Maximum Entropy Methods for Biological Sequence Modeling

Eugen C. Buehler[*]
CIS, University of Pennsylvania
Philadelphia, PA 19104
buehler@gradient.cis.upenn.edu

Lyle H. Ungar
CIS, University of Pennsylvania
Philadelphia, PA 19104
ungar@cis.upenn.edu

ABSTRACT

Many of the same modeling methods used in natural languages, specifically Markov models and HMM's, have also been applied to biological sequence analysis. In recent years, natural language models have been improved upon by using maximum entropy methods which allow information based upon the entire history of a sequence to be considered. This is in contrast to the Markov models, whose predictions generally are based on some fixed number of previous emissions, that have been the standard for most biological sequence models. To test the utility of Maximum Entropy modeling for biological sequence analysis, we used these methods to model amino acid sequences. Our results show that there is significant long-distance information in amino acid sequences and suggests that maximum entropy techniques may be beneficial for a range of biological sequence analysis problems.

Keywords

maximum entropy, amino acids, sequence analysis

1. INTRODUCTION

Comparisons between biological sequence data and natural languages are commonplace, in part because these comparisons help to frame our understanding of nucleic and peptide sequences within a topic, language, of which most of us have an intuitive sense. So when newspapers describe biology to the masses, genomes become books, chromosomes become chapters, genes become sentences, and so on. These comparisons are more than useful metaphors for conceptualization. Many of the statistical models that have proven successful for language have been adapted for use in analyzing biological sequences. For example, Markov models and Hidden Markov Models are popular tools both in natural language processing [9] and in computational biology [8], where they have been used for gene finding [3] and profiling protein domains and families [11].

In spite of the popularity of Markov models, it has become increasingly popular in natural language processing to use models that deviate from the Markov assumption, that is that the next member of a sequence will be dependent on a short, fixed length of prior emissions. Rosenfeld [18] showed recently that he could significantly reduce the perplexity of a probability model of the next word in a document by using "triggers", words that cause other words to occur at some later, unspecified distance in the document. So for example, seeing the word "bank" increases the likelihood of the word "bank" occurring again later in the document, as well as related words such as "check" and "loan".

To incorporate "long distance" features such as triggers into his model, Rosenfeld used Maximum Entropy (ME) modeling. A major benefit of the Maximum Entropy modeling technique is that a diverse range of features can be incorporated into a single probability model, without any requirement of independence among the features. Rosenfeld compared these models to interpolated Markov models, where several models are trained separately and then combined, and concluded that Maximum Entropy models are superior in their performance. The ability of ME Models to incorporate diverse, sometimes overlapping features into a single probability model has inspired their adoption for other tasks in natural languages, such as translation [2], document classification [12], identifying sentence boundaries [17], prepositional phrase attachment [16], and part-of-speech tagging [14].

Given the rising popularity of ME models in natural languages, we might expect them to be applicable to biological sequence models as well. As a first test case of applying Maximum Entropy techniques to biological sequences, we built a model for amino acid sequences. We chose amino acid sequences in part because they require a less complex model than eukaryotic genes (with exons, introns, promoters, and other features to consider). While not requiring the complexity of a eukaryotic gene model, amino acid sequences (or peptide sequences) are good candidates for the utility of "long distance" information, since the amino acids within a protein interact over variable distances due to the bending of the protein into its secondary and tertiary structure. Finally, it has been suggested that better statistical models of amino acid sequences may lead to better gene prediction models [19].

1.1 Theoretical Basis for Maximum Entropy

One of the most attractive aspects of Maximum Entropy is its simplicity. It specifies that an ideal model should match what we know about the empirical distribution we are attempting to model without making any unwarranted assumptions. Phrased differently, the technique involves first constraining the modeled distribution to match certain fea-

[*]Corresponding Author

tures of the empirical data (training data), and then choosing the probability function that satisfies these constraints while being as uniform as possible.

We begin by selecting a set of n features that are salient to the problem at hand. We will limit ourselves to real valued features (functions) that evaluate to between zero and one inclusive: $f_i : \varepsilon \rightarrow [0, 1]$. A special case of this is binary features which are either present or absent within a given event or instance: $f_i : \varepsilon \rightarrow \{0, 1\}$. Our goal is to end with a probability model $\hat{p}$ that, given an event, can estimate its probability based on its features. A reasonable way to constrain our model is to require it to have the same expectation value for each feature as the observed probability $\tilde{p}$ in the training data, T, which is our approximation of the real (unobservable) distribution p. That is, for each of the n features indexed by i, $1 < i < n$:

$$\sum_{x \in \varepsilon} \hat{p}(x) f_i(x) = \sum_{x \in T} \tilde{p}(x) f_i(x) = \frac{1}{|T|} \sum_{x \in T} f_i(x) \qquad (1)$$

where $|T|$ is the number of examples in our training set. The set of constraints will not, in general, specify a unique probability distribution. Rather, there will be a set of distributions that satisfy these constraints, from which we must choose one. We will choose the distribution that has maximum entropy, where entropy of $\hat{p}$ is defined by

$$H(\hat{p}) = -\sum_{x \in \varepsilon} \hat{p}(x) \log \hat{p}(x) \qquad (2)$$

This can also be viewed as the information content of the distribution, and thus is a measurement of what we do not know about the world. The smaller $H(p)$ is, the more we know. Maximizing this function is then comparable to assuming as little as possible about the world.

It can be shown that there is always a unique, maximum entropy distribution that satisfies the constraints imposed by our training set and choice of features, and that this probability distribution will always have the exponential form:

$$\hat{p}(x) = \pi \prod_{i=1}^{n} \alpha_i^{f_i(x)} \qquad (3)$$

where π is simply a normalization constant to guarantee that $\sum_{x \in \varepsilon} \hat{p}(x) = 1$ [15]. Each α_i can be viewed as the weight of the feature f_i in the model. A very large α_i corresponds to a more frequent feature, whereas a very small α_i corresponds to a feature that occurs rarely. Note that both can be useful information, although some models focus exclusively on frequent features because of the size of the feature space.

There is a duality to the Maximum Entropy approach that is appealing. It can be shown that finding the probability distribution that satisfies the chosen feature constraints while maximizing entropy is equivalent to finding the exponential distribution that maximizes the likelihood of the training data [15]. Thus, while our ME model is the most uniform distribution that will satisfy our constraints, it is also the most likely exponential distribution with respect to our data.

1.2 Estimating Parameters

Since the Maximum Entropy distribution must have the exponential form detailed above, once features have been selected the task of modeling is reduced to finding the parameter (α) values that satisfy the constraints generated by the selected features and the training data. As a rule, parameter values cannot be solved for directly and must instead be estimated by an iterative, hill-climbing method. Two such methods are available: Generalized Iterative Scaling [7] and Improved Iterative Scaling [13].

Both GIS and IIS begin with all α_i initialized to one, unless a previous solution for a similar set of features is available as a starting point. Both algorithms then update the α_i's iteratively, each iteration resulting in a new probability distribution that is closer to satisfying the constraints imposed by the expectation values of the selected features. GIS requires that for any event the sum of features will be a constant. This requirement can be met for any set of features by introducing a dummy feature such that:

$$F_{dummy}(x) = C - \sum_{i=1}^{n} F_i(x) \qquad (4)$$

The constant C is chosen to be equal to the maximum value for the sum of features 1 though n. IIS differs from GIS in that it does not require model features to sum to a constant, and the IIS algorithm often runs slightly faster than GIS. Since IIS and GIS result in equivalent models, we chose to use GIS for our parameter estimation because it is slightly easier to code.

2. METHODS

2.1 Model

Our goal is to model the probability distribution $p(a \mid h)$, the probability that a will be the next amino acid, given the history h of amino acids that have already occurred. This probability will be calculated from the probability of the next amino acid and the history $p(a, h)$, normalized over all possible selections for the next amino acid.

$$\hat{p}(a \mid h) = \frac{\hat{p}(a, h)}{\sum_x \hat{p}(x, h)} \qquad (5)$$

In practice, the event space (a, h) is too large to practically calculate the expectation value of a feature in $\hat{p}$, so we estimate it using the training set [18].

A standard method used to measure the success of a probability model $\hat{p}$ in modeling a probability distribution p is the cross-entropy of the two distributions. The cross-entropy of two probability distributions for sequences S with tokens a taken from a language L is:

$$H(p, \hat{p}) = \lim_{n \to \infty} \frac{1}{n} \sum_{S \in L, |S| = n} p(a_1, \ldots, a_n) \log \hat{p}(a_1, \ldots, a_n) \qquad (6)$$

This cross-entropy gives an upper-bound for the actual entropy of the "real" distribution. If we compare the cross-entropy of two different models (that is, comparing $H(p, \hat{p}_1)$ to $H(p, \hat{p}_2)$), the more accurate model will have a lower cross-entropy. By making certain simplifying assumptions about the modeled probability distribution [10], specifically that it is ergodic and stationary, we can assume that a long enough sequence from p will be representative of its distribution, rather than summing over all possible sequences. Making this assumption, we can write:

$$H(p, \hat{p}) = \lim_{n \to \infty} -\frac{1}{n} \log \hat{p}(a_1, \ldots, a_n) \approx \frac{1}{|T|} \log \hat{p}\left(a_1, \ldots, a_{|T|}\right) \quad (7)$$

where T is our test set of amino acid sequences.

We use the perplexity (actually cross-perplexity) of the models created with test data as our measure of success. Cross-Perplexity, which is equal to two to the cross-entropy, is an intuitively pleasing measure of how well we have modeled a probability distribution. The upper-bound perplexity per amino acid of this system is 20, corresponding to the 20 possible amino acids. Thus, if the cross-entropy of our model with a set of test data is 19, we have increased our knowledge of amino acid sequences such that we could encode protein sequences with 19 symbols instead of 20 and still use one symbol (on average) to encode each amino acid.

2.2 Features

Models were created with various sets of features, drawing upon both short distance (unigrams and bigrams) and long distance (triggers and caches) information. Below, each of the feature sets is described and then expressed as a set of functions. Subscripts in our feature functions serve to identify the type of function (u for unigram), and superscripts provide information specific to that type of feature (the type of unigram). For example, $f_u^c(a, h)$ represents the feature function corresponding to the unigram frequency of the amino acid C (Cysteine).

Conventional Unigrams and Bigrams The frequency of amino acid x is:

$$f_u^x(a, h) = \begin{cases} 1 & \text{if } a = x \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

There are 20 unigram features, one for each amino acid.

The frequency of amino acid pair xy is:

$$f_b^{x,y}(a, h) = \begin{cases} 1 & \text{if } a = y \text{ and } h \text{ ends in } x \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

There are four hundred possible bigram features.

Unigram Cache The observed frequency of the next amino acid in the history.

$$f_{uc}(a, h) = \frac{N(a, h)}{|h|} \quad (10)$$

This is a single feature which is not specific to a given amino acid. $N(a, h)$ refers to the number of occurrence of a in h.

Self-Triggers If the next amino acid is x, the frequency of x in the history, and zero otherwise. There are 20 possible self-triggers, one for each amino acid.

$$f_t^x(a, h) = \begin{cases} \frac{N(x,h)}{|h|} & \text{if } a = x \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

There are 20 possible self-triggers, one for each amino acid. Note that these triggers are different from the binary valued triggers used in some natural language applications [18], in which a single occurrence of a word "triggers" its occurrence later in the document. Because there are only 20 amino acids, we are more likely to have a useful metric if we consider the number of times an amino acid has occurred in the history rather than whether it has occurred at all.

Class-Based Self-Triggers If the next amino acid belongs to class x, the frequency of class x in the history, and zero otherwise.

$$f_{ct}^x(a, h) = \begin{cases} \frac{N(x,h)}{|h|} & \text{if } a \in x \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

Amino acids are partitioned into one of five possible classes: Non-polar aliphatic (AVLIP), aromatic (FYW), polar uncharged (STCMNQ), positively charged (KRH), and negatively charged (DE).

2.3 Training and Test Sets

Our training and test sets were derived from a set of genes from Burset et al. [5] that have become a standard benchmark for gene-finding software. Their set included 571 proteins, some of which were highly homologous. All 571 proteins were compared to the other proteins in the set using BLAST [1], after which the set was screened so that no two proteins with any significant homology would be included together in the screened set. The resulting set of 204 non-homologous proteins was randomly split into a training set of 102 proteins with 30,438 amino acids and a test set of 102 proteins with 29,158 amino acids.

3. RESULTS AND DISCUSSION

Perplexity measures for models trained with various combinations of features are summarized in Figure 1. Note that the addition of the 400 possible bigram features to the unigram model actually increases test set perplexity because of over-fitting. In contrast to natural languages, where bigrams are very useful information sources, the previous amino acid tells us little or nothing about what amino acid will follow it. While knowledge of the previous amino acid is uninformative, there is significant information available in the entire amino acid sequence, as witnessed by the 30% decrease in perplexity for our best model over the perplexity decrease of the unigram model alone.

As we might expect, individual self-triggers for each amino acid performed better than the Unigram cache, which attempts to encode the same information in one parameter instead of twenty. By examining the calculated parameters for the amino acid self-triggers, we can see that the presence of some amino acids in the history carry more information than others. For example, the Cysteine self-trigger had the largest parameter associated with it, which is in keeping with the biological literature that certain proteins are "Cysteine rich".

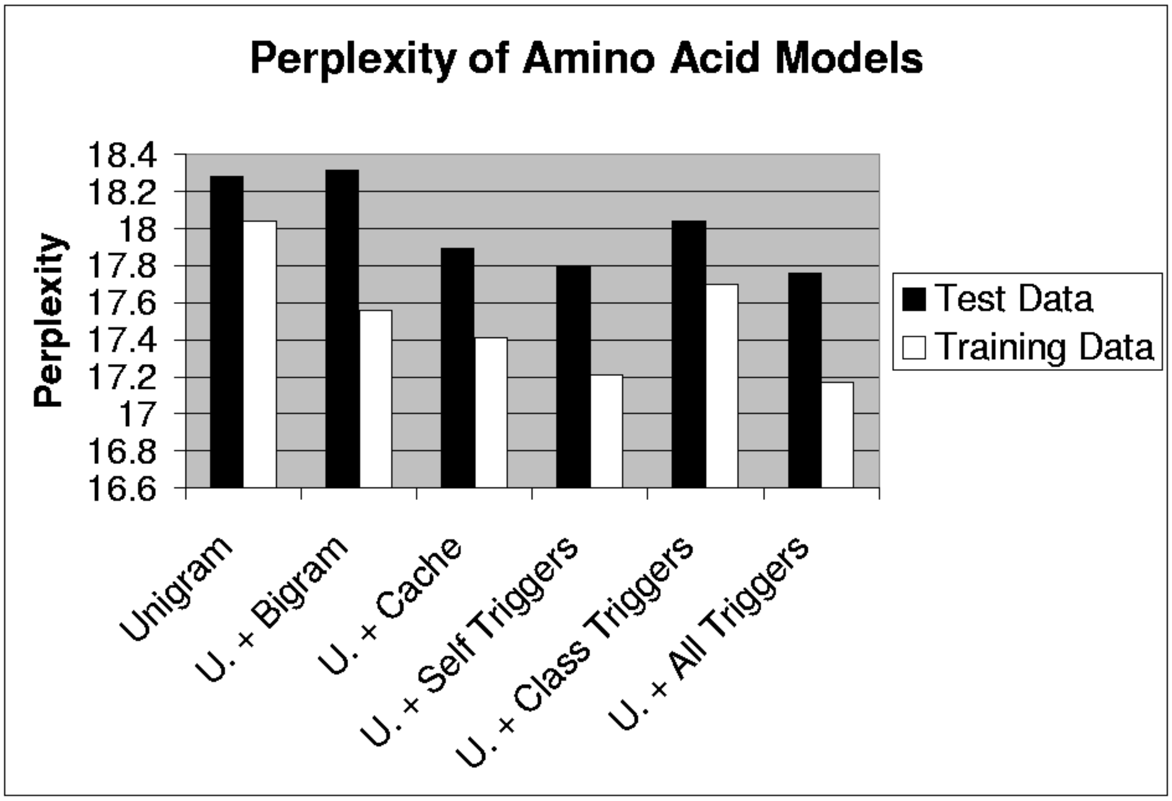

Figure 1: Training and test set perplexity for ME Models using the following feature sets: Unigram, Unigram and Bigram, Unigram and Unigram Cache, Unigram and Self-Triggers, Unigram and Class-Based Self-Triggers, and Unigram and All Triggers (Normal and Class-Based).

Our best model resulted from using both self-triggers and class-based self-triggers combined with unigrams, which performed slightly better on the test set than the unigram and self-triggers model. Thus, the knowledge of category frequency, for example that many previous amino acids have been positively charged, carries information not available in the frequency of each amino acid alone.

To test if our model might improve existing gene prediction techniques, Genscan [4] was run on the Burset benchmark set of genes. We produced a set of 169 predicted exons with low Genscan scores (less than 0.5), and grouped them into categories of correct if the exon was predicted exactly correct, and incorrect if the predicted exon did not overlap with any known coding region. We did not consider partially correct exons. Scores were then assigned to each exon based on the log of the ratio of the likelihood of the sequence given our best model to that of a baseline model, which we chose to be the unigram model.

$$Score = \log\left(\frac{P\left(protein \mid BestModel\right)}{P\left(protein \mid UnigramModel\right)}\right) \tag{13}$$

A logistic regression was performed using our score and the Genscan assigned probability to predict the category of each exon (correct or incorrect). Both the Genscan assigned probability and our score were statistically significant predictors of exon correctness, with null hypothesis probabilities of 0.0016 for Genscan's probability and 0.019 for our score. We therefore conclude that our model contains information useful for exon prediction not already present in Genscan's probability assignment.

4. CONCLUSION

We have shown that the use of “long distance” features can improve a statistical model of amino acid sequences. Even better results may be possible by including other features that are calculated across the history of the protein, for example the net charge of the encoded protein. It might also be possible to improve our model by weighting distance features to rely more heavily on the recent history, since many proteins have multiple “domains” with different functional roles and thus different compositions. Improvement may also be possible by smoothing either the entire model or features of the model, as has been attempted in some natural language ME models. [6]

Our results strongly suggest that this type of model could improve existing gene finding programs. A logical next step would be to build a nucleotide coding model that accounts both for codon preference and for the long-range patterns expected of the encoded protein. Maximum Entropy is amenable to this kind of joint model, since there is no problem with dependent feature sets such as amino acid frequency and hexamer nucleotide frequency.

Finally, there are a host of open biological sequence analysis problems that may be amenable to ME modeling. Promoter analysis is made difficult by the need to use contextual clues to identify functional transcriptional elements from random consensus matches. This problem is not unlike that of using contextual clues for language translation, to which ME modeling techniques have already been applied. Similarly, protein secondary structure prediction is roughly analogous to part-of-speech tagging, for which ME models have also been built. Because of the similarity of language and biological sequences, it may behoove computational biologists to continue borrowing the techniques that prove successful for natural language.

5. ACKNOWLEDGEMENTS

This work was supported by a training grant in computational genomics from the National Institutes of Health (1-T32-HG-00046).

6. REFERENCES

[1] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[2] A. Berger, S. A. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March 1996.

[3] M. Borodovsky and J. McIninch. Genemark: Parallel gene recognition for both DNA strands. *Computers & Chemistry*, 17:123–133, 1993.

[4] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78–94, April 1997.

[5] M. Burset and R. Guigo. Evaluation of gene structure prediction programs. *Genomics*, 34(3):353–367, June 1996.

[6] S. F. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Carnegie Mellon University, Pittsburgh, PA 15213, February 1999.

[7] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Statistics*, 43:1470–1480, 1972.

[8] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids.* Cambridge University Press, 1998.

[9] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall, 2000.

[10] D. Jurafsky and J. H. Martin. *Speech and Language Processing.* Prentice Hall, Upper Saddle River, NJ 07458, 2000.

[11] A. Krogh, M. Brown, I. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology. applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–31, Feb 1994.

[12] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.

[13] S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, April 1997.

[14] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey, 1996.

[15] A. Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. Technical Report 97–08, IRCS, University of Pennsylvania, 3401 Walnut Street, Suite 400A, Philadelphia, PA 19104-6228, May 1997.

[16] A. Ratnaparkhi and S. Roukos. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the ARPA Workshop on Human Language Technology*, 1994.

[17] J. Reynar and A. Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, 1997.

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

[19] E. C. Thayer, C. Bystroff, and D. Baker. Detection of protein coding sequences using a mixture model for local protein amino acid sequence. *Journal of Computational Biology*, 7(1/2):317–327, 2000.