Investigating GIS and Smoothing for Maximum Entropy Taggers

James R. Curran and **Stephen Clark**
School of Informatics
University of Edinburgh
2 Buccleuch Place, Edinburgh. EH8 9LW
{jamesc,stephenc}@cogsci.ed.ac.uk

Abstract

This paper investigates two elements of Maximum Entropy tagging: the use of a correction feature in the Generalised Iterative Scaling (GIS) estimation algorithm, and techniques for model smoothing. We show analytically and empirically that the correction feature, assumed to be required for the correctness of GIS, is unnecessary. We also explore the use of a Gaussian prior and a simple cutoff for smoothing. The experiments are performed with two tagsets: the standard Penn Treebank POS tagset and the larger set of lexical types from Combinatory Categorial Grammar.

1 Introduction

The use of maximum entropy (ME) models has become popular in Statistical NLP; some example applications include part-of-speech (POS) tagging (Ratnaparkhi, 1996), parsing (Ratnaparkhi, 1999; Johnson et al., 1999) and language modelling (Rosenfeld, 1996). Many tagging problems have been successfully modelled in the ME framework, including POS tagging, with state of the art performance (van Halteren et al., 2001), "supertagging" (Clark, 2002) and chunking (Koeling, 2000).

Generalised Iterative Scaling (GIS) is a very simple algorithm for estimating the parameters of a ME model. The original formulation of GIS (Darroch and Ratcliff, 1972) required the sum of the feature values for each event to be constant. Since this is not the case for many applications, the standard method is to add a "correction", or "slack", feature to each event. Improved Iterative Scaling (IIS) (Berger et al., 1996; Della Pietra et al., 1997) eliminated the correction feature to improve the convergence rate of the algorithm. However, the extra book keeping required for IIS means that GIS is often faster in practice (Malouf, 2002). This paper shows, by a simple adaptation of Berger's proof for the convergence of IIS (Berger, 1997), that GIS does not require a correction feature. We also investigate how the use of a correction feature affects the performance of ME taggers.

GIS and IIS obtain a maximum likelihood estimate (MLE) of the parameters, and, like other MLE methods, are susceptible to overfitting. A simple technique used to avoid overfitting is a frequency cutoff, in which only frequently occurring features are included in the model (Ratnaparkhi, 1998). However, more sophisticated smoothing techniques exist, such as the use of a Gaussian prior on the parameters of the model (Chen and Rosenfeld, 1999). This technique has been applied to language modelling (Chen and Rosenfeld, 1999), text classification (Nigam et al., 1999) and parsing (Johnson et al., 1999), but to our knowledge it has not been compared with the use of a feature cutoff. We explore the combination of Gaussian smoothing and a simple cutoff for two tagging tasks.

The two taggers used for the experiments are a POS tagger, trained on the WSJ Penn Treebank, and a "supertagger", which assigns tags from the much larger set of lexical types from Combinatory Categorial Grammar (CCG) (Clark, 2002). Elimination of the correction feature and use of appropriate smoothing methods result in state of the art performance for both tagging tasks.

2 Maximum Entropy Models

A conditional ME model, also known as a log-linear model, has the following form:

$$p(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n} \lambda_i f_i(x,y)\right) \quad (1)$$

where the functions f_i are the features of the model, the λ_i are the parameters, or weights, and $Z(x)$ is a normalisation constant. This form can be derived by choosing the model with maximum entropy (i.e. the most uniform model) from a set of models that satisfy a certain set of constraints. The constraints are that the expected value of each feature f_i according to the model p is equal to some value K_i (Rosenfeld, 1996):

$$\sum_{x,y} p(x,y) f_i(x,y) = K_i \quad (2)$$

Calculating the expected value according to p requires summing over all contexts x, which is not possible in practice. Therefore we use the now standard approximation (Rosenfeld, 1996):

$$\sum_{x,y} p(x,y) f_i(x,y) \approx \sum_{x,y} \tilde{p}(x) p(y|x) f_i(x,y) \quad (3)$$

where $\tilde{p}(x)$ is the relative frequency of context x in the data. This is convenient because $\tilde{p}(x)$ is zero for all those events not seen in the training data.

Finding the maximum entropy model that satisfies these constraints is a constrained optimisation problem, which can be solved using the method of Lagrange multipliers, and leads to the form in (1) where the λ_i are the Lagrange multipliers.

A natural choice for K_i is the empirical expected value of the feature f_i:

$$E_{\tilde{p}} f_i = \sum_{x,y} \tilde{p}(x,y) f_i(x,y) \quad (4)$$

which leads to the following set of constraints:

$$\sum_{x,y} \tilde{p}(x) p(y|x) f_i(x,y) = E_{\tilde{p}} f_i \quad (5)$$

An alternative motivation for this model is that, starting with the log-linear form in (1) and deriving (conditional) MLEs, we arrive at the same solution as the ME model which satisfies the constraints in (5).

3 Generalised Iterative Scaling

GIS is a very simple algorithm for estimating the parameters of a ME model. The algorithm is as follows, where $E_{\tilde{p}} f_i$ is the empirical expected value of f_i and $E_p f_i$ is the expected value according to model p:

- Set $\lambda_i^{(0)}$ equal to some arbitrary value, say:

$$\lambda_i^{(0)} = 0 \quad (6)$$

- Repeat until convergence:

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \frac{1}{C} \log \frac{E_{\tilde{p}} f_i}{E_{p^{(t)}} f_i} \quad (7)$$

where (t) is the iteration index and the constant C is defined as follows:

$$C = \max_{x,y} \sum_{i=1}^{n} f_i(x,y) \quad (8)$$

In practice C is maximised over the (x,y) pairs in the training data, although in theory C can be any constant greater than or equal to the figure in (8). However, since $\frac{1}{C}$ determines the rate of convergence of the algorithm, it is preferable to keep C as small as possible.

The original formulation of GIS (Darroch and Ratcliff, 1972) required the sum of the feature values for each event to be constant. Since this is not the case for many applications, the standard method is to add a “correction”, or “slack”, feature to each event, defined as follows:

$$f_c(x,y) = C - \sum_{i=1}^{n} f_i(x,y) \quad (9)$$

For our tagging experiments, the use of a correction feature did not significantly affect the results. Moreover, we show in the Appendix, by a simple adaptation of Berger's proof for the convergence of IIS (Berger, 1997), that GIS converges to the maximum likelihood model without a correction feature.[1]

The proof works by introducing a correction feature with fixed weight of 0 into the IIS convergence proof. This feature does not contribute to the model and can be ignored during weight update. Introducing this null feature still satisfies Jensen's inequality, which is used to provide a lower bound on the change in likelihood between iterations, and the existing GIS weight update (7) can still be derived analytically.

An advantage of GIS is that it is a very simple algorithm, made even simpler by the removal of the correction feature. This simplicity means that, although GIS requires more iterations than IIS to reach convergence, in practice it is significantly faster (Malouf, 2002).

4 Smoothing Maximum Entropy Models

Several methods have been proposed for smoothing ME models (see Chen and Rosenfeld (1999)). For taggers, a standard technique is to eliminate low frequency features, based on the assumption that they are unreliable or uninformative (Ratnaparkhi, 1998). Studies of infrequent features in other domains suggest this assumption may be incorrect (Daelemans et al., 1999). We test this for ME taggers by replacing the cutoff with the use of a Gaussian prior, a technique which works well for language models (Chen and Rosenfeld, 1999).

When using a Gaussian prior, the objective function is no longer the likelihood, $L_{\tilde{p}}(\Lambda)$, but has the form:

$$L'_{\tilde{p}}(\Lambda) = L_{\tilde{p}}(\Lambda) + \sum_i \log \frac{1}{\sqrt{2\pi\sigma_i^2}} - \frac{\lambda_i^2}{2\sigma_i^2} \quad (10)$$

Maximising this function is a form of maximum *a posteriori* estimation, rather than maximum likelihood estimation. The effect of the prior is to penalise models that have very large positive or negative weights. This can be thought of as relaxing the constraints in (5), so that the model fits the data less exactly. The parameters σ_i are usually collapsed into one parameter which can be set using heldout data.

[1]We note that Goodman (2002) suggests that the correction feature may not be necessary for convergence.

CCG lexical category	Description
$(S\backslash NP)/NP$	transitive verb
$S\backslash NP$	intransitive verb
NP/N	determiner
N/N	nominal modifier
$(S\backslash NP)\backslash(S\backslash NP)$	adverbial modifier

Table 1: Example CCG lexical categories

The new update rule for GIS with a Gaussian prior is found by solving the following equation for the λ_i update values (denoted by δ_i), which can easily be derived from (10) by analogy with the proof in the Appendix:

$$E_{\tilde{p}} f_i = E_p f_i \, e^{C\delta_i} + \frac{\lambda_i + \delta_i}{\sigma_i^2} \quad (11)$$

This equation does not have an analytic solution for δ_i and can be solved using a numerical solver such as Newton-Raphson. Note that this new update rule is still significantly simpler than that required for IIS.

5 Maximum Entropy Taggers

We reimplemented Ratnaparkhi's publicly available POS tagger MXPOST (Ratnaparkhi, 1996; Ratnaparkhi, 1998) and Clark's CCG supertagger (Clark, 2002) as a starting point for our experiments. CCG supertagging is more difficult than POS tagging because the set of "tags" assigned by the supertagger is much larger (398 in this implementation, compared with 45 POS tags). The supertagger assigns CCG lexical categories (Steedman, 2000) which encode subcategorisation information. Table 1 gives some examples.

The features used by each tagger are binary valued, and pair a tag with various elements of the context; for example:

$$f_j(x, y) = \begin{cases} 1 & \text{if } \texttt{word}(x) = \texttt{the} \ \& \ y = \mathrm{DT} \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

$\texttt{word}(x) = \texttt{the}$ is an example of what Ratnaparkhi calls a *contextual predicate*. The contextual predicates used by the two taggers are given in Table 2, where w_i is the ith word and t_i is the

Condition	Contextual predicate
$freq(w_i) \geq 5$	$w_i = X$
$freq(w_i) < 5$ (POS tagger)	X is prefix of w_i, $\|X\| \leq 4$ X is suffix of w_i, $\|X\| \leq 4$ w_i contains a digit w_i contains uppercase char w_i contains a hyphen
$\forall w_i$	$t_{i-1} = X$ $t_{i-2}t_{i-1} = XY$ $w_{i-1} = X$ $w_{i-2} = X$ $w_{i+1} = X$ $w_{i+2} = X$
$\forall w_i$ (supertagger)	$\text{POS}_i = X$ $\text{POS}_{i-1} = X$ $\text{POS}_{i-2} = X$ $\text{POS}_{i+1} = X$ $\text{POS}_{i+2} = X$

Table 2: Contextual predicates used in the taggers

ith tag. We insert a special end of sentence symbol at sentence boundaries so that the features looking forwards and backwards are always defined.

The supertagger uses POS tags as additional features, which Clark (2002) found improved performance significantly, and does not use the morphological features, since the POS tags provide equivalent information. For the supertagger, t_i is the lexical category of the ith word.

The conditional probability of a tag sequence $y_1 \ldots y_n$ given a sentence $w_1 \ldots w_n$ is approximated as follows:

$$p(y_1 \ldots y_n | w_1 \ldots w_n) \approx \prod_{i=1}^{n} p(y_i | x_i) \qquad (13)$$

$$= \prod_{i=1}^{n} \frac{1}{Z(x_i)} \exp \left(\sum_j \lambda_j f_j(x_i, y_i) \right) \qquad (14)$$

where x_i is the context of the ith word. The tagger returns the most probable sequence for the sentence. Following Ratnaparkhi, beam search is used to retain only the 20 most probable sequences during the tagging process;[2] we also use a "tag dictionary", so that words appearing 5 or more times in the data can only be assigned those tags previously seen with the word.

[2] Ratnaparkhi uses a beam width of 5.

Split	DATA	# SENT.	# WORDS
Develop	WSJ 00	1921	46451
Train	WSJ 02-21	39832	950028
Test	WSJ 23	2416	56684

Table 3: WSJ training, testing and development

Tagger	ACC	UWORD	UTAG	AMB
MXPOST	96.59	**85.81**	30.04	94.82
BASE	96.58	85.70	29.28	94.82
– CORR	**96.60**	85.58	**31.94**	**94.85**

Table 4: Basic tagger performance on WSJ 00

6 POS Tagging Experiments

We develop and test our improved POS tagger (C&C) using the standard parser development methodology on the Penn Treebank WSJ corpus. Table 3 shows the number of sentences and words in the training, development and test datasets.

As well as evaluating the overall accuracy of the taggers (ACC), we also calculate the accuracy on previously unseen words (UWORD), previously unseen word-tag pairs (UTAG) and ambiguous words (AMB), that is, those with more than one tag over the testing, training and development datasets. Note that the unseen word-tag pairs do not include the previously unseen words.

We first replicated the results of the MXPOST tagger. In doing so, we discovered a number of minor variations from Ratnaparkhi (1998):

- MXPOST adds a default contextual predicate which is true for every context;
- MXPOST does not use the cutoff values described in Ratnaparkhi (1998).

MXPOST uses a cutoff of 1 for the current word feature and 5 for other features. However, the current word must have appeared at least 5 times with any tag for the current word feature to be included; otherwise the word is considered rare and morphological features are included instead.

7 POS Tagging Results

Table 4 shows the performance of MXPOST and our reimplementation.[3] The third row shows a mi-

[3] By examining the MXPOST model files, we discovered a minor error in the counts for prefix and suffix features, which may explain the slight difference in performance.

Tagger	ACC	UWORD	UTAG	AMB
BASE $\alpha = 2.05$	96.75	86.74	33.08	**95.06**
$w \geq 2, \alpha = 2.06$	96.71	86.62	33.46	95.00
$w \geq 3, \alpha = 2.05$	96.68	86.51	**34.22**	94.94
$pw \geq 2, \alpha = 1.50$	**96.76**	87.02	32.70	**95.06**
$pw \geq 3, \alpha = 1.75$	**96.76**	**87.14**	33.08	**95.06**

Table 5: WSJ 00 results with varying current and previous word feature cutoffs

Tagger	ACC	UWORD	UTAG	AMB
$\geq 1, \alpha = 1.95$	**96.82**	**87.20**	30.80	**95.07**
$\geq 2, \alpha = 1.98$	96.77	87.02	31.18	95.00
$\geq 3, \alpha = 1.73$	96.72	86.62	31.94	94.94
$\geq 4, \alpha = 1.50$	96.72	87.08	**34.22**	94.96

Table 6: WSJ 00 results with varying cutoffs

nor improvement in performance when the correction feature is removed. We also experimented with the default contextual predicate but found it had little impact on the performance. For the remainder of the experiments we use neither the correction nor the default features.

The rest of this section considers various combinations of feature cutoffs and Gaussian smoothing. We report optimal results with respect to the smoothing parameter α, where $\alpha = N\sigma^2$ and N is the number of training instances. We found that using $\alpha \approx 2$ gave the most benefit to our basic tagger, improving performance by about 0.15% on the development set. This result is shown in the first row of Table 5.

The remainder of Table 5 shows a minimal change in performance when the current word (*w*) and previous word (*pw*) cutoffs are varied. This led us to reduce the cutoffs for all features simultaneously. Table 6 gives results for cutoff values between 1 and 4. The best performance (in row 1) is obtained when the cutoffs are eliminated entirely.

Gaussian smoothing has allowed us to retain all of the features extracted from the corpus and reduce overfitting. To get more information into the model, more features must be extracted, and so we investigated the addition of the current word feature for *all* words, including the rare ones. This resulted in a minor improvement, and gave the best performance on the development data: 96.83%.

Tagger	ACC	UWORD	UTAG	AMB
MXPOST	97.05	83.63	**30.20**	95.44
C&C	**97.27**	**85.21**	28.98	**95.69**

Table 7: Tagger performance on WSJ 23

Tagger	# PREDICATES	# FEATURES
BASE	44385	121557
C&C	254038	685682

Table 8: Model size

Table 7 shows the final performance on the test set, using the best configuration on the development data (which we call C&C), compared with MXPOST. The improvement is 0.22% overall (a reduction in error rate of 7.5%) and 1.58% for unknown words (a reduction in error rate of 9.7%).

The obvious cost associated with retaining all the features is the significant increase in model size, which slows down both the training and tagging and requires more memory. Table 8 shows the difference in the number of contextual predicates and features between the original and final taggers.

8 POS Tagging Validation

To ensure the robustness of our results, we performed 10-fold cross-validation using the whole of the WSJ Penn Treebank. The 24 sections were split into 10 equal components, with 9 used for training and 1 for testing. The final result is an average over the 10 different splits, given in Table 9, where σ is the standard deviation of the overall accuracy. We also performed 10-fold cross-validation using MXPOST and TNT, a publicly available Markov model POS tagger (Brants, 2000).

The difference between MXPOST and C&C represents a reduction in error rate of 4.3%, and the

Tagger	ACC	σ	UWORD	UTAG	AMB
MXPOST	96.72	0.12	85.50	**32.16**	95.00
TNT	96.48	0.13	85.31	0.00	94.26
C&C	**96.86**	0.12	**86.43**	30.42	**95.08**

Table 9: 10-fold cross-validation results

Tagger	ACC	UWORD	UTAG	AMB
COLLINS	**97.07**	-	-	-
C&C	96.93	87.28	34.44	95.31
T&M	96.86	**86.91**	-	-
C&C	**97.10**	86.43	34.84	95.52

Table 10: Comparison with other taggers

difference between TNT and C&C a reduction in error rate of 10.8%.

We also compare our performance against other published results that use different training and testing sections. Collins (2002) uses WSJ 00-18 for training and WSJ 22-24 for testing, and Toutanova and Manning (2000) use WSJ 00-20 for training and WSJ 23-24 for testing. Collins uses a linear perceptron, and Toutanova and Manning (T&M) use a ME tagger, also based on MXPOST. Our performance (in Table 10) is slightly worse than Collins', but better than T&M (except for unknown words). We noticed during development that unknown word performance improves with larger α values at the expense of overall accuracy – and so using separate α's for different types of contextual predicates may improve performance. A similar approach has been shown to be successful for language modelling (Goodman, p.c.).

9 Supertagging Experiments

The lexical categories for the supertagging experiments were extracted from CCGbank, a CCG version of the Penn Treebank (Hockenmaier and Steedman, 2002). Following Clark (2002), all categories that occurred at least 10 times in the training data were used, resulting in a tagset of 398 categories. Sections 02-21, section 00, and section 23 were used for training, development and testing, as before.

Our supertagger used the same configuration as our best performing POS tagger, except that the α parameter was again optimised on the development set. The results on section 00 and section 23 are given in Tables 11 and 12.[4] C&C outperforms Clark's supertagger by 0.43% on the test set, a reduction in error rate of 4.9%.

Supertagging has the potential to benefit more from Gaussian smoothing than POS tagging because the feature space is sparser by virtue of the much larger tagset. Gaussian smoothing would also allow us to incorporate rare longer range dependencies as features, without risk of overfitting. This may further boost supertagger performance.

[4]The results in Clark (2002) are slightly lower because these did not include punctuation.

Tagger	ACC	UWORD	UTAG	AMB
CLARK	90.97	90.86	28.48	89.84
C&C $\alpha = 1.52$	**91.45**	**91.16**	**28.79**	**90.38**

Table 11: Supertagger WSJ 00 results

Tagger	ACC	UWORD	UTAG	AMB
CLARK	91.27	88.48	32.20	90.32
C&C $\alpha = 1.52$	**91.70**	**88.92**	**32.30**	**90.78**

Table 12: Supertagger WSJ 23 results

10 Conclusion

This paper has demonstrated, both analytically and empirically, that GIS does not require a correction feature. Eliminating the correction feature simplifies further the already very simple estimation algorithm. Although GIS is not as fast as some alternatives, such as conjugate gradient and limited memory variable metric methods (Malouf, 2002), our C&C POS tagger takes less than 10 minutes to train, and the space requirements are modest, irrespective of the size of the tagset.

We have also shown that using a Gaussian prior on the parameters of the ME model improves performance over a simple frequency cutoff. The Gaussian prior effectively relaxes the constraints on the ME model, which allows the model to use low frequency features without overfitting. Achieving optimal performance with Gaussian smoothing and without cutoffs demonstrates that low frequency features can contribute to good performance.

Acknowledgements

We would like to thank Joshua Goodman, Miles Osborne, Andrew Smith, Hanna Wallach, Tara Murphy and the anonymous reviewers for their comments on drafts of this paper. This research is supported by a Commonwealth scholarship and a Sydney University Travelling scholarship to the first author, and EPSRC grant GR/M96889.

References

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Adam Berger. 1997. The improved iterative scaling algorithm: A gentle introduction. Unpublished manuscript.

Thorsten Brants. 2000. TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*.

Stanley Chen and Ronald Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University, Pittsburgh, PA.

Stephen Clark. 2002. A supertagger for Combinatory Categorial Grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 19–24, Venice, Italy.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP Conference*, pages 1–8, Philadelphia, PA.

Walter Daelemans, Antal Van Den Bosch, and Jakub Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34(1-3):11–43.

J. N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 19(4):380–393.

Joshua Goodman. 2002. Sequential conditional generalized iterative scaling. In *Proceedings of the 40th Meeting of the ACL*, pages 9–16, Philadelphia, PA.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third LREC Conference*, Las Palmas, Spain.

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

Rob Koeling. 2000. Chunking with maximum entropy models. In *Proceedings of the CoNLL Workshop 2000*, pages 139–141, Lisbon, Portugal.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Workshop on Natural Language Learning*, pages 49–55, Taipei, Taiwan.

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, Stockholm, Sweden.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the EMNLP conference*, Hong Kong.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 2001. Improving accuracy in wordclass tagging through combination of machine learning systems. *Computational Linguistics*, 27(2):199–229.

Appendix A: Correction free GIS

This proof of GIS convergence without the correction feature is based on the IIS convergence proof by Berger (1997).

Start with some initial model with arbitrary parameters $\Lambda \equiv \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. Each iteration of the GIS algorithm finds a set of new parameters $\Lambda' \equiv \Lambda + \Delta \equiv \{\lambda_1 + \delta_1, \lambda_2 + \delta_2, \ldots, \lambda_n + \delta_n\}$ which increases the log-likelihood of the model.

The change in log-likelihood is as follows:

$$\begin{aligned} L_{\tilde{p}}(\Lambda + \Delta) &- L_{\tilde{p}}(\Lambda) \\ &= \sum_{x,y} \tilde{p}(x,y) \log p_{\Lambda'}(y|x) - \sum_{x,y} \tilde{p}(x,y) \log p_{\Lambda}(y|x) \\ &= \sum_{x,y} \tilde{p}(x,y) \sum_{i=1}^{n} \delta_i f_i(x,y) - \sum_{x} \tilde{p}(x) \log \frac{Z_{\Lambda'}(x)}{Z_{\Lambda}(x)} \end{aligned} \tag{15}$$

As in Berger (1997), use the inequality $-\log \alpha \geq 1 - \alpha$ to establish a lower bound on the change in likelihood:

$$L_{\tilde{p}}(\Lambda + \Delta) - L_{\tilde{p}}(\Lambda) \geq$$

$$\sum_{x,y} \tilde{p}(x,y) \sum_{i=1}^{n} \delta_i f_i(x,y) + \sum_{x} \tilde{p}(x) \left(1 - \frac{Z_{\Lambda'}(x)}{Z_\Lambda(x)}\right)$$

$$= \sum_{x,y} \tilde{p}(x,y) \sum_{i=1}^{n} \delta_i f_i(x,y) + 1 - \sum_{x} \tilde{p}(x) \frac{Z_{\Lambda'}(x)}{Z_\Lambda(x)}$$

$$= 1 + \sum_{x,y} \tilde{p}(x,y) \sum_{i=1}^{n} \delta_i f_i(x,y)$$

$$- \sum_{x} \tilde{p}(x) \sum_{y} \frac{1}{Z_\Lambda(x)} \exp \sum_{i=1}^{n} (\lambda_i + \delta_i) f_i(x,y)$$

$$= 1 + \sum_{x,y} \tilde{p}(x,y) \sum_{i=1}^{n} \delta_i f_i(x,y)$$

$$- \sum_{x} \tilde{p}(x) \sum_{y} p_\Lambda(y|x) \exp \sum_{i=1}^{n} \delta_i f_i(x,y) \quad (16)$$

Call the right hand side of this last equation $\mathcal{A}(\Delta|\Lambda)$. If we can find a Δ for which $\mathcal{A}(\Delta|\Lambda) > 0$, then $L_{\tilde{p}}(\Lambda + \Delta)$ is an improvement over $L_{\tilde{p}}(\Lambda)$. The obvious approach is to maximise $\mathcal{A}(\Delta|\Lambda)$ with respect to each δ_i, but this cannot be performed directly, since differentiating $\mathcal{A}(\Delta|\Lambda)$ with respect to δ_i leads to an equation containing all elements of Δ.

The trick is to rewrite $\mathcal{A}(\Delta|\Lambda)$ as follows, with an extra term which will be used to satisfy Jensen's inequality:

$$\mathcal{A}(\Delta|\Lambda) = 1 + \sum_{x,y} \tilde{p}(x,y) \sum_{i=1}^{n} \delta_i f_i(x,y)$$

$$- \sum_{x} \tilde{p}(x) \sum_{y} p_\Lambda(y|x) \exp \left(\sum_{i=1}^{n+1} \frac{f_i(x,y)}{C} \, C \, \delta_i \right) \quad (17)$$

where C is previously defined in equation 8, $f_{n+1}(x,y) = f_c(x,y)$ as in (9), and δ_{n+1} is defined to be zero. Note that the correction feature has been introduced but has been given a constant weight of zero.

This reformulation of $\mathcal{A}(\Delta|\Lambda)$ is similar to Berger's for the IIS proof, but with a crucial difference: Berger introduces $f^\# = \sum_{x,y} f_i(x,y)$ into the equation rather than C, and does not have the correction feature.

The next part of the proof introduces another, less tight, lower bound on the change in likelihood, by using Jensen's inequality, which can be stated as follows:

Let f be a convex function on the interval I. If $x_1, x_2, \ldots x_n \in I$ and $t_1, t_2, \ldots t_n$ are non-negative real numbers such that $\sum_{i=1}^{n} t_i = 1$, then

$$f\left(\sum_{i=1}^{n} t_i x_i\right) \leq \sum_{i=1}^{n} t_i f(x_i) \quad (18)$$

Since $\sum_{i=1}^{n+1} \frac{f_i(x,y)}{C} = 1$ and the exponential function is convex, we can apply Jensen's inequality to give a new form of $\mathcal{A}(\Delta|\Lambda)$:

$$\mathcal{A}(\Delta|\Lambda) \geq 1 + \sum_{x,y} \tilde{p}(x,y) \sum_{i} \delta_i f_i(x,y) -$$

$$\sum_{x} \tilde{p}(x) \sum_{y} p_\Lambda(y|x) \sum_{i=1}^{n+1} \frac{f_i(x,y)}{C} \exp(C\,\delta_i) \quad (19)$$

Call this bound $\mathcal{B}(\Delta|\Lambda)$. Della Pietra et al. (1997) give extra conditions on the continuity and derivative of the lower bound, in order to guarantee convergence. These conditions can be verified for $\mathcal{B}(\Delta|\Lambda)$ in a similar way to Della Pietra et al. (1997).

Differentiating $\mathcal{B}(\Delta|\Lambda)$ with respect to each weight update δ_i ($1 \leq \delta_i \leq n$) gives:

$$\frac{\partial \mathcal{B}(\Delta|\Lambda)}{\partial \delta_i} = \sum_{x,y} \tilde{p}(x,y) f_i(x,y)$$

$$- \sum_{x} \tilde{p}(x) \sum_{y} p_\Lambda(y|x) f_i(x,y) \exp(C\,\delta_i) \quad (20)$$

The effect of introducing C rather than $f^\#$ is that solving $\frac{\partial \mathcal{B}(\Delta|\Lambda)}{\partial \delta_i} = 0$ can be done analytically (at the cost of a slower convergence rate), giving the following:

$$\begin{aligned} \delta_i &= \frac{1}{C} \log \frac{\sum_{x,y} \tilde{p}(x,y) f_i(x,y)}{\sum_x \tilde{p}(x) \sum_y p_\Lambda(y|x) f_i(x,y)} \\ &= \frac{1}{C} \log \frac{E_{\tilde{p}} f_i}{E_{p^{(j)}} f_i} \end{aligned} \quad (21)$$

which leads to the update rule in (7).