CABIOS

Vol. 12 no. 4 1996
Pages 327–345

Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology

Kimmen Sjölander[3], Kevin Karplus, Michael Brown, Richard Hughey, Anders Krogh[1], I.Saira Mian[2] and David Haussler

Abstract

We present a method for condensing the information in multiple alignments of proteins into a mixture of Dirichlet densities over amino acid distributions. Dirichlet mixture densities 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, so that remotely related family members can be more reliably recognized by the model. This paper corrects the 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.

Introduction

One of the main techniques used in protein sequence analysis is the identification of *homologous* proteins—proteins which share a common evolutionary history and almost invariably have similar overall structure and function (Doolittle, 1986). Homology is straightforward to infer when two sequences share 25% residue identity over a stretch of 80 or more residues (Sander and Schneider, 1991). Recognizing *remote homologs*, with lower primary sequence identity, is more difficult. Finding these remote homologs is one of the primary motivating forces behind the development of statistical models for protein families and domains, such as *profiles* and their many offshoots (Waterman and Perlwitz, 1986; Gribskov *et al.* 1987, 1990; Altschul *et al.*, 1990; Barton and Sternberg, 1990; Bowie *et al.*, 1991; Lüthy *et al.*, 1991; Thompson *et al.*, 1994a,b; Bucher *et al.*, 1996), *Position-Specific Scoring Matrices* (Henikoff *et al.*, 1990), and *hidden Markov models (HMMs)* (Churchill, 1989; Baldi *et al.*, 1992; Asai *et al.*, 1993; Stultz *et al.*, 1993; Baldi and Chauvin, 1994; Krogh *et al.*, 1994; White *et al.*, 1994; Eddy, 1995, 1996; Eddy *et al.*, 1995; Hughey and Krogh, 1996).

We address this problem by incorporating prior information about amino acid distributions that typically occur in columns of multiple alignments into the process of building a statistical model. We present a method to condense the information in databases of multiple alignments into a mixture of *Dirichlet* densities (Berger, 1985; Santner and Duffy, 1989; Bernardo and Smith, 1994) over amino acid distributions, and to combine this prior information with the observed amino acids to form more effective estimates of the expected distributions. Multiple alignments used in these experiments were taken from the Blocks database (Henikoff and Henikoff, 1991). We use *Maximum Likelihood* (Duda and Hart, 1973; Dempster *et al.*, 1977; Nowlan, 1990), 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 columns of multiple alignments.

With accurate prior information about which kinds of amino acid distributions are reasonable in columns of alignments, it is possible with only a few sequences to identify which prototypical distribution may have generated the amino acids observed in a particular column. Using this informed guess, we adjust the expected amino acid probabilities to include the possibility of amino acids that may not have been seen but are consistent with observed amino acid distributions. The statistical models produced are more effective at generalizing to previously unseen data, and are often superior at database search and discrimination experiments (Brown *et al.*, 1993; Tatusov *et al.*, 1994; Bailey and Elkan, 1995; Karplus, 1995a; Hughey and Krogh, 1996; Henikoff and Henikoff, 1996; Wang *et al.*, 1996).

Baskin Center for Computer Engineering and Information Sciences, Applied Sciences Building, University of California at Santa Cruz, Santa Cruz, CA 95064, USA, [1]*The Sanger Centre, Hinxton Hall, Hinxton, Cambs CB10 1RQ, UK and* [2]*Life Sciences Division (Mail Stop 29-100), Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720, USA*

[3]*To whom correspondence should be addressed. E-mail: kimmen@cse.ucsc.edu*

Database search using statistical models

Statistical models for proteins capture the statistics defining a protein family or domain. These models have two essential aspects: (i) parameters for every position in the molecule or domain that express the probabilities of the amino acids, gap initiation and extension, and so on; and (ii) a scoring function for sequences with respect to the model.

Statistical models do not use percentage residue identity to determine homology. Instead, these models assign a probability score [some methods report a cost rather than a probability score; these are closely related (Altschul, 1991)] to sequences, and then compare the score to a cutoff. Most models (including HMMs and profiles) make the simplifying assumption that each position in the protein is generated independently. Under this assumption, the score for a sequence aligning to a model is equal to the product of the probabilities of aligning each residue in the protein to the corresponding position in the model. In homolog identification by percentage residue identity, if a protein aligns a previously unseen residue at a position it is not penalized; it loses that position, but can still be recognized as homologous if it matches at a sufficient number of other positions in the search protein. However, in statistical models, if a protein aligns a zero-probability residue at a position, the probability of the sequence with respect to the model is zero, regardless of how well it may match the rest of the model. Because of this, most statistical models rigorously avoid assigning residues probability zero, and accurate estimates for the amino acids at each position are particularly important.

Using prior information about amino acid distributions

The parameters of a statistical model for a protein family or domain are derived directly from sequences in the family or containing the domain. When we have few sequences, or a skewed sample, the raw frequencies are a poor estimate of the distributions which we expect to characterize the homologs in the database. Skewed samples 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. In this kind of skew, sequence weighting schemes are necessary to compensate for the bias in the data. Models that use these raw frequencies may recognize the sequences used to train the model, but will not generalize well to recognizing remoter homologs.

It is illuminating to consider the analogous problem of assessing the fairness of a coin. A coin is said to be fair if Prob (*heads*) = Prob (*tails*) = $1/2$. If we toss a coin three times, and it comes up heads each time, what should our estimate be of the probability of heads for this coin? If we use the observed raw frequencies, we would set the probability of heads to one, but, if we assume that most coins are fair, then we are unlikely to change this *a priori* assumption based on only a few tosses. Given little data, we will believe our prior assumptions remain valid. On the other hand, if we toss the coin an additional thousand times and it comes up heads each time, few will insist that the coin is indeed fair. Given an abundance of data, we will discount any previous assumptions, and believe the data.

When we estimate the expected amino acids in each position of a statistical model for a protein family, we encounter virtually identical situations. 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, and $|\vec{n}| = \sum_i n_i$. The estimated probability of amino acid i is denoted $\hat{p}_i$. If the raw frequencies are used to set the probabilities, then $\hat{p}_i := n_i/|\vec{n}|$. Note that we use the symbol ':=' to denote assignment, to distinguish it from equality, since we compute $\hat{p}_i$ differently in different parts of the paper.

Consider the following two scenarios. In the first, we have only three sequences from which to estimate the parameters of the model. In the alignment of these three sequences, we have a column containing only isoleucine, and no other amino acids. With such a small sample, we cannot rule out the possibility that homologous proteins may have different amino acids at this position. In particular, we know that isoleucine is commonly found in buried β-strand environments, and leucine and valine often substitute for it in these environments. Thus, our estimate of the expected distribution at this position would sensibly include these amino acids, and perhaps the other amino acids as well, albeit with smaller probabilities.

In a second scenario, we have an alignment of 100 varied sequences and again find a column containing only isoleucine, and no other amino acids. In this case, we have much more evidence that isoleucine is indeed conserved at this position, and thus generalizing the distribution at this position to include similar residues is probably not a good idea. In this situation, it makes more sense to give less importance to prior beliefs about similarities among amino acids, and more importance to the actual counts observed.

Table I. Parameters of mixture prior Blocks9

	Comp. 1	Comp. 2	Comp. 3	Comp. 4	Comp. 5	Comp. 6	Comp. 7	Comp. 8	Comp. 9
q	0.1829	0.0576	0.0898	0.0792	0.0831	0.0911	0.1159	0.0660	0.2340
$\lvert\vec{\alpha}\rvert$	1.1806	1.3558	6.6643	2.0814	2.0810	2.5681	1.7660	4.9876	0.0995
A	0.2706	0.0214	0.5614	0.0701	0.0411	0.1156	0.0934	0.4521	0.0051
C	0.0398	0.0103	0.0454	0.0111	0.0147	0.0373	0.0047	0.1146	0.0040
D	0.0175	0.0117	0.4383	0.0194	0.0056	0.0124	0.3872	0.0624	0.0067
E	0.0164	0.0108	0.7641	0.0946	0.0102	0.0181	0.3478	0.1157	0.0061
F	0.0142	0.3856	0.0873	0.0131	0.1536	0.0517	0.0108	0.2842	0.0034
G	0.1319	0.0164	0.2591	0.0480	0.0077	0.0172	0.1058	0.1402	0.0169
H	0.0123	0.0761	0.2149	0.0770	0.0071	0.0049	0.0497	0.1003	0.0036
I	0.0225	0.0353	0.1459	0.0329	0.2996	0.7968	0.0149	0.5502	0.0021
K	0.0203	0.0139	0.7622	0.5766	0.0108	0.0170	0.0942	0.1439	0.0050
L	0.0307	0.0935	0.2473	0.0722	0.9994	0.2858	0.0277	0.7006	0.0059
M	0.0153	0.0220	0.1186	0.0282	0.2101	0.0758	0.0100	0.2765	0.0014
N	0.0482	0.0285	0.4415	0.0803	0.0061	0.0145	0.1878	0.1185	0.0041
P	0.0538	0.0130	0.1748	0.0376	0.0130	0.0150	0.0500	0.0974	0.0090
Q	0.0206	0.0230	0.5308	0.1850	0.0197	0.0113	0.1100	0.1266	0.0036
R	0.0236	0.0188	0.4655	0.5067	0.0145	0.0126	0.0386	0.1436	0.0065
S	0.2161	0.0291	0.5834	0.0737	0.0120	0.0275	0.1194	0.2789	0.0031
T	0.0654	0.0181	0.4455	0.0715	0.0357	0.0883	0.0658	0.3584	0.0036
V	0.0654	0.0361	0.2270	0.0425	0.1800	0.9443	0.0254	0.6617	0.0029
W	0.0037	0.0717	0.0295	0.0112	0.0127	0.0043	0.0032	0.0615	0.0027
Y	0.0096	0.4196	0.1210	0.0287	0.0264	0.0167	0.0187	0.1993	0.0026

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) for each component. The second row gives the $|\vec{\alpha}| = \sum_i \alpha_i$ for each component. Rows A (alanine) through Y (tyrosine) contain the values of each of the components' $\vec{\alpha}$ parameters for that amino acid. See the section 'What is a Dirichlet mixture', in the Introduction, for details on how to interpret these values.

It is informative to examine this table and Table II in unison. The mixture coefficients (q) of the densities reveal that, in this mixture, the components peaked around the aromatic and the uncharged hydrophobic residues (components 2 and 8) 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 captured by any single component.

Examining the $|\vec{\alpha}|$ of each component shows that the two components with the largest values of $|\vec{\alpha}|$ (and so the most mixed distributions) represent the polars (component 3) and the uncharged hydrophobics (component 8), respectively. The component with the smallest $|\vec{\alpha}|$ (component 9) gives probability to pure distributions.

This mixture prior is available via anonymous ftp at our ftp site, `ftp://ftp.cse.ucsc.edu/pub/protein/dirichlet/` and at our World-Wide Web site `http://www.cse.ucsc.edu/research/compbio/dirichlet.html`.

The natural solution is to introduce prior information into the construction of the statistical model. The method we propose interpolates smoothly between reliance on the prior information concerning likely amino acid distributions, in the absence of data, and confidence in the amino acid frequencies observed at each position, given sufficient data.

What is a Dirichlet density?

A Dirichlet density ρ (Berger, 1985; Santner and Duffy, 1989) is a probability density over the set of all probability vectors $\vec{p}$ (i.e. $p_i \geq 0$ and $\sum_i p_i = 1$). Proteins have a 20 letter alphabet, with p_i = Prob (amino acid i). Each vector $\vec{p}$ represents a possible probability distribution over the 20 amino acids.

A Dirichlet density has parameters $\vec{\alpha} = \alpha_1, \ldots, \alpha_{20}$, with $\alpha_i > 0$. 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 sum to unity. The mean value of p_i given a Dirichlet density with parameters $\vec{\alpha}$ is

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

where $|\vec{\alpha}| = \sum_i \alpha_i$.

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

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

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

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

Table II. Preferred amino acids of Blocks9

Component	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		

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 $(\alpha_{j,i}/|\vec{\alpha}|)/(\sum_k q_k \alpha_{k,i}/|\vec{\alpha}_k|)$.
An analysis of the amino acids favored by each component reveals the following:
Component 1 favors small neutral residues.
Component 2 favors the aromatics.
Component 3 gives high probability to most of the polar residues (except for C, Y and W).
Component 4 gives high probability to positively charged amino acids and residues with NH_2 groups.
Component 5 gives high probability to residues that are aliphatic or large and non-polar.
Component 6 prefers I and V (aliphatic residues commonly found in β sheets), and allows substitutions with L and M.
Component 7 gives high probability to negatively charged residues, allowing substitutions with certain of the hydrophilic polar residues.
Component 8 gives high probability to uncharged hydrophobics, with the exception of glycine.
Component 9 gives high probability to distributions peaked around individual amino acids (especially P, G, W and C).

We chose Dirichlet densities because of their mathematical convenience: a Dirichlet density is a conjugate prior, i.e. the posterior of a Dirichlet density has the same form as the prior (see, for example, section A.4 in the Appendix).

What is a Dirichlet mixture?

A mixture of Dirichlet densities is a collection of individual Dirichlet densities that function jointly to assign probabilities to distributions. For any distribution of amino acids, the mixture as a whole assigns a probability to the distribution by using a weighted combination of the probabilities given the distribution by each of the components in the mixture. These weights are called *mixture coefficients*. Each individual density in a mixture is called a *component* of the mixture.

A Dirichlet mixture density ρ with l components has the form

$$\rho = q_1 \rho_1 + \cdots + q_l \rho_l \qquad (5)$$

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 the mixture coefficients and sum to one.

The symbol Θ refers 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, $\Theta = (\vec{\alpha})$.

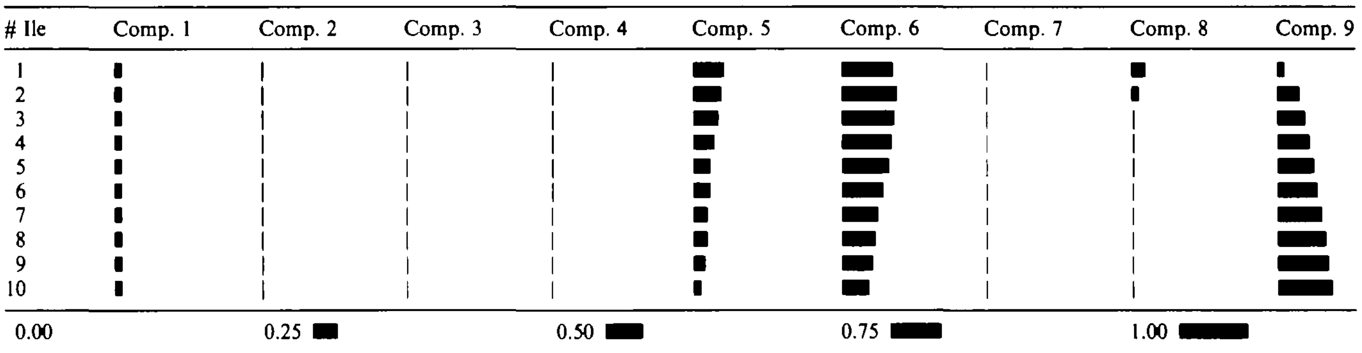
Interpreting the parameters of a Dirichlet mixture. Since a Dirichlet mixture describes the typical distributions of amino acids in the data used to estimate the mixture, it is useful to look in some detail at each individual component of the mixture to see what distributions of amino acids it favors. We include in this paper a nine-component mixture estimated on the Blocks database (Henikoff and Henikoff, 1991), a close variant of which has been used in experiments elsewhere (Tatusov *et al.*, 1994; Henikoff and Henikoff, 1996).

A couple of comments about how we estimated this mixture density are in order.

First, the decision to use nine components was somewhat arbitrary. As in any statistical model, a balance must be struck between the complexity of the model and the data available to estimate the parameters of the model. A mixture with too few components will have a limited ability to represent different contexts for the amino acids. On the other hand, there may not be sufficient data to estimate precisely the parameters of the mixture if it has too many components. We have experimented with mixtures with anywhere from one to 30 components; in practice, nine components appears to be the best compromise with the data we have available. Also, a nine-component mixture uses 188 parameters, slightly fewer than the 210 of a symmetric substitution matrix, so that better results with the Dirichlet mixture cannot be attributed to having more parameters to fit to the data.

Second, there are many different mixtures having the same number of components that give basically the same results. This reflects the fact that Dirichlet mixture densities attempt to fit a complex space, and there are many ways to fit this space. Optimization problems such

Table III. The posterior probability of each component in Dirichlet mixture *Blocks9* [equation (16)] given 1–10 isoleucines. Initially, component 6, which prefers I and V found jointly, is most likely, followed by components 5 and 8, which like aliphatic residues in general. As the number of observed isoleucines increases, component 9, which favors pure distributions of any type, increases in probability, but component 6 remains fairly probable. The more mixed distributions become increasingly unlikely as the number of observed isoleucines increases



as this are notoriously difficult, and we make no claim that this mixture is globally optimal. This mixture works quite well, however, and is better than many other nine-component local optima that we have found.

Table I gives the parameters of this mixture. Table II lists the preferred amino acids for each component in the mixture, in order by the ratio of the mean probability of the amino acids in a component to the background probability of the amino acids. An alternative 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 were given in the previous section 'What is a Dirichlet density?'.

The value $|\vec{\alpha}| = \sum_{i=1}^{20} \alpha_i$ is a measure of the variance 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 mixtures we have estimated, components having high $|\vec{\alpha}|$ tend to give high probability to *combinations* of amino acids which have similar physio-chemical characteristics and are known to substitute readily for each other in particular environments. By contrast, when $|\vec{\alpha}|$ is small, the component favors pure distributions conserved around individual amino acids. A residue may be represented primarily by one component (as proline is) or by several components (as isoleucine and valine are). When we estimate mixtures with many components, we sometimes find individual components with high $|\vec{\alpha}|$ that give high probability to pure distributions of particular amino acids. However, this is unusual in mixtures with relatively few components.

Comparison with other methods for computing these probabilities

In this section, we compare the different results obtained when estimating the expected amino acids using three methods: Dirichlet mixture priors, substitution matrices and pseudocounts. A brief analysis of the differences is contained in the subsections below. In addition, we give examples of the different results produced by these methods in several tables. Tables IV–VII show the different amino acid estimates produced by each method for the cases where 1–10 isoleucines are aligned in a column, with no other amino acids.

Substitution matrices. The need for incorporating prior information about amino acid distributions into protein alignment motivated the development of amino acid substitution matrices. These have been used effectively in database search and discrimination tasks (George *et al.*, 1990; Altschul, 1991; Henikoff and Henikoff, 1992; Jones *et al.*, 1992; Henikoff and Henikoff, 1993; Claverie, 1994; Rodionov and Johnson, 1994).

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, a phenylalanine is a phenylalanine is a phenylalanine. However, a phenylalanine seen in one context, for instance, a position requiring an aromatic residue, will have different substitution probabilities than a phenylalanine seen in a context requiring a large non-polar residue. 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 any pure phenylalanine column, whether it contains 1, 3 or 100 phenylalanines. All three situations are treated identically, and the estimates produced are indistinguishable.

Pseudocount methods. Pseudocount methods can be viewed as 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 normalized. More precisely, the formula used is $\hat{p}_i := (n_i + z_i)/(\sum_j n_j + z_j)$, where each z_j is some

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

	Substitution matrices		Pseudocount	Dirichlet mixture
	Blosum62	SM-Opt.	PC-Opt.	Blocks9
A	0.055	0.028	0.046	0.037
C	0.018	0.008	0.010	0.010
D	0.020	0.006	0.026	0.008
E	0.020	0.011	0.031	0.012
F	0.044	0.025	0.021	0.027
G	0.022	0.011	0.033	0.012
H	0.010	0.005	0.014	0.006
I	**0.253**	**0.517**	**0.495**	**0.472**
K	0.027	0.011	0.031	0.014
L	**0.147**	**0.117**	**0.046**	**0.117**
M	0.037	0.026	0.017	0.030
N	0.017	0.009	0.025	0.010
P	0.018	0.008	0.018	0.008
Q	0.016	0.008	0.023	0.010
R	0.019	0.010	0.027	0.012
S	0.027	0.015	0.039	0.020
T	0.048	0.025	0.033	0.028
V	**0.171**	**0.146**	**0.042**	**0.149**
W	0.006	0.003	0.006	0.004
Y	0.024	0.011	0.017	0.013

Tables IV–VII 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 include two substitution matrices: *Blosum62*, which does Gribskov average score (Gribskov *et al.*, 1987) using the Blosum-62 matrix (Henikoff and Henikoff, 1992), and *SM-Opt*, which does matrix multiply with matrix optimized for the Blocks database (Karplus, 1995a); one pseudocount method, *PC-Opt*, which is a single-component Dirichlet density optimized for the Blocks database (Karplus, 1995a); and the Dirichlet mixture *Blocks9*, the nine-component Dirichlet mixture given in Tables I and II.

In order to interpret the changing amino acid probabilities produced by the Dirichlet mixture, *Blocks9*, we recommend examining this table in conjunction with Table III which shows the changing contribution of the components in the mixture as the number of isoleucines increases. In the estimate produced by the Dirichlet mixture, isoleucine has a probability just under 0.5 when a single isoleucine is observed, and the other aliphatic residues have significant probability. This reveals the influence of components 5 and 6, with their preference for allowing substitutions with valine, leucine and methionine. By 10 observations, the number of isoleucines observed dominates the pseudocounts added for other amino acids, and the amino acid estimate is peaked around isoleucine.

The pseudocount method *PC-Opt* also converges to the observed frequencies in the data, as the number of isoleucines increases, but does not give any significant probability to the other aliphatic residues when the number of isoleucines is small.

By contrast, the substitution matrices give increased probability to the aliphatic residues, but the estimated probabilities remain fixed as the number of isoleucines increases.

constant. Pseudocount methods have many of the desirable properties of Dirichlet mixtures—in particular, that the estimated amino acids converge to the observed frequencies as the number of observations increases—but because they have only a single component, they are unable to represent as complex a set of prototypical distributions.

Table V. Estimated amino acid probabilities using various methods, given three isoleucines. See the caption to Table IV for details

	Substitution matrices		Pseudocount	Dirichlet mixture
	Blosum62	SM-Opt.	PC-Opt.	Blocks9
A	0.055	0.028	0.024	0.018
C	0.018	0 008	0.005	0.005
D	0.020	0.006	0.014	0.003
E	0.020	0.011	0.016	0.004
F	0.044	0.025	0.011	0.013
G	0.022	0.011	0.017	0.006
H	0.010	0.005	0.007	0.002
I	**0.253**	**0.517**	**0.737**	**0.737**
K	0.027	0 011	0.016	0.005
L	**0.147**	**0.117**	**0.024**	**0.059**
M	0.037	0.026	0.009	0.015
N	0.017	0.009	0.013	0.004
P	0.018	0.008	0.009	0.004
Q	0.016	0.008	0.012	0.004
R	0.019	0.010	0.014	0.004
S	0.027	0.015	0.020	0.008
T	0.048	0.025	0.017	0.013
V	**0.171**	**0.146**	**0.022**	**0.089**
W	0.006	0.003	0.003	0.002
Y	0.024	0.011	0.009	0.006

Dirichlet mixtures. Dirichlet mixtures address the problems encountered in substitution matrices and in pseudocounts.

The inability of both substitution matrices and pseudocount methods to represent more than one context for the amino acids is addressed by the multiple components of Dirichlet mixtures. These components enable a

Table VI. Estimated amino acid probabilities using various methods, given five isoleucines. See the caption to Table IV for details

	Substitution matrices		Pseudocount	Dirichlet mixture
	Blosum62	SM-Opt.	PC-Opt.	Blocks9
A	0.055	0.028	0.016	0.010
C	0.018	0.008	0.004	0.003
D	0.020	0.006	0.009	0.002
E	0.020	0.011	0.011	0.002
F	0.044	0.025	0.008	0.007
G	0.022	0.011	0.012	0.004
H	0.010	0.005	0.005	0.001
I	**0.253**	**0.517**	**0.822**	**0.846**
K	0.027	0.011	0.011	0.003
L	**0.147**	**0.117**	**0.016**	**0.034**
M	0.037	0.026	0.006	0.008
N	0.017	0.009	0.009	0.002
P	0.018	0.008	0.006	0.002
Q	0.016	0.008	0.008	0.002
R	0.019	0.010	0.009	0.002
S	0.027	0.015	0.014	0.004
T	0.048	0.025	0.012	0.007
V	**0.171**	**0.146**	**0.015**	**0.054**
W	0.006	0.003	0.002	0.001
Y	0.024	0.011	0.006	0.003

Table VII. Estimated amino acid probabilities using various methods, given 10 isoleucines. See the caption to Table IV for details

	Substitution matrices		Pseudocount	Dirichlet mixture
	Blosum62	Blocks-Opt.	Blocks-Opt.	Blocks9
A	0.055	0.028	0.009	0.004
C	0.018	0.008	0.002	0.001
D	0.020	0.006	0.005	0.001
E	0.020	0.011	0.006	0.001
F	0.044	0.025	0.004	0.003
G	0.022	0.011	0.006	0.002
H	0.010	0.005	0.003	0.001
I	**0.253**	**0.517**	**0.902**	**0.942**
K	0.027	0.011	0.006	0.001
L	**0.147**	**0.117**	**0.009**	**0.012**
M	0.037	0.026	0.003	0.003
N	0.017	0.009	0.005	0.001
P	0.018	0.008	0.003	0.001
Q	0.016	0.008	0.005	0.001
R	0.019	0.010	0.005	0.001
S	0.027	0.015	0.008	0.002
T	0.048	0.025	0.006	0.003
V	**0.171**	**0.146**	**0.008**	**0.020**
W	0.006	0.003	0.001	0.001
Y	0.024	0.011	0.003	0.001

mixture to represent a variety of contexts for each amino acid. It is important to note that the components in the mixture do not always represent prototypical distributions and are, instead, used in combination to give high probability to these commonly found distributions. Sometimes a component will represent a prototypical distribution, at other times such a distribution is represented by a combination of components; in some cases, multiple distributions will be represented by a single component.

For example, the mixture density shown in Tables I and II presents several contexts for isoleucine. A pure isoleucine distribution would be given high probability by component 9, which gives high probability to all conserved distributions. Components 5, 6 and 8 prefer isoleucine found in combination with other amino acids. In producing an estimate for the expected amino acids, the formula [equation (15)] gives those components that are most likely to have generated the observed amino acids the greatest impact on the estimation. Table III shows the change in the posterior probabilities of the components as a variable number of isoleucines are observed (with no other amino acids).

Dirichlet mixtures also address the second drawback associated with substitution matrices—the importance of the actual number of residues observed—in the formula used to compute the expected amino acids. In this formula, given no observations, the estimated amino acid probabilities approximate the background distribution, but as more data become available, the estimate for a column becomes increasingly 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 that a residue is conserved at a particular position, the expected amino acid probabilities produced by this method will remain focused on that residue, instead of being modified to include all the residues that substitute on average for the conserved residue.

Dirichlet mixtures were shown to give superior results in encoding multiple alignments and in database discrimination experiments in comparison with various pseudocount and substitution matrix-based methods in (Brown *et al.*, 1993; Tatusov *et al.*, 1994; Karplus, 1995a; Henikoff and Henikoff, 1996).

Algorithm

Computing amino acid probabilities

The raw frequencies in small samples are often poor approximations to the distribution of amino acids among all proteins which the model is supposed to represent. This section will show how to use Dirichlet priors to form $\hat{p}_i$ estimates that are good approximations of the actual p_i values.

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 prior probabilities for each amino acid distribution with the observed amino acid counts to form estimates, $\hat{p}_i$, of the probabilities of each amino acid i at that position.

We assume that the distribution of amino acids can be modeled by the following generative stochastic process:

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 Prob $(\vec{p}|\vec{\alpha}_j)$, the probability defined by component j over all such distributions.
3. Finally, the observed amino acids are generated independently according to the distribution $\vec{p}$. Thus, the count vector $\vec{n}$ summarizing the observed amino acids in a column will be distributed according to the multinomial distribution with parameters $\vec{p}$.

When Θ consists of a single component, the probability of the components is one, and the stochastic process consists of steps 2 and 3.

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

$$\hat{p}_i := \text{Prob (amino acid } i|\Theta,\vec{n}) \quad (6)$$

$$= \int_{\vec{p}} \text{Prob (amino acid } i|\vec{p}) \text{ Prob } (\vec{p}|\Theta,\vec{n})\mathrm{d}\vec{p} \quad (7)$$

The first term in the integral, Prob (amino acid $i|\vec{p}$), is simply p_i, the ith element of the distribution vector $\vec{p}$. The second term, Prob ($\vec{p}|\Theta,\vec{n}$), represents the posterior probability of the distribution $\vec{p}$ under the Dirichlet density with parameters Θ, given that we have observed amino acid counts $\vec{n}$. The integral 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*.

Computing probabilities using a single density (pseudocounts). 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

$$\hat{p}_i = \int_{\vec{p}} \text{Prob (amino acid } i|\vec{p}) \text{ Prob } (\vec{p}|\vec{\alpha},\vec{n})\mathrm{d}\vec{p} \quad (8)$$

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

Lemma 4:

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

where Γ is the Gamma function, the continuous generalization of the integer factorial function (i.e. $\Gamma(x+1)=x!$).

Here we can substitute p_i for Prob (amino acid $i|\vec{p}$) and the result of Lemma 4 into equation (8), giving

$$\hat{p}_i := \frac{\Gamma(|\vec{\alpha}|+|\vec{n}|)}{\prod_{k=1}^{20}\Gamma(\alpha_k+n_k)} \int_{\vec{p}} p_i \prod_{k=1}^{20} p_k^{\alpha_k+n_k-1}\mathrm{d}\vec{p} \quad (9)$$

Now, noting the contributions of the p_i term within the integral, and using equation (47) from Lemma 2, giving $\int_{\vec{p}}\prod_i p^{\alpha_i-1}\mathrm{d}\vec{p} = \prod_i\Gamma(\alpha_i)/\Gamma(|\vec{\alpha}|)$, we have

$$\hat{p}_i := \frac{\Gamma(|\vec{\alpha}|+|\vec{n}|)}{\Gamma(|\vec{\alpha}|+|\vec{n}|+1)} \frac{\Gamma(\alpha_i+n_i+1)\prod_{k\neq i}\Gamma(\alpha_k+n_k)}{\prod_{k=1}^{20}\Gamma(\alpha_k+n_k)} \quad (10)$$

Since $\Gamma(n+1)/\Gamma(n) = n!/(n-1)! = n$, we obtain

$$\hat{p}_i := \int_{\vec{p}} p_i \text{ Prob } (\vec{p}|\vec{\alpha},\vec{n})\mathrm{d}\vec{p} = \frac{n_i+\alpha_i}{|\vec{n}|+|\vec{\alpha}|} \quad (11)$$

The method in the case of a single Dirichlet density can thus be seen as adding a vector $\vec{\alpha}$ of *pseudocounts* to the vector $\vec{n}$ of observed counts, and then normalizing so that $\sum_i \hat{p}_i = 1$.

Note, when $|\vec{n}| = 0$, 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. While not necessarily the background frequency of the amino acids in the training set, this mean is often a close approximation. Thus, in the absence of data, our estimate of the expected amino acid probabilities will be close to the background frequencies. The computational simplicity of the pseudocount method is one of the reasons why Dirichlet densities are so attractive.

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

$$\hat{p}_i := \text{Prob (amino acid } i|\Theta,\vec{n})$$

$$= \int_{\vec{p}} \text{Prob (amino acid } i|\vec{p}) \text{ Prob } (\vec{p}|\Theta,\vec{n})\mathrm{d}\vec{p} \quad (12)$$

As in the case of the single density, we can substitute p_i for Prob (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 find Prob ($\vec{p}|\Theta,\vec{n}$) obtaining

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

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

Pulling out terms not depending on $\vec{p}$ from inside the integral gives us

$$\hat{p}_i := \sum_{j=1}^{l} \text{Prob } (\vec{\alpha}_j|\vec{n},\Theta) \int_{\vec{p}} \text{Prob } (\vec{p}|\vec{\alpha}_j,\vec{n})\mathrm{d}\vec{p} \quad (14)$$

Substituting equation (11) [equation (15) was misreported in previous work (Brown *et al.*, 1993; Karplus, 1995a,b)]:

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

Thus, 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. Then, each component contributes pseudocounts proportional to the posterior probability that it produced the observed counts. This probability is calculated using Bayes' Rule:

$$\text{Prob}\,(\vec{\alpha}_j|\vec{n},\Theta) = \frac{q_j\,\text{Prob}\,(\vec{n}|\vec{\alpha}_j,|\vec{n}|)}{\text{Prob}\,(\vec{n}|\Theta,|\vec{n}|)} \qquad (16)$$

Prob $(\vec{n}|\vec{\alpha}_j,|\vec{n}|)$ is the probability of the count vector $\vec{n}$ given the jth component of the mixture, and is derived in section A.3 in the Appendix. The denominator, Prob $(\vec{n}|\Theta,|\vec{n}|)$, is defined

$$\text{Prob}\,(\vec{n}|\Theta,|\vec{n}|) = \sum_{k=1}^{l} q_k\,\text{Prob}\,(\vec{n}|\vec{\alpha}_k,|\vec{n}|) \qquad (17)$$

Equation (15) reveals a smooth transition between reliance on the prior information, in the absence of sufficient data, and confidence in the observed frequencies as the number of observations increases. When $|\vec{n}| = 0$, $\hat{p}_i$ 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. As the number of observations increases, the n_i values dominate the α_i values, and this estimate approaches the maximum likelihood estimate, $\hat{p}_i := n_i/|\vec{n}|$.

When a component has a very small $|\vec{\alpha}|$, it adds a very small bias to the observed amino acid frequencies. 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 given relatively small numbers of observed counts.

By contrast, components having a larger $|\vec{\alpha}|$ tend to favor mixed distributions, i.e. 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.

Estimation of Dirichlet densities

In this section, we give the derivation of the procedure to estimate the parameters of a mixture prior. 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, instead of beginning with secondary structure or other column labeling, our approach takes unlabeled training data (i.e. columns from multiple alignments with no information attached) and attempts to discover those classes of distributions of amino acids that are intrinsic to the data. The statistical method 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. aromatic residues), but we do not set out *a priori* to find distributions representing known environments.

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 need only estimate 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. (This two-stage process is not necessary; we have also implemented an algorithm for mixture estimation that optimizes all parameters simultaneously. However, the performance of these mixtures is no better, and the math is more complex.)

As the derivations that follow can become somewhat complex, we provide two tables in the Appendix to help the reader: Table VIII summarizes our notation, and Table IX contains an index to key derivations and definitions.

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 data set. 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 have used Maximum Likelihood to estimate the parameters Θ from the set of count vectors, i.e. we seek those parameters that maximize the probability of occurrence of the observed count vectors. We assume that the three-stage stochastic model described in the section on computing amino acid probabilities 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 $\prod_{t=1}^{m} \text{Prob}\ (\vec{n}_t|\Theta, |\vec{n}_t|)$. If we take the negative logarithm of this quantity, we obtain the *encoding cost* of all the count vectors under the mixture. Since the encoding cost of the count vectors is inversely related to their probability, we can equivalently seek a mixture density with parameters Θ that *minimizes* the encoding cost

$$f(\Theta) = -\sum_{t=1}^{m} \log \text{Prob}\ (\vec{n}_t|\Theta, |\vec{n}_t|) \qquad (18)$$

In the simplest case, we fix the number of components l in the Dirichlet mixture to a particular value and then estimate the $21l-1$ parameters (20 α_i values for each of the components, and $l-1$ mixture coefficients). In other experiments, we attempt to estimate l as well. The simplest method to estimate l involves estimating several Dirichlet mixtures for each number of components, and choosing the smallest mixture that performs well enough for our purposes. Unfortunately, even for fixed l, there does not appear to be an efficient method of estimating these parameters that is guaranteed to find the maximum likelihood estimate. However, a variant of the standard expectation-maximization (EM) algorithm for mixture density estimation works well in practice: 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). [An introduction to this method of mixture density estimation is given in Duda and Hart (1973). We have modified their procedure to estimate a mixture of Dirichlet rather than Gaussian densities.] Since there are many rather different local optima with similar performance, no optimization technique is likely to find the global optimum. The mixture described in Tables I and II is the best local optimum we have found in many different optimizations.

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 f [equation (18)] with respect to $w_{j,i}$ is

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

We will use two lemmas in this section, the proofs for which are given in the Appendix:

Lemma 5:

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

Lemma 6:

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

where $\Psi(x) = \Gamma'(x)/\Gamma(x)$. Using Lemma 5, we obtain

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

Using Lemma 6, and the fact that $\partial \alpha_{j,i}/\partial w_{j,i} = \alpha_{j,i}$, we obtain

$$\frac{\partial f(\Theta)}{\partial w_{j,i}} = -\sum_{t=1}^{m} \alpha_{j,i} \text{Prob}\ (\vec{\alpha}_j|\vec{n}_t, \Theta) \times (\Psi(|\vec{\alpha}_j|) - \Psi(|\vec{n}_t| + |\vec{\alpha}_j|) + \Psi(n_{t,i} + \alpha_{j,i}) - \Psi(\alpha_{j,i})) \qquad (21)$$

To optimize 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, $0 < \eta \ll 1$) during each iteration of the learning cycle. Thus, the gradient descent rule in the mixture case can now be defined as follows:

$$w_{j,i}^{new} := w_{j,i}^{old} - \eta \frac{\partial f(\Theta)}{\partial w_{j,i}} \qquad (22)$$

$$:= w_{j,i}^{old} + \eta \sum_{t=1}^{m} \alpha_{j,i} \text{Prob}\ (\vec{\alpha}_j|\vec{n}_t, \Theta) \times (\Psi(|\vec{\alpha}_j|) - \Psi(|\vec{n}_t| + |\vec{\alpha}_j|) + \Psi(n_{t,i} + \alpha_{j,i}) - \Psi(\alpha_{j,i})) \qquad (23)$$

Now, letting $S_j = \sum_{t=1}^{m} \text{Prob}\ (\vec{\alpha}_j|\vec{n}_t, \Theta)$, this gives us

$$w_{j,i}^{new} := w_{j,i}^{old} + \eta \alpha_{j,i} \Bigg(S_j(\Psi(|\vec{\alpha}_j|) - \Psi(\alpha_{j,i})) + \sum_{t=1}^{m} \text{Prob}\ (\vec{\alpha}_j|\vec{n}_t, \Theta) \times (\Psi(n_{t,i} + \alpha_{j,i}) - \Psi(|\vec{n}_t| + |\vec{\alpha}_j|)) \Bigg) \qquad (24)$$

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

$$w_i^{new} := w_i^{old} + \eta\alpha_i \times \left(m(\Psi(|\vec{\alpha}|) - \Psi(\alpha_i)) + \sum_{t=1}^{m}(\Psi(n_{t,i} + \alpha_i) - \Psi(|\vec{n}_t| + |\vec{\alpha}|))\right) \quad (25)$$

After each update of the weights, the $\vec{\alpha}$ parameters are reset, and the process continued until the change in the encoding cost [equation (18)] falls below some pre-defined cutoff.

Mixture coefficient estimation. In the case of a mixture of Dirichlet densities, the mixture coefficient, q, of the component is also estimated. However, since we require that the mixture coefficients must be non-negative and sum to one, we first reparameterize, setting $q_j = Q_j/|Q|$, where the Q, are constrained to be strictly positive, and $|Q| = \sum_{j=1}^{l} Q_j$. As in the first stage, we want to maximize the probability of the data given the model, which is equivalent to minimizing the encoding cost [equation (18)]. In this stage, we take the derivative of f with respect to Q_j. 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 $q_j = Q_j/|Q|$ that maximize the probability of the data. As we will see, however, the new q_j are a function of the previous q_j; thus, this estimation process must also be iterated.

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

$$\frac{\partial f(\Theta)}{\partial Q_j} = -\sum_{t=1}^{m}\frac{\partial \log \text{Prob }(\vec{n}_t|\Theta,|\vec{n}_t|)}{\partial Q_j} \quad (26)$$

We introduce Lemma 8 (the proof for which is found in section A.8 in the Appendix).

Lemma 8:

$$\frac{\partial \log \text{Prob }(\vec{n}|\Theta,|\vec{n}|)}{\partial Q_j} = \frac{\text{Prob }(\vec{\alpha}_j|\vec{n},\Theta)}{Q_j} - \frac{1}{|Q|}$$

Using Lemma 8, we obtain

$$\frac{\partial f(\Theta)}{\partial Q_j} = -\sum_{t=1}^{m}\left(\frac{\text{Prob }(\vec{\alpha}_j|\vec{n}_t,\Theta)}{Q_j} - \frac{1}{|Q|}\right) \quad (27)$$

$$= \frac{m}{|Q|} - \frac{\sum_{t=1}^{m}\text{Prob }(\vec{\alpha}_j|\vec{n}_t,\Theta)}{Q_j} \quad (28)$$

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_j is

$$q_j := \frac{Q_j}{|Q|} \quad (29)$$

$$:= \frac{1}{m}\sum_{t=1}^{m}\text{Prob }(\vec{\alpha}_j|\vec{n}_t,\Theta) \quad (30)$$

Here, the re-estimated mixture coefficients are functions of the old mixture coefficients, so we iterate this process until the change in the encoding cost falls below the predefined cutoff. (It is easy to confirm that these coefficients sum to one, as required, since $\sum_{j=1}^{l}\sum_{t=1}^{m}$ Prob $(\vec{\alpha}_j|\vec{n}_t,\Theta) = \sum_{t=1}^{m}\sum_{j=1}^{l}$ Prob $(\vec{\alpha}_j|\vec{n}_t,\Theta) = \sum_{t=1}^{m} 1 = m$.)

In summary, when estimating the parameters of a mixture prior, we alternate between re-estimating 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.

Implementation

Implementing Dirichlet mixture priors for use in HMMs 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 splits the implementation details into two groups: those that are essential for getting working Dirichlet mixture code (next section), and those that increase efficiency, but are not essential (section on 'Efficiency improvements').

Essential details

In the section 'Computing amino acid probabilities', 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:

$$\hat{p}_i := \frac{n_i + \alpha_i}{|\vec{n}| + |\vec{\alpha}|} \quad (31)$$

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),

$$\hat{p}_i = \sum_{j=1}^{l}\text{Prob }(\vec{\alpha}_j|\vec{n},\Theta)\frac{n_i + \alpha_{j,i}}{|\vec{n}| + |\vec{\alpha}_j|} \quad (32)$$

The interesting part for computation comes in computing Prob $(\vec{\alpha}_j|\vec{n},\Theta)$ [see equation (16)]. We can expand Prob $(\vec{n}|\Theta,|\vec{n}|)$ using equation (17) to obtain

$$\text{Prob } (\vec{\alpha}_j|\vec{n},\Theta) = \frac{q_j \text{ Prob } (\vec{n}|\vec{\alpha}_j,|\vec{n}|)}{\sum_{k=1}^{l} q_k \text{ Prob } (\vec{n}|\vec{\alpha}_k,|\vec{n}|)} \tag{33}$$

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

First, we can expand Prob $(\vec{n}|\vec{\alpha}_j,|\vec{n}|)$ using Lemma 3 (the proof of which is found in section A.3 in the Appendix):

$$\text{Prob } (\vec{n}|\vec{\alpha}_j,|\vec{n}|) = \frac{\Gamma(|\vec{n}|+1)\Gamma(|\vec{\alpha}_j|)}{\Gamma(|\vec{n}|+|\vec{\alpha}_j|)} \prod_{i=1}^{20} \frac{\Gamma(n_i+\alpha_{j,i})}{\Gamma(n_i+1)\Gamma(\alpha_{j,i})} \tag{34}$$

If we rearrrange some terms, we obtain

$$\text{Prob } (\vec{n}|\vec{\alpha}_j,|\vec{n}|) = \frac{\prod_{i=1}^{20}\Gamma(n_i+\alpha_{j,i})}{\Gamma(|\vec{n}|+|\vec{\alpha}_j|)} \frac{\Gamma(|\vec{\alpha}_j|)}{\prod_{i=1}^{20}\Gamma(\alpha_{j,i})} \times \frac{\Gamma(|\vec{n}|+1)}{\prod_{i=1}^{20}\Gamma(n_i+1)} \tag{35}$$

The first two terms are most easily expressed using the Beta function: $\text{B}(x) = \prod_{i=1}^{20}\Gamma(x_i)/\Gamma(|\vec{x}|)$, where, as usual, $|\vec{x}| = \sum_i x_i$. This simplifies the expression to

$$\text{Prob } (\vec{n}|\vec{\alpha}_j,|\vec{n}|) = \frac{\text{B}(\vec{n}+\vec{\alpha}_j)}{\text{B}(\vec{\alpha}_j)} \frac{\Gamma(|\vec{n}|+1)}{\prod_{i=1}^{20}\Gamma(n_i+1)} \tag{36}$$

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

$$\text{Prob } (\vec{\alpha}_j|\vec{n},\Theta) = \frac{q_j \frac{\text{B}(\vec{n}+\vec{\alpha}_j)}{\text{B}(\vec{\alpha}_j)}}{\sum_{k=1}^{l} q_k \frac{\text{B}(\vec{n}+\vec{\alpha}_k)}{\text{B}(\vec{\alpha}_k)}} \tag{37}$$

Plugging this formula into equation (32) gives us

$$\hat{p}_i := \frac{\sum_{j=1}^{l} q_j \frac{\text{B}(\vec{n}+\vec{\alpha}_j)}{\text{B}(\vec{\alpha}_j)} \frac{n_i+\alpha_{j,i}}{|\vec{n}|+|\vec{\alpha}_j|}}{\sum_{k=1}^{l} q_k \frac{\text{B}(\vec{n}+\vec{\alpha}_k)}{\text{B}(\vec{\alpha}_k)}} \tag{38}$$

Since the dominator of equation (38) is independent of i, we can compute $\hat{p}_i$ by normalizing

$$X_i = \sum_{j=1}^{l} q_j \frac{\text{B}(\vec{n}+\vec{\alpha}_j)}{\text{B}(\vec{\alpha}_j)} \frac{n_i+\alpha_{j,i}}{|\vec{n}|+|\vec{\alpha}_j|} \tag{39}$$

to sum to one. That is

$$\hat{p}_i = \frac{X_i}{\sum_{k=1}^{20} X_k} \tag{40}$$

The biggest problem that implementers 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:

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

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:

$$\begin{aligned}X_i &= \sum_{j=1}^{l} q_j \frac{\text{B}(\vec{\alpha}_j+\vec{n})}{\text{B}(\vec{\alpha}_j)} \frac{\alpha_{j,i}+n_i}{|\vec{\alpha}_j|+|\vec{n}|} \\ &= \sum_{j=1}^{l} q_j e^{(\log \text{B}(\vec{\alpha}_j+\vec{n}) - \log \text{B}(\vec{\alpha}_j))} \frac{\alpha_{j,i}+n_i}{|\vec{\alpha}_j|+|\vec{n}|}\end{aligned}$$

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 divide 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 $\log \text{B}(\vec{\alpha}_j+\vec{n}) - \log \text{B}(\vec{\alpha}_j)$

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. (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.)

Efficiency improvements

The previous section gave simple computation formulas for $\hat{p}_i$ [equations (39) and (40)]. When computations of $\hat{p}_i$ are done infrequently (e.g. 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.

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 pre-compute $\log \mathrm{B}(\vec{\alpha}_j)$, cutting the computation time almost in half.

If the n_i values are mainly small integers (zero 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. Pre-computation 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 pre-computation 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 are possible when x is close to previously computed values, by using interpolation rather than calling `lgamma`.

Discussion

The methods employed to estimate and use Dirichlet mixture priors are shown to be firmly based on Bayesian statistics. While biological knowledge has been introduced only indirectly from the multiple alignments used to estimate the mixture parameters, the mixture priors produced agree with accepted biological understanding. The effectiveness of Dirichlet mixtures for increasing the ability of statistical models to recognize homologous sequences has been demonstrated experimentally (Brown *et al.*, 1993; Tatusov *et al.*, 1994; Bailey and Elkan, 1995; Karplus, 1995a; Henikoff and Henikoff, 1996; Hughey and Krogh, 1996; Wang *et al.*, 1996).

The mixture priors we have estimated thus far have been on unlabeled multiple alignment columns—columns with no secondary structure or other information attached. Previous work deriving structurally informed distributions, such as that by Lüthy *et al.* (1991), has 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 (Bowie *et al.*, 1991). However, these distributions cannot be used in a Bayesian framework, since there is no measure of the variance associated with each distribution, and Bayes' rule requires that the observed frequency counts be modified inversely proportional to the variance in the distribution. Thus, to use these structural distributions one must assign a variance arbitrarily. We plan to estimate Dirichlet mixtures for particular environments, and to make these mixtures available on the World-Wide Web.

Dirichlet mixture priors address two primary weaknesses of substitution matrices: considering only 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 matrices 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 corrects these problems. When little data are available, the amino acids predicted are those that are known to be associated in different contexts with the amino acids observed. As the available data increase, the amino acid probabilities produced by this method converge to the observed frequencies in the data. In particular, when evidence exists that a particular amino acid is conserved at a given position, the expected amino acid estimates reflect this preference.

Because of the sensitivity of Dirichlet mixtures to the number of observations, any significant correlation among the sequences must be handled carefully. One way to compensate for sequence correlation is by the use of a sequence weighting scheme (Sibbald and Argos, 1990; Thompson *et al.*, 1994a,b; Henikoff and Henikoff, 1996). Dirichlet mixtures interact with sequence weighting in two ways. First, sequence weighting changes the expected distributions somewhat, making mixed distributions more uniform. Second, the total weight allotted the sequences plays a critical role when Dirichlet densities are used. If the data are highly correlated, and this is not compensated for in the weighting scheme (by reducing the total counts), the estimated amino acid distributions will be too close to the raw frequencies in the data, and not generalized to include similar residues. Since most sequence weighting methods are concerned only with relative weights, and pay little attention to the total weight allotted the sequences, we are developing sequence weighting schemes that coordinate the interaction of Dirichlet mixtures and sequence weights.

Since the mixture presented in this paper was estimated and tested on alignments of fairly close homologs [the BLOCKS (Henikoff and Henikoff, 1991) and HSSP (Sander and Schneider, 1991) alignment databases], it may not accurately reflect the distributions we would expect from more remote homologs. We are planning to train a Dirichlet mixture specifically to recognize true remote homologies, by a somewhat different training technique on a database of structurally aligned sequences.

Finally, as the detailed analysis of Karplus (1995a,b) 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, and assuming independence between columns and between sequences for a given column. 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.

The Dirichlet mixture prior from Table I is available electronically at our World-Wide Web site `http://www.cse.ucsc.edu/research/compbio/`. In addition to the extensions described above, we plan to make programs for using and estimating Dirichlet mixture densities available on our World-Wide Web and ftp sites later this year. See our World-Wide Web site for announcements.

Acknowledgements

We gratefully acknowledge the input and suggestions of Stephen Altschul, Tony Fink, Lydia Gregoret, Steven and Jorja Henikoff, Graeme Mitchison and Chris Sander. Richard Lathrop made numerous suggestions that improved the quality of the manuscript greatly, as did the anonymous referees. 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 National Science Foundation Graduate Research Fellowship and funds granted by the UCSC Division of Natural Sciences. The Sanger Centre is supported by the Wellcome Trust. This paper is dedicated to the memory of Tal Grossman, a dear friend and a true mensch.

References

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

Altschul,S.F., Gish,W., Miller,W., Meyers,E.W. and Lippman,D.J. (1990) Basic local alignment search tool. *JMB*, **215**, 403–410.

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, pp. 783–791.

Bailey,T.L. and Elkan,C. (1995) The value of prior knowledge in discovering motifs with MEME. In *ISMB-95*, Menlo Park, CA. AAAI/MIT Press, pp. 21–29.

Baldi,P. and Chauvin,Y. (1994) Smooth on-line learning algorithms for hidden Markov models. *Neural Computation*, **6**, 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**, 389–402.

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

Bernardo,J.M. and Smith,A.F.M. (1994) *Bayesian Theory*, 1st edn. John Wiley and Sons.

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. (eds), *ISMB-93*, Menlo Park, CA. AAAI/MIT Press, pp. 47–55.

Bucher,P., Karplus,K., Moeri,N. and Hoffman,K. (1996) A flexible motif search technique based on generalized profiles. *Computers Chem.*, **20**, 3–24.

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

Claverie,J.-M. (1994) Some useful statistical properties of position-weight matrices. *Computers Chem.*, **18**, 287–294.

Dempster,A.P., Laird,N.M. and Rubin,D.B. (1977) Maximum likelihood from incomplete data via the *EM* algorithm, *J. R. 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, CA.

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

Eddy,S.R. (1995) Multiple alignment using hidden Markov models. In *ISMB-95*, Menlo Park, CA. AAAI/MIT Press, pp. 114–120

Eddy,S.R. (1996) Hidden Markov models. *Curr. Opin. Struct. Biol.* **6**, 361–365.

Eddy,S.R., Mitchison,G. and Durbin,R. (1995) Maximum discrimination hidden Markov models of sequence consensus. *J. Comput. Biol.*, **2**, 9–23

George,D.G., Barker,W.C. and Hunt,L.T. (1990) Mutation data matrix and its uses. *Methods Enzymol.*, **183**, 333–351.

Gradshteyn,I.S. and Ryzhik,I.M. (1965) *Table of Integrals, Series, and Products*, 4th edn. Academic Press.

Gribskov,M., McLachlan,A.D. and Eisenberg,D. (1987) Profile analysis: Detection of distantly related proteins. *Proc. Natl Acad. Sci. USA*, **84**, 4355–4358.

Gribskov,M., Lüthy,R. and Eisenberg,D. (1990) Profile analysis. *Methods Enzymol.*, **183**, 146–159.

Henikoff,S. and Henikoff,J.G. (1991) Automated assembly of protein blocks for database searching. *Nucleic Acids Res.*, **19**, 6565–6572.

Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA*, **89**, 10915–10919.

Henikoff,S. and Henikoff,J.G. (1993) Performance evaluation of amino acid substitution matrices. *Proteins: Structure, Function Genet.*, **17**, 49–61.

Henikoff,J.G. and Henikoff,S. (1996) Using substitution probabilities to improve position-specific scoring matrices. *Comput. Applic. Biosci.* **12**, 135–143.

Henikoff,S., Wallace,J.C. and Brown,J.P. (1990) Finding protein similarities with nucleotide sequence databases. *Methods Enzymol.*, **183**, 111–132.

Hughey,R. and Krogh,A. (1996) Hidden Markov models for sequence analysis: Extension and analysis of the basic method. *Comput. Applic. Biosci.*, **12**, 95–107.

Jones,D.T., Taylor,W.R. and Thornton,J.M. (1992) The rapid generation of mutation data matrices from protein sequences. *Comput. Applic. Biosci.*, **8**, 275–282.

Karplus,K. (1995a) Regularizers for estimating distributions of amino acids from small samples. In *ISMB-95*, Menlo Park, CA. AAAI/MIT Press.

Karplus,K. (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.

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, Gent.*, **10**, 229–239.

Nowlan,S. (1990) Maximum likelihood competitive learning. In Touretsky,D. (ed.), *Advances in Neural Information Processing Systems*, Volume 2. Morgan Kaufmann, pp. 574–582.

Rodionov,M.A. and Johnson,M.S. (1994) Residue-residue contact substitution probabilities derived from aligned three-dimensional structures and the identification of common folds. *Protein Sci.*, **3**, 2366–2377.

Sander,C. and Schneider,R. (1991) Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins*, **9**, 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 Sci.*, **2**, 305–315.

Tatusov,R.L., Altschul,S.F. and Koonin,E.V. (1994) Detection of conserved segments in proteins: Iterative scanning of sequence databases with alignment blocks. *Proc. Natl Acad. Sci. USA*, **91**, 12091–12095.

Thompson,J.D., Higgins,D.G. and Gibson,T.J. (1994a) Improved sensitivity of profile searches through the use of sequence weights and gap excision. *Comput. Applic. Biosci.*, **10**, 19–29.

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

Wang,J.T.L., Marr,T.G., Shasha,D., Shapiro,B., Chirn,G.-W. and Lee,T.Y. (1996) Complementary classification approaches for protein sequences. *Protein Eng.* **9**, 381–386

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

White,J.V., Stultz,M. and Smith,T.F. (1994) Protein classification by stochastic modeling and optimal filtering of amino-acid sequences. *Math. Biosci.*, **119**, 35–75.

Received on April 15, 1996; revised and accepted on June 4, 1996

Appendix

Table VIII. Summary of notation

$|\vec{x}| = \sum_i x_i$, where $\vec{x}$ is any vector.
$\vec{n} = n_1, \ldots, n_{20}$ is a vector of counts from a column in a multiple alignment. The symbol n_i refers to the number of amino acids i in the column. The tth such observation in the data is denoted $\vec{n}_t$.
$\vec{p} = (p_1, \ldots, p_{20})$, $\sum p_i = 1$, $p_i \geq 0$, 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. The parameters of the jth component of a Dirichlet mixture are denoted $\vec{\alpha}_j$. The symbol $\alpha_{j,i}$ refers to the ith parameter of the jth component of a mixture.
$q_j = \text{Prob}\ (\vec{\alpha}_j)$ is the *mixture coefficient* of the jth component of a 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 *weight* vectors, used during gradient descent to train the Dirichlet density parameters $\vec{\alpha}$. After each training cycle, $\alpha_{j,i}$ is set to $e^{w_{j,i}}$. The symbol $w_{j,i}$ is the value of the ith parameter of the jth *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 a mixture.
η = *eta*, the learning rate used to control the size of the step taken during each iteration of gradient descent.

Table IX. Index to key derivations and definitions

$f(\Theta)$	$= -\sum_{t=1}^{m} \log(\mathrm{Prob}(\vec{n}_t \mid \Theta, \lvert\vec{n}_t\rvert))$ (the encoding cost of all the count vectors given the mixture—the function minimized)	(18)
$\Gamma(\lvert\vec{n}\rvert+1)$	$= n!$ (for integer $n \geq 0$) (Gamma function)	(43)
$\Psi(x)$	$= \dfrac{\partial \log \Gamma(x)}{\partial x} = \dfrac{\Gamma'(x)}{\Gamma(x)}$ (Psi function)	(64)
$\mathrm{Prob}(\vec{n} \mid \vec{p}, \lvert\vec{n}\rvert)$	$= \Gamma(\lvert\vec{n}\rvert+1) \prod_{i=1}^{20} \dfrac{p_i^{n_i}}{\Gamma(n_i+1)}$ (the probability of $\vec{n}$ under the multinomial distribution with parameters $\vec{p}$)	(44)
$\mathrm{Prob}(\vec{n} \mid \vec{\alpha}, \lvert\vec{n}\rvert)$	$= \dfrac{\Gamma(\lvert\vec{n}\rvert+1)\Gamma(\lvert\vec{\alpha}\rvert)}{\Gamma(\lvert\vec{n}\rvert+\lvert\vec{\alpha}\rvert)} \prod_{i=1}^{20} \dfrac{\Gamma(n_i+\alpha_i)}{\Gamma(n_i+1)\Gamma(\alpha_i)}$ (the probability of $\vec{n}$ under the Dirichlet density with parameters $\vec{\alpha}$)	(51)
$\mathrm{Prob}(\vec{n} \mid \Theta, \lvert\vec{n}\rvert)$	$= \sum_{k=1}^{l} q_k \, \mathrm{Prob}(\vec{n} \mid \vec{\alpha}_k, \lvert\vec{n}\rvert)$ (the probability of $\vec{n}$ given the entire mixture prior)	(17)
$\mathrm{Prob}(\vec{\alpha}_j \mid \vec{n}, \Theta)$	$= \dfrac{q_j \, \mathrm{Prob}(\vec{n} \mid \vec{\alpha}_j, \lvert\vec{n}\rvert)}{\mathrm{Prob}(\vec{n} \mid \Theta, \lvert\vec{n}\rvert)}$ (shorthand for the posterior probability of the jth component of the mixture given the vector of counts $\vec{n}$)	(16)

(A.1) Lemma 1

$$\mathrm{Prob}(\vec{n} \mid \vec{p}, \lvert\vec{n}\rvert) = \Gamma(\lvert\vec{n}\rvert+1) \prod_{i=1}^{20} \frac{p_i^{n_i}}{\Gamma(n_i+1)}$$

Proof:
For a given vector of counts $\vec{n}$, with p_i being the probability of seeing the ith amino acid, and $\lvert\vec{n}\rvert = \sum_i n_i$, there are $\lvert\vec{n}\rvert!/(n_1!n_2!\ldots n_{20}!)$ distinct permutations of the amino acids which result in the count vector $\vec{n}$. If we allow for the simplifying assumption that each amino acid is generated independently (i.e. the sequences in the alignment are uncorrelated), 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

$$\mathrm{Prob}(\vec{n} \mid \vec{p}, \lvert\vec{n}\rvert) = \frac{\lvert\vec{n}\rvert!}{n_1!n_2!\ldots n_{20}!} \prod_{i=1}^{20} p_i^{n_i} \quad (41)$$

$$= \lvert\vec{n}\rvert! \prod_{i=1}^{20} \frac{p_i^{n_i}}{n_i!}. \quad (42)$$

To enable us to handle real-valued data (such as those 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,

$$\Gamma(n+1) = n! \quad (43)$$

Substituting the Gamma function, we obtain the equivalent form

$$\mathrm{Prob}(\vec{n} \mid \vec{p}, \lvert\vec{n}\rvert) = \Gamma(\lvert\vec{n}\rvert+1) \prod_{i=1}^{20} \frac{p_i^{n_i}}{\Gamma(n_i+1)}. \quad (44)$$

(A.2) Lemma 2

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

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:

$$\mathrm{Prob}(\vec{p} \mid \vec{\alpha}) = \frac{\prod_{i=1}^{20} p_i^{\alpha_i - 1}}{\int_{\vec{p} \in \mathcal{P}} p_i^{\alpha_i - 1} \mathrm{d}\vec{p}}. \quad (45)$$

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

$$\mathrm{B}(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\mathrm{d}t$$

$$= \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$

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

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

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

$$\int_{\vec{p}\in\mathscr{P}} \prod_i p^{\alpha_i-1}\mathrm{d}\vec{p} = \mathrm{B}(\alpha_1, \alpha_2+\cdots+\alpha_{20}) \times \mathrm{B}(\alpha_2, \alpha_3+\cdots+\alpha_{20}) \ldots \mathrm{B}(\alpha_{19}, \alpha_{20}) \tag{46}$$

$$= \frac{\prod_i \Gamma(\alpha_i)}{\Gamma(|\vec{\alpha}|)}. \tag{47}$$

We can now given an explicit definition of the probability of the amino acid distribution $\vec{p}$ given the Dirichlet density with parameters $\vec{\alpha}$:

$$\text{Prob }(\vec{p}|\vec{\alpha}) = \frac{\Gamma(|\vec{\alpha}|)}{\prod_{i=1}^{20}\Gamma(\alpha_i)} \prod_{i=1}^{20} p_i^{\alpha_i-1}. \tag{48}$$

(A.3) Lemma 3

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

Proof:
We can substitute equations (44) and (48) into the identity

$$\text{Prob }(\vec{n}|\vec{\alpha}, |\vec{n}|) = \int_{\vec{p}\in\mathscr{P}} \text{Prob }(\vec{n}|\vec{p}, |\vec{n}|) \text{ Prob }(\vec{p}|\vec{\alpha})\mathrm{d}\vec{p}, \tag{49}$$

giving

$$= \int_{\vec{p}\in\mathscr{P}} \frac{\Gamma(|\vec{n}|+1)\Gamma(|\vec{\alpha}|)}{\prod_{i=1}^{20}(\Gamma(n_i+1)\Gamma(\alpha_i))} \prod_{i=1}^{20} p^{n_i+\alpha_i-1}\mathrm{d}\vec{p}. \tag{50}$$

Pulling out terms not depending on $\vec{p}$ from inside the integral, using the result from equation (47), and rearranging terms, we obtain

$$= \frac{\Gamma(|\vec{n}|+1)\Gamma(|\vec{\alpha}|)}{\Gamma(|\vec{n}|+|\vec{\alpha}|)} \prod_{i=1}^{20} \frac{\Gamma(n_i+\alpha_i)}{\Gamma(n_i+1)\Gamma(\alpha_i)}. \tag{51}$$

(A.4) Lemma 4

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

Proof:
By repeated application of the rule for conditional probability, 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

$$\text{Prob }(\vec{p}|\vec{\alpha}, \vec{n}) = \frac{\text{Prob }(\vec{p}, \vec{\alpha}, \vec{n}\,|\,|\vec{n}|)}{\text{Prob }(\vec{\alpha}, \vec{n}\,|\,|\vec{n}|)} \tag{52}$$

$$= \frac{\text{Prob }(\vec{n}|\vec{p}, \vec{\alpha}, |\vec{n}|) \text{ Prob }(\vec{p}, \vec{\alpha})}{\text{Prob }(\vec{n}|\vec{\alpha}, |\vec{n}|) \text{ Prob }(\vec{\alpha})} \tag{53}$$

$$= \frac{\text{Prob }(\vec{n}|\vec{p}, \vec{\alpha}, |\vec{n}|) \text{ Prob }(\vec{p}|\vec{\alpha})}{\text{Prob }(\vec{n}|\vec{\alpha}, |\vec{n}|)}. \tag{54}$$

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

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

At this point, we apply the results from previous derivations for quantities Prob $(\vec{n}|\vec{p}, |\vec{n}|)$ [equation (44)], Prob $(\vec{p}|\vec{\alpha})$ [equation (48)] and Prob $(\vec{n}|\vec{\alpha}, |\vec{n}|)$ [equation (51)]. This gives us

$$\text{Prob }(\vec{p}|\vec{\alpha}, \vec{n}) = \frac{\left(\frac{\Gamma(|\vec{n}|+1)}{\prod_i \Gamma(n_i+1)} \prod_i p_i^{n_i}\right)\left(\frac{\Gamma(|\vec{\alpha}|)}{\prod_i \Gamma(\alpha_i)} \prod_i p_i^{\alpha_i-1}\right) \Gamma(|\vec{n}|+|\vec{\alpha}|) \prod_i \Gamma(n_i+1)\Gamma(\alpha_i)}{\Gamma(|\vec{n}|+1)\Gamma(|\vec{\alpha}|) \prod_i \Gamma(n_i+\alpha_i)} \tag{56}$$

Most of the terms cancel, and we have

$$\text{Prob }(\vec{p}|\vec{\alpha}, \vec{n}) = \frac{\Gamma(|\vec{\alpha}|+|\vec{n}|)}{\prod_{i=1}^{20}\Gamma(\alpha_i+n_i)} \prod_{i=1}^{20} p_i^{\alpha_i+n_i-1}. \tag{57}$$

Note that this is the expression for a Dirichlet density with parameters $\vec{\alpha}+\vec{n}$. 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

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

Proof:
The derivative with respect to $\alpha_{j,i}$ of the log likelihood of each count vector $\vec{n}$ given the mixture is

$$\frac{\partial \log\ \text{Prob}\ (\vec{n}|\Theta, |\vec{n}|)}{\partial \alpha_{j,i}} = \frac{1}{\text{Prob}\ (\vec{n}|\Theta, |\vec{n}|)}\frac{\partial\ \text{Prob}\ (\vec{n}|\Theta, |\vec{n}|)}{\partial \alpha_{j,i}}. \tag{58}$$

Applying equation (17), this gives us

$$\frac{\partial \log\ \text{Prob}\ (\vec{n}|\Theta, |\vec{n}|)}{\partial \alpha_{j,i}} = \frac{1}{\text{Prob}\ (\vec{n}|\Theta, |\vec{n}|)}\frac{\partial \sum_{k=1}^{L} q_k\ \text{Prob}\ (\vec{n}|\vec{\alpha}_k, |\vec{n}|)}{\partial \alpha_{j,i}}. \tag{59}$$

Since the derivative of Prob $(\vec{n}|\vec{\alpha}_k, |\vec{n}|)$ 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

$$\frac{\partial \log\ \text{Prob}\ (\vec{n}|\Theta, |\vec{n}|)}{\partial \alpha_{j,i}} = \frac{q_j}{\text{Prob}\ (\vec{n}|\Theta, |\vec{n}|)}\frac{\partial\ \text{Prob}\ (\vec{n}|\vec{\alpha}_j, |\vec{n}|)}{\partial \alpha_{j,i}}. \tag{60}$$

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

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

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

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

(A.6) Lemma 6

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

Proof:
In this proof, we use Lemma 3, giving

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

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

$$\frac{\partial \log\ \text{Prob}\ (\vec{n}|\vec{\alpha}, |\vec{n}|)}{\partial \alpha_i} = \frac{\partial \log \Gamma(|\vec{\alpha}|)}{\partial \alpha_i} - \frac{\partial \log \Gamma(|\vec{n}|+|\vec{\alpha}|)}{\partial \alpha_i} + \frac{\partial \log \Gamma(n_i+\alpha_i)}{\partial \alpha_i} - \frac{\partial \log \Gamma(\alpha_i)}{\partial \alpha_i}. \tag{63}$$

Now, if we substitute the shorthand

$$\Psi(x) = \frac{\partial \log \Gamma(x)}{\partial x} = \frac{\Gamma'(x)}{\Gamma(x)}, \tag{64}$$

we have

$$\frac{\partial \log\ \text{Prob}\ (\vec{n}|\vec{\alpha}, |\vec{n}|)}{\partial \alpha_i} = \Psi(|\vec{\alpha}|) - \Psi(|\vec{n}| + |\vec{\alpha}|) + \Psi(n_i + \alpha_i) - \Psi(\alpha_i). \tag{65}$$

(A.7) Lemma 7

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

Proof:
Substituting equation (17), giving Prob $(\vec{n}|\Theta, |\vec{n}|) = \sum_{j=1}^{l} q_j$ Prob $(\vec{n}|\vec{\alpha}_j, |\vec{n}|)$ and replacing q_j by $Q_j/|Q|$, we obtain

$$\frac{\partial\ \text{Prob}\ (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_i} = \frac{\partial \sum_{j=1}^{l}(Q_j/|Q|)\ \text{Prob}\ (\vec{n}|\vec{\alpha}_j, |\vec{n}|)}{\partial Q_i}. \tag{66}$$

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

$$\frac{\partial\ \text{Prob}\ (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_i} = \sum_{j=1}^{l}\left((Q_j/|Q|)\frac{\partial\ \text{Prob}\ (\vec{n}|\vec{\alpha}_j, |\vec{n}|)}{\partial Q_i} + \text{Prob}\ (\vec{n}|\vec{\alpha}_j, |\vec{n}|)\frac{\partial (Q_j/|Q|)}{\partial Q_i}\right). \tag{67}$$

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

$$\frac{\partial \text{ Prob } (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_i} = \sum_{j=1}^{l} \text{Prob } (\vec{n}|\vec{\alpha}_j, |\vec{n}|) \frac{\partial (Q_j/|Q|)}{\partial Q_i}. \tag{68}$$

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

$$\frac{\partial (Q_j/|Q|)}{\partial Q_i} = |Q|^{-1}\frac{\partial Q_j}{\partial Q_i} + Q_j\frac{\partial |Q|^{-1}}{\partial Q_i} \tag{69}$$

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

$$\frac{\partial \text{ Prob } (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_i} = \frac{\text{Prob } (\vec{n}|\vec{\alpha}_i, |\vec{n}|)}{|Q|} - \sum_{j=1}^{l} \text{Prob } (\vec{n}|\vec{\alpha}_j, |\vec{n}|) \frac{Q_j}{|Q|^2}. \tag{70}$$

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

$$= \frac{\text{Prob } (\vec{n}|\vec{\alpha}_i, |\vec{n}|) - \sum_{j=1}^{l} q_j \text{ Prob } (\vec{n}|\vec{\alpha}_j, |\vec{n}|)}{|Q|}. \tag{71}$$

At this point, we use equation (17) and obtain

$$\frac{\partial \text{ Prob } (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_i} = \frac{\text{Prob } (\vec{n}|\vec{\alpha}_i, |\vec{n}|) - \text{ Prob } (\vec{n}|\Theta, |\vec{n}|)}{|Q|}. \tag{72}$$

(A.8) Lemma 8

$$\frac{\partial \log \text{ Prob } (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_j} = \frac{\text{Prob } (\vec{\alpha}_j|\vec{n}, \Theta)}{Q_j} - \frac{1}{|Q|}$$

Proof:
Using Lemma 7, we can derive

$$\frac{\partial \log \text{Prob } (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_j}$$

$$= \frac{1}{\text{Prob } (\vec{n}|\Theta, |\vec{n}|)} \frac{\partial \text{ Prob } (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_j} \tag{73}$$

$$= \frac{1}{\text{Prob } (\vec{n}|\Theta, |\vec{n}|)} \times \frac{\text{Prob } (\vec{n}|\vec{\alpha}_j, |\vec{n}|) - \text{Prob } (\vec{n}|\Theta, |\vec{n}|)}{|Q|} \tag{74}$$

$$= \left(\frac{\text{Prob } (\vec{n}|\vec{\alpha}_j, |\vec{n}|)}{\text{Prob } (\vec{n}|\Theta, |\vec{n}|)} - 1\right)/|Q|. \tag{75}$$

If we rearrange equation (16), we obtain $\text{Prob}(\vec{n}|\vec{\alpha}_j, |\vec{n}|)/\text{Prob}(\vec{n}|\Theta, |\vec{n}|) = \text{Prob}(\vec{\alpha}_j|\vec{n}, \Theta)/q_j$. This allows us to write

$$\frac{\partial \log \text{ Prob } (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_j} = \left(\frac{\text{Prob } (\vec{\alpha}_j|\vec{n}, \Theta)}{q_j} - 1\right)/|Q|. \tag{76}$$

Now we can use the identity $q_j = Q_j/|Q|$, obtaining the equivalent

$$\frac{\partial \log \text{ Prob } (\vec{n}|\Theta, |\vec{n}|)}{\partial Q_j} = \frac{\text{Prob } (\vec{\alpha}_j|\vec{n}, \Theta)}{Q_j} - \frac{1}{|Q|}. \tag{77}$$