a_2$ and $a_1 < a_2$ separately. The following iterative algorithm generalizes the result to n terms $a_1, \ldots, a_n$:

LOG-SUM-EXP$(a_1, \ldots, a_n)$
1 $b_1 \leftarrow a_1$
2 **for** $i \leftarrow 2$ **to** n
3 **do** $b_i \leftarrow \max\{b_{i-1}, a_i\} + \log\left(1 + e^{-|b_{i-1} - a_i|}\right)$
4 **return** b_n

This allows one to compute $\log Z$ and $\log \widehat{Z}$ exactly, even if Z and $\widehat{Z}$ are too large to represent, using $O(n)$ time and $O(1)$ space. The toolkit described next makes use of this extensively.

4. Toolkit for fitting joint maxent models

Although various software tools currently exist to fit conditional maxent models, such as [10] and [11], none supports Monte Carlo estimation of Z or $\mathbb{E}_p f_i(S)$ for for fitting joint models for sentences or larger units. This section describes such a toolkit using Numerical Python [12], the sparse matrix module by Roman Geus [13], and an implementation of Polak-Ribière conjugate gradient optimization inspired by *Numerical Recipes* [8]. The source code and documentation for the `ftwmaxent` module are available at `http://textmodeller.sf.net`.

4.1. A simple walkthrough

Consider the problem posed by Berger *et al.* [4] of estimating a simple language model for translating the English preposition 'in' into French. Suppose that in a corpus of parallel texts, such as the proceedings of the European or Canadian parliaments, that we observe the following:

$$p(\text{dans}) + p(\text{en}) + p(\text{à}) + p(\text{au cours de}) + p(\text{pendant}) = 1$$
$$p(\text{dans}) + p(\text{en}) = 0.3$$
$$p(\text{dans}) + p(\text{à}) = 0.5$$

and we wish to determine the maxent distribution satisfying these relations. Figure 1 shows code for this.

```
Python 2.3.3 (#1, Mar 25 2004, 16:29:43)
>>> import math, ftwmaxent, ftwutils
>>> a_grave = u'\u00e0'
>>> samplespace = ['dans', 'en', a_grave,
        'au cours de', 'pendant']
>>> def f0(x):
...     return (x in samplespace)
...
>>> def f1(x):
...     return x=='dans' or x=='en'
...
>>> def f2(x):
...     return x=='dans' or x==a_grave
...
>>> f = [f0, f1, f2]
>>> model = ftwmaxent.Model(f,
                samplespace)
>>> K = [1.0, 0.3, 0.5]
>>> model.fit(K)
```

Figure 1: Example code to fit the translation example of Berger *et al.* [4] with the `ftwmaxent` module

Note that we can use Python's Unicode support to handle the à character correctly. Note also that although we pass the sample space explicitly, it may represent an infinite object (as a Python iterator), and we can modify it dynamically (for example, to add words to the lexicon).

The code in Figure 1 fits the model without simulation. We can alternatively fit the model using simulation by substituting the last command in Figure 1 with `model.fitapprox(K)`. In this case a uniform instrumental distribution is used for sampling by default. A different sampling distribution can otherwise be specified, as explained in the documentation.

This is an illustration of a toolkit designed explicitly for innumerable sample spaces with the smallest example I could find. This is slightly silly, but the example is easy to understand and verify, and its small sample space allows a comparison of the routines using estimators with those using exact expressions. Table 1 shows trials of how well the constraints are satisfied and how many iterations are required to reach convergence as a function of sample size for the naive and gradient-based methods and their exact versions. All trials in Table 1 are based on a uniform instrumental sampling distribution $p(\text{dans}) = \cdots = p(\text{au cours de}) = \frac{1}{6}$ and convergence

tolerance $\varepsilon = 10^{-6}$. All figures are averages over 100 trials. Source code for repeating the tests is included with the toolkit.

Table 1: Fitted models for the translation example of Berger *et al*. [4].

# samples	Avg. # iterations	Avg. $\|\|\mathbb{E}_p f(S) - k\|\|^2$
Without gradient, exact:		
—	113	7.2×10^{-15}
With gradient, exact:		
—	8	5.5×10^{-18}
Without gradient, simulated:		
10^2 samples	122.4	4.4×10^{-3}
10^3 samples	119.4	4.6×10^{-4}
10^4 samples	115.0	5.0×10^{-5}
10^5 samples	114.2	4.3×10^{-6}
With gradient, simulated:		
10^2 samples	7.9	4.8×10^{-3}
10^3 samples	7.5	5.1×10^{-4}
10^4 samples	7.4	3.9×10^{-5}
10^5 samples	7.5	4.7×10^{-6}

5. Conclusions

This paper brings the theory of whole-sentence maxent language models up-to-date with recent progress in parameter estimation for general maxent models in language processing. At the same time it extends the theory of parameter estimation for maxent models to domains with large sample spaces as exemplified by whole-sentence models. Two useful results are that the importance sampler is a natural choice of sampler for estimating feature expectations of exponential models; and that the normalization terms in maxent models can be computed and approximated efficiently in the log domain to avoid numerical difficulties in large tasks.

The accompanying implementation of this theory should also bring some practical benefits. Maxent language modelling for sentences and larger tokens will become easier, since until now no suitable modelling toolkit has been publicly available. A speech scientist, linguist, or teenage hacker for whom the theory of these models is inaccessible should be able to experiment with linguistic features and build successful models using the toolkit and accompanying documentation. I would also hope that the toolkit's generic structure and Open Source license help it find use in speech recognition, machine translation, and diverse problems in machine learning.

6. Acknowledgments

I would like to thank Konstantinos Koumpis for helpful comments on a draft of this paper.

This work was partially funded under the Austrian government's K–*plus* Competence Center Program.

7. References

[1] E. T. Jaynes. *The Relation of Bayesian and Maximum Entropy Methods*, page 25. American Institute of Physics, 1988.

[2] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley, 1991.

[3] I. Csiszar. Why least squares and maximum entropy? *Annals of Statistics*, 19(4):2032–2056, 1991.

[4] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Comp. Linguistics*, 22(1):39–71, 1996.

[5] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1994.

[6] R. Rosenfeld, S. F. Chen, and X. Zhu. Whole-sentence exponential language models: A vehicle for linguistic-statistical integration. *Computer Speech and Language*, 15(1), 2001.

[7] T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. Technical Report AITR-1668, MIT AI Laboratory, October 1999.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. CUP, 1992.

[9] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. CoNLL-2002*, pages 49–55. Taipei, Taiwan, 2002.

[10] Z. Le. Maximum entropy modeling toolkit for Python and C++, `http://www.nlplab.cn/zhangle/maxent_toolkit.html`, 2003.

[11] J. Baldridge, T. Morton, and G. Bierner. OpenNLP maxent package in Java, 2002.

[12] P. F. Dubois, K. Hinsen, and J. Hugunin. Numerical Python. *Journal of Comp. Physics*, 10(3):262–267, May/June 1996.

[13] R. Geus. *The Jacobi-Davidson algorithm for solving large sparse symmetric eigenvalue problems.*, chapter 9. PhD Thesis, ETH Zurich, 2002.