Efficient functional clustering of protein sequences using the Dirichlet process

Duncan P. Brown1,2

1Department of Bioengineering, UC Berkeley and 2Merck & Co., Inc., 1700 Owens St, San Francisco, CA 94158, USA

Received on March 10, 2008; revised and accepted on May 22, 2008

ABSTRACT

Motivation: Automatic clustering of protein sequences is an important problem in computational biology. The recent explosion in genome sequences has given biological researchers a vast number of novel protein sequences. However, the majority of these sequences have no experimental evidence for their molecular function in the cell, and the responsibility for correctly annotating these sequences falls upon the bioinformatics community. Ideally, we would like to be able to group sequences of similar or identical molecular function in an automatic fashion, without relying on experimental evidence.

Results: In this article I present a novel probabilistic framework that models subfamilies within a known protein family. Given a multiple sequence alignment, the model uses Dirichlet mixture densities to estimate amino acid preferences within subfamily clusters, and places a Dirichlet process prior on the overall set of clusters. Based on results from several datasets, the model breaks data accurately into functional subgroups.

Availability: The algorithm is implemented as c++ software available at bpg-research.berkeley.edu/~duncanb/dpccluster/

Contact: duncan_brown@merck.com

Supplementary information: Supplementary data are available at Bioinformatics online.

1 INTRODUCTION

The vast majority of known protein sequences have no experimental evidence for their function, but rather are annotated by homology to sequences of known function. As sequence databases continue to grow, the field of comparative genomics will only become more important. As a result, there has been considerable interest in automatic methods for grouping sequences of similar function.

Many methods for protein clustering depend on the derivation of a tree or graph linking the given set of sequences (Abascal and Valencia, 2002; Enright and Ouzounis, 2000; Krause and Vingron, 1998; Li and Godzik, 2001; Yona et al., 1998). These methods tend to use pairwise sequence similarity to define the graph and the clusters, since it is computationally inexpensive, and require setting a similarity threshold parameter to cut the tree into clusters. Sjölander (1998) proposed a truly automatic method that minimizes an encoding cost calculated from a Dirichlet mixture. However, all these algorithms depend on producing a graph topology that adequately reflects the functional groups, which is by no means guaranteed.

A recent publication described a probabilistic model that uses a mixture of Gaussian distributions under a Dirichlet process (DP) prior to detect sequence clusters (Duube et al., 2004) within protein families. The model allows the number of clusters to be defined by the data, and is not constrained by tree topology. However, categorical data such as protein sequences are not natively amenable to inference using the Gaussian. The algorithm transforms sequences into Fisher score vectors using parameters from an HMM trained on the input sequences and then greatly reduces the dimensionality of the data using principal component analysis.

This article proposes a novel solution to the protein clustering problem that is also based on the DP mixture model, but which models the sequence data directly. I focus on the problem of subfamily detection within a known protein family, given a multiple sequence alignment as input. The model efficiently detects latent classes of sequences within a protein family, which very often correspond to functionally distinct subtypes. I compare its performance to several available algorithms on three distinct datasets.

2 APPROACH

2.1 Dirichlet processes

I give here a brief intuitive overview of the DP mixture model; for more details see Neal (2000). Let \( G_0 \) be a probability measure defined over a measurable space \( \Theta \). \( G(a, G_0) \) is defined to be a DP if, for any finite partition \( A = (A_1, \ldots, A_N) \) of \( \Theta \), the probability of \( A \) is distributed as a Dirichlet density with parameters \((a G_0(A_1), \ldots, a G_0(A_N))\). This definition is precise, but not particularly helpful in visualizing the DP object itself or in understanding its utility. The DP functions as a distribution on distributions; a particular draw from a DP is itself a complete probability distribution from which we can sample. Sethuraman (1994) showed that any draw from a DP must be a countably infinite discrete distribution, even when \( G_0 \) is continuous. This 'discretization' property makes the DP extremely useful as a prior in clustering applications, since it allows multiple samples to possess the same value, and does not fix a priori the number of clusters in the model.

Blackwell and MacQueen (1973) showed that it is possible to generate data samples \( \theta_i \) from a DP according to a Pólya urn model:

\[
\begin{align*}
\theta_i & \sim G_0 \\
\theta_i & | \theta_{<i} \sim \frac{1}{1 + a} \sum_{j=1}^{i-1} \delta(\theta_j) + \frac{a}{1 + a} G_0 
\end{align*}
\]
Gc model to the set of cluster indicators $Gc$ draws according to Equation (1). Here, $\theta$ is a countably infinite mixture, in which unique component values $\phi_i$ are drawn from $Gc_0$; cluster sizes ($n_c$, not to scale) are a consequence of Equation (1).

where $\theta_{c,i}$ is the set of $\theta$ values preceding $i$, and $\delta$ is the delta function. The probability of assigning $y_j$ to an existing $c$ is proportional to the number of points already assigned to $c$, and the probability of assigning it to a new cluster is proportional to $\alpha$. This process generates a set of clusters, where all $\theta$ assigned to a given cluster $c$ have the same value, $\phi_c$. The limiting distribution of the $\theta$ is a countably infinite mixture, in which unique component values $\phi_i$ are drawn from $Gc_0$, and the component proportions correspond to the number of $\theta$ in each cluster (Fig. 1). The cluster sizes can also be modeled directly with a ‘stick-breaking’ distribution (Neal, 2000).

We can utilize this clustering effect by incorporating the DP into a mixture model framework. We treat a set of data $y=(y_1, \ldots, y_n)$ as independent draws from a mixture of distributions $\text{F}(y_i, \theta)$, where $\theta$ is drawn from $Gc$ (and therefore from the pool of discrete $\phi$), rather than from $Gc_0$, as would normally be the case. As with the standard model, when the underlying distribution $Gc_0$ is conjugate to $F$, we may integrate out $\theta$ to obtain the posterior distribution of the clusters given only the observed data.

$$P(c \in (c_{-i}, y_{-i}) \propto \alpha_{i-1+\alpha} \int \text{F}(y_i, \phi) dH_{c,i}(\phi)$$

$$P(c \notin (c_{-i}, y_{-i}) \propto \alpha_{i-1+\alpha} \int \text{F}(y_i, \phi) dG_{c_0}(\phi)$$

where $\alpha_{c,i}$ is the set of $\alpha$ values preceding $i$, and $\delta$ is the delta function. The probability of assigning $y_j$ to an existing $c$ is proportional to the number of points already assigned to $c$, and the probability of assigning it to a new cluster is proportional to $\alpha$. This process generates a set of clusters, where all $\theta$ assigned to a given cluster $c$ have the same value, $\phi_c$. The limiting distribution of the $\theta$ is a countably infinite mixture, in which unique component values $\phi_i$ are drawn from $Gc_0$, and the component proportions correspond to the number of $\theta$ in each cluster (Fig. 1). The cluster sizes can also be modeled directly with a ‘stick-breaking’ distribution (Neal, 2000).

We can utilize this clustering effect by incorporating the DP into a mixture model framework. We treat a set of data $y=(y_1, \ldots, y_n)$ as independent draws from a mixture of distributions $\text{F}(y_i, \theta)$, where $\theta$ is drawn from $Gc$ (and therefore from the pool of discrete $\phi$), rather than from $Gc_0$, as would normally be the case. As with the standard model, when the underlying distribution $Gc_0$ is conjugate to $F$, we may integrate out $\theta$ to obtain the posterior distribution of the clusters given only the observed data.

The integral in (3) becomes

$$\int \text{F}(y_i, \phi) dG_{c_0}(\phi) = \frac{\prod_{j=1}^{L} \text{Mult}(y_{ij}, \phi_j)}{Z(1)} = \frac{\prod_{j=1}^{L} \phi_j^{y_{ij}}}{\sum_{k=1}^{M} \phi_k^{y_{kj}} Z(k)}$$

where $\phi_k$ is a multinomial distribution for column $j$. The most straightforward sampling algorithm for this model is Gibbs sampling, although the Gibbs algorithm mixes poorly and must be
assigning all other points probability proportional to the likelihood of \( y_k \) given in the Supplementary Materials.

2.3.2 Split-merge sampling The Gibbs sampler is straightforward to implement, but when the dimensionality of the data becomes large, as in this model, the Gibbs sampler can often fail to mix well and will become stuck for long periods of time in one configuration. This occurs because Gibbs sampling updates only single data points; the sampler must pass through low-probability states in which a small number of data points ‘break away’ to establish their own cluster. Jain and Neal (2000) proposed a Metropolis–Hastings algorithm that overcomes this issue by updating multiple data points at once, providing effective mixing within the state space.

Their algorithm chooses points \( y_i \) and \( y_j \) at random from the dataset and if these are in the same cluster proposes a split move, assigning all other points \( y_k \) in the cluster to join \( y_i \) or \( y_j \) with probability proportional to the likelihood of \( y_k \) under either new cluster. Conversely, if the points are in different clusters, the algorithm proposes to merge the two clusters. The method uses split and merge updates to complement Gibbs sampling scans; the split–merge update is efficient in jumping between cluster configurations, while the Gibbs scan is efficient in assigning single data points to their most probable cluster. For results reported here, I chose to use a variant of the split–merge algorithm proposed by Dahl (2003), which was significantly faster for this model. Algorithm details and equations for the proposal, prior and likelihood distributions are given in the Supplementary Materials.

The split–merge samplers described in Jain and Neal (2000) and Dahl (2003) alternate between performing split–merge updates and full Gibbs scans across the whole dataset. However, I found this to be inefficient for the protein model. Sequences were infrequently moved between clusters during Gibbs scans, and these usually followed accepted split-merge updates. I reduced the frequency of Gibbs sampling scans to one per 50 split–merge updates. This seemed to be a good balance, in that most Gibbs scans still did not reassign data points (and thus were not too infrequent), and the average time per iteration was reduced approximately 20-fold. Note that this procedure still produces a valid chain, since the split–merge updates and Gibbs scans each sample from the posterior.

3 RESULTS

3.1 Datasets

The goal of the algorithm is to accurately cluster functionally similar sequences within a protein family. I tested the performance of the split–merge sampler on two datasets that have been previously used to evaluate sequence protein clustering solutions (Brown et al., 2007), and one novel set. The EXPERT set consisted of five extensively curated protein families from three different resources, with additional subdivisions of two of the families to create a total of eight classifications of expert-defined subtypes. These included the amine and secretin families from the GPCRDB (Horn et al., 2003), the nuclear hormone receptor (NHR) family from NucleaRDB (Horn et al., 2001), and the enolase and coronase enzyme families from the Structure–Function Linkage Database (SFLD; Pegg et al. (2006)), a manually curated resource that incorporates mechanistic, sequence and structural information for several diverse enzyme families to derive functional subtypes based on conserved chemical mechanism.

The second dataset was a larger, uncurated set of 57 PFAM family alignments, matched to SCOP superfamilies by scoring superfamily members against the PFAM HMM. The set is non-redundant at the level of SCOP fold. Families were divided into functional subtypes based on all four digits of members’ EC number (Webb et al., 1992). Families with fewer than two EC numbers were removed. The algorithms were given the entire family as input, but were only scored on sequences which possessed EC numbers and aligned over at least 75% of the alignment.

For the final dataset, I examined the algorithms’ ability to divide sequences in SCOP superfamilies into their family designations. I chose SCOP superfamilies having five or more families, for a total of 50 superfamilies in the set. Superfamily sequences were taken from Astral 90 (Chandonia et al., 2007) and aligned with Muscle (Edgar, 2004) before submission to the algorithms.

3.2 Comparing partitions

I used several scoring functions to evaluate the performance of clustering methods. These were also used in (Brown et al., 2007). Subfamily purity is measured as the fraction of subfamilies that contain only one expert subtype or EC number.

The variation of information (VI) is a distance metric on partitions (Meila, 2003). As such, it obeys the triangle inequality: \( VI(A, B) + VI(B, C) \geq VI(A, C) \) for partitions \( A, B, C \). Given two partitions, the VI index measures the amount of information in each partition that is not shared between them. It is calculated as

\[
VI = H(S) + H(S') - 2I(S, S')
\]

where \( H \) is the entropy of a partition, and \( I \) is the mutual information between two partitions:

\[
H(S) = -\sum_{k=1}^{K} \frac{n_k}{N} \log \frac{n_k}{N}
\]

\[
I(S, S') = \sum_{k=1}^{K} \sum_{k' \neq k} \frac{n_k n_{k'}}{N^2} \log \frac{n_k n_{k'}}{n_k' n_k}
\]

Here, \( n_k \) is the number of items in cluster \( k \) of partition \( S \), \( n_{k'} \) is the number of overlapping items between cluster \( k \) in partition \( S \) and cluster \( k' \) in partition \( S' \), \( k \) and \( k' \) are the total number of clusters in partitions \( S \) and \( S' \), respectively, and \( N \) is the total number of items in the set.

The edit distance is defined as the minimum number of split or merge operations required to transform one partition into the other. A split or merge affecting multiple data points is considered one operation. The edit distance is calculated as

\[
\text{Edit} = 2\sum_{k, k'} n_{k,k'} - K - K'
\]

where \( n_{k,k'} \) equals 1 if clusters \( k \) and \( k' \) have items in common, and 0 otherwise. Like the VI distance, the edit distance is a metric on partitions.

The purity and distance scores are complimentary: an algorithm that places each sequence in its own cluster will have perfect purity, but very poor distance scores. The relationship between the VI and edit distances is somewhat more subtle. The VI distance places more weight on large clusters, producing high scores when they
are incorrect, whereas the edit distance weights each cluster equally, therefore penalizing methods producing many singletons.

I ran the sampler for 10,000 iterations on each family alignment, performing a Gibbs scan over all sequences every 50 iterations. I then evaluated the posterior likelihood of each distinct ε state and chose the state with the highest likelihood for assessment.

### 3.3 EXPERT set

Using the EXPERT set, I compared DPCluster, SCI-PHY (Brown et al., 2007), Ncut (Abascal and Valencia, 2002), Secator (Wicker et al., 2001) and two similarity thresholds for CD-HIT (Li and Godzik, 2006): 40% and 70% within-cluster identity. The results show that DPCluster is definitively the best of the methods tested (Table 1). CD-HIT 70 consistently achieved the highest purity, but also greatly over-divided the families, based on its high distance scores. DPCluster had the next-best subfamily purity in four of the eight classifications, and also achieved the lowest VI distance in three of the eight, and the lowest edit distance in five. No method performed as consistently well overall.

The performance of the Ncut algorithm was quite inconsistent. In some cases, such as for the crotonase family, it had low VI and edit distance, but relatively poor purity. In the enolases, however, the algorithm greatly over-divided the family, receiving high purity, but also quite high distance scores. In the amine family, Ncut produced only eight non-singleton subfamilies, seven of which were pure. However, the one impure subfamily contained 317 sequences, 89% of the family.

DPCluster results for the crotonase family were particularly good; only one (cluster 10) of the 26 clusters mixed subtypes, and 88 of the 89 sequences in cluster 10 were of the same subtype. The one sequence in error was the only member of its subtype. The singleton was an outlier within its cluster; it shared at most 35% identity with any other sequence. However, cluster 10 as a whole showed conservation at several positions that were not conserved across the whole family, and agreement at these positions likely explains why the singleton was placed in the cluster. This illustrates the point that in general, singleton clusters will be unlikely under this model, unless the sequence is quite different from the other members of the family. This is part of the reason why the Gibbs sampler performs poorly on its own. In fact, DPCluster produced the fewest singletons in every family. This is a significant improvement over the other methods, which tended to produce far too many singletons for this dataset. This can be a problem if the true classification contains many singletons, but note that DPCluster had the lowest VI and edit distances for the very specific NHR Level 3 classification, which was divided into 77 subtypes.

To demonstrate this anti-singleton effect, I split cluster 10 to place the singleton into its own cluster and then assessed the likelihood of the new cluster state. The new configuration received a substantially lower likelihood than the original. I then introduced an additional copy of the singleton into both states, including the copy into either cluster 10 or the new cluster. With two copies, the split configuration received a higher likelihood than incorporating both copies into cluster 10. This bias could be addressed by setting α to a higher value, but this is not a general solution. The likelihood surface is extremely steep; very small changes in cluster structure (such as creation of a singleton) tended to cause tens or hundreds of log value changes in the likelihood. Setting α to a particular (high) value to counteract this effect would not generalize to other families. However, it may be possible to address the issue by placing a prior on α and integrating it out.

The functional subtypes defined in these families are generally representative of the types of divisions that are useful to biologists, but the concept of ‘functional similarity’ is somewhat arbitrary. For example, the three levels of classification within the nuclear receptor family represent increasingly more specific functional distinctions: Level 1 divides the families into general classes of ligand, such as estrogen-like and thyroid-hormone-like; Level 2 subdivides the Level 1 subtypes into finer gradations, for instance separating the estrogen-like group into estrogen, estrogen-related and gluco-corticoid receptors; finally Level 3 splits Level 2 groups into subtypes that bind specific ligand molecules. As shown in Table 1, the set of clusters produced by the DPCluster algorithm fell between Levels 2 and 3 in specificity. The performance of the algorithm may be tuned to reflect a particular level of functional specificity through the exact choice of the Dirichlet mixture prior Go. It may also be possible to adjust for the anti-singleton effect seen in the Crotonase family in the same way. I explore this idea further in Section 3.6.

### 3.4 EC and SCOP sets

Results for the EC dataset, while not as definitive as for the EXPERT set, support the same conclusions (Table 2). DPCluster received higher purity scores than SCI-PHY, while maintaining quite similar distance scores. In fact, DPCluster was the only algorithm to show both better purity and better edit distance than another method. In addition, DPCluster shows similar purity scores to CD-HIT 40 (P < 0.06), and much better VI and edit distance (P < 10⁻⁵ in both cases).

All algorithms except Secator (which collapsed most families together) received high purity scores on the SCOP dataset (Table 3). Both CD-HIT versions had perfect purity but quite high distance scores, suggesting that they over-divided the families. DPCluster achieved the next-highest purity (0.93) and the lowest distance scores among the four high-purity algorithms.

### 3.5 DPCluster sampling efficiency

Overall, DPCluster is fast; 10,000 iterations required between 19 and 132 min on a standard Mac for the families in the Expert set, depending on the size of the alignment. This compares favorably with clustering based on phylogenetic trees, which can require significantly more computation time to estimate (although DPCluster is still far slower than distance methods such as neighbor joining). The run time per iteration is O(n_c,cL,M) where M is the number of components in the Dirichlet mixture Go, L is the number of columns in the input alignment and n_c,c is the number of sequences in the cluster(s) under consideration in the current iteration. Clearly, the clustering structure of the set features significantly in the run time: when the number of clusters is large, the cost is reduced significantly.

Based on trace plots, the sampler quickly reaches equilibrium even when the input alignment is large (Fig. 2), and for most runs the cluster structure was essentially unchanged after 2000 iterations. Figure 2 also shows that the chain mixes well between alternate
Efficient protein sequence clustering

Table 1. DPCluster performance in subfamily identification for the EXPERT set

<table>
<thead>
<tr>
<th></th>
<th>Amine L1</th>
<th>Amine L2</th>
<th>Crotonase</th>
<th>Enolase</th>
<th>NHR L1</th>
<th>NHR L2</th>
<th>NHR L3</th>
<th>Secretin</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subtypes/Sequences</strong></td>
<td>7/358</td>
<td>31/358</td>
<td>10/365</td>
<td>8/472</td>
<td>8/412</td>
<td>27/412</td>
<td>77/412</td>
<td>15/153</td>
</tr>
<tr>
<td><strong>Pure/Total Subfamilies (%)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPCluster</td>
<td>40/41 (98)</td>
<td>35/41 (85)</td>
<td>25/26 (96)</td>
<td>28/30 (93)</td>
<td>37/38 (97)</td>
<td>37/38 (97)</td>
<td>22/38 (58)</td>
<td>17/19 (89)</td>
</tr>
<tr>
<td>SCI-PHY</td>
<td>36/37 (97)</td>
<td>32/37 (87)</td>
<td>15/16 (94)</td>
<td>26/26 (100)</td>
<td>29/29 (100)</td>
<td>26/29 (97)</td>
<td>11/29 (38)</td>
<td>14/16 (88)</td>
</tr>
<tr>
<td>NCUT</td>
<td>7/8 (88)</td>
<td>7/8 (88)</td>
<td>14/16 (88)</td>
<td>65/66 (98)</td>
<td>24/25 (96)</td>
<td>22/25 (88)</td>
<td>10/25 (40)</td>
<td>8/10 (80)</td>
</tr>
<tr>
<td>SECATOR</td>
<td>10/14 (71)</td>
<td>6/14 (42)</td>
<td>3/7 (43)</td>
<td>8/9 (89)</td>
<td>5/5 (100)</td>
<td>1/5 (20)</td>
<td>0/5 (0)</td>
<td>3/6 (50)</td>
</tr>
<tr>
<td>CD-HIT 40</td>
<td>22/33 (67)</td>
<td>16/33 (49)</td>
<td>47/47 (100)</td>
<td>16/18 (89)</td>
<td>28/28 (100)</td>
<td>24/28 (86)</td>
<td>50/28 (30)</td>
<td>7/11 (64)</td>
</tr>
<tr>
<td>CD-HIT 70</td>
<td>67/67 (100)</td>
<td>66/67 (99)</td>
<td>65/65 (100)</td>
<td>73/73 (100)</td>
<td>57/57 (100)</td>
<td>57/57 (100)</td>
<td>41/57 (72)</td>
<td>31/31 (100)</td>
</tr>
<tr>
<td><strong>Edit Distance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPCluster</td>
<td>38</td>
<td>32</td>
<td>18</td>
<td>28</td>
<td>38</td>
<td>27</td>
<td>53</td>
<td>10</td>
</tr>
<tr>
<td>SCI-PHY</td>
<td>38</td>
<td>36</td>
<td>32</td>
<td>70</td>
<td>38</td>
<td>21</td>
<td>54</td>
<td>15</td>
</tr>
<tr>
<td>NCUT</td>
<td>40</td>
<td>62</td>
<td>23</td>
<td>99</td>
<td>30</td>
<td>23</td>
<td>57</td>
<td>14</td>
</tr>
<tr>
<td>SECATOR</td>
<td>49</td>
<td>61</td>
<td>53</td>
<td>32</td>
<td>6</td>
<td>23</td>
<td>71</td>
<td>9</td>
</tr>
<tr>
<td>CD-HIT 40</td>
<td>64</td>
<td>84</td>
<td>55</td>
<td>28</td>
<td>30</td>
<td>21</td>
<td>69</td>
<td>25</td>
</tr>
<tr>
<td>CD-HIT 70</td>
<td>107</td>
<td>85</td>
<td>213</td>
<td>177</td>
<td>90</td>
<td>71</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td><strong>VI Distance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPCluster</td>
<td>1.78</td>
<td>0.82</td>
<td>1.43</td>
<td>1.53</td>
<td>1.98</td>
<td>0.74</td>
<td>0.42</td>
<td>0.42</td>
</tr>
<tr>
<td>SCI-PHY</td>
<td>1.55</td>
<td>0.90</td>
<td>1.05</td>
<td>1.37</td>
<td>1.62</td>
<td>0.39</td>
<td>0.95</td>
<td>0.56</td>
</tr>
<tr>
<td>NCUT</td>
<td>2.08</td>
<td>3.01</td>
<td>0.63</td>
<td>2.75</td>
<td>1.38</td>
<td>0.57</td>
<td>1.21</td>
<td>1.15</td>
</tr>
<tr>
<td>SECATOR</td>
<td>1.44</td>
<td>1.14</td>
<td>1.29</td>
<td>0.87</td>
<td>0.43</td>
<td>1.32</td>
<td>2.39</td>
<td>0.91</td>
</tr>
<tr>
<td>CD-HIT 40</td>
<td>1.94</td>
<td>1.37</td>
<td>2.28</td>
<td>0.56</td>
<td>1.47</td>
<td>0.45</td>
<td>1.16</td>
<td>1.25</td>
</tr>
<tr>
<td>CD-HIT 70</td>
<td>2.53</td>
<td>1.31</td>
<td>3.77</td>
<td>3.19</td>
<td>2.55</td>
<td>1.25</td>
<td>0.70</td>
<td>0.99</td>
</tr>
<tr>
<td><strong>Singletons</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPCluster</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>SCI-PHY</td>
<td>6</td>
<td>6</td>
<td>22</td>
<td>52</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>6</td>
</tr>
<tr>
<td>NCUT</td>
<td>27</td>
<td>27</td>
<td>9</td>
<td>35</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>SECATOR</td>
<td>34</td>
<td>34</td>
<td>46</td>
<td>29</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>CD-HIT 40</td>
<td>4</td>
<td>4</td>
<td>18</td>
<td>14</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>CD-HIT 70</td>
<td>47</td>
<td>47</td>
<td>158</td>
<td>112</td>
<td>41</td>
<td>41</td>
<td>41</td>
<td>19</td>
</tr>
</tbody>
</table>

DPCluster was compared to SCI-PHY, NCUT, SECATOR, CD-HIT 40 and CD-HIT 70. Subtypes/Sequences: number of expert-derived subtypes / number of sequences in each classification. Percentage of pure subfamilies is given in parentheses. Pure/total subfamilies: fraction of pure non-singleton subfamilies / total number of non-singleton subfamilies.

Singletons: number of single-sequence clusters.

high-probability states. Runs of up to 100 000 iterations showed very similar findings: no major changes in the clustering structure were apparent after the initial 'burn-in' phase of approximately 2000 iterations (data not shown).

3.6 The effect of the Dirichlet mixture prior \( G_0 \)

It is worth noting the effect of the Dirichlet mixture prior \( G_0 \) on the performance of the model. Standard Dirichlet mixture priors are trained on large sets of alignment data and reflect the types of conservative substitutions observed in evolution. Each component of the mixture represents a group of amino acids that tend to substitute for each other, observations that match a particular prior component will give that component high posterior probability compared to the other prior components. The prior then in effect provides pseudocounts for amino acids similar to those observed in the data, allowing the posterior to generalize accurately even when there are few observations. In most models, such as standard HMM parameter estimation, the contribution of the prior will quickly shrink as the number of observations grows, and the posterior distribution will tend to reflect the observed amino acid frequencies in this case. However, in the DP model, each cluster generates its own posterior and the number of observed amino acids is dependent only on the size of the cluster. This means that the prior can have a large effect on the posterior likelihood, no matter how many sequences are present in the alignment.

To demonstrate this, I reran the algorithm using a 'null' prior having just one Dirichlet component, with small total mass. This prior essentially rejects any substitutions; posterior amino acid distributions will tend to be based solely on the observed frequencies. Runs for most families produced between 50% and 150% more clusters, most of which had high internal conservation. The large effect of the choice of prior on the results allows us to tune performance depending on the extent of clustering we expect to see. Very specific functional attributes, such as ligand specificity, or fine taxonomic distinctions, might require a prior that rewards individual amino acid conservation, while more general attributes would use a more forgiving prior. This is analogous to the choice of substitution matrices in pairwise homology searches.
Wilcoxon signed rank tests for de novo subfamily detection on the EC dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Purity E Value</th>
<th>Edit E Value</th>
<th>VI E Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scop</td>
<td>0.97</td>
<td>2.07</td>
<td>3.00</td>
</tr>
<tr>
<td>Sci-phy</td>
<td>0.91</td>
<td>18.48</td>
<td>0.84</td>
</tr>
<tr>
<td>DPCluster</td>
<td>0.93</td>
<td>17.94</td>
<td>0.84</td>
</tr>
<tr>
<td>CDHIT 40</td>
<td>1.00</td>
<td>23.24</td>
<td>0.84</td>
</tr>
<tr>
<td>CDHIT 70</td>
<td>0.97</td>
<td>2.07</td>
<td>3.00</td>
</tr>
</tbody>
</table>

Diagonal entries are the mean scores for the method. P: purity score; V: variation of information distance; E: edit distance. Off-diagonal entries are the Wilcoxon signed rank P-values between methods for the corresponding score. In comparison of two methods for a particular score, parentheses around the P-value indicates that the method listed in the column is better; plain values indicate that the row method is better.

4 CONCLUSION

I have presented a highly accurate probabilistic model for clustering protein sequences. The algorithm, based on the DP infinite mixture model, is able to separate sequences into functionally distinct clusters, given a multiple sequence alignment, and achieves excellent accuracy on a set of five manually classified protein families.

For all families, multiple runs produced very similar output, suggesting that the algorithm is converging to optimal or near-optimal maxima despite the very large state space. As noted earlier, due to the high dimensionality of the data, the likelihood surface is very steep, and favorable proposals often have an acceptance probability of unity. This suggests that the columns of the alignment may be redundant; it may be possible to achieve a faster algorithm by reducing the dimensionality of the data under consideration at each update step.

Finally, as with many sequence-based models, the quality of the input alignment is naturally critical to the success of the algorithm; incorrect alignments will degrade performance and lead to artificial clustering. Several algorithms have been proposed that marry alignment sampling to another model (Lunter et al., 2005; Metzler et al., 2001), and I believe that it is possible to extend this algorithm in a similar manner.

ACKNOWLEDGEMENTS

Deep thanks to Kimmen Sjölander for support during this work. Thanks also to several anonymous reviewers for their insights.

Conflict of Interest: none declared.

REFERENCES

Efficient protein sequence clustering


Luntif, G. et al. (2005) Bayesian coestimation of phylogeny and sequence alignment. BMC Bioinformatics, 6, 83.


