Weighting hidden Markov models for maximum discrimination

Rachel Karchin and Richard Hughey

Department of Computer Engineering, Jack Baskin School of Engineering, University of California, Santa Cruz, CA 95064, USA

Received on January 23, 1998; revised on March 16, 1998; accepted on March 19, 1998

Abstract

Motivation: Hidden Markov models can efficiently and automatically build statistical representations of related sequences. Unfortunately, training sets are frequently biased toward one subgroup of sequences, leading to an insufficiently general model. This work evaluates sequence weighting methods based on the maximum-discrimination idea.

Results: One good method scales sequence weights by an exponential that ranges between 0.1 for the best scoring sequence and 1.0 for the worst. Experiments with a curated data set show that while training with one or two sequences performed worse than single-sequence Probabilistic Smith–Waterman, training with five or ten sequences reduced errors by 20% and 51%, respectively. This new version of the SAM HMM suite outperforms HMMer (17% reduction over PSW for 10 training sequences), Meta-MEME (28% reduction), and unweighted SAM (31% reduction).

Availability: A WWW server, as well as information on obtaining the Sequence Alignment and Modeling (SAM) software suite and additional data from this work, can be found at http://www.cse.ucsc.edu/research/compbio/sam.html

Contact: rph@cse.ucsc.edu

Introduction

Linear hidden Markov models (HMMS) (Haussler et al., 1993; Krogh et al., 1994) or generalized profiles (Bucher and Bairoch, 1994) have been demonstrated to be very effective in detecting conserved patterns in multiple sequences (Hughey and Krogh, 1996; Baldi et al., 1994; Eddy et al., 1995; Eddy, 1995; Bucher et al., 1996; McClure et al., 1996; Karplus et al., 1997). When provided with a set of unaligned or aligned ‘training’ sequences that contain members of a protein family, the resulting HMM can identify the positions of amino acids which describe conserved first-order structure of the family.

The HMM can also be used to generate a multiple alignment of the unaligned sequences from which the model was built by using a dynamic programming method. The new multiple alignment reveals the regions in the trained sequences that the process found to be conserved and can provide new information about the common structure underlying the sequences in the family. In addition, the HMM can discriminate between family and non-family members in a sequence database search.

This paper studies several sequence weighting methods that improve the ability of an HMM to discriminate between members and non-members of a protein family. Sequence weighting addresses a potential weakness in the model building process. If there is a high degree of redundancy in the set of sequences being used, the model will over-represent the similar sequences. By assigning larger weights to poorly scoring sequences during model training we have been able to significantly reduce the number of errors made when the resulting model is used to recognize members of a particular family.

The methods we have considered are based entirely on the log-odds score (Altschul, 1991) of the sequence against the model being trained. This investigation was motivated by HMMer’s maximum-discrimination training method (Eddy et al., 1995; Eddy, 1995). HMMer uses a simulated annealing approach, in which the number of sequences actively in the training set is reduced over the course of training. During the course of this training, when maximum discrimination training is used, sequence weights are set according to their log-odds score so that sequences that poorly match the model are given higher weight. As training iterations proceed, the model is modified with data from each new sequence as it is scored against the model.

SAM uses expectation-maximization and its batch mode of training (Krogh et al., 1994; Hughey and Krogh, 1996). In this case, the entire training set is aligned to the model, and a completely new model is generated by combining the information from these alignments and the transition regularizer and a Dirichlet mixture prior (Sjölander et al., 1996). (SAM uses the full forward–backward algorithm, so more precisely, the sum of probabilities across all possible alignments is calculated for each sequence.) SAM can produce a model from unaligned sequences, as was done in these experiments, or SAM can be given an initial seed alignment, which often improves results.

Since it has been known for some time that weighting is a problem in SAM, we added the ability to weight sequences
to the new version of SAM. This came in two phases. The first is external weighting, in which a file of weights can be read in and assigned to sequences. Our group currently uses this and either position-specific weighting (Henikoff and Henikoff, 1994) or an entropy-based method (Karplus, personal communication), using an unpublished method to assign the total weight (without regularization, only relative weight matters; with regularization, required for small sample sizes, total weight will influence how strongly the sequences can overpower the regularizer (Karplus, 1995)). Both of these methods are based on analyzing a multiple alignment of the training sequences. When one is not available, a rough alignment is created using unweighted HMM training, the sequences are weighted based on that alignment, and then a new HMM is trained using those weights.

In an effort to add standalone weighting (i.e. without the extra steps for the file-based weighting), we examined a number of maximum-discrimination style weighting schemes, that is schemes that depend only on the log-odds score of the sequence against the model and in which weights are continuously updated during HMM training. This analysis has led us to make two of the weighting methods, described below, permanent additions to the Sequence Alignment and Modeling software system.

On completing our evaluation of these different weighting schemes, we performed a more general analysis of our best weighting schemes based on the recent work of Agarwal and States (1998). Using 67 superfamilies from an annotated version of PIR (Pearson, 1995), they compared several pairwise sequence comparison methods, leading to the observation that Probabilistic Smith–Waterman (PSW) (Bucher and Hoffmann, 1996) was the best discrimination technique tested for whole proteins. To their PSW results, we add evaluations of SAM’s new internal weighting, HMMer’s maximum discrimination training, and Meta-MEME’s combination of MEME motif location with HMMer hidden Markov model scoring (Bailey and Elkan, 1994; Grundy et al., 1997; Grundy, 1998).

**Systems and methods**

All work described in this paper was performed using the SAM HMM program suite version 2.0 on DEC Alpha 255 workstations and a DEC AlphaServer. SAM is programmed in ANSI C for Unix workstations. The HMMer experiments were performed with HMMer version 1.8.1. The Meta-MEME experiments used Meme and Mast version 2.0 and Meta-MEME version 1.0.

A typical training time for 10 sequences, such as with the cytochrome P450 superfamily with average length 498, was 10.7 min for SAM, 2.5 min for HMMer, and 4.2 min for MetaMeme. Search times against the reference database were 19.9 min for SAM, 12.0 min for HMMer, and 0.4 min for MetaMeme.

**Algorithm**

**Hidden Markov models**

Given a linear hidden Markov model $M$, a dynamic programming calculation can be used to calculate, for a given sequence, the probability that sequence $a$ was generated by the model, $P(a|M)$. The question of interest is, however, does that model match the sequence? That is, is the sequence more likely to be generated by the hidden Markov model than some other, less structured null model, $\phi$. Making the assumption that the models $M$ and $\phi$ are a priori equally likely, this reduces to the log-odds probability of

$$
P(M|a) = \frac{P(a|M)}{P(a|\phi)}.
$$

Typically, a log-odds score $S$ is used instead (Altschul, 1991):

$$
S_a = \ln \frac{P(a|M)}{P(a|\phi)} = \frac{1}{1 + e^{-S}}.
$$

This score measures whether the probability that the sequence was generated by the model is greater than the probability it was generated by a null model (in SAM’s case, the geometric average of the modeled sequence positions (Barrett et al., 1997)). This log-odds score is used to calculate the weight of a sequence. (SAM actually uses negative log-odds, but for this discussion, hopefully consistently, we use positive log-odds scores.)

HMMer’s maximum discrimination algorithm gives to each sequence a weight of $1 - P(M|a)$. Thus, if a sequence perfectly matches the model $M$, it is given no weight at all, while a sequence that poorly matches the model is given unity weight. In score terms, weights are set to $1/(1 + e^S)$. One potential problem of this score function is that a model could, for example, alternately score one group of sequences well and a second poorly and then, because of the weighting, score the second group well after retraining. The HMMer maximum discrimination algorithm deals with this in two ways. First, training is done serially; the model is updated after each sequence is scored against the model. Second, a mixing parameter is used to ensure that any given sequence does not change the model too much.

**Weighting methods**

For SAM, we considered several different weighting functions, all loosely based on the maximum discrimination equation above. We had four criteria in selecting candidate weighting functions:
1. In general, poorly scoring sequences should receive larger weights than well scoring sequences.
2. Weights should not get too large.
3. Weighting should not produce excessive iterations when a model is built.
4. The weighted model should have an improved ability to recognize sequences that belong to a particular protein family.

To achieve these ends, we designed eight ‘candidate’ functions described below. We also included weight normalization and weight ‘mixing’ capabilities to optimize our results.

The methods are numbered in roughly chronological order, perhaps revealing some of our thought processes in creating the experiments (Table 1).

**Table 1. Notation**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>A sequence’s score</td>
</tr>
<tr>
<td>S</td>
<td>Weight of the best scoring sequence (user defined)</td>
</tr>
<tr>
<td>K</td>
<td>The best (lowest) score in the training set</td>
</tr>
<tr>
<td>b</td>
<td>The worst (highest) score in the training set</td>
</tr>
<tr>
<td>m</td>
<td>The median score in the training set</td>
</tr>
<tr>
<td>o</td>
<td>User-defined offset value</td>
</tr>
<tr>
<td>W_S</td>
<td>Weight of an individual sequence</td>
</tr>
<tr>
<td>N</td>
<td>Size of the training set</td>
</tr>
<tr>
<td>x</td>
<td>Mix of old weight and new weight</td>
</tr>
</tbody>
</table>

The first weighting function is directly from HMMer’s maximum discrimination method, but with additional parameters. Here, the weighting can be adjusted using an offset parameter \( o \) that is not necessarily zero. Any sequence with score better than \( o \) has reduced weight, while any sequence that scores worse than \( o \) has increased weight. We typically set \( o \) to the log of the database size, following the ideas of algorithmic significance (Milosavljevic and Jurka, 1993).

Thus, this offset encourages the model to score sequences not just better than the null model, but significantly better than the null model. The Method 1 formula has a second parameter, \( 0 < K < 1 \), that is used to temper the effects of weighting on the sequences:

\[
W = 1 - \frac{K}{1 + e^{(o-S)}}.
\]  

The second weighting function is the first of our relative weighting functions which attempt to remove the offset parameter to reduce the number of free variables. The worst score \( w \) and the best score \( b \) are used to decide on two extreme weights. The worst scoring sequence will have a weight of 1, while the best scoring sequence will have a weight of \( K \), typically in the 0.01 to 0.1 range:

\[
W = e^{(w - b) \frac{\log K}{b - w}}. 
\]  

Method 3 is again similar to the maximum discrimination formula, but instead of using an arbitrary offset, the median of all scores in the training set is used. Thus, sequences that score worse than the median \( m \) receive a weight close to unity, while those that score better than the median receive a small weight:

\[
W = 1 - \frac{K}{e^{(o-m)}}. 
\]  

Method 4 is basically Method 2 with an offset used to scale the scores, rather than the worst score. Here, sequences that score better than the offset receive weights less than one, while sequences that score worse than the offset can receive very high weights:

\[
W = e^{(S-o) \frac{\log K}{b-o}}. 
\]  

Continuing with variations on a theme, Method 5 uses the median and the best score to determine the exponential weight curve:

\[
W = e^{(S-m) \frac{\log K}{b-m}}. 
\]  

One problem with Methods 2, 4 and 5 is that the scaling depends very strongly on the worst score. This can be particularly a problem if the training set is polluted: e.g. if a non-family member is included in the training set. In an effort to deal with these possible outliers, Method 6 is a piecewise weighting function. The primary weighting function,

\[
W = e^{(S-m) \frac{\log K}{b-m}},
\]  

is applied to scores that are at most 3 sample deviations below the mean scores, while the poorest scoring sequences receive a weight that is linear in terms of their score, allowing for the fact that they may not really be family members:

\[
W = 1 - \frac{S - w}{b - w}. 
\]  

Method 7 is a similar function based on the best and worst score, rather than the median and worst score. Here, the equation

\[
W = e^{(w - m) \frac{\log K}{b - m}}
\]  

is applied to scores no more than 3 deviations below the median, while

\[
W = 1 - \frac{S - w}{b - w}
\]  

is applied to the remaining sequences.

Finally, Method 8 is a simple linear function,
\[ W = K + \frac{b - S}{b - w}. \]

### Weight normalization and mixing

Whenever the total weight of all the sequences exceeds the number of sequences (it is unlikely that \( N \) sequences could ever contain more than \( N \) sequences worth of information!), the weights are renormalized to sum to \( N \). To deal with the opposite case of the sequence weights becoming too small (below \( \ln N \)), the weights are renormalized as well.

\[
\text{total} > N \Rightarrow W_a = \frac{W_a}{\text{total}} \cdot N \\
\text{total} < \log N \Rightarrow W_a = \frac{W_a}{\text{total}} \cdot \log N
\]

To potentially account for weights varying too rapidly between training iterations, we experimented with an annealing schedule similar to that used by SAM for noise injection. This weight mixing function decides how much influence the calculated \( W_{\text{new}} \) can have on the previous iteration’s \( W_{\text{old}} \):

\[
W = (1 - x)W_{\text{old}} + x \cdot W_{\text{new}}
\]

where \( x \) is the ‘weight mix’ parameter that is decreased linearly from one to zero over the training iterations depending on a factor, either default or specified by the user.

### Method evaluations

Following previous work (Barrett et al., 1997), we initially tested these eight methods on three protein families: globins, ferredoxins, and EF-hands. Models were generated from large training sets of known family members using SAM’s buildmodel program. We would expect such models to be quite good at discriminating between family and non-family members.

For each family, we built five models using each prospective weighting method and a variety of parameters. We wanted to compare their discrimination performance with unweighted SAM models.

We tested discrimination performance by using SAM’s hmm.score program with each model to calculate the log-odds scores of sequences in two discrimination sets. The ‘true’ set is composed of members of the given protein family; the ‘false’ set is composed of non-family members.

The true sets were assembled by using a Prosite keyword search on the SwissProt 32 database (685 globins, 100 ferredoxins, 147 EF-hands). The false sets consisted of all remaining sequences in the SwissProt 32 database after extraction of the true set. Although our true and false sets undoubtedly had errors, the same errors were present for all methods.

We then plotted the resulting number of false positives versus number of false negatives over a range of score thresholds. False positives are non-family sequences which the model identifies as family members; that is, their log-odds score is better than the worst scoring family member. False negatives are family members which the model misses. They score worse than the best-scoring non-family members.

The point where the number of false positives equals the number of false negatives is called the equivalence number (Pearson, 1995; Agarwal and States, 1998). We calculated the equivalence number \( (EN) \) for each of our test and control models. This number is a good standard of discrimination performance. It provides an upper bound on the minimum number of discrimination errors \( (MER) \)

\[
MER < = 2 \cdot EN
\]

and it balances model sensitivity and selectivity; false negatives occur when a model is not sensitive enough and false positives occur when a model is not selective enough.

To evaluate computational performance, we kept track of the number of iterations required to build each model and the range of weights assigned to sequences in each experiment.

Four of our methods had poor computational performance (1, 3, 5, 6). We eliminated methods and parameter settings that made more discrimination errors than an unweighted model, added excessive iterations to the model building process, or were numerically unstable. However, four of the methods (2, 4, 7, 8) performed very well with \( K \) (weight of the best scoring sequence before normalization) set at 0.1, so we continued our study using these.

At this point, we also revised the training procedure with the use of a different Dirichlet mixture prior (mall-opt.9comp) and an improved local (Smith–Waterman style) scoring algorithm (Smith and Waterman, 1981; Tarnas and Hughey, 1998). All other parameters, apart from weighting parameters, were defaults.

Further tests with size 15 training sets showed that for globins and ferredoxins, methods 2 and 4 had the best discrimination performance. For EF-hands, position specific weighting and method 7 performed better than methods 2 and 4. Computationally method 4 performed the worst, requiring two and three times as many model building iterations as the other internal methods. We restricted further experiments to method 2, method 7, unweighted, external entropy, and external position-specific (Henikoff and Henikoff, 1994). As mentioned, the two external methods consist of training an unweighted SAM model, using that model to form a multiple alignment, and then passing the multiple alignment to the weighting algorithm.

### Training set size and discrimination errors

Our next concern was to see if training set size made a difference in a model’s discrimination accuracy. To speed method evaluation, we created smaller discrimination sets by training an unweighted SAM model on each entire family (Swiss-
Fig. 1. Globins family. Comparison of weighted and unweighted models’ average equivalence numbers by training set size. Weight mixing used.

Fig. 2. Ferredoxins family. Comparison of weighted and unweighted models’ average equivalence numbers by training set size. Weight mixing used.

Fig. 3. EF-hands family. Comparison of weighted and unweighted models’ average equivalence numbers by training set size. Weight mixing used.

sProt keyword extraction) and then scoring this model on the entire SwissProt32 database. The true set was created by selecting the 100 worst scoring true sequences from each family and the false set was created by selecting the 2000 best scoring false. Thus, these sets included the sequences that were difficult for the unweighted HMM to categorize.

We studied training set sizes of 2, 5, 10, 15, 30, 50, 65, 85 and 100. For each size, we randomly formed five different training sets of each size and calculated the minimum, maximum and average equivalence numbers of the models created with each method.

While running the experiments on small training sets, we observed that sequence scores were oscillating indefinitely, a known problem with maximum discrimination score functions.

When weight mixing was added, the oscillations were successfully damped. We tried several $x$ parameters in equation (10) and discovered that equivalence numbers were minimized when $x = 0.8$; as a result, we have included weight mixing with parameter 0.8 as the default when SAM internal weighting is used.

Figures 1, 2 and 3 show the average equivalence number of each weighting method, as training set size increases from 2 to 100, by its difference from the average equivalence number of an unweighted SAM model calculated over the same five training sets.

This lets us see clearly how the weighting method improves on the discrimination performance of an unweighted model. The improvement can be measured by decreases in average equivalence numbers from unweighted to weighted models.

For the globins family, SAM methods 2 and 7 clearly have the best overall performance.

For ferredoxins, SAM methods 2 and 7 do best overall, however entropy weighting does an outstanding job for size 50 training sets and position specific weighting is best by a small margin for size 100 training sets.

For EF-hands, position specific weighting has the best overall performance and its margin of superiority increases as training sets get larger. Method 2 also does well. For size 85 sets it does better than position specific, however at size 100 it does poorly. For very small training sets (less than 15
Fig. 4. Average and range of equivalence numbers for each size group of ferredoxin training sets for several weighting methods. The internal weighting methods 2 and 7 perform somewhat better than the unweighted training, especially on small training sets. The off-line position-specific and entropy-based weighting produce a wide range of results.

Range of errors made by different methods

In addition to the average equivalence numbers, we looked at the ranges between minimum and maximum equivalence numbers for training sets of the same size.

We observed that position specific and entropy weighting methods had more unstable results than the SAM internal
R.Karchin and R.Hughey

methods. For training sets where these methods have excellent average equivalence numbers, the actual range of equivalence numbers is frequently quite large. This variation could be due to sensitivity to the single alignment (from an HMM trained with unweighted sequences) from which the external weights are calculated. Equivalence number ranges for the ferredoxins family are shown in Figure 4.

All methods tested were somewhat unstable on the EF-hands family; external position specific weighting was the most unstable.

Dynamic behavior of weights

One great advantage of internal weighting methods is that the weights are dynamically changed as the model is trained. Training 100 ferredoxins (Figure 5), when the sequences are initially compared against the untrained model in iteration 1, there are two outliers that receive particularly high weights. In the second iteration, the sequences are much more clustered. Because of the weight mixing, however, the two outliers’ weights are only partly reduced to that of the typical sequence receiving the same score. After 10 iterations, weights are nearly fixed for the remainder of the training process. At the last iteration, 26 in this case, the best scoring sequence is one that during one of the initial iterations was hard to model, and thus received a higher weight.

Additional protein families

To find out whether or not our methods had value beyond these three protein families and to compare our discrimination results with other methods in general use, we ran another

Fig. 5. Weight versus score snapshots during the training process of 100 ferredoxins. The score range changes between plots as the model specializes. Training iterations 1, 2, 10 and 26 (the last) are shown.
Table 2. Comparison of Agarwal and States MER of Best Method to SAM MER of Best Method. Includes family number, family name, length of single-sequence query, number of sequences in the family, and the training set size used to get SAM best results.

<table>
<thead>
<tr>
<th>#</th>
<th>Superfamily</th>
<th>Length</th>
<th>Size</th>
<th>Agarwal/States</th>
<th>SAM</th>
<th>Best method</th>
<th>Best MER</th>
<th>Trainset size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hemoglobin alpha/beta</td>
<td>141</td>
<td>505</td>
<td>PS</td>
<td>14</td>
<td>m2</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Ig kappa chain V region</td>
<td>108</td>
<td>280</td>
<td>PS</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>G-protein coupled receptors</td>
<td>348</td>
<td>165</td>
<td>SW</td>
<td>26</td>
<td>pepent</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>cytchrome C</td>
<td>105</td>
<td>142</td>
<td>PS</td>
<td>18</td>
<td>m2</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>snake neurotoxin</td>
<td>74</td>
<td>109</td>
<td>PS/FA/B2</td>
<td>0</td>
<td>mw/m2</td>
<td>m2</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>calcium binding EF-hand</td>
<td>158</td>
<td>106</td>
<td>PS/FA/B2</td>
<td>11</td>
<td>mw</td>
<td>m2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>glutathione transferase</td>
<td>222</td>
<td>106</td>
<td>PS</td>
<td>3</td>
<td>pep</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>protein kinase, C2H2</td>
<td>391</td>
<td>97</td>
<td>SW</td>
<td>70</td>
<td>ext</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>ferredoxin</td>
<td>54</td>
<td>93</td>
<td>SW</td>
<td>27</td>
<td>m2</td>
<td>m2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>ribose-phosphohosphate carboxylase</td>
<td>139</td>
<td>77</td>
<td>all</td>
<td>2</td>
<td>mw/m2</td>
<td>m2</td>
<td>0</td>
</tr>
</tbody>
</table>

Agarwal and States have analyzed five single-sequence search algorithms using a single sequence from each of the 67 superfamilies (Agarwal and States, 1998). They analyzed Smith and Waterman SSEARCH (Smith and Waterman, 1981; Pearson and Lipman, 1988), FASTA (FA) (Pearson and Lipman, 1988), Probabilistic Smith–Waterman (PS) (Bucher and Hoffmann, 1996), BLASTP 1.4 (BP) (Altschul et al., 1990), and WU-BLASTP2 (WU) (Altschul et al., 1990; Altschul and Gish, 1996). The general result of this work was that MERs are an effective evaluation criterion, and that Probabilistic Smith–Waterman had the best overall performance on complete proteins.

The table summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamliy, the best MER obtained by Agarwal and States, and the method which provided the best MER. 

set of experiments using training and discrimination sets based Pearson's annotated version of the PIR 39 database with 12 219 proteins total (Pearson, 1995). 

Agarwal and States have analyzed five single-sequence search algorithms using a single sequence from each of the 67 superfamilies (Agarwal and States, 1998). They analyzed Smith and Waterman SSEARCH (Smith and Waterman, 1981; Pearson and Lipman, 1988), FASTA (FA) (Pearson and Lipman, 1988), Probabilistic Smith–Waterman (PS) (Bucher and Hoffmann, 1996), BLASTP 1.4 (BP) (Altschul et al., 1990), and WU-BLASTP2 (WU) (Altschul et al., 1990; Altschul and Gish, 1996). The general result of this work was that MERs are an effective evaluation criterion, and that Probabilistic Smith–Waterman had the best overall performance on complete proteins.

Table 2 summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamily, the best MER obtained by Agarwal and States, and the method which provided the best MER. 

Table 2 summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamily, the best MER obtained by Agarwal and States, and the method which provided the best MER.

Table 2 summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamily, the best MER obtained by Agarwal and States, and the method which provided the best MER.

Table 2 summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamily, the best MER obtained by Agarwal and States, and the method which provided the best MER.

Table 2 summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamily, the best MER obtained by Agarwal and States, and the method which provided the best MER.

Table 2 summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamily, the best MER obtained by Agarwal and States, and the method which provided the best MER.

Table 2 summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamily, the best MER obtained by Agarwal and States, and the method which provided the best MER.

Table 2 summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamily, the best MER obtained by Agarwal and States, and the method which provided the best MER.

Table 2 summarizes the results from both our experiments and those of Agarwal and States. Each row lists one of the 67 superfamilies, length of the sequence used in the single-sequence search, size of the superfamily, the best MER obtained by Agarwal and States, and the method which provided the best MER.
Table 3. Comparison of Probabilistic Smith–Waterman (from Agarwal and States) to unweighted, weighted method 7, and PSP SAM HMM training, HMMer, and Meta-MEME (using MEME, Mast, and HMMer) with various training set sizes. The training sets always include the single query sequence used for Probabilistic Smith–Waterman. SAM internal weighting with one sequence is algorithmically the same as unweighted SAM. The final rows indicate total number of errors, number of families better than and worse than PSW, net change in number of errors over PSW, and net number of improved families over PSW for each method.

<table>
<thead>
<tr>
<th>#</th>
<th>Len</th>
<th>Size</th>
<th>SAM No Weights</th>
<th>SAM Internal</th>
<th>HMMer</th>
<th>Meta-MEME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 2 5 10</td>
<td>1 2 5 10</td>
<td>1 2 5 10</td>
<td>1 2 5 10</td>
</tr>
<tr>
<td>1</td>
<td>141</td>
<td>505</td>
<td>14</td>
<td>57</td>
<td>118</td>
<td>116</td>
</tr>
<tr>
<td>2</td>
<td>106</td>
<td>286</td>
<td>11</td>
<td>56</td>
<td>127</td>
<td>116</td>
</tr>
<tr>
<td>3</td>
<td>348</td>
<td>165</td>
<td>29</td>
<td>56</td>
<td>127</td>
<td>105</td>
</tr>
<tr>
<td>4</td>
<td>109</td>
<td>142</td>
<td>18</td>
<td>30</td>
<td>41</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>74</td>
<td>109</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>159</td>
<td>106</td>
<td>15</td>
<td>9</td>
<td>58</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>222</td>
<td>106</td>
<td>0</td>
<td>16</td>
<td>80</td>
<td>29</td>
</tr>
<tr>
<td>8</td>
<td>351</td>
<td>97</td>
<td>74</td>
<td>71</td>
<td>78</td>
<td>80</td>
</tr>
<tr>
<td>9</td>
<td>54</td>
<td>93</td>
<td>58</td>
<td>58</td>
<td>74</td>
<td>80</td>
</tr>
</tbody>
</table>

Produced that MER. We added this data to the best SAM method for each family, the best MER, and the size of the training set that produced that MER.

We calculated the SAM MERs by building models trained on the single superfamily search sequence used by Agarwal and States and models trained on small training sets (2, 5 and 10) containing the search sequence and a few other randomly-selected superfamily members from PIR 39. As in the Agarwal and States result, the discrimination sets consisted of the superfamily members from PIR 39 and the remaining (non-family) sequences in the database.

To compare our results with other maximum discrimination methods, we repeated the experiments using HMMer and Meta-MEME (Table 3). We built the HMMer models
from the training sets used in the SAM experiments and scored the same discrimination sets with these models, using HMMer’s hmmsw program. We used the default HMMer configuration. The impending rewrite of HMMer can be expected to have improved performance.

We also built HMMs using a combined MEME (Bailey and Elkan, 1994) and HMM method called Meta-MEME (Grundy et al., 1997; Grundy, 1998). In this procedure, MEME is used to locate common motifs in the training set, MAST finds the sequences in the training set with the strongest matches to these common motifs, and Meta-MEME builds an HMM in HMMer format based on the motifs and ‘diagrams’ which describe the distribution of the motifs in the matching sequences. We again used the same training and discrimination sets. We ran the programs from the MEME suite using the settings described in the literature (Grundy, 1998): a minimum motif width of 12, maximum motif width of 55, the ZOOPS motif model (zero or one motif occurrence per sequence), ten selected motifs from each training set, and use of only those motifs that appeared in more than half of the sequences in the training set to build the HMM.

The final rows of Table 3 include the minimum number of errors across the entire experiment and the number of families that have a better or worse MER than PSW. As can be seen, the HMM-based methods require 5 or 10 sequences to gain any improvement over the PSW results. For training sets of 1 or 2 sequences, PSW performs better than SAM (46–66% higher MER), HMMer (190–270% higher), or Meta-MEME (150–240%), indicating further need for tuning the model building procedure for small data sets. For training sets of 5 sequences, unweighted SAM has a 10% better MER and weighted SAM has a 20% better MER compared to PSW. HMMer and Meta-MEME MERs are 44% and 14% worse than PSW. At 10 training sequences, all HMM methods do better than PSW, with net MER improvements over PSW of 17, 28, 31 and 51%, corresponding to MERs of 296, 255, 244, and 173, for HMMer, Meta-MEME, unweighted SAM, and weighted SAM. Method 2 performed slightly worse than method 7 with, for example, a total MER of 187 (47% better than PSW) for a size 10 training set.

Based on these data, we conclude that SAM’s new internal sequence weighting method is a success, significantly extending the discriminatory effectiveness of hidden Markov models.

Discussion

Our experiments lead us to believe that we can consistently improve the discrimination performance of a SAM HMM by using maximum discrimination weighting functions during model building. We have added methods 2 and 7 to our current upgrade of SAM (2.0).

Our improvement in discrimination is not always dramatic, but it is very cheap. There is no noticeable increase in the time it takes to build the model when internal weighting is used.

We believe that a careful choice of Dirichlet mixture priors and transition regularizers could further improve results for a particular protein family or training set size. However, experiments to determine the best choice are beyond the scope of this paper.

The position-specific and entropy-based ‘external’ SAM methods often performed very well but were not as stable as our standalone methods. It is possible that internally implemented versions of these methods, with continuous updating of weights during model training, would be more consistent and discriminate better than our current internal methods.

None of the HMM-based methods performs as well as Probabilistic Smith–Waterman with only one or two training sequences. Because HMMs are a generalization of the PSW method, there is no intrinsic reason why this should be the case. Instead, it indicates that our current library of Dirichlet mixture priors and transition regularizers is more finely tuned to training HMMs for groups of sequences. Further study of this problem could result in a range of different prior probabilities for different training set sizes and discrimination goals.

The comparison of single-sequence PSW with multiple-sequence HMMs is somewhat unfair. A more complete comparison would include a modified version of PSW that uses all sequences in the training set and, for example, using an iterative search technique (Pearson, 1997) or averaging the scores (Grundy, 1998). As our primary goal was to evaluate the effectiveness of internal weighting methods in SAM, this work was considered beyond the scope of the paper.

Acknowledgments

We gratefully acknowledge Christian Barrett, David Haussler, and Kevin Karplus for their assistance and commentary on this work. This work was supported in part by NSF grant DBI-9408579 and its REU supplement, DOE grant DE-FG0395ER62112, and a grant from Digital Equipment Corporation.

References


