Dirichlet Mixtures: A Method for Improving Detection of Weak but Significant Protein Sequence Homology

Kimmen Sjölander[†]
Computer Science
U.C. Santa Cruz
kimmen@cse.ucsc.edu

Kevin Karplus
Computer Engineering
U.C. Santa Cruz
karplus@cse.ucsc.edu

Michael Brown
Computer Science
U.C. Santa Cruz
mpbrown@cse.ucsc.edu

Richard Hughey
Computer Engineering
U.C. Santa Cruz
rph@cse.ucsc.edu

Anders Krogh
The Sanger Centre
England
krogh@sanger.ac.uk

I. Saira Mian
Lawrence Berkeley Laboratory
U.C. Berkeley
saira@cse.ucsc.edu

David Haussler
Computer Science
U.C. Santa Cruz
haussler@cse.ucsc.edu

UCSC Technical Report
UCSC-CRL-96-09

Abstract

This paper presents the mathematical foundations of Dirichlet mixtures, which have been used to improve database search results for homologous sequences, when a variable number of sequences from a protein family or domain are known. We present a method for condensing the information in a protein database into a mixture of Dirichlet densities. These mixtures are designed to be combined with observed amino acid frequencies, to form estimates of expected amino acid probabilities at each position in a profile, hidden Markov model, or other statistical model. These estimates give a statistical model greater generalization capacity, such that remotely related family members can be more reliably recognized by the model. Dirichlet mixtures have been shown to outperform substitution matrices and other methods for computing these expected amino acid distributions in database search, resulting in fewer false positives and false negatives for the families tested. This paper corrects a previously published formula for estimating these expected probabilities, and contains complete derivations of the Dirichlet mixture formulas, methods for optimizing the mixtures to match particular databases, and suggestions for efficient implementation.

Keywords: Substitution matrices, pseudocount methods, Dirichlet mixture priors, profiles, hidden Markov models.

[†]To whom correspondence should be addressed. Mailing address: Baskin Center for Computer Engineering and Information Sciences, Applied Sciences Building, University of California at Santa Cruz, Santa Cruz, CA 95064. Phone: (408) 459-3430, Fax: (408) 459-4829.

1 Introduction

Recently, the first complete genome for a free-living organism was sequenced. On July 28, 1995, The Institute for Genomic Research (TIGR) announced in *Science* the complete DNA sequence for *Haemophilus influenzae* (R.D.Fleischmann, 1995). Along with this sequence were 1743 protein genes. It is not every day that the protein databases get such a large influx of novel proteins, and within days protein scientists were hard at work analyzing the data (Casari *et al.*, 1995). One of the main techniques used to analyze these proteins is to find similar proteins in the database whose structure or function are already known. When two sequences share at least 25% residue identity and each is at least 80 residues in length, then the two sequences are said to be *homologous*, i.e., they share the same overall structure (Doolittle, 1986). If the structure of one of the sequences has been determined experimentally, then the structure of the new protein can be inferred from the other. If one is fortunate, and a large number of homologous sequences are found, then it may be possible to tackle the somewhat more difficult problem, inferring the new protein's function(s).

However, requiring a minimum residue identity of 25% can mean that no sequences of known structure are deemed homologous to the new sequence. Does this mean that we can then assume that the three-dimensional structure of this new sequence is in a class of its own? This may be the case some fraction of the time. But it is more likely that some *remote homolog* exists in the database, sharing a common structure, but having less than 25% residue identity.

Moreover, the problem of finding homologous sequences, close or remote, is not limited to the case where one has a single protein. One may have several sequences available for a given family, but expect that other family members exist in the databases, and want to locate these putative members. Finding these remote homologs is one of the primary motivating forces behind the development of new types of statistical models for protein families and domains in recent years. It is also a key motivation for the work presented here.

1.1 Database search using statistical models

Statistical models for proteins are objects, like profiles, that capture the statistics defining a protein family or domain. Along with parameters expressing the expected amino acids at each position in the molecule or domain, and possibly other parameters as well, a statistical model will have a scoring function for sequences with respect to the model. These models come in various forms. Profiles and their many offshoots (Gribskov *et al.*, 1987; Gribskov *et al.*, 1990; Bucher *et al.*, 1996; Barton and Sternberg, 1990; Altschul *et al.*, 1990; Waterman and Perlwitz, 1986; Thompson *et al.*, 1994a; Thompson *et al.*, 1994b; Barton and Sternberg, 1990; Bowie *et al.*, 1991; Lüthy *et al.*, 1991; Bucher *et al.*, 1996), Position-Specific Scoring Matrices (Henikoff *et al.*, 1990), and hidden Markov models (HMMs) (Churchill, 1989; White *et al.*, 1994; Stultz *et al.*, 1993; Krogh *et al.*, 1994; Hughey, 1993; Baldi *et al.*, 1992; Baldi and Chauvin, 1994; Asai *et al.*, 1993), have all been proposed and demonstrated effective for particular tasks under certain conditions.

In contrast with homology determination by residue identity, statistical models use a very different technique to determine whether two sequences share a common structure. During database search with these models, each sequence in the database is assigned a score, (or, negatively, a cost), generally by adding the score (or cost) at each position in the model. For instance, a typical cost for aligning residue a at position i, is $-\log \text{Prob}(a \,|\, \text{position } i)$, where the base of the logarithm is arbitrary. A sequence is determined to belong to the family, or contain the domain, if the cost of aligning the sequence to the model falls below a cutoff. This cutoff can be determined experimentally, for instance, by setting it to the maximum cost for any of the known members of the family, or it can be pre-determined[1].

Because these parameters are used to score each sequence in the database, careful tuning of the parameters representing the expected amino acids becomes essential, and zero probabilities are particularly problematic. Allowing zero probabilities at positions gives an infinite penalty to sequences having the zero-probability residues at those positions. Even if a sequence is homologous to those used in training the model, a single mismatch at such a position would render that sequence unrecognizable by the model. On the other hand, the costs at each position are additive, so small improvements in predicting the expected amino acids at each position accumulate over the length of the sequence, and can boost a model's effectiveness significantly. Since each of these statistical models relies on having sufficient data to estimate its parameters, modeling protein families or domains for which few sequences have been identified is quite difficult. Methods that

[1]Two examples of pre-setting the cutoff are choosing a cost that is a certain number of standard deviations below the mean cost of all the proteins in the database (in which case, the number of standard deviations is pre-determined), and setting the cutoff based on the statistical significance of choosing the model over a null model.

increase the accuracy of estimating the expected amino acids at each position are thus of primary importance for these models.

We tread a thin line between specificity and sensitivity in estimating these parameters. If a model is highly specific, but does not generalize well, it will recognize only a fraction of those sequences in the family. In database discrimination, this model will generate *false negatives*—sequences that should be labeled as family members, but are instead labeled as not belonging to the family. The model is too strict, and database search with this model produces little new information. The reverse situation occurs when we sacrifice specificity for sensitivity. In this case, the model categorizes sequences which are not in the family as family members. These *false positives* are obtained through models that are too lax, and while true remote homologs may be included in the set identified as family members, they may be hard to identify as such if the pool is simply too large. One of the tests of the effectiveness of a statistical modeling technique, in fact, is how well it reduces the numbers of false negatives and false positives in database discrimination.

1.2 Issues in estimating expected amino acid probabilities

The following examples illustrate the kinds of issues encountered in estimating amino acid probabilities. In the first scenario, imagine that a multiple alignment of 100 sequences has a column containing only isoleucine, and no other amino acids. In the second scenario, an alignment of three sequences also has a column containing only isoleucine, and no other amino acids. If we estimate the expected probabilities of the amino acids in these columns to be equal to the observed frequencies, then the estimate of the expected probability of each amino acid i is simply the fraction of times i is observed (*i.e.*, $\hat{p}_i = n_i / |\vec{n}|$, where n_i is the frequency of amino acid i in the column, and $|\vec{n}| = \sum_i n_i$). Using this method of estimating the probabilities, we would assign a probability of 1 to isoleucine and zero to all the other amino acids for both of these columns. But is this estimate reasonable?

It is illuminating to consider the analogous problem of assessing the fairness of a coin. A coin is said to be fair if $\text{Prob}(heads) = \text{Prob}(tails) = 1/2$. Equivalently, we expect that if we toss a fair coin n times, obtaining h heads and t tails, we expect h/n and t/n to each come closer and closer to $1/2$ as n approaches infinity, in accordance with the law of large numbers. Now, if we pick a coin at random, and toss it three times, and it comes up heads each time, what should our estimate of the probability of heads for this coin be? If we assume that most coins are fair, then we are unlikely to change this *a priori* assumption based on only a few tosses. On the other hand, if we toss the coin an additional thousand times and it comes up heads each time, at this point very few of us would insist that the coin was indeed fair. Our estimate of this coin's probability of heads is going to be 1, or quite close to it. Given an abundance of data, we will discount any previous assumptions, and believe the data.

In the first scenario, for the column containing 100 isoleucines and no other amino acids, the evidence is strong that isoleucine is conserved at this position. Allowing any substitutions in this position is clearly not optimal, and giving isoleucine probability 1, or close to it, appears sensible.

In the second scenario, with an alignment of only three sequences, we cannot rule out the possibility that proteins in the family not included in the training set may have other amino acids at this position. In this case, we might not want to assign isoleucine probability 1, and require that all sequences in the family (or containing the domain) must have an isoleucine at this position. Instead, we might want to use prior knowledge about amino acid distributions, and modify our estimate about the expected distribution to reflect that prior knowledge. In this case, we know that where isoleucine is found, other hydrophobic residues are often found, especially leucine and valine. Our estimate of the expected distribution at this position would sensibly include these residues, and perhaps the other amino acids as well, albeit with much smaller probabilities. By contrast, when we have many sequences multiply aligned, we expect the estimate $\hat{p}_i = n_i / |\vec{n}|$ to be a close approximation of the true underlying probabilities, and any prior information about typical amino acid distributions is relatively unimportant.

Thus, the natural solution is to introduce prior information into the construction of the statistical model, interpolating smoothly between reliance on the prior information concerning likely amino acid distributions, in the absence of data, to confidence in the amino acid frequencies observed at each position, given abundant data. Our aim in this work is to provide a statistically well-founded, Bayesian framework for obtaining this prior information and for combining this prior information with observed amino acid frequencies.

One final comment concerning skew is in order. A skewed sample can arise in two ways. In the first, the sample is skewed simply from the luck of the draw. This kind of skew is common in small samples, and is akin to tossing a fair coin three times and observing three heads in a row. The second type of skew is more insidious, and can occur even when large samples are drawn. In this kind of skew, one subfamily is over-represented, such that a large fraction of the sequences used to train the statistical model are minor variants of each other. This disparity among the number of sequences available from different subfamilies for a given protein is the basis for the widespread use of weighting schemes (Sibbald and Argos, 1990; Thompson *et al.*, 1994a; Thompson *et al.*, 1994b; Henikoff and Henikoff, 1994). If one has reason to believe that the available data over-represents some subfamilies, Dirichlet mixtures can be used in conjunction with any weighting scheme desired to produce more accurate amino acid estimates. Simply weight the sequences prior to computing the expected amino acids for each position using a Dirichlet mixture. Each column in the weighted data will be a vector of counts, though probably real-valued rather than integral. Because we assume weighted data may be used as input, we have incorporated this possibility in the formula given in Section 3.2 to compute the expected amino acid distributions.

1.3 Obtaining and using prior knowledge of amino acid distributions

Fortunately, even when data from a particular family may be limited, there is no lack of data in the protein sequence databases concerning the kinds of distributions which are likely or unlikely in particular positions in proteins. In this work, we have attempted to condense the enormous wealth of information in the databases into the form of a mixture of densities. These densities assign a probability to every possible distribution of the amino acids. We use *Maximum Likelihood* (Duda and Hart, 1973; Nowlan, 1990; Dempster *et al.*, 1977) to estimate these mixtures—i.e., we seek to find a mixture that maximizes the probability of the observed data. Often, these densities capture some prototypical distributions. Taken as an ensemble, they explain the observed distributions in the databases.

There are many different commonly occurring distributions. Some of these reflect a preference for hydrophobic amino acids, some for small amino acids, and some for more complex combinations of physiochemical features. Certain combinations of these features are commonly found, while others are much rarer. Degrees of conservation differ, due to the presence or absence of structural or functional constraints. In the extreme case, when an amino acid is highly conserved at a certain position in the protein family, such as the proximal histidine that coordinates the heme iron in hemoglobin, the distribution of amino acids in the corresponding column of the multiple alignment is sharply peaked on that one amino acid, whereas in other cases the distribution may be spread over many possible amino acids.

With accurate prior information about which kinds of amino acid distributions are reasonable in columns of alignments, it is possible even with only a few sequences to identify which of the prototypical distributions characterizing positions in proteins may have generated the amino acids observed in a particular column of the emerging statistical model. Using this informed guess, we can adjust the expected amino acid probabilities so that the estimate of the amino acids for that position includes the possibility of amino acids that may not have been seen at all in that position, but are consistent with observed amino acid distributions in the protein databases. This has the effect of moving estimated amino acid distributions toward known distributions, and away from distributions that are unusual biologically. The models produced are more effective at generalizing to previously unseen data, and are often superior at database search and discrimination experiments (Karplus, 1995a; Tatusov *et al.*, 1994; Bailey and Elkan, 1995; Brown *et al.*, 1993).

1.3.1 Comparison with other methods for computing these probabilities

We are certainly not the first group to notice the need for incorporating prior information about such amino acid distributions into the parameter estimation process. Indeed, our present work has several conceptual similarities with profile methods, particularly in regard to seeking meaningful amino acid distributions for use in database search and multiple alignment (Waterman and Perlwitz, 1986; Barton and Sternberg, 1990; Gribskov *et al.*, 1990; Bowie *et al.*, 1991; Lüthy *et al.*, 1991; Claverie, 1993; Claverie, 1994). This work also has much in common with amino acid substitution matrices, which have been used effectively in database search and discrimination tasks (Henikoff and Henikoff, 1992; Altschul, 1991).

There are two drawbacks associated with the use of substitution matrices. First, each amino acid has a fixed substitution probability with respect to every other amino acid. In any particular substitution matrix, to paraphrase Gertrude Stein, an isoleucine is an isoleucine is an isoleucine. However, an isoleucine seen in one context, for instance, in a position that is functionally conserved, will have different substitution probabilities than an isoleucine seen in another context, where any hydrophobic residue may be allowed. Second, only the relative frequency of amino acids is considered, while the actual number observed is ignored. Thus, in substitution-matrix-based methods, the expected amino acid probabilities are identical for an apparently conserved column containing 100 isoleucines (and no other amino acids) and a column containing three isoleucines, or even a single isoleucine. All three situations are treated identically, and the estimates produced are indistinguishable.

The method described here addresses both of these issues. A Dirichlet mixture prior can be decomposed into individual *components*, each of which is a probability density over all the possible combinations of amino acids occurring at positions in proteins. Common distributions, determined by functional or structural constraints, are captured by these components; these then provide position-specific substitution probabilities. In producing an estimate for the expected amino acids, the formula employed (equation 15 in Section 3.2) gives those components which are most likely to have generated the actual amino acids observed the greatest impact on the estimation.

For example, in Tables 1 and 2 we give a nine-component mixture estimated on the Blocks database (Henikoff and Henikoff, 1991). In this mixture, isoleucine is seen in several contexts. Component 9 gives high probability to all conserved distributions (i.e., distributions where a single residue is preferred over all others). Component 6 represents distributions preferring isoleucine and valine, but allowing leucine and methionine (i.e., this component gives high probability to aliphatic residues found in beta sheets). Component 5 reverses the order of residues preferred from component 6, preferring leucine and methionine to isoleucine, and allowing phenylalanine and valine as less likely substitutions. Component 8 favors methionine, but allows isoleucine and the other aliphatic residues as well as phenylalanine (and a few other residues). (A full description of how to interpret these mixtures in general is given in Section 2.)

When only one or two isoleucines are observed, the lion's share of the probability is shared by two components: component 6 starts off with the highest probability at 0.45, while component 5 comes in second with just under 0.20 probability. (Components 8 and 9 both have relatively low probability, at 0.12 and 0.08, respectively.) However, the information in the column increases rapidly as the number of sequences grows, and the probabilities of each of the components changes. Components 5, 6, and 8 decrease in probability, while component 9, which favors conserved distributions, grows very rapidly in probability. At ten observed isoleucines, component 9 has probability 0.62, component 6 has probability 0.24, component 5 has probability 0.07, and component 8 has one of the lowest probabilities of all the components, at 0.002. This process is demonstrated in Table 3.

The estimates of the expected amino acids reflect the changing contribution of these components. Given a single observation, isoleucine has probability 0.47, valine has probability 0.15, leucine has probability 0.12, and methionine has probability 0.03. This reveals the influence of components 5 and 6, with their preference for allowing substitutions with valine, leucine and methionine. By ten observations, isoleucine has probability 0.94, valine has probability 0.02, leucine has probability 0.01, and methionine has probability 0.003. We can still see the contribution of component 6, with its bias toward allowing valine to substitute for isoleucine. But the predominant signal is that isoleucine is required at this position.

Moreover, the second issue—the importance of the actual number of residues observed—is addressed in the estimation formula as well. Here, as the number of observations increases, the contribution of the prior information is lessened. Even if a mixture prior does not give high probability to a particular type of distribution, as the number of sequences aligned increases, the estimate for a column becomes more and more peaked around the maximum likelihood estimate for that column (i.e., $\hat{p}_i$ approaches $n_i / |\vec{n}|$ as $|\vec{n}|$ increases). Importantly, when the data indicate a residue is conserved at a particular position (i.e., most or all of the sequences in an alignment contain a given residue in one position, and a sufficient number of observations are available), the expected amino acid probabilities produced by this method will remain peaked around that residue, instead of being modified to include all the residues that substitute *on average* for the conserved residue, as is the case with substitution matrices. (See, for example, the estimated amino acid probabilities produced by two substitution matrix-based methods in Tables 4-7.)

Pseudocount methods are a special case of Dirichlet mixtures, where the mixture consists of a single component. In these methods, a fixed value is added to each observed amino acid count, and then the counts are renormalized (i.e., $\hat{p}_i = (n_i + z_i)/(\sum_j n_j + z_j)$, where z_j can be the same constant for every amino acid j, or can vary from one amino acid to the next[2]. They have some of the desirable properties of Dirichlet

[2] A comparison of Dirichlet mixtures with data-dependent pseudocount methods is given in (Karplus, 1995a) and (Tatusov *et al.*, 1994), where Dirichlet mixtures were shown to give superior results.

mixtures, but because they have only a single component, they are unable to represent as complex a set of prototypical distributions. We include in Tables 4-7 probability estimates for two popular pseudocount methods which add the same constant for each amino acid, and can thus be called *zero-offset* methods: *Add-One*, where $z_i = 1$ for all i, and *Add-Share*, where $z_i = 0.05$ for all i. The Dirichlet density *1-comp*, a single-component Dirichlet density estimated on the Blocks database, is also a pseudocount method where z_i is a closely related to the background frequency of amino acid i.

The work of Lüthy, McLachlan, and Eisenberg (Lüthy *et al.*, 1991) also has some interesting similarities to that presented here. They analyzed multiple alignments containing secondary structure information to construct a set of nine probability distributions, which we call the *LME* distributions, describing the distribution of amino acids in nine different structural environments[3]. LME distributions have been shown to increase the accuracy of profiles in both database search and multiple alignment by enabling them to take advantage of prior knowledge of secondary structure.

These distributions cannot always be used, since in many cases structural information is not available, or the statistical model employed is not designed to take advantage of such information. For example, our method for training an HMM assumes unaligned sequences are given as input to the program, and that no secondary structure information for the sequences is available. Thus, distributions associated with particular secondary structural environments, such as the LME distributions, are inappropriate for our use. Moreover, we have an additional problem using the LME distributions in this Bayesian framework. As we will show in Section 3.2, Bayes' rule requires that in computing the amino acid probabilities the observed frequency counts be modified less strongly when the prior distribution has a very high variance. Thus, when there is no measure of the variance associated with a distribution, as is the case with the LME distributions, one must assign a variance arbitrarily in order to use the distribution to compute the expected probabilities.

In this paper, we propose the use of mixtures of Dirichlet densities (see e.g., (Berhardo and Smith, 1994)) as a means of representing prior information about expected amino acid distributions. In Section 2 we give a description of ways to interpret these mixtures. The mathematical foundations of the method described in this paper are given in Section 3. Dirichlet densities are described in Section 3.1. For those wishing to use these mixtures, we present in Section 3.2 a Bayesian method for combining observed amino acids with these priors to produce posterior estimates of the probabilities of the amino acids. Section 3.3 contains the mathematical derivation of the learning rule for estimating Dirichlet mixtures. In Section 4, we present an overview of work done both at Santa Cruz (Karplus, 1995a; Karplus, 1995b; Brown *et al.*, 1993) and elsewhere (Tatusov *et al.*, 1994; Bailey and Elkan, 1995; Henikoff and Henikoff, 1995) that demonstrates the effectiveness of these densities in a variety of statistical models, and the superiority of this technique in general over others tried. Some pointers to help users avoid underflow and overflow problems, as well as speed up the computation of mixture estimation, are treated in Section 5.

We also want to emphasize, perhaps obviously, that the method described in this paper is general, and applies not only to data drawn from columns of multiple alignments of protein sequences, but can be used to characterize distributions over other alphabets as well. For example, we have done some experiments developing Dirichlet mixtures for RNA, both for single-column statistics and for pairs of columns, and we have estimated Dirichlet densities over transition probabilities between states in hidden Markov models.

For a review of the essentials of the HMM methodology we use, including architecture, parameter estimation, multiple alignments, and database searches, see (Krogh *et al.*, 1994).

[3] In more recent work, they have used 18 different distributions (Bowie *et al.*, 1991).

2 Interpreting Dirichlet Mixtures

We include in this paper a 9-component mixture estimated on the Blocks database (Henikoff and Henikoff, 1991) which has given some of the best results of any mixture estimated using the techniques described here[4]. Table 1 gives the parameters of this mixture.

Since a Dirichlet mixture describes the expected distributions of amino acids in the data used to estimate the mixture, it is useful to look in some detail at each individual component of the mixture to see what distributions of amino acids it favors.

Two kinds of parameters are associated with each component: the mixture coefficient, q, and the $\vec{\alpha}$ parameters which define the distributions preferred by the component. For any distribution of amino acids, the mixture as a whole assigns a probability to the distribution by combining the probabilities given the distribution by each of the components in the mixture.

One way to characterize a component is by giving the mean expected amino acid probabilities and the variance around the mean. Formulas to compute these quantities are given in Section 3.1. We can also list the amino acids for each component in order by the ratio of the mean frequency of the amino acids in a component to the background frequency of the amino acids. Table 2 lists the preferred amino acids for each component in the mixture.

The mixture coefficient q associated with a component is equal to the probability of that component given the data, averaged over all the data—i.e., it expresses the fraction of the data represented by the component. In this mixture, the components peaked around the aromatic and the non-polar hydrophobic residues represent the smallest fraction of the columns used to train the mixture, and the component representing all the highly conserved residues (component number 9) represents the largest fraction of the data.

The value $|\vec{\alpha}| = \sum_{i=1}^{20} \alpha_i$ is a measure of the peakedness of the component about the mean. Higher values of $|\vec{\alpha}|$ indicate that distributions must be close to the mean of the component in order to be given high probability by that component. In our experience, when we allow a large number of components, we often find that many of the components that result are peaked around individual residues, and have high $|\vec{\alpha}|$, but this may be an artifact of our optimization technique. However, when we estimate mixtures having a limited number of components (for instance, ten or fewer components), we find that one component tends to have a very small $|\vec{\alpha}|$, allowing this component to give high probability to all essentially pure distributions. This kind of component has high probability in most of the mixtures we have estimated— evidence that nearly pure distributions are common in the databases we have used to estimate these mixtures. Since the Blocks database was selected to favor highly conserved columns, it is not surprising that the individual components of a Dirichlet mixture tuned for the Blocks database also favor conserved columns. Mixtures tuned for the HSSP set of alignments, which contains full proteins, rather than just highly conserved blocks, show similar behavior, although the $|\vec{\alpha}|$ of the components of these mixtures are not quite as low as the $|\vec{\alpha}|$ of mixtures estimated on the Blocks database.

We often find that $|\vec{\alpha}|$ and q are inversely proportional to each other. For instance, the component in Table 1 which has the largest mixture coefficient (meaning the most common distributions) also has the smallest value of $|\vec{\alpha}|$. The amino acids favored by this component (component 9)—tryptophan, glycine, proline and cysteine—are indeed the most highly conserved ones. However, as Table 3 shows, this component gives high probability to pure distributions centered around other residues as well.

Groups of amino acids that frequently substitute for each other will tend to have one component that assigns a high probability to the members of the group and a low probability to other amino acids. These components tend to have higher $|\vec{\alpha}|$. For instance, in Table 1, the two components with the largest values of $|\vec{\alpha}|$ (and so the most mixed distributions) represent the polars and the non-polar hydrophobics, respectively. A residue may be represented primarily by one component (as proline is) or by several components (as isoleucine and valine are).

[4]A close variant of this mixture was used in experiments elsewhere (Tatusov *et al.*, 1994; Henikoff and Henikoff, 1995)

3 Mathematical Foundations

3.1 What are Dirichlet densities?

A Dirichlet density ρ is a probability density over the set of all probability vectors $\vec{p}$ (i.e., $p_i \geq 0$ and $\sum_i p_i = 1$) (Berger, 1985; Santner and Duffy, 1989). In the case of proteins, with a 20-letter alphabet, $\vec{p} = p_1, \ldots, p_{20}$ and $p_i = \mathrm{Prob}(\text{amino acid } i)$. Here, each vector $\vec{p}$ represents a possible probability distribution over the 20 amino acids. A Dirichlet density has parameters $\vec{\alpha} = \alpha_1, \ldots, \alpha_{20}$, $\alpha_i > 0$. The value of the density for a particular vector $\vec{p}$ is

$$\rho(\vec{p}) = \frac{\prod_{i=1}^{20} p_i^{\alpha_i - 1}}{Z} \tag{1}$$

where Z is the normalizing constant that makes ρ integrate to unity. The mean value of p_i given a Dirichlet density with parameters $\vec{\alpha}$ is $\alpha_i / |\vec{\alpha}|$ (where $|\vec{\alpha}| = \sum_i \alpha_i$). The mean value for $\vec{p}$ is

$$E p_i = \alpha_i / |\vec{\alpha}| \tag{2}$$

The second moment $E p_i p_j$, for the case $i \neq j$ is given by

$$E p_i p_j = \frac{\alpha_j \alpha_i}{|\vec{\alpha}| \left(|\vec{\alpha}| + 1\right)}. \tag{3}$$

When $i = j$, the second moment $E p_i^2$ is given by

$$E p_i^2 = \frac{\alpha_i \left(\alpha_i + 1\right)}{|\vec{\alpha}| \left(|\vec{\alpha}| + 1\right)} \tag{4}$$

In the case of a mixture prior, we assume that ρ is a mixture of Dirichlet densities, and hence has the form

$$\rho = q_1 \rho_1 + \ldots + q_l \rho_l \tag{5}$$

where each ρ_j is a Dirichlet density specified by parameters $\vec{\alpha}_j = (\alpha_{j,1}, \ldots, \alpha_{j,20})$ and the numbers $q_1, \ldots, q_l$ are positive and sum to 1. A density of this form is called a *mixture density* (or, in this specific case, a *Dirichlet mixture density*), and the q_j values are called *mixture coefficients*. Each of the densities ρ_j is called a *component* of the mixture.

The mean of a mixture is the weighted sum of the means of each of the components in the mixture, weighted by their mixture coefficients. That is, $E p_i = \sum_j q_j \alpha_{j,i} / |\vec{\alpha}_j|$.

We use the symbol Θ to refer to the entire set of parameters defining a prior. In the case of a mixture, $\Theta = \vec{\alpha}_1, \ldots, \vec{\alpha}_l, q_1 \ldots, q_l$, whereas in the case of a single density, $\Theta = \vec{\alpha}$.

3.2 Computing Expected Amino Acid Probabilities

As described in Section 1, in predicting the expected probabilities of amino acids at each position in a protein family or domain, one is often hampered by insufficient or skewed data. The amino acid frequencies in the available data may be far from accurate reflections of the amino acid frequencies in all the family members.

Fortunately, we are in a position to take advantage of information contained in a Dirichlet prior. As we explained in Section 3.1, a Dirichlet density with parameters $\Theta = \vec{\alpha}_1, \ldots, \vec{\alpha}_l,\ q_1 \ldots,\ q_l$ defines a probability distribution ρ_Θ over all the possible distributions of amino acids. Given a column in a multiple alignment, we can combine the information in the prior with the observed amino acid counts to form estimates $\hat{p}_i$ of the probabilities of each amino acid i at that position. These estimates, $\hat{p}_1, \ldots, \hat{p}_{20}$, of the actual p_i values will differ from the estimate $\hat{p}_i = n_i / |\vec{n}|$, and should be much better when the the number of observations is small.

Let us suppose that we fix a numbering of the amino acids from 1 to 20. Then, each column in a multiple alignment can be represented by a vector of counts of amino acids of the form $\vec{n} = (n_1, \ldots, n_{20})$, where n_i is the number of times amino acid i occurs in the column represented by this count vector.

At this point, we must explain some assumptions we have made concerning how the observed data were generated. The mathematical formulae for estimating and using Dirichlet mixture priors described in the following sections follow directly from these assumptions. We assume that the hidden process generating each count vector $\vec{n}$, can be modeled by the following stochastic process[5]:

1. First, a component j from the mixture Θ is chosen at random according to the mixture coefficient q_j.
2. Then a probability distribution $\vec{p}$ is chosen independently according to $\text{Prob}\left(\vec{p} \mid \vec{\alpha}_j\right)$, the probability defined by component j over all such distributions.
3. Finally, the count vector $\vec{n}$ is generated according to the multinomial distribution with parameters $\vec{p}$.

Obviously, when Θ consists of a single component, the first step is trivial, since the probability of the single component is 1. In this case, the stochastic process consists of steps 2 and 3.

We can now define the estimated probability $\hat{p}_i$ of amino acid i, given a Dirichlet density with parameters Θ and observed amino acid counts $\vec{n}$ as follows:

$$\hat{p}_i = \text{Prob}\left(\text{amino acid } i \,\middle|\, \Theta, \vec{n}\right) = \int_{\vec{p}} \text{Prob}\left(\text{amino acid } i \,\middle|\, \vec{p}\right) \text{Prob}\left(\vec{p} \,\middle|\, \Theta, \vec{n}\right)\, d\vec{p} \tag{6}$$

The first term in the integral, $\text{Prob}\left(\text{amino acid } i \mid \vec{p}\right)$, is simply p_i, the i^{th} element of the distribution vector $\vec{p}$. The second term, $\text{Prob}\left(\vec{p} \mid \Theta, \vec{n}\right)$, represents the posterior probability of the distribution $\vec{p}$ under the Dirichlet density with parameters Θ, given that we have observed amino acid counts $\vec{n}$. Taken together, the integral $\int_{\vec{p}} \text{Prob}\left(\text{amino acid } i \mid \vec{p}\right) \text{Prob}\left(\vec{p} \mid \Theta, \vec{n}\right)\, d\vec{p}$ represents the contributions from each probability distribution $\vec{p}$, weighted according to its posterior probability, of amino acid i. An estimate of this type is called a *mean posterior estimate*.

3.2.1 Computing probabilities using a single density (pseudocounts)

While we find the best results in computing these expected amino acid distributions come from employing mixtures of Dirichlet densities, it is enlightening to consider the posterior estimate of an amino acid i in the case of a single density.

In the case of a single-component density with parameters $\vec{\alpha}$, the mean posterior estimate of the probability of amino acid i is defined

$$\hat{p}_i = \int_{\vec{p}} \text{Prob}\left(\text{amino acid } i \,\middle|\, \vec{p}\right) \text{Prob}\left(\vec{p} \,\middle|\, \vec{\alpha}, \vec{n}\right)\, d\vec{p} \tag{7}$$

By Lemma 4 (the proof of which is found in the Appendix) the posterior probability of each distribution $\vec{p}$, given the count data $\vec{n}$ and the density with parameters $\vec{\alpha}$, is

[5] Note: we could instead choose to select amino acids independently with probability p_i; the optimization problem for optimizing Θ comes out the same.

Lemma 4**:**

$$\mathrm{Prob}\left(\vec{p} \,\middle|\, \vec{\alpha}, \vec{n}\right) = \frac{\Gamma(|\vec{\alpha}| + |\vec{n}|)}{\prod_{i=1}^{20} \Gamma(\alpha_i + n_i)} \prod_{i=1}^{20} p_i^{\alpha_i + n_i - 1}$$

.

Here, as usual, $|\vec{\alpha}| = \sum_i \alpha_i$, $|\vec{n}| = \sum_i n_i$, and Γ is the Gamma function, the continuous generalization of the integer factorial function (i.e., $\Gamma(x+1) = x!$).

Now, if we substitute p_i for $\mathrm{Prob}\left(\text{amino acid } i \,\middle|\, \vec{p}\right)$ and the result of Lemma 4 into equation 7 we have

$$\hat{p}_i = \int_{\vec{p}} p_i \frac{\Gamma(|\vec{\alpha}| + |\vec{n}|)}{\prod_j \Gamma(\alpha_j + n_j)} \prod_j p_j^{\alpha_j + n_j - 1} \, d\vec{p}. \tag{8}$$

Here, we can pull those terms not depending on $\vec{p}$ out of the integral, obtaining

$$\hat{p}_i = \frac{\Gamma(|\vec{\alpha}| + |\vec{n}|)}{\prod_j \Gamma(\alpha_j + n_j)} \int_{\vec{p}} p_i \prod_j p_j^{\alpha_j + n_j - 1} \, d\vec{p}. \tag{9}$$

Now, noting the contribution of the p_i term within the integral, and using equation (49) from Lemma 2, giving $\int_{\vec{p}} \prod_i p_i^{\alpha_i - 1} \, d\vec{p} = \frac{\prod_i \Gamma(\alpha_i)}{\Gamma(|\vec{\alpha}|)}$, we have

$$\hat{p}_i = \frac{\Gamma(|\vec{\alpha}| + |\vec{n}|)}{\Gamma(|\vec{\alpha}| + |\vec{n}| + 1)} \frac{\Gamma(\alpha_i + n_i + 1) \prod_{j \neq i} \Gamma(\alpha_j + n_j)}{\prod_j \Gamma(\alpha_j + n_j)} \tag{10}$$

At this point we can cancel out most of the terms, and take advantage of the fact that $\frac{\Gamma(n+1)}{\Gamma(n)} = \frac{n!}{(n-1)!} = n$, obtaining

$$\hat{p}_i = \frac{n_i + \alpha_i}{|\vec{n}| + |\vec{\alpha}|} \tag{11}$$

These Dirichlet densities can thus be seen as vectors of *pseudocounts*: probability estimates are formed by adding constants to the observed counts for each amino acid, and then renormalizing. Pseudocount methods are widely used to avoid zero probabilities in building statistical models. Note, when $n = 0$, in the absence of data, the estimate produced is simply $\alpha_i / |\vec{\alpha}|$, the normalized values of the parameters $\vec{\alpha}$, which are the means of the Dirichlet density. This mean, while not necessarily the background frequency of the amino acids in the training set, is often a close approximation to it. Thus, in the absence of data, our estimate of the expected amino acid probabilities will be close to the background frequencies. The simplicity of the pseudocount method is one of the reasons Dirichlet densities are so attractive.

Programs to compute the expected amino acid frequencies are available via anonymous ftp from our ftp site, ftp.cse.ucsc.edu, and on our web site at http://www.cse.ucsc.edu/research/compbio.

3.2.2 Computing probabilities using mixture densities

In the case of a mixture density, we compute the amino acid probabilities in a similar way:

$$\hat{p}_i = \mathrm{Prob}\left(\text{amino acid } i \,\middle|\, \Theta, \vec{n}\right) = \int_{\vec{p}} \mathrm{Prob}\left(\text{amino acid } i \,\middle|\, \vec{p}\right) \mathrm{Prob}\left(\vec{p} \,\middle|\, \Theta, \vec{n}\right) \, d\vec{p} \tag{12}$$

As in the case of the single density, we can substitute p_i for $\mathrm{Prob}(\text{amino acid } i \,|\, \vec{p})$. In addition, since Θ is a mixture of Dirichlet densities, by the definition of a mixture (equation 5), we can expand $\mathrm{Prob}(\vec{p} \,|\, \Theta, \vec{n})$ obtaining

$$\hat{p}_i = \int_{\vec{p}} p_i \left(\sum_{j=1}^{l} \mathrm{Prob}\left(\vec{p} \,\middle|\, \vec{\alpha}_j, \vec{n}\right) \mathrm{Prob}\left(\vec{\alpha}_j \,\middle|\, \vec{n}, \Theta\right) \right) d\vec{p} \tag{13}$$

In this equation, $\text{Prob}\left(\vec{\alpha}_j \mid \vec{n}, \Theta\right)$ is the posterior probability of the j^{th} component of the density, given the vector of counts $\vec{n}$ (equation 16 below). It captures our assessment that the j^{th} component was chosen in step 1 of the stochastic process generating these observed amino acids. The first term, $\text{Prob}(\vec{p} \mid \vec{\alpha}_j, \vec{n})$, then represents the probability of each distribution $\vec{p}$, given component j and the count vector $\vec{n}$.

We can pull out terms not depending on $\vec{p}$ from inside the integral, giving us

$$\hat{p}_i = \sum_{j=1}^{l} \text{Prob}\left(\vec{\alpha}_j \,\middle|\, \vec{n}, \Theta\right) \int_{\vec{p}} p_i \text{Prob}(\vec{p} \mid \vec{\alpha}_j, \vec{n}) d\vec{p} \tag{14}$$

At this point, we use the result from equation (11), and obtain[6]

$$\hat{p}_i = \sum_{j=1}^{l} \text{Prob}\left(\vec{\alpha}_j \,\middle|\, \vec{n}, \Theta\right) \frac{n_i + \alpha_{j,i}}{|\vec{n}| + |\vec{\alpha}_j|} \tag{15}$$

Hence, instead of identifying one single component of the mixture that accounts for the observed data, we determine how likely each individual component is to have produced the data. Each component then contributes pseudocounts proportional to the posterior probability that it produced the observed counts. In this case, when $n = 0$, $\hat{p}_i$ is simply $\sum_j q_j \alpha_{j,i} / |\vec{\alpha}_j|$, the weighted sum of the mean of each Dirichlet density in the mixture.

When a component has a very small $|\vec{\alpha}|$, it adds a very small bias to the observed amino acid frequencies. As we show in Section 2, such components give high probability to all distributions peaked around individual amino acids. The addition of such a small bias allows these components to not shift the estimated amino acids away from conserved distributions, even when relatively small amounts of data are available.

By contrast, components having a larger $|\vec{\alpha}|$ tend to favor mixed distributions, that is, combinations of amino acids. In these cases, the individual $\alpha_{j,i}$ values tend to be relatively large for those amino acids i preferred by the component. When such a component has high probability given a vector of counts, these $\alpha_{j,i}$ have a corresponding influence on the expected amino acids predicted for that position. The estimates produced may include significant probability for amino acids not seen at all in the count vector under consideration.

Moreover, examining equation 15 reveals a smooth transition between reliance on the prior information, in the absence of sufficient data, and confidence that the observed frequencies in the available training data represent the expected probabilities in the family as a whole, as the number of observations increases. When the number of observations is small, the mixture prior has the greatest effect in determining the posterior estimate. But as the number of observations increases, the n_i values will dominate the α_i values. Importantly, as the number of observations increases, this estimate approaches the maximum likelihood estimate, $\hat{p}_i = n_i / |\vec{n}|$.

Thus, in the case of a mixture density, we will first want to calculate the quantity $\text{Prob}\left(\vec{\alpha}_j \mid \vec{n}, \Theta\right)$ for each j between 1 and l. This quantity is computed from Bayes' rule as

$$\text{Prob}\left(\vec{\alpha}_j \,\middle|\, \vec{n}, \Theta\right) = \frac{q_j \,\text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)} \;. \tag{16}$$

$\text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)$ is the probability of the count vector $\vec{n}$ given the j^{th} component of the mixture, and is derived in Section A.3. The denominator, $\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)$, is defined

$$\text{Prob}\left(\vec{n} \,\middle|\, \Theta, |\vec{n}|\right) = \sum_k q_k \text{Prob}\left(\vec{n} \,\middle|\, \vec{\alpha}_k, |\vec{n}|\right) \;. \tag{17}$$

[6]This formula was misreported in previous work (Brown *et al.*, 1993; Karplus, 1995a; Karplus, 1995b).

3.3 Derivation of Dirichlet Densities

As noted earlier, much statistical analysis has been done on amino acid distributions found in particular secondary structural environments in proteins. However, our primary focus in developing these techniques for protein modeling has been to rely as little as possible on previous knowledge and assumptions, and instead to use statistical techniques that uncover the underlying key information in the data.

Consequently, our approach, instead of beginning with secondary structure, is to take unlabeled training data (i.e., columns from multiple alignments with no secondary structure information attached) and attempt to discover those classes of distributions of amino acids that are intrinsic to the data. The statistical method employed directly estimates the most likely Dirichlet mixture density through clustering observed counts of amino acids. In most cases, the common amino acid distributions we find are easily identified (e.g., a large non-polar), but we do not set out *a priori* to find distributions representing known structural environments.

Given a set of m columns from a variety of multiple alignments, we tally the frequency of each amino acid in each column, with the end result being a vector of counts of each amino acid for each column in the dataset. Thus, our primary data is a set of m count vectors. Many multiple alignments of different protein families are included, so m is typically in the thousands. We fix a numbering of the amino acids from 1 to 20, so each count vector has the form $\vec{n} = (n_1, \ldots, n_{20})$, where n_i is the number of times amino acid i occurs in the column represented by this count vector.

We have used Maximum Likelihood to estimate the parameters Θ of ρ from the set of count vectors; that is, we seek those parameters that maximize the probability of occurrence of the observed count vectors. We assume the three-stage stochastic model described in Section 3.2 was used independently to generate each of the count vectors in our observed set of count vectors. Under this assumption of independence, the probability of the entire set of observed frequency count vectors is equal to the product of their individual probabilities. Thus, we seek to find the model that maximizes $\prod_{t=1}^{m} \mathrm{Prob}\left(\vec{n}_t \;\middle|\; \Theta, |\vec{n}_t|\right)$. Since the negative logarithm of the probability is inversely proportional to the probability, this is equivalent to finding the $\vec{\alpha}$ that minimizes the *objective function*

$$f(\Theta) = -\sum_{t=1}^{m} \log \mathrm{Prob}\left(\vec{n}_t \;\middle|\; \Theta, |\vec{n}_t|\right). \tag{18}$$

In the simplest case, we have simply fixed the number of components l in the Dirichlet mixture to a particular value and then estimated the $21l - 1$ parameters (twenty α_i values for each of the components, and $l - 1$ mixture coefficients). In other experiments, we tried to estimate l as well. Unfortunately, even for fixed l, there does not appear to be an efficient method of estimating these parameters that is guaranteed to always find the maximum likelihood estimate. However, a variant of the standard estimation-maximization (EM) algorithm for mixture density estimation works well in practice[7]. EM has been proved to result in closer and closer approximations to a local optimum with every iteration of the learning cycle; a global optimum, unfortunately, is not guaranteed (Dempster *et al.*, 1977)[8].

As the derivations that follow can become somewhat complex, we provide two tables in the Appendix to help the reader follow the derivations. Table 8 contains a summary of the notation we use and Table 9 contains an index to where certain key quantities are derived or defined.

In this section we give the derivation of the procedure to estimate the parameters of a mixture prior. As we will show, the case where the prior consists of a single density follows directly from the general case of a mixture. In the case of a mixture, we have two sets of parameters to estimate: the $\vec{\alpha}$ parameters for each component, and the q, or mixture coefficient, for each component. In the case of a single density, we estimate only the $\vec{\alpha}$ parameters.

In our practice, we estimate these parameters in a two-stage process: first we estimate the $\vec{\alpha}$, keeping the mixture coefficients q fixed, then we estimate the q, keeping the $\vec{\alpha}$ parameters fixed. This two-stage process is iterated until all estimates stabilize.

---

[7] An introduction to this method of mixture density estimation is given in the book by Duda and Hart (Duda and Hart, 1973). We have modified their procedure to estimate a mixture of Dirichlet rather than Gaussian densities.

[8] This method for parameter estimation has also been used for other problems in biosequence analysis (Lawrence and Reilly, 1990; Cardon and Stormo, 1992).

3.3.1 Deriving the $\vec{\alpha}$ parameters

Since we require that the α_i be strictly positive, and we want the parameters upon which we will do gradient descent to be unconstrained, we reparameterize, setting $\alpha_{j,i} = e^{w_{j,i}}$, where $w_{j,i}$ is an unconstrained real number. Then, the partial derivative of the objective function (equation 18) with respect to $w_{j,i}$ is

$$\frac{\partial f(\Theta)}{\partial w_{j,i}} = -\sum_{t=1}^{m} \frac{\partial \log \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial \alpha_{j,i}} \frac{\partial \alpha_{j,i}}{\partial w_{j,i}} \tag{19}$$

Here, we introduce Lemma 5 (the proof of which is found in the Appendix), giving

Lemma 5:

$$\frac{\partial \log \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial \alpha_{j,i}} = \text{Prob}\left(\vec{\alpha}_j \;\middle|\; \vec{n}, \Theta\right) \frac{\partial \log \text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{\partial \alpha_{j,i}}$$

to obtain

$$\frac{\partial f(\Theta)}{\partial w_{j,i}} = -\sum_{t=1}^{m} \text{Prob}\left(\vec{\alpha}_j \;\middle|\; \vec{n}_t, \Theta\right) \frac{\partial \log \text{Prob}\left(\vec{n}_t \mid \vec{\alpha}_j, |\vec{n}_t|\right)}{\partial \alpha_{j,i}} \frac{\partial \alpha_{j,i}}{\partial w_{j,i}} \tag{20}$$

Using the fact that $\frac{\partial \alpha_{j,i}}{\partial w_{j,i}} = \alpha_{j,i}$, and introducing Lemma 6 (the proof of which is found in the Appendix) giving

Lemma 6:

$$\frac{\partial \log \text{Prob}\left(\vec{n} \mid \vec{\alpha}, |\vec{n}|\right)}{\partial \alpha_i} = \Psi(|\vec{\alpha}|) - \Psi(|\vec{n}| + |\vec{\alpha}|) + \Psi(n_i + \alpha_i) - \Psi(\alpha_i) \tag{21}$$

we obtain

$$\frac{\partial f(\Theta)}{\partial w_{j,i}} = -\sum_{t=1}^{m} \alpha_{j,i} \text{Prob}\left(\vec{\alpha}_j \;\middle|\; \vec{n}_t, \Theta\right) \left(\Psi(|\vec{\alpha}_j|) - \Psi(|\vec{n}_t| + |\vec{\alpha}_j|) + \Psi(n_{t,i} + \alpha_{j,i}) - \Psi(\alpha_{j,i})\right) \tag{22}$$

In optimizing the $\vec{\alpha}$ parameters of the mixture, we do gradient descent on the weights $\vec{w}$, taking a step in the direction of the negative gradient (controlling the size of the step by the variable η, $0 \le \eta \ll 1$) during each iteration of the learning cycle. Thus, the gradient descent rule in the mixture case can now be defined as follows:

$$w_{j,i}^{new} := w_{j,i}^{old} - \eta \frac{\partial f(\Theta)}{\partial w_{j,i}} \tag{23}$$

$$:= w_{j,i}^{old} + \eta \sum_{t=1}^{m} \alpha_{j,i} \text{Prob}\left(\vec{\alpha}_j \;\middle|\; \vec{n}_t, \Theta\right)\left(\Psi(|\vec{\alpha}_j|) - \Psi(\vec{n}_t + |\vec{\alpha}_j|) + \Psi(n_{t,i} + \alpha_{j,i}) - \Psi(\alpha_{j,i})\right) \tag{24}$$

Now, letting $S_j = \sum_{t=1}^{m} \text{Prob}\left(\vec{\alpha}_j \mid \vec{n}_t, \Theta\right)$, this is

$$w_{j,i}^{new} := w_{j,i}^{old} + \eta\, \alpha_{j,i} \left(S_j \left(\Psi(|\vec{\alpha}_j|) - \Psi(\alpha_{j,i})\right) + \sum_{t=1}^{m} \text{Prob}\left(\vec{\alpha}_j \;\middle|\; \vec{n}_t, \Theta\right) \left(\Psi(n_{t,i} + \alpha_{j,i}) - \Psi(\vec{n}_t + |\vec{\alpha}_j|)\right)\right) \tag{25}$$

In the case of a single density, $\text{Prob}(\vec{\alpha} \mid \vec{n}, \Theta) = 1$ for all vectors $\vec{n}$, thus $S_j = \sum_{t=1}^{m} \text{Prob}\left(\vec{\alpha} \mid \vec{n}_t, \Theta\right) = m$, and the gradient descent rule for a single density can be written as

$$w_i^{new} := w_i^{old} + \eta\, \alpha_i \left(m \left(\Psi(|\vec{\alpha}|) - \Psi(\alpha_i)\right) + \sum_{t=1}^{m} \left(\Psi(n_{t,i} + \alpha_i) - \Psi(\vec{n}_t + |\vec{\alpha}|)\right)\right) \tag{26}$$

After each update of the weights, the $\vec{\alpha}$ parameters are reset, and the process continued until the change in the objective function falls below some pre-defined cutoff.

3.3.2 Mixture coefficient estimation

In the case of a mixture of Dirichlet densities, the mixture coefficients, q, of each component are also estimated. However, since we require that the mixture coefficients must be non-negative and sum to 1, we first reparameterize, setting $q_i = Q_i / |Q|$, where the Q_i are constrained to be strictly positive, and $|Q| = \sum_i Q_i$. As in the first stage, we want to maximize the probability of the data given the model, which is equivalent to minimizing the objective function (equation 18), $f(\Theta) = -\sum_{t=1}^m \log \text{Prob}\left(\vec{n}_t \mid \Theta, |\vec{n}_t|\right)$. In this stage, we take the derivative of f with respect to Q_i. However, instead of having to take iterative steps in the direction of the negative gradient, as we did in the first stage, we can set the derivative to zero, and solve for those $q_i = Q_i / |Q|$ that maximize the probability of the data. As we will see, however, the new q_i are a function of the previous q_i; thus, this estimation process must also be iterated.

Taking the gradient of f with respect to Q_i, we obtain

$$\frac{\partial f(\Theta)}{\partial Q_i} = -\sum_{t=1}^{m} \frac{\partial \log \text{Prob}\left(\vec{n}_t \mid \Theta, |\vec{n}_t|\right)}{\partial Q_i} \tag{27}$$

This allows us to focus on the partial derivative of the log likelihood of a single count vector with respect to Q_i. By Lemma 8 (the proof for which is found in Section A.8),

Lemma 8:

$$\frac{\partial \log \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} = \frac{\text{Prob}\left(\vec{\alpha}_i \mid \vec{n}, \Theta)\right)}{Q_i} - \frac{1}{|Q|}$$

When we sum over all observations $\vec{n}$, we obtain that in the case of a mixture,

$$\begin{aligned} \frac{\partial f(\Theta)}{\partial Q_i} &= -\sum_{t=1}^{m} \left(\frac{\text{Prob}\left(\vec{\alpha}_i \mid \vec{n}_t, \Theta)\right)}{Q_i} - \frac{1}{|Q|} \right) && (28) \\ &= \frac{m}{|Q|} - \frac{\sum_{t=1}^m \text{Prob}\left(\vec{\alpha}_i \mid \vec{n}_t, \Theta\right)}{Q_i} && (29) \end{aligned}$$

Since the gradient must vanish for those mixture coefficients giving the maximum likelihood, we set the gradient to zero, and solve. Thus, the maximum likelihood setting for q_i is

$$\begin{aligned} q_i &:= \frac{Q_i}{|Q|} && (30) \\ &:= \frac{1}{m} \sum_{t=1}^{m} \text{Prob}\left(\vec{\alpha}_i \,\middle|\, \vec{n}_t, \Theta\right)) && (31) \end{aligned}$$

Note that since $\sum_i \sum_{t=1}^m \text{Prob}\left(\vec{\alpha}_i \mid \vec{n}_t, \Theta\right) = \sum_{t=1}^m \sum_i \text{Prob}\left(\vec{\alpha}_i \mid \vec{n}_t, \Theta\right) = \sum_{t=1}^m 1 = m$, the mixture coefficients sum to 1, as required.

Since the reestimated mixture coefficients are functions of the old mixture coefficients, we iterate this process until the change in the objective function falls below the predefined cutoff.

In summary, when estimating the parameters of a mixture prior, we alternate between reestimating the $\vec{\alpha}$ parameters of each density in the mixture, by gradient descent on the $\vec{w}$, resetting $\alpha_{j,i} = e^{w_{j,i}}$ after each iteration, followed by re-estimating and resetting the mixture coefficients as described above, until the process converges.

4 Results

The problem of estimating expected distributions over the amino acids in the absence of large amounts of data is not unique to hidden Markov models. Thus other researchers have experimented with Dirichlet mixture priors, both those which we reported in (Brown *et al.*, 1993), and those which we developed and made available afterwards. In addition to the experiments we reported in (Brown *et al.*, 1993), and which we summarize below, three independent groups of researchers, (Tatusov *et al.*, 1994; Henikoff and Henikoff, 1995; Bailey and Elkan, 1995), used these mixtures in database search and discrimination experiments, while the work of Karplus (Karplus, 1995a; Karplus, 1995b) is more information theoretic, comparing the number of bits to encode the posterior probability estimates of the amino acids given different methods and different sample sizes.

4.1 HMM experiments

In our original paper on the use of Dirichlet mixture priors (Brown *et al.*, 1993), we described a series of experiments on building HMMs for the EF-hand motif. EF-hands are an approximately 29-residue structure present in cytosolic calcium-modulated proteins (Nakayama *et al.*, 1992; Persechini *et al.*, 1989; Moncrief *et al.*, 1990). We chose EF-hands to demonstrate the ability of mixture priors to compensate for limited sample sizes because the motif's small size allowed many experiments to be performed relatively rapidly. For these experiments we used the June 1992 database of EF-hand sequences maintained by Kretsinger and co-workers (Nakayama *et al.*, 1992). We extracted the EF-hand structures from each of the 242 sequences in the database, obtaining 885 EF-hand motifs having an average length of 29. HMM training sets were constructed by randomly extracting subsets of size 5, 10, 20, 40, 60, 80, and 100.

The Dirichlet priors we used for these experiments were derived from two sources of multiple alignments: a subset of alignments from the HSSP database suggested in (Sander and Schneider, 1991) and multiple alignments we generated using HMMs to model the kinase, globin, and elongation factor families (Haussler *et al.*, 1993; Krogh *et al.*, 1994).

Using the maximum likelihood procedure described in Section 3.3 we estimated the parameters of a one-component and a nine-component Dirichlet mixture density from the 5670 count vectors obtained from the HSSP multiple alignments. We call these Dirichlet mixtures *HSSP1*, and *HSSP9* respectively. Similar experiments were done for the HMM alignments, obtaining Dirichlet mixture priors with one component and nine components (*HMM1, HMM9*).

In addition to the priors we estimated via maximum likelihood estimation, we tested the effectiveness of some additional priors: the standard uniform prior called *Add-One* (see Section 1.3.1), priors obtained directly from amino acid distributions estimated by Lüthy, McLachlan, and Eisenberg (which we call the *LME* distributions) for nine different structural environments (1991), and a 29-component *EF-hand custom* prior in which each component is derived from a column in our EF-hand multiple alignment. The prior derived from the nine-component LME distributions was obtained by forming Dirichlet densities for each of the nine LME amino acid distributions with the same means as the original distributions. Since there is no measure of the expected variance around the mean associated with these distributions, we arbitrarily set the $|\vec{\alpha}|$ for each component to 10, and set the mixture coefficients uniformly. The 29-component EF-hand custom prior was designed to determine a bound on the best possible performance for any Dirichlet mixture for this family.

For each training set size and each prior, several HMMs were built using the method described in (Krogh *et al.*, 1994). We evaluated each HMM on a separate test set containing EF-hand sequences not in the training set, yielding an average negative log likelihood (*NLL*) score over all test sequences for each model. Lower scores represent more accurate models. For every combination of training sample size and prior used, we took the average test-set NLL score across all models, and the standard deviation of the test-set NLL scores.

In these experiments, the EF-hand custom prior performed the best, followed by HMM9, HSSP9, LME[9], HMM1 and HSSP1. Add-One performed the worst. For example, at 20 training sequences, the average test-set NLL score for HMMs trained using HMM9 was lower than the average NLL score for the HMMs trained using the Add-One prior for all training set sizes up to 60. Details of the results of these experiments are given in (Brown *et al.*, 1993).

[9] In retrospect, the high $|\vec{\alpha}|$ of the LME prior may have handicapped this density in these experiments; $|\vec{\alpha}|$ closer to 1 might have been more effective.

In our previous work, the NLL score has always been almost perfectly correlated with superior multiple alignments and database search. To further demonstrate the latter point, we tested some of the HMMs built from various priors on their ability to discriminate sequences containing the EF-hand domain from those not containing the domain. To do this we choose models built from training samples of sizes 5, 10, and 20, using the Add one, HMM1, HMM9 and EF-hand custom priors. For each sample size and prior, we built an HMM as above and then used it to search the SWISS-PROT database for sequences that contain the EF-hand motif, using the method described in (Krogh *et al.*, 1994). The results of these database discrimination experiments confirmed the ordering of the priors by NLL score. Unfortunately, only one test was done for each combination of sample size and prior, so the results are not as statistically significant as those for NLL score.

Finally, we note that in these experiments, data used to train HMM1 and HMM9 contained no EF-hand-specific proteins, yet these mixtures still produced a substantial increase in performance for the EF-Hand HMMs estimated using these priors. This confirmed that these priors do indeed capture some universal aspect of amino acid distributions that are meaningful across different protein families.

4.2 Experiments with other statistical models

Karplus's work, (Karplus, 1995a; Karplus, 1995b), compared the relative costs of encoding multiple alignments using the estimated posterior probabilities $\hat{p}_i$ of each amino acid i in samples of various sizes drawn from count vectors from the BLOCKS database (Henikoff and Henikoff, 1991) for several methods[10]. Karplus noted the sample size(s) for which each method was superior to the others, and whether a method's posterior probability estimate approaches the maximum-likelihood estimate in the limit, as the number of observations grows unboundedly large. Karplus compared several methods:

1. *Zero-offset* (of which one variant is the popular 'Add-One') where a small positive constant is added to all amino acid counts.
2. *Pseudocounts*, in which a different positive value is added for each amino acid, rather than one fixed constant for all amino acids.
3. *Gribskov profile*, or *average score method*, where the scores are logarithmic, comparing the probability estimate of an amino acid in a particular context to the global (or background) probability of that amino acid. This method has been used by various researchers (Tatusov *et al.*, 1994; Gribskov *et al.*, 1984), employing any of several scoring matrices, such as the popular Dayhoff (Dayhoff *et al.*, 1978) and Blosum (Henikoff and Henikoff, 1992) matrices.
4. *Substitution matrices*, which encode the cost for substituting amino acid i for amino acid j, and comparing two variants on this basic technique, adding scaled counts and/or pseudocounts. These methods are similar to those employed in method 3 above, but use matrix multiplication to compute Prob(amino acid i) rather than $\log$ Prob(amino acid i) scores.
5. *Dirichlet mixture priors*, with several mixture priors compared against each other.

For this problem, Dirichlet mixtures were always superior for sample sizes 2 or more, and were very close to optimal for sample size 1 (where substitution matrices were optimal), and sample size zero (where pseudocount methods based on background frequency were optimal).

Tatusov, Altschul and Koonin propose a technique in (Tatusov *et al.*, 1994) for iterative refinement of protein profiles that is able to start with very few aligned sequences (or even a single protein segment), and repeatedly compute a probability distribution over the amino acids for each column in the alignment, search a sequence database for protein segments that match the amino acid distributions specified by the model, according to some criterion, and multiply align all new protein segments to the model, until no new sequences scoring above a given cutoff are found. They tested several methods for estimating the expected distributions over the amino acids in the first part of this iterative model-building process. The resulting models were then tested at database discrimination tasks, and their relative performances compared. The methods compared in this paper were:

[10]More recently, Karplus duplicated these experiments on columns drawn from the HSSP protein database, and confirmed a similar ordering of these methods. Also, the 9-component Dirichlet density reported in Section 2 of this paper performs very well in his tests for all sample sizes tested.

1. *Average score method*, incorporating the use of amino acid substitution matrices, such as PAM (Altschul, 1991), or BLOSUM (Henikoff and Henikoff, 1992) (identical to the third method tested by Karplus);
2. *Log-odds Bayesian prediction using pseudocounts* (identical to the second method tested by Karplus);
3. *Data-dependent pseudocount method*, where the pseudocounts are calculated using a substitution matrix (equivalent to one of the substitution matrix methods tested by Karplus);
4. *Dirichlet mixture method*, which incorporates a Dirichlet mixture prior into the log-odds Bayesian prediction method.

Tatusov et al. reported that the use of Dirichlet mixture priors (specifically, a nine-component mixture prior estimated from the Blocks database (Henikoff and Henikoff, 1991) quite similar to Blocks9 given in Tables 1 and 2, resulted in protein models with the highest accuracy in database discrimination, yielding the fewest false negatives and false positives overall of any of the methods compared.

Steven and Jorja Henikoff conducted a series of tests on the same methods, using a testing strategy similar to that described in (Henikoff and Henikoff, 1994) and confirm these results (personal communication). Good results with these mixtures are also reported by (Wang *et al.*, To appear), who, in a related set of experiments, created an expanded set of blocks using the same mixtures used in (Tatusov *et al.*, 1994), and then used these blocks to classify protein sequences.

In (Bailey and Elkan, 1995), the authors report several extensions to their motif-finding tool MEME which incorporate prior information into the parameter estimation process. While the authors did not compare different methods for computing posterior estimates of amino acid densities (the other prior information introduced concerned motif width, presence or absence of the motif in sequences being searched, and whether, as in the case of DNA sequences, the motif is expected to be a palindrome), they reported that the use of a Dirichlet mixture prior (in this case, a 30-component mixture estimated from the BLOCKS database) boosted their protein database search accuracy significantly, especially in the case where few (< 20) training sequences were available.

5 Implementation details

Implementing Dirichlet mixture priors for use in hidden Markov models or other stochastic models of biological sequences is not difficult, but there are many details that can cause problems if not handled carefully.

This section will split the implementation details into two groups: those that are essential for getting working Dirichlet mixture code (Section 5.1), and those that increase efficiency, but are not essential (Section 5.2).

5.1 Essential details

In Section 3.2, we gave the formulas for computing the amino acid probabilities in the cases of a single density (equation 11) and of a mixture density (equation 15).

For a single Dirichlet component, the estimation formula is trivial:

$$\hat{p}_i = \frac{n_i + \alpha_i}{|\vec{n}| + |\vec{\alpha}|} \ , \tag{32}$$

and no special care is needed in the implementation. For the case of a multi-component mixture, the implementation is not quite so straightforward.

As we showed in the derivation of equation 15,

$$\hat{p}_i = \sum_{j=1}^{l} \text{Prob}\left(\vec{\alpha}_j \;\middle|\; \vec{n}, \Theta\right) \frac{n_i + \alpha_{j,i}}{|\vec{n}| + |\vec{\alpha}_j|} \ . \tag{33}$$

The interesting part for computation comes in computing $\text{Prob}\left(\vec{\alpha}_j \mid \vec{n}, \Theta\right)$, whose formula is repeated here from Equation 16:

$$\text{Prob}\left(\vec{\alpha}_j \;\middle|\; \vec{n}, \Theta\right) = \frac{q_j \text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)} \ . \tag{34}$$

We can expand $\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)$ using equation 17 to obtain

$$\text{Prob}\left(\vec{\alpha}_j \;\middle|\; \vec{n}, \Theta\right) = \frac{q_j \text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{\sum_{k=1}^{l} q_k \text{Prob}\left(\vec{n} \mid \vec{\alpha}_k, |\vec{n}|\right)} \ . \tag{35}$$

Note that this is a simple renormalization of $q_j \text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)$ to sum to one. Rather than carry the normalization through all the equations, we can work directly with $\text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)$, and put everything back together at the end.

First, we can expand it using Lemma 3 (the proof of which is found in Section A.3):

$$\text{Prob}\left(\vec{n} \;\middle|\; \vec{\alpha}_j, |\vec{n}|\right) = \frac{\Gamma(|\vec{n}|+1)\Gamma(|\vec{\alpha}_j|)}{\Gamma(|\vec{n}|+|\vec{\alpha}_j|)} \prod_i \frac{\Gamma(n_i + \alpha_{j,i})}{\Gamma(n_i+1)\Gamma(\alpha_{j,i})} \ . \tag{36}$$

If we rearrange some terms, we obtain

$$\text{Prob}\left(\vec{n} \;\middle|\; \vec{\alpha}_j, |\vec{n}|\right) = \frac{\prod_i \Gamma(n_i + \alpha_{j,i})}{\Gamma(|\vec{n}|+|\vec{\alpha}_j|)} \frac{\Gamma(|\vec{\alpha}_j|)}{\prod_i \Gamma(\alpha_{j,i})} \frac{\Gamma(|\vec{n}|+1)}{\prod_i \Gamma(n_i+1)} \ . \tag{37}$$

The first two terms are most easily expressed using the Beta function: $\text{B}(x) = \frac{\prod_i \Gamma(x_i)}{\Gamma(|\vec{x}|)}$, where, as usual, $|\vec{x}| = \sum_i x_i$. This simplifies the expression to

$$\text{Prob}\left(\vec{n} \;\middle|\; \vec{\alpha}_j, |\vec{n}|\right) = \frac{\text{B}\left(\vec{n}+\vec{\alpha}_j\right)}{\text{B}\left(\vec{\alpha}_j\right)} \frac{\Gamma(|\vec{n}|+1)}{\prod_i \Gamma(n_i+1)} \ . \tag{38}$$

The remaining Gamma functions are not easily expressed with a Beta function, but they don't need to be. Since they depend only on $\vec{n}$ and not on j, when we do the normalization to make the $\text{Prob}\left(\vec{\alpha}_j \mid \vec{n}, \Theta\right)$ sum to one, this term will cancel out, giving us

$$\text{Prob}\left(\vec{\alpha}_j \;\middle|\; \vec{n}, \Theta\right) = \frac{q_j \frac{\text{B}(\vec{n}+\vec{\alpha}_j)}{\text{B}(\vec{\alpha}_j)}}{\sum_{k=1}^{l} q_k \frac{\text{B}(\vec{n}+\vec{\alpha}_k)}{\text{B}(\vec{\alpha}_k)}} \ . \tag{39}$$

Plugging this formula into Equation 33 gives us

$$\hat{p}_i = \frac{\sum_{j=1}^{l} q_j \frac{\mathrm{B}(\vec{n}+\vec{\alpha}_j)}{\mathrm{B}(\vec{\alpha}_j)} \frac{n_i+\alpha_{j,i}}{|\vec{n}|+|\vec{\alpha}_j|}}{\sum_{k=1}^{l} q_k \frac{\mathrm{B}(\vec{n}+\vec{\alpha}_k)}{\mathrm{B}(\vec{\alpha}_k)}} . \tag{40}$$

Since the denominator of Equation 40 is independent of i, we can compute $\hat{p}_i$ by normalizing

$$X_i = \sum_{j=1}^{l} q_j \frac{\mathrm{B}(\vec{n}+\vec{\alpha}_j)}{\mathrm{B}(\vec{\alpha}_j)} \frac{n_i+\alpha_{j,i}}{|\vec{n}|+|\vec{\alpha}_j|} \tag{41}$$

to sum to one. That is,

$$\hat{p}_i = \frac{X_i}{\sum_{k=1}^{20} X_k} . \tag{42}$$

The biggest problem that implementors run into is that these Beta functions can get very large or very small—outside the range of the floating-point representation of most computers. The obvious solution is to work with the logarithm of the Beta function:

$$\begin{aligned} \log \mathrm{B}(x) &= \log \frac{\prod_i \Gamma(x(i))}{\Gamma(|\vec{x}|)} \\ &= \sum_i \log \Gamma(x(i)) - \log \Gamma(|\vec{x}|) . \end{aligned}$$

Most libraries of mathematical routines include the **lgamma** function which implements $\log \Gamma(x)$, and so using the logarithm of the Beta function is not difficult.

We could compute each X_i using only the logarithmic notation, but it turns out to be slightly more convenient to use the logarithms just for the Beta functions:

$$\begin{aligned} X_i &= \sum_j q_j \frac{\mathrm{B}(\vec{\alpha}_j+\vec{n})}{\mathrm{B}(\vec{\alpha}_j)} \frac{\alpha_{j,i}+n_i}{|\vec{\alpha}_j|+|\vec{n}|} \\ &= \sum_j q_j e^{(\log \mathrm{B}(\vec{\alpha}_j+\vec{n}) - \log \mathrm{B}(\vec{\alpha}_j))} \frac{\alpha_{j,i}+n_i}{|\vec{\alpha}_j|+|\vec{n}|} . \end{aligned}$$

Some care is needed in the conversion from the logarithmic representation back to floating-point, since the ratio of the Beta functions may be so large or so small that it cannot be represented as floating-point numbers. Luckily, we do not really need to compute X_i, only $\hat{p}_i = X_i / \sum_{k=1}^{20} X_k$. This means that we can multiply X_i by any constant and the normalization will eliminate the constant. Equivalently, we can freely subtract a constant (independent of j and i) from $\log \mathrm{B}(\vec{\alpha}_j+\vec{n}) - \log \mathrm{B}(\vec{\alpha}_j)$ before converting back to floating-point.

If we choose the constant to be $\max_j (\log \mathrm{B}(\vec{\alpha}_j+\vec{n}) - \log \mathrm{B}(\vec{\alpha}_j))$, then the largest logarithmic term will be zero, and all the terms will be reasonable.[11]

5.2 Efficiency improvements

The previous section gave simple computation formulas for $\hat{p}_i$ (Equations 42 and 41). When computations of $\hat{p}_i$ are done infrequently (for example, for profiles, where $\hat{p}_i$ only needs to be computed once for each column of the profile), those equations are perfectly adequate.

When recomputing $\hat{p}_i$ frequently, as may be done in a Gibbs sampling program or training a hidden Markov model, it is better to have a slightly more efficient computation. Since most of the computation time is spent in the **lgamma** function used for computing the log Beta functions, the biggest efficiency gains come from avoiding the **lgamma** computations.

[11] We could still get floating-point underflow to zero for some terms, but the $\hat{p}$ computation will still be about as good as can be done within floating-point representation.

If we assume that the $\alpha_{j,i}$ and q_j values change less often than the values for $\vec{n}$ (which is true of almost every application), then it is worthwhile to precompute $\log \mathrm{B}(\vec{\alpha}_j)$, cutting the computation time almost in half.

If the n_i values are mainly small integers (0 is common in all the applications we've looked at), then it is worth pre-computing $\log \Gamma(\alpha_{j,i})$, $\log \Gamma(\alpha_{j,i} + 1)$, $\log \Gamma(\alpha_{j,i} + 2)$, and so on, out to some reasonable value. Precomputation should also be done for $\log \Gamma(|\vec{\alpha}_j|)$, $\log \Gamma(|\vec{\alpha}_j| + 1)$, $\log \Gamma(|\vec{\alpha}_j| + 2)$, and so forth. If all the $\vec{n}$ values are small integers, this precomputation almost eliminates the **`lgamma`** function calls.

In some cases, it may be worthwhile to build a special-purpose implementation of $\log \Gamma(x)$ that caches all calls in a hash table, and does not call **`lgamma`** for values of x that it has seen before. Even larger savings can be had when x is close to previously computed values, by using interpolation rather than calling **`lgamma`**.

6 Conclusions and Future Research

Dirichlet mixture priors have been demonstrated to be more effective at forming accurate estimates for expected amino acid distributions than substitution matrix-based methods, pseudocounts, and other such methods. In particular, the method presented in this paper has been shown to fix two primary weaknesses of substitution matrix-based methods: focusing only on the relative frequency of the amino acids while ignoring the actual number of amino acids observed, and having fixed substitution probabilities for each amino acid. One of the potentially most problematic consequences of these drawbacks is that substitution matrix-based methods do not produce estimates that are conserved, or mostly conserved, where the evidence is clear that an amino acid is conserved.

The method presented here addresses these issues. Given abundant training data, the estimate produced by these methods is very close to the actual frequencies observed. When little data is available, the amino acids predicted are those that are known to be associated in different contexts with the amino acids observed. In particular, when evidence exists that a particular amino acid is conserved at a given position, the expected amino acid estimates reflect this preference.

In database search for homologous sequences, Dirichlet mixtures have been shown to maximize sensitivity without sacrificing specificity. As a result experiments using Dirichlet mixtures to estimate the expected amino acid distributions in a variety of statistical models for proteins result in fewer false negatives and false positives than when other methods are used.

The methods employed to estimate and use these mixtures have been shown to be firmly based on Bayesian statistics. While no biological knowledge has been introduced into the parameter-estimation process, the mixture priors that result agree with accepted biological understanding.

In order to be able to use these mixtures to find true remote homologs, mixtures should be estimated on alignments containing more distant homologs, rather than estimated from databases where fairly close homologs are aligned, as is the case for both the BLOCKS and HSSP databases. Another key area needing research is the weighting of sequences to remove bias. Previous work has concentrated on relative weighting schemes, but the total weight is also relevant when using Dirichlet mixtures.

Since the method for estimating these mixtures (EM) is sensitive to the initial parameter settings, we are exploring heuristics that enable us to explore the parameter space more effectively, and obtain better mixtures. We are also exploring methods to compensate for the assumption that each column is generated independently, which although simplifying the math, is without biological basis. However, as the detailed analysis of Karplus (Karplus, 1995a; Karplus, 1995b) shows, the Dirichlet mixtures already available are close to optimal as far as their capacity for assisting in computing estimates of amino acid distributions, given a single column context. Thus, further work in this area will perhaps profit by focusing on obtaining information from relationships among the sequences (for instance, as revealed in a phylogenetic tree), or in inter-columnar interactions.

Acknowledgments

We gratefully acknowledge the input and suggestions of Stephen Altschul, Tony Fink, Lydia Gregoret, Steven and Jorja Henikoff, Graeme Mitchison, and Chris Sander. Special thanks to friends at Laforia, Université de Pierre et Marie Curie, in Paris, and the Biocomputing Group at the European Molecular Biology Laboratory at Heidelberg, who provided workstations, support, and scientific inspiration during the early stages of writing this paper. This work was supported in part by NSF grants CDA-9115268, IRI-9123692, and BIR-9408579; DOE grant 94-12-048216, ONR grant N00014-91-J-1162, NIH grant GM17129, a grant from the Danish Natural Science Research Council, a National Science Foundation Graduate Research Fellowship, and funds granted by the UCSC Division of Natural Sciences. This paper is dedicated to the memory of Tal Grossman, a dear friend and a true mensch.

References

Altschul, Stephen F.; Gish, Warren; Miller, Webb; Meyers, Eugene W.; and Lippman, David J. 1990. Basic local alignment search tool. *JMB* 215:403–410.

Altschul, Stephen F. 1991. Amino acid substitution matrices from an information theoretic perspective. *JMB* 219:555–565.

Asai, K.; Hayamizu, S.; and Onizuka, K. 1993. HMM with protein structure grammar. In *Proceedings of the Hawaii International Conference on System Sciences*, Los Alamitos, CA. IEEE Computer Society Press. 783–791.

Bailey, Timothy L. and Elkan, Charles 1995. The value of prior knowledge in discovering motifs with MEME. In *ISMB-95*, Cambridge, England. ??

Baldi, P. and Chauvin, Y. 1994. Smooth on-line learning algorithms for hidden Markov models. *Neural Computation* 6(2):305–316.

Baldi, P.; Chauvin, Y.; Hunkapiller, T.; and McClure, M. A. 1992. Adaptive algorithms for modeling and analysis of biological primary sequence information. Technical report, Net-ID, Inc., 8 Cathy Place, Menlo Park, CA 94305.

Barton, G. J. and Sternberg, M. J. 1990. Flexible protein sequence patterns: A sensitive method to detect weak structural similarities. *JMB* 212(2):389–402.

Berger, J. 1985. *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, New York.

Berhardo, J.M. and Smith, A.F.M. 1994. *Bayesian Theory.* John Wiley and Sons, first edition.

Bowie, J. U.; Lüthy, R.; and Eisenberg, D. 1991. A method to identify protein sequences that fold into a known three-dimensional structure. *Science* 253:164–170.

Brown, M. P.; Hughey, R.; Krogh, A.; Mian, I. S.; Sjölander, K.; and Haussler, D. 1993. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In Hunter, L.; Searls, D.; and Shavlik, J., editors 1993, *ISMB-93*, Menlo Park, CA. AAAI/MIT Press. 47–55.

Bucher, Philipp; Karplus, Kevin; Moeri, Nicolas; and Hoffman, Kay 1996. A flexible motif search technique based on generalized profiles. *Computers and Chemistry* 20(1):3–24.

Cardon, L. R. and Stormo, G. D. 1992. Expectation maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *JMB* 223:159–170.

Casari, G.; Andrade, M.; Bork, P.; Boyle, J.; Daruvar, A.; Ouzounis, C.; Schneider, R.; Tamames, J.; Valencia, A.; and Sander, C. 1995. Scientific correspondence to nature. *Nature.*

Churchill, G. A. 1989. Stochastic models for heterogeneous DNA sequences. *Bull Math Biol* 51:79–94.

Claverie, Jean-Michael 1993. Information enhancement methods for large scale sequence analysis. *Computers and Chemistry* 17(2):191–201.

Claverie, Jean-Michael 1994. Some useful statistical properties of position-weight matrices. *Computers and Chemistry* 18(3):287–294.

Dayhoff, M. O.; Schwartz, R. M.; and Orcutt, B. C. 1978. A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure.* National Biomedical Research Foundation, Washington, D. C. chapter 22, 345–358.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *J. Roy. Statist. Soc. B* 39:1–38.

Doolittle, R. F. 1986. *Of URFs and ORFs: A primer on how to analyze derived amino acid sequences.* University Science Books, Mill Valley, California.

Duda, R. O. and Hart, P. E. 1973. *Pattern Classification and Scene Analysis.* Wiley, New York.

Gradshteyn, I. S. and Ryzhik, I. M. 1965. *Table of Integrals, Series, and Products.* Academic Press, fourth edition.

Gribskov, M.; Devereux, J.; and Burgess, R. 1984. The codon preference plot: Graphic analysis of protein coding sequences and prediction of gene expression. *NAR* 12:539–549.

Gribskov, Michael; McLachlan, Andrew D.; and Eisenberg, David 1987. Profile analysis: Detection of distantly related proteins. *PNAS* 84:4355–4358.

Gribskov, M.; Lüthy, R.; and Eisenberg, D. 1990. Profile analysis. *Methods in Enzymology* 183:146–159.

Haussler, D.; Krogh, A.; Mian, I. S.; and Sjölander, K. 1993. Protein modeling using hidden Markov models: Analysis of globins. In *Proceedings of the Hawaii International Conference on System Sciences*, volume 1, Los Alamitos, CA. IEEE Computer Society Press. 792–802.

Henikoff, Steven and Henikoff, Jorja G. 1991. Automated assembly of protein blocks for database searching. *NAR* 19(23):6565–6572.

Henikoff, Steven and Henikoff, Jorja G. 1992. Amino acid substitution matrices from protein blocks. *PNAS* 89:10915–10919.

Henikoff, Steven and Henikoff, Jorja G. 1994. Position-based sequence weights. *JMB* 243(4):574–578.

Henikoff, Steven and Henikoff, Jorja G. 1995. Personal communication.

Henikoff, Steven; Wallace, James C.; and Brown, Joseph P. 1990. Finding protein similarities with nucleotide sequence databases. *Methods in Enzymology* 183:111–132.

Hughey, Richard 1993. Massively parallel biosequence analysis. Technical Report UCSC-CRL-93-14, University of California, Santa Cruz, CA.

Karplus, Kevin 1995a. Regularizers for estimating distributions of amino acids from small samples. In *ISMB-95*, Cambridge, England.

Karplus, Kevin 1995b. Regularizers for estimating distributions of amino acids from small samples. Technical Report UCSC-CRL-95-11, University of California, Santa Cruz. URL ftp://ftp.cse.ucsc.edu/pub/tr/ucsc-crl-95-11.ps.Z.

Krogh, A.; Brown, M.; Mian, I. S.; Sjölander, K.; and Haussler, D. 1994. Hidden Markov models in computational biology: Applications to protein modeling. *JMB* 235:1501–1531.

Lawrence, C. E. and Reilly, A. A. 1990. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins* 7:41–51.

Lüthy, R.; McLachlan, A. D.; and Eisenberg, D. 1991. Secondary structure-based profiles: Use of structure-conserving scoring table in searching protein sequence databases for structural similarities. *Proteins: Structure, Function, and Genetics* 10:229–239.

Moncrief, N. D.; Kretsinger, R. H.; and Goodman, M. 1990. Evolution of EF-hand calcium-modulated proteins. I. relationships based on amino acid sequences. *Journal of Molecular Evolution* 30:522–562.

Nakayama, S.; Moncrief, N. D.; and Kretsinger, R. H. 1992. Evolution of EF-hand calcium-modulated proteins. ii. domains of several subfamilies have diverse evolutionary histories. *Journal of Molecular Evolution* 34:416–448.

Nowlan, S. 1990. Maximum likelihood competitive learning. In Touretsky, D., editor 1990, *Advances in Neural Information Processing Systems*, volume 2. Morgan Kaufmann. 574–582.

Persechini, A.; Moncrief, N. D.; and Kretsinger, R. H. 1989. The EF-hand family of calcium-modulated proteins. *Trends in Neurosciences* 12(11):462–467.

R.D.Fleischmann, 1995. Whole-genome random sequencing and assemply of *haemophilus influenzae rd*. *Science* 269:496–512.

Sander, C. and Schneider, R. 1991. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins* 9(1):56–68.

Santner, T. J. and Duffy, D. E. 1989. *The Statistical Analysis of Discrete Data*. Springer Verlag, New York.

Sibbald, P. and Argos, P. 1990. Weighting aligned protein or nucleic acid sequences to correct for unequal representation. *JMB* 216:813–818.

Stultz, C. M.; White, J. V.; and Smith, T. F. 1993. Structural analysis based on state-space modeling. *Protein Science* 2:305–315.

Tatusov, Roman L.; Altschul, Stephen F.; and Koonin, Eugen V. 1994. Detection of conserved segments in proteins: Iterative scanning of sequence databases with alignment blocks. *PNAS* 91:12091–12095.

Thompson, Julie D.; Higgins, Desmond G.; and Gibson, Toby J. 1994a. Improved sensitivity of profile searches through the use of sequence weights and gap excision. *CABIOS* 10(1):19–29.

Thompson, Julie D.; Higgins, Desmond G.; and Gibson, Toby J. 1994b. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties, and weight matrix choice. *NAR* 22(22):4673–4680.

Wang, Jason T.L.; Marr, Thomas G.; Shasha, Dennis; Shapiro, Bruce; Chirn, Gung-Wei; and Lee, T.Y. ppear. Complementary classification approaches for protein sequences. *Protein Engr.*

Waterman, M. S. and Perlwitz, M. D. 1986. Line geometries for sequence comparisons. *Bull. Math. Biol.* 46:567–577.

White, James V.; Stultz, Collin M.; and Smith, Temple F. 1994. Protein classification by stochastic modeling and optimal filtering of amino-acid sequences. *Mathematical Biosciences* 119:35–75.

7 Tables

Blocks9									
	Blocks9.1	Blocks9.2	Blocks9.3	Blocks9.4	Blocks9.5	Blocks9.6	Blocks9.7	Blocks9.8	Blocks9.9
q	0.178091	0.056591	0.0960191	0.0781233	0.0834977	0.0904123	0.114468	0.0682132	0.234585
$\|\vec{\alpha}\|$	1.180650	1.355830	6.664360	2.081410	2.081010	2.568190	1.766060	4.987680	0.099500
A	0.270671	0.021465	0.561459	0.070143	0.041103	0.115607	0.093461	0.452171	0.005193
C	0.039848	0.010300	0.045448	0.011140	0.014794	0.037381	0.004737	0.114613	0.004039
D	0.017576	0.011741	0.438366	0.019479	0.005610	0.012414	0.387252	0.062460	0.006722
E	0.016415	0.010883	0.764167	0.094657	0.010216	0.018179	0.347841	0.115702	0.006121
F	0.014268	0.385651	0.087364	0.013162	0.153602	0.051778	0.010822	0.284246	0.003468
G	0.131916	0.016416	0.259114	0.048038	0.007797	0.017255	0.105877	0.140204	0.016931
H	0.012391	0.076196	0.214940	0.077000	0.007175	0.004911	0.049776	0.100358	0.003647
I	0.022599	0.035329	0.145928	0.032939	0.299635	0.796882	0.014963	0.550230	0.002184
K	0.020358	0.013921	0.762204	0.576639	0.010849	0.017074	0.094276	0.143995	0.005019
L	0.030727	0.093517	0.247320	0.072293	0.999446	0.285858	0.027761	0.700649	0.005990
M	0.015315	0.022034	0.118662	0.028240	0.210189	0.075811	0.010040	0.276580	0.001473
N	0.048298	0.028593	0.441564	0.080372	0.006127	0.014548	0.187869	0.118569	0.004158
P	0.053803	0.013086	0.174822	0.037661	0.013021	0.015092	0.050018	0.097470	0.009055
Q	0.020662	0.023011	0.530840	0.185037	0.019798	0.011382	0.110039	0.126673	0.003630
R	0.023612	0.018866	0.465529	0.506783	0.014509	0.012696	0.038668	0.143634	0.006583
S	0.216147	0.029156	0.583402	0.073732	0.012049	0.027535	0.119471	0.278983	0.003172
T	0.065438	0.018153	0.445586	0.071587	0.035799	0.088333	0.065802	0.358482	0.003690
V	0.065438	0.036100	0.227050	0.042532	0.180085	0.944340	0.025430	0.661750	0.002967
W	0.003758	0.071770	0.029510	0.011254	0.012744	0.004373	0.003215	0.061533	0.002772
Y	0.009621	0.419641	0.121090	0.028723	0.026466	0.016741	0.018742	0.199373	0.002686

Table 1: Parameters of Mixture Prior Blocks9

This table contains the parameters defining a nine-component mixture prior estimated on unweighted columns from the Blocks database. The first row gives the mixture coefficient (q_j) for each component. The second row gives the $|\vec{\alpha}| = \sum_i \alpha_i$ for each component. This value reflects how peaked the distribution is around the mean. The higher the value of $|\vec{\alpha}|$, the lower the variance around the mean. Rows A (alanine) through Y (tyrosine) contain the values of each of the components' $\vec{\alpha}$ parameters for that amino acid.

This mixture is available via anonymous ftp from our ftp site, ftp.cse.ucsc.edu, and on our web site at http://www.cse.ucsc.edu/research/compbio.

Analysis of 9-Component Dirichlet Mixture Prior Blocks9								
Comp.	Ratio (r) of amino acid frequency relative to background frequency							
	$8 \leq r$	$4 \leq r \leq 8$	$2 \leq r \leq 4$	$1 \leq r \leq 2$	$1/2 \leq r < 1$	$1/4 \leq r < 1/2$	$1/8 \leq r < 1/4$	$r < 1/8$
1			SAT	CGP	NVM	QHRIKFLDW	EY	
2	Y	FW	H		LM	NQICVSR	TPAKDGE	
3			QE	KNRSHDTA	MPYG	VLIWCF		
4		KR	Q	H	NETMS	PWYALGVCI	DF	
5		LM	I	FV		WYCTQ	APHR	KSENDG
6		IV		LM	CTA	F	YSPWN	EQKRDGH
7		D	EN	QHS	KGPTA	RY	MVLFWIC	
8			M	IVLFTYCA	WSHQRNK	PEG	D	
9			PGW	CHRDE	NQKFYTLAM	SVI		

Table 2: Preferred amino acids of Blocks9

The function used to compute the ratio of the frequency of amino acid i in component j relative to the background frequency predicted by the mixture as a whole is $\frac{\alpha_{j,i}/|\vec{\alpha}_j|}{\sum_k q_k \alpha_{k,i}/|\vec{\alpha}_k|}$.

An analysis of the amino acids favored by each component reveals the following:

1. Component 1 favors small neutral residues.
2. Component 2 favors the aromatics.
3. Component 3 gives high probability to most of the polar residues (except for C, Y, and W). Since cysteine can play two roles, either as a disulfide, or as a free thiol, in this component it is apparently appearing in its disulfide role.
4. Component 4 gives high probability to positively charged amino acids (especially K and R) and Q—favoring residues with long sidechains that can function as hydrogen donors.
5. Component 5 gives high probability to residues that are both large and non-polar.
6. Component 6 prefers I and V (aliphatic residues commonly found in Beta sheets), and allows substitutions with L and M.
7. Component 7 gives high probability to negatively charged residues, allowing substitutions with certain of the hydrophilic polar residues.
8. Component 8 gives high probability to methionine, but allows substitution with most neutral residues, especially the aliphatics.
9. Component 9 gives high probability to distributions peaked around individual amino acids (especially P, G, W, and C).

Posterior Probability of the components of Blocks9									
# Ile	Blocks9.1	Blocks9.2	Blocks9.3	Blocks9.4	Blocks9.5	Blocks9.6	Blocks9.7	Blocks9.8	Blocks9.9
1	0.0550	0.0238	0.0339	0.0200	0.1941	0.4529	0.0157	0.1215	0.0831
2	0.0548	0.0222	0.0108	0.0142	0.1738	0.4843	0.0122	0.0668	0.1609
3	0.0547	0.0212	0.0042	0.0111	0.1538	0.4656	0.0102	0.0383	0.2409
4	0.0537	0.0200	0.0019	0.0090	0.1356	0.4311	0.0088	0.0231	0.3168
5	0.0520	0.0188	0.0009	0.0075	0.1197	0.3929	0.0076	0.0146	0.3860
6	0.0500	0.0176	0.0005	0.0063	0.1059	0.3558	0.0067	0.0096	0.4477
7	0.0478	0.0165	0.0003	0.0053	0.0941	0.3216	0.0059	0.0065	0.5020
8	0.0455	0.0154	0.0002	0.0046	0.0839	0.2909	0.0053	0.0046	0.5496
9	0.0433	0.0144	0.0001	0.0040	0.0753	0.2637	0.0047	0.0033	0.5913
10	0.0412	0.0135	0.0001	0.0035	0.0678	0.2396	0.0042	0.0024	0.6277

Table 3: The posterior probability of each component (Prob $\left(\vec{\alpha}_j \mid \vec{n}, \Theta\right)$), equation 16) in Blocks9 given 1 to 10 isoleucines. Initially, component 6, which favors I and V, is most likely. But, as more isoleucines are seen without any valines, component 9, which favors distributions peaked around a single residue, becomes more likely.

	Methods used to estimate amino acid probabilities					
	Pseudocount methods		Substitution Matrices		Dirichlet densities	
Amino Acid	Add_one	Add_share	Blosum62	Subst	1-comp	Blocks9
A	0.047619	0.025000	0.036775	0.023388	0.042183	0.037347
C	0.047619	0.025000	0.037993	0.007416	0.010497	0.009717
D	0.047619	0.025000	0.019712	0.005109	0.025597	0.008526
E	0.047619	0.025000	0.019217	0.008594	0.029950	0.012252
F	0.047619	0.025000	0.054988	0.022703	0.021800	0.027068
G	0.047619	0.025000	0.015991	0.008718	0.035132	0.012359
H	0.047619	0.025000	0.018971	0.004003	0.012552	0.006202
I	**0.095238**	**0.525000**	**0.232458**	**0.565594**	**0.518022**	**0.470946**
K	0.047619	0.025000	0.023046	0.009486	0.029355	0.014450
L	**0.047619**	**0.025000**	**0.098515**	**0.105160**	**0.046082**	**0.117353**
M	0.047619	0.025000	0.085919	0.025844	0.014432	0.029961
N	0.047619	0.025000	0.019068	0.007490	0.022574	0.010000
P	0.047619	0.025000	0.022365	0.006738	0.018882	0.008189
Q	0.047619	0.025000	0.022265	0.006414	0.019882	0.010463
R	0.047619	0.025000	0.020627	0.008294	0.025707	0.012287
S	0.047619	0.025000	0.025767	0.012484	0.034161	0.019641
T	0.047619	0.025000	0.045341	0.022306	0.030533	0.027728
V	**0.047619**	**0.025000**	**0.140556**	**0.137617**	**0.039777**	**0.148299**
W	0.047619	0.025000	0.023772	0.002822	0.006091	0.003858
Y	0.047619	0.025000	0.036653	0.009820	0.016790	0.013352

Table 4: Estimated amino acid probabilities using various methods, given one isoleucine.

Tables 4, 5, 6 and 7 give amino acid probability estimates produced by different methods, given a varying number of isoleucines observed (and no other amino acids). Methods used to estimate these probabilities are *Add_one* which adds 1 to each count, and then renormalizes; *Add_share*, which adds 0.05 to each count, and renormalizes; *Blosum62*, which does Gribskov average score (Gribskov *et al.*, 1987) using the Blosum-62 matrix (Henikoff and Henikoff, 1992) (natural log base, 3 decimal places); *Subst*, which does matrix multiply with an optimized matrix. *1-comp*, which is a single-component Dirichlet density optimized for the Blocks database (Karplus, 1995a); *Blocks9*, which is the nine-component Dirichlet mixture given in Tables 1 and 2.

	Methods used to estimate amino acid probabilities					
	Pseudocount methods		Substitution Matrices		Dirichlet densities	
Amino Acid	Add_one	Add_share	Blosum62	Subst	1-comp	Blocks9
A	0.043478	0.012500	0.036775	0.023388	0.021437	0.017725
C	0.043478	0.012500	0.037993	0.007416	0.005335	0.005062
D	0.043478	0.012500	0.019712	0.005109	0.013008	0.003382
E	0.043478	0.012500	0.019217	0.008594	0.015221	0.004413
F	0.043478	0.012500	0.054988	0.022703	0.011079	0.012761
G	0.043478	0.012500	0.015991	0.008718	0.017854	0.005918
H	0.043478	0.012500	0.018971	0.004003	0.006379	0.002293
I	**0.173913**	**0.762500**	**0.232458**	**0.565594**	**0.755058**	**0.736501**
K	0.043478	0.012500	0.023046	0.009486	0.014918	0.004964
L	**0.043478**	**0.012500**	**0.098515**	**0.105160**	**0.023419**	**0.059162**
M	0.043478	0.012500	0.085919	0.025844	0.007334	0.014584
N	0.043478	0.012500	0.019068	0.007490	0.011472	0.003835
P	0.043478	0.012500	0.022365	0.006738	0.009596	0.003860
Q	0.043478	0.012500	0.022265	0.006414	0.010104	0.003694
R	0.043478	0.012500	0.020627	0.008294	0.013064	0.004495
S	0.043478	0.012500	0.025767	0.012484	0.017361	0.007892
T	0.043478	0.012500	0.045341	0.022306	0.015517	0.012980
V	**0.043478**	**0.012500**	**0.140556**	**0.137617**	**0.020215**	**0.089090**
W	0.043478	0.012500	0.023772	0.002822	0.003095	0.001704
Y	0.043478	0.012500	0.036653	0.009820	0.008533	0.005685

Table 5: Estimated amino acid probabilities using various methods, given three isoleucines. See the caption for Table 4 for details.

	Methods used to estimate amino acid probabilities					
	Pseudocount methods		Substitution Matrices		Dirichlet densities	
Amino Acid	Add_one	Add_share	Blosum62	Subst	1-comp	Blocks9
A	0.040000	0.008333	0.036775	0.023388	0.014370	0.010315
C	0.040000	0.008333	0.037993	0.007416	0.003576	0.003050
D	0.040000	0.008333	0.019712	0.005109	0.008720	0.002014
E	0.040000	0.008333	0.019217	0.008594	0.010203	0.002471
F	0.040000	0.008333	0.054988	0.022703	0.007426	0.007256
G	0.040000	0.008333	0.015991	0.008718	0.011968	0.003864
H	0.040000	0.008333	0.018971	0.004003	0.004276	0.001283
I	**0.240000**	**0.841667**	**0.232458**	**0.565594**	**0.835808**	**0.845797**
K	0.040000	0.008333	0.023046	0.009486	0.010000	0.002645
L	**0.040000**	**0.008333**	**0.098515**	**0.105160**	**0.015699**	**0.033869**
M	0.040000	0.008333	0.085919	0.025844	0.004917	0.008248
N	0.040000	0.008333	0.019068	0.007490	0.007690	0.002169
P	0.040000	0.008333	0.022365	0.006738	0.006432	0.002433
Q	0.040000	0.008333	0.022265	0.006414	0.006773	0.001987
R	0.040000	0.008333	0.020627	0.008294	0.008757	0.002480
S	0.040000	0.008333	0.025767	0.012484	0.011637	0.004445
T	0.040000	0.008333	0.045341	0.022306	0.010402	0.007471
V	**0.040000**	**0.008333**	**0.140556**	**0.137617**	**0.013550**	**0.054004**
W	0.040000	0.008333	0.023772	0.002822	0.002075	0.001004
Y	0.040000	0.008333	0.036653	0.009820	0.005720	0.003195

Table 6: Estimated amino acid probabilities using various methods, given five isoleucines. See the caption for Table 4 for details.

	Methods used to estimate amino acid probabilities					
	Pseudocount methods		Substitution Matrices		Dirichlet densities	
Amino Acid	Add_one	Add_share	Blosum62	Subst	1-comp	Blocks9
A	0.033333	0.004545	0.036775	0.023388	0.007878	0.003909
C	0.033333	0.004545	0.037993	0.007416	0.001960	0.001229
D	0.033333	0.004545	0.019712	0.005109	0.004780	0.000921
E	0.033333	0.004545	0.019217	0.008594	0.005593	0.001031
F	0.033333	0.004545	0.054988	0.022703	0.004071	0.002629
G	0.033333	0.004545	0.015991	0.008718	0.006561	0.002006
H	0.033333	0.004545	0.018971	0.004003	0.002344	0.000554
I	**0.366667**	**0.913636**	**0.232458**	**0.565594**	**0.909991**	**0.942399**
K	0.033333	0.004545	0.023046	0.009486	0.005482	0.001015
L	**0.033333**	**0.004545**	**0.098515**	**0.105160**	**0.008606**	**0.011797**
M	0.033333	0.004545	0.085919	0.025844	0.002695	0.002855
N	0.033333	0.004545	0.019068	0.007490	0.004216	0.000893
P	0.033333	0.004545	0.022365	0.006738	0.003526	0.001182
Q	0.033333	0.004545	0.022265	0.006414	0.003713	0.000772
R	0.033333	0.004545	0.020627	0.008294	0.004801	0.001026
S	0.033333	0.004545	0.025767	0.012484	0.006380	0.001732
T	0.033333	0.004545	0.045341	0.022306	0.005702	0.002782
V	**0.033333**	**0.004545**	**0.140556**	**0.137617**	**0.007428**	**0.019612**
W	0.033333	0.004545	0.023772	0.002822	0.001137	0.000441
Y	0.033333	0.004545	0.036653	0.009820	0.003136	0.001216

Table 7: Estimated amino acid probabilities using various methods, given ten isoleucines. See the caption for Table 4 for details.

A Appendix

$|\vec{x}| = \sum_i x_i$, where x is any vector.
$\vec{n} = n_1, \ldots, n_{20}$ is a vector of counts from a column in a multiple alignment.
$\vec{n}_t$ is the t^{th} such observation in the data set.
$|\vec{n}| = \sum_i n_i$, the number of amino acids observed in a given column of a multiple alignment.
$|\vec{n}_t| = \sum_i^{20} n_{t,i}$ is the number of amino acids observed in the t^{th} count vector ($\vec{n}_t$).
$\vec{p} = (p_1, \ldots, p_{20})$, $\sum p_i = 1$, $p_i \geq 0$, are the parameters of the multinomial distributions from which the $\vec{n}$ are drawn.
$\mathcal{P}$ is the set of all such $\vec{p}$.
$\vec{\alpha} = (\alpha_1, \ldots, \alpha_{20})$ s.t. $\alpha_i > 0$, are the parameters of a Dirichlet density.
$|\vec{\alpha}| = \sum_{i=1}^{20} \alpha_i$ is a measure of the peakedness of the Dirichlet density with parameters $(\alpha_1, \ldots, \alpha_{20})$.
$\vec{\alpha}_j$ are the parameters of the j^{th} component of the Dirichlet density.
$\alpha_{j,i}$ is the value of the i^{th} parameter of the j^{th} component of the Dirichlet mixture.
α_i is the value of the i^{th} element of a Dirichlet density.
$q_j = \mathrm{Prob}(\vec{\alpha}_j)$ is the *mixture coefficient* of the j^{th} component of the mixture.
$\Theta = \{q_1, \ldots, q_l, \vec{\alpha}_1, \ldots, \vec{\alpha}_l\}$ = all the parameters of the Dirichlet mixture.
$\vec{w} = (w_1, \ldots, w_{20})$, are unconstrained values upon which we do gradient descent during training. After each training cycle, $\alpha_{j,i}$ is set to $e^{w_{j,i}}$.
$w_{j,i}$ is the value of the i^{th} parameter of the j^{th} *weight* vector. The nomenclature *weights* comes from artificial neural networks.
m = the number of columns from multiple alignments used in training.
l = the number of components in the mixture.
η = *eta*, the learning rate used to control the size of the step taken during each iteration of gradient descent.

Table 8: Summary of notation.

$f(\Theta)$	$= -\sum_{t=1}^{m} \log(\mathrm{Prob}\left(\vec{n}_t \mid \Theta, \lvert\vec{n}_t\rvert\right))$ (the objective function minimized)	(18)
$\Gamma(\lvert\vec{n}\rvert+1)$	$= n!$ (for integer $n \geq 0$) (Gamma function)	(45)
$\Psi(x)$	$= \frac{\partial \log \Gamma(x)}{\partial x} = \frac{\Gamma'(x)}{\Gamma(x)}$ (Psi function)	(65)
$\mathrm{Prob}\left(\vec{n} \mid \vec{p}, \lvert\vec{n}\rvert\right)$	$= \Gamma(\lvert\vec{n}\rvert+1) \prod_{i=1}^{20} \frac{p_i^{n_i}}{\Gamma(n_i+1)}$ (the probability of $\vec{n}$ under the multinomial distribution with parameters $\vec{p}$)	(46)
$\mathrm{Prob}\left(\vec{n} \mid \vec{\alpha}, \lvert\vec{n}\rvert\right)$	$= \frac{\Gamma(\lvert\vec{n}\rvert+1)\,\Gamma(\lvert\vec{\alpha}\rvert)}{\Gamma(\lvert\vec{n}\rvert+\lvert\vec{\alpha}\rvert)} \prod_{i=1}^{20} \frac{\Gamma(n_i+\alpha_i)}{\Gamma(n_i+1)\Gamma(\alpha_i)}$ (the probability of $\vec{n}$ under the Dirichlet density with parameters $\vec{\alpha}$)	(54)
$\mathrm{Prob}\left(\vec{n} \mid \Theta, \lvert\vec{n}\rvert\right)$	$= \sum_{k=1}^{l} q_k \, \mathrm{Prob}(\vec{n} \mid \vec{\alpha}_k, \lvert\vec{n}\rvert)$ (the probability of $\vec{n}$ given the entire mixture prior)	(17)
$\mathrm{Prob}\left(\vec{\alpha}_j \mid \vec{n}, \Theta\right)$	$= \frac{q_j \, \mathrm{Prob}(\vec{n} \mid \vec{\alpha}_j, \lvert\vec{n}\rvert)}{\mathrm{Prob}\left(\vec{n} \mid \Theta, \lvert\vec{n}\rvert\right)}$ (shorthand for the posterior probability of the j^{th} component of the mixture given the vector of counts $\vec{n}$)	(16)

Table 9: Index to key derivations and definitions.

A.1 Lemma 1. $\mathbf{Prob}\,(\vec{n} \mid \vec{p}, |\vec{n}|) = \Gamma(|\vec{n}|+1)\prod_{i=1}^{20} \frac{p_i^{n_i}}{\Gamma(n_i+1)}$

Proof:

For a given vector of counts $\vec{n}$, with p_i being the probability of seeing the i^{th} amino acid, and $|\vec{n}| = \sum_i n_i$, there are $\frac{|\vec{n}|!}{n_1!n_2!\ldots n_{20}!}$ distinct permutations of the amino acids which result in the count vector $\vec{n}$. If we allow for the assumption that each column is generated independently, then each such permutation has probability $\prod_{i=1}^{20} p_i^{n_i}$. Thus, the probability of a given count vector $\vec{n}$ given the multinomial parameters $\vec{p}$ is

$$\mathrm{Prob}\left(\vec{n} \,\middle|\, \vec{p}, |\vec{n}|\right) = \frac{|\vec{n}|!}{n_1!n_2!\ldots n_{20}!}\prod_{i=1}^{20} p_i^{n_i} \tag{43}$$

$$= |\vec{n}|!\prod_{i=1}^{20} \frac{p_i^{n_i}}{n_i!} \tag{44}$$

Since we may need to handle real-valued data (such as that obtained from using a weighting scheme on the sequences in the training set), we introduce the Gamma function, the continuous generalization of the integer factorial function,

$$\Gamma(n+1) = n! \tag{45}$$

Substituting the Gamma function, we obtain the equivalent form

$$\mathrm{Prob}\left(\vec{n} \,\middle|\, \vec{p}, |\vec{n}|\right) = \Gamma(|\vec{n}|+1)\prod_{i=1}^{20} \frac{p_i^{n_i}}{\Gamma(n_i+1)} \tag{46}$$

A.2 Lemma 2. $\textbf{Prob}\,(\vec{p} \mid \vec{\alpha}) = \frac{\Gamma(|\vec{\alpha}|)}{\prod_{i=1}^{20}\Gamma(\alpha_i)} \prod_{i=1}^{20} p_i^{\alpha_i - 1}$

Proof:

Under the Dirichlet density with parameters $\vec{\alpha}$, the probability of the distribution $\vec{p}$ (where $p_i \geq 0$, and $\sum_i p_i = 1$) is defined as follows:

$$\mathrm{Prob}(\vec{p} \mid \vec{\alpha}) = \frac{\prod_{i=1}^{20} p_i^{\alpha_i - 1}}{\int_{\vec{p} \in \mathcal{P}} \prod_i p_i^{\alpha_i - 1} d\vec{p}} \tag{47}$$

We introduce two formulas concerning the Beta function—its definition (Gradshteyn and Ryzhik, 1965, p. 948):

$$\begin{aligned} \mathrm{B}(x, y) &= \int_0^1 t^{x-1}(1-t)^{y-1}\, dt \\ &= \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \end{aligned}$$

and the combining formula (Gradshteyn and Ryzhik, 1965, p. 285):

$$\int_0^b t^{x-1}(b-t)^{y-1}\, dt = b^{x+y-1}\mathrm{B}(x, y)\ .$$

This allows us to write the integral over all $\vec{p}$ vectors as a multiple integral, rearrange some terms, and obtain

$$\int_{\vec{p} \in \mathcal{P}} \prod_i p_i^{\alpha_i - 1} d\vec{p} = \mathrm{B}(\alpha_1, \alpha_2 + \ldots + \alpha_{20})\mathrm{B}(\alpha_2, \alpha_3 + \ldots + \alpha_{20}) \ldots \mathrm{B}(\alpha_{19}, \alpha_{20}) \tag{48}$$

$$= \frac{\prod_i \Gamma(\alpha_i)}{\Gamma(|\vec{\alpha}|)} \tag{49}$$

This allows us to now give an explicit definition of the probability of the point $\vec{p}$ given the Dirichlet density with parameters $\vec{\alpha}$:

$$\mathrm{Prob}(\vec{p} \mid \vec{\alpha}) = \frac{\Gamma(|\vec{\alpha}|)}{\prod_{i=1}^{20}\Gamma(\alpha_i)} \prod_{i=1}^{20} p_i^{\alpha_i - 1} \tag{50}$$

A.3 Lemma 3. $\textbf{Prob}\,(\vec{n} \mid \vec{\alpha}, |\vec{n}|) = \frac{\Gamma(|\vec{n}|+1)\,\Gamma(|\vec{\alpha}|)}{\Gamma(|\vec{n}|+|\vec{\alpha}|)} \prod_{i=1}^{20} \frac{\Gamma(n_i+\alpha_i)}{\Gamma(n_i+1)\Gamma(\alpha_i)}$

Proof:

Since

$$\text{Prob}\left(\vec{n} \;\middle|\; \vec{\alpha}, |\vec{n}|\right) = \int_{\vec{p}\in\mathcal{P}} \text{Prob}\left(\vec{n} \;\middle|\; \vec{p}, |\vec{n}|\right) \text{Prob}(\vec{p} \,|\, \vec{\alpha})\, d\vec{p} \tag{51}$$

Substituting equations (46) and (50) into equation (51), we obtain

$$\text{Prob}\left(\vec{n} \;\middle|\; \vec{\alpha}, |\vec{n}|\right) = \int_{\vec{p}\in\mathcal{P}} \frac{\Gamma(|\vec{n}|+1)\Gamma(|\vec{\alpha}|)}{\prod_{i=1}^{20}(\Gamma(n_i+1)\,\Gamma(\alpha_i))} \prod_{i=1}^{20} p_i^{n_i+\alpha_i-1}\, d\vec{p} \tag{52}$$

Pulling out terms not depending on $\vec{p}$ from inside the integral, and using the result from Equation (49), we obtain

$$= \frac{\Gamma(|\vec{n}|+1)\,\Gamma(|\vec{\alpha}|)}{\prod_{i=1}^{20}(\Gamma(n_i+1)\,\Gamma(\alpha_i))} \frac{\prod_{i=1}^{20}\Gamma(n_i+\alpha_i)}{\Gamma(|\vec{n}|+|\vec{\alpha}|)} \tag{53}$$

At this point, we can simply rearrange a few terms and obtain the equivalent form

$$\text{Prob}\left(\vec{n} \;\middle|\; \vec{\alpha}, |\vec{n}|\right) = \frac{\Gamma(|\vec{n}|+1)\,\Gamma(|\vec{\alpha}|)}{\Gamma(|\vec{n}|+|\vec{\alpha}|)} \prod_{i=1}^{20} \frac{\Gamma(n_i+\alpha_i)}{\Gamma(n_i+1)\Gamma(\alpha_i)} \tag{54}$$

A.4 Lemma 4. $\mathbf{Prob}\,(\vec{p} \mid \vec{\alpha}, \vec{n}) = \frac{\Gamma(|\vec{\alpha}|+|\vec{n}|)}{\prod_{i=1}^{20}\Gamma(\alpha_i+n_i)} \prod_{i=1}^{20} p_i^{\alpha_i+n_i-1}$

Proof:

By Bayes' Rule, the probability of the distribution $\vec{p}$, given the Dirichlet density with parameters $\vec{\alpha}$, and the observed amino acid count vector $\vec{n}$ is defined

$$\text{Prob}\left(\vec{p}\,\middle|\,\vec{\alpha},\vec{n}\right) = \frac{\text{Prob}\left(\vec{n}\,\middle|\,\vec{p},\vec{\alpha},|\vec{n}|\right)\text{Prob}\left(\vec{p}\,\middle|\,\vec{\alpha}\right)}{\text{Prob}\left(\vec{n}\,\middle|\,\vec{\alpha},|\vec{n}|\right)} \tag{55}$$

However, once the point $\vec{p}$ is fixed, the probability of $\vec{n}$ no longer depends on $\vec{\alpha}$. Hence,

$$\text{Prob}\left(\vec{p}\,\middle|\,\vec{\alpha},\vec{n}\right) = \frac{\text{Prob}\left(\vec{n}\,\middle|\,\vec{p},|\vec{n}|\right)\text{Prob}\left(\vec{p}\,\middle|\,\vec{\alpha}\right)}{\text{Prob}\left(\vec{n}\,\middle|\,\vec{\alpha},|\vec{n}|\right)} \tag{56}$$

At this point, we apply the results from previous derivations for quantities $\text{Prob}\left(\vec{n}\,\middle|\,\vec{p},|\vec{n}|\right)$ (equation 46), $\text{Prob}\left(\vec{p}\,\middle|\,\vec{\alpha}\right)$ (equation 50), and $\text{Prob}\left(\vec{n}\,\middle|\,\vec{\alpha},|\vec{n}|\right)$ (equation 54). This gives us

$$\text{Prob}\left(\vec{p}\,\middle|\,\vec{\alpha},\vec{n}\right) = \frac{\left(\frac{\Gamma(|\vec{n}|+1)}{\prod_i\Gamma(n_i+1)}\prod_i p_i^{n_i}\right)\left(\frac{\Gamma(|\vec{\alpha}|)}{\prod_i\Gamma(\alpha_i)}\prod_i p_i^{\alpha_i-1}\right)\Gamma(|\vec{n}|+|\vec{\alpha}|)\prod_i\Gamma(n_i+1)\Gamma(\alpha_i)}{\Gamma(|\vec{n}|+1)\,\Gamma(|\vec{\alpha}|)\prod_i\Gamma(n_i+\alpha_i)} \tag{57}$$

Most of the terms cancel, and we have

$$\text{Prob}\left(\vec{p}\,\middle|\,\vec{\alpha},\vec{n}\right) = \frac{\Gamma(|\vec{\alpha}|+|\vec{n}|)}{\prod_{i=1}^{20}\Gamma(\alpha_i+n_i)}\prod_{i=1}^{20} p_i^{\alpha_i+n_i-1} \tag{58}$$

Note that this is the expression for a Dirichlet density with parameters $\vec{\alpha}$ and $\vec{n}$. This property, that the posterior density of ρ is from the same family as the prior, characterizes all conjugate priors, and is one of the properties that make Dirichlet densities so attractive.

A.5 Lemma 5. $\frac{\partial \log \mathbf{Prob}(\vec{n}|\Theta,|\vec{n}|)}{\partial \alpha_{j,i}} = \mathbf{Prob}(\vec{\alpha}_j \mid \vec{n}, \Theta) \frac{\partial \log \mathbf{Prob}(\vec{n}|\vec{\alpha}_j,|\vec{n}|)}{\partial \alpha_{j,i}}$

Proof:

The derivative of the logarithm of the probability of each individual observation $\vec{n}$ given the mixture with respect to $\alpha_{j,i}$ is:

$$\frac{\partial \log \text{ Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial \alpha_{j,i}} = \frac{1}{\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)} \frac{\partial \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial \alpha_{j,i}} \tag{59}$$

Applying equation 17, this gives us

$$\frac{\partial \log \text{ Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial \alpha_{j,i}} = \frac{1}{\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)} \frac{\partial \sum_{k=1}^{L} q_k \text{ Prob}\left(\vec{n} \mid \vec{\alpha}_k, |\vec{n}|\right)}{\partial \alpha_{j,i}} \tag{60}$$

Since the derivative of $\text{Prob}\left(\vec{n} \mid \vec{\alpha}_k, |\vec{n}|\right)$ with respect to $\alpha_{j,i}$ is zero for all $k \neq j$, and the mixture coefficients (the q_k) are independent parameters, this yields

$$\frac{\partial \log \text{ Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial \alpha_{j,i}} = \frac{q_j}{\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)} \frac{\partial \text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{\partial \alpha_{j,i}} \tag{61}$$

We rearrange equation (16) somewhat, and replace $\frac{q_j}{\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}$ by its equivalent, obtaining,

$$\frac{\partial \log \text{ Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial \alpha_{j,i}} = \frac{\text{Prob}\left(\vec{\alpha}_j \mid \vec{n}, \Theta\right)}{\text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)} \frac{\partial \text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{\partial \alpha_{j,i}} \tag{62}$$

Here, again using the fact that $\frac{\partial \log(f(x))}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$, we obtain the final form

$$\frac{\partial \log \text{ Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial \alpha_{j,i}} = \text{Prob}\left(\vec{\alpha}_j \,\middle|\, \vec{n}, \Theta\right) \frac{\partial \log \text{ Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{\partial \alpha_{j,i}} \tag{63}$$

A.6 Lemma 6. $\frac{\partial \log \mathbf{Prob}(\vec{n}|\vec{\alpha},|\vec{n}|)}{\partial \alpha_i} = [\Psi(|\vec{\alpha}|) - \Psi(|\vec{n}| + |\vec{\alpha}|) + \Psi(n_i + \alpha_i) - \Psi(\alpha_i)]$

Proof:

In this proof, we use Lemma 3 giving

$$\text{Prob}\left(\vec{n} \;\middle|\; \vec{\alpha}, |\vec{n}|\right) = \frac{\Gamma(|\vec{n}|+1)\,\Gamma(|\vec{\alpha}|)}{\Gamma(|\vec{n}|+|\vec{\alpha}|)} \prod_{i=1}^{20} \frac{\Gamma(n_i+\alpha_i)}{\Gamma(n_i+1)\Gamma(\alpha_i)}$$

Since the derivative of terms not depending on α_i are zero, we obtain that for a single vector of counts $\vec{n}$,

$$\frac{\partial \log\ \text{Prob}\left(\vec{n} \;\middle|\; \vec{\alpha}, |\vec{n}|\right)}{\partial \alpha_i} = \frac{\partial \log \Gamma(|\vec{\alpha}|)}{\partial \alpha_i} - \frac{\partial \log\ \Gamma(|\vec{n}|+|\vec{\alpha}|)}{\partial \alpha_i} + \frac{\partial \log\ \Gamma(n_i+\alpha_i)}{\partial \alpha_i} - \frac{\partial \log\ \Gamma(\alpha_i)}{\partial \alpha_i}\,. \tag{64}$$

Now, if we substitute the shorthand

$$\Psi(x) = \frac{\partial \log\ \Gamma(x)}{\partial x} = \frac{\Gamma'(x)}{\Gamma(x)} \tag{65}$$

we have

$$\frac{\partial \log\ \text{Prob}\left(\vec{n} \;\middle|\; \vec{\alpha}, |\vec{n}|\right)}{\partial \alpha_i} = \Psi(|\vec{\alpha}|) - \Psi(|\vec{n}|+|\vec{\alpha}|) + \Psi(n_i+\alpha_i) - \Psi(\alpha_i) \tag{66}$$

A.7 Lemma 7. $\frac{\partial \mathbf{Prob}(\vec{n}|\Theta,|\vec{n}|)}{\partial Q_i} = \frac{\mathbf{Prob}(\vec{n}|\vec{\alpha}_i,|\vec{n}|) - \mathbf{Prob}(\vec{n}|\Theta,|\vec{n}|)}{|Q|}$

Proof:

Substituting equation (17), giving $\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right) = \sum_{j=1}^{l} q_j \,\text{Prob}(\vec{n} \mid \vec{\alpha}_j,\, |\vec{n}|)$ and replacing q_j by $Q_j/|Q|$, this gives us

$$\frac{\partial \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} = \frac{\partial \sum_j (Q_j/|Q|) \text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{\partial Q_i} \tag{67}$$

As the derivative of a sum is the sum of the derivatives, we can use the standard product rule for differentiation, and obtain

$$\frac{\partial \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} = \sum_j \left[(Q_j/|Q|) \frac{\partial \text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{\partial Q_i} + \text{Prob}\left(\vec{n} \,\middle|\, \vec{\alpha}_j, |\vec{n}|\right) \frac{\partial (Q_j/|Q|)}{\partial Q_i} \right] \tag{68}$$

$$\tag{69}$$

Since $\frac{\partial \text{Prob}\left(\vec{n} \middle| \vec{\alpha}_j, |\vec{n}|\right)}{\partial Q_i} = 0$ for all j, this gives us

$$\frac{\partial \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} = \sum_j \text{Prob}\left(\vec{n} \,\middle|\, \vec{\alpha}_j, |\vec{n}|\right) \frac{\partial (Q_j/|Q|)}{\partial Q_i} \tag{70}$$

Taking the derivative of the fraction $(Q_j/|Q|)$ with respect to Q_i, we obtain

$$\frac{\partial (Q_j/|Q|)}{\partial Q_i} = |Q|^{-1} \frac{\partial Q_j}{\partial Q_i} + Q_j \frac{\partial |Q|^{-1}}{\partial Q_i}$$

.

The first term, $|Q|^{-1} \frac{\partial Q_j}{\partial Q_i}$, is zero when $j \neq i$, and is $\frac{1}{|Q|}$ when $j = i$. The second term, $Q_j \frac{\partial |Q|^{-1}}{\partial Q_i}$, is simply $\frac{-Q_j}{|Q|^2}$. Thus, this is

$$\frac{\partial \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} = \frac{\text{Prob}\left(\vec{n} \mid \vec{\alpha}_i, |\vec{n}|\right)}{|Q|} + \sum_j \text{Prob}\left(\vec{n} \,\middle|\, \vec{\alpha}_j, |\vec{n}|\right) \frac{-Q_j}{|Q|^2} \tag{71}$$

Here, $q_j = Q_j/|Q|$ allows us to replace $Q_j/|Q|^2$ with $q_j/|Q|$, giving us

$$= \frac{\text{Prob}\left(\vec{n} \mid \vec{\alpha}_i, |\vec{n}|\right) - \sum_j q_j \text{Prob}\left(\vec{n} \mid \vec{\alpha}_j, |\vec{n}|\right)}{|Q|} \tag{72}$$

$$\tag{73}$$

At this point, we use equation 17 and obtain

$$\frac{\partial \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} = \frac{\text{Prob}\left(\vec{n} \mid \vec{\alpha}_i, |\vec{n}|\right) - \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{|Q|} \tag{74}$$

A.8 Lemma 8. $\frac{\partial \log \mathbf{Prob}(\vec{n}|\Theta,|\vec{n}|)}{\partial Q_i} = \frac{\mathbf{Prob}(\vec{\alpha}_i|\vec{n},\Theta)}{Q_i} - \frac{1}{|Q|}$

Proof:

$$\frac{\partial \log \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} = \frac{1}{\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)} \frac{\partial \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} \tag{75}$$

Here we use Lemma 7, which allows us to express the derivative with respect to Q_i of the log likelihood of a single observation $\vec{n}$, given the mixture, as

$$\begin{aligned} \frac{\partial \log \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} &= \frac{1}{\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)} \frac{\text{Prob}\left(\vec{n} \mid \vec{\alpha}_i, |\vec{n}|\right) - \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{|Q|} && (76) \\ &= \left(\frac{\text{Prob}\left(\vec{n} \mid \vec{\alpha}_i, |\vec{n}|\right)}{\text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)} - 1 \right) / \; |Q| && (77) \end{aligned}$$

If we rearrange equation 16, we obtain $\frac{\text{Prob}\left(\vec{n} \middle| \vec{\alpha}_i, |\vec{n}|\right)}{\text{Prob}\left(\vec{n} \middle| \Theta, |\vec{n}|\right)} = \frac{\text{Prob}(\vec{\alpha}_i \mid \vec{n})}{q_i}$. This allows us to write

$$\frac{\partial \log \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} = \left(\frac{\text{Prob}(\vec{\alpha}_i \mid \vec{n})}{q_i} - 1 \right) / \; |Q| \tag{78}$$

Now we can use the identity $q_i = Q_i / \, |Q|$, obtaining the equivalent

$$\frac{\partial \log \text{Prob}\left(\vec{n} \mid \Theta, |\vec{n}|\right)}{\partial Q_i} = \frac{\text{Prob}\left(\vec{\alpha}_i \mid \vec{n}, \Theta\right)}{Q_i} - \frac{1}{|Q|} \tag{79}$$