UNIVERSITY OF CALIFORNIA, SAN DIEGO

Information-Theoretic and Hypothesis-Based Clustering in Bioinformatics

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science and Engineering

by

Sean Michael O'Rourke

2009

EPIGRAPH

We were hoping for the best, but it turned out as always.
—Victor Chernomyrdin

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would first like to thank my advisor for his support and patience, for sparing me from administrative nightmares, and for a confidence that usually exceeded my own. The tireless Julie Conner was instrumental in navigating the administrative course of the degree. I am also grateful to Noah Zaitlen and Robin Friedman for their grunt-work on the *Alu* and Information Bottleneck projects. Noah deserves additional thanks for sacrificing time, sleep, and an elbow to my Sierra ambitions. I would also like to thank the other members of UCSD's Zarlab for, among other things, expanding my culinary horizons: Hyun Min Kang, Buhm Han, and Jimmie Ye.

I would like to thank (alphabetically) Gal Chechik at Stanford, David Heckerman and Nebojsa Jojic at Microsoft Research, and Alkes Price at UCSD for their good ideas and valuable discussion.

My work has been done mostly or entirely using Free Software, without which it would have been not only more expensive, but also much more difficult. Free programs rarely earn their authors much money or credit, especially in an academic environment. I would therefore like to thank Larry Wall (Perl), John Eaton (Octave), Richard Stallman (GNU Emacs, GCC), Don Knuth (TEX), Leslie Lamport (LATEX), Till Tantau (Beamer), and the many contributors to these and other Free programs I have used.

Finally, I am indebted to Professor Nogar for support and advice when graduating seemed impossible or absurd. Coffee's on me next time.

Chapter 2 is based on work with Gal Chechik, Robin Friedman, and Eleazar Eskin, and published in *BMC Bioinformatics*, 7(Suppl 1):S8, March 2006. Chapter 3 is based on work with Gal Chechik and Eleazar Eskin and published in the proceedings of the NIPS workshop on new methods and problems in computational biology, 2005. Chapter 4 is based on work with Nebojsa Jojic, Noah Zaitlen, and Eleazar Eskin and published in the proceedings of RECOMB 2007.

VITA

1999	B. S. in Computer Science *cum laude*, Rice University, Houston
2002-2008	Graduate Teaching Assistant, University of California, San Diego
2009	Ph. D. in Computer Science, University of California, San Diego

PUBLICATIONS

S O'Rourke and E Eskin. A finite state transducer approach to haplotype phasing. In *NIPS workshop on new methods and problems in computational biology*, 2007.

S O'Rourke, N Zaitlen, N Jojic, and E Eskin. Reconstructing the phylogeny of mobile elements. In TP Speed and H Huang, editors, *RECOMB*, volume 4453 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2007.

S O'Rourke, G Chechik, R Friedman, and E Eskin. Discrete profile comparison using information bottleneck. *BMC Bioinformatics*, 7(Suppl 1):S8, Mar 2006.

S O'Rourke, G Chechik, and E Eskin. Separation of overlapping subpopulations by mutual information. In *NIPS workshop on new methods and problems in computational biology*, 2005.

S O'Rourke, G Chechik, R Friedman, and E Eskin. Discrete profile alignment via constrained information bottleneck. In *NIPS*, 2004.

M Zander, J Hall, J Painter, and S O'Rourke. Component architecture of the tecolote framework. In D Caromel, RR Oldehoeft, and M Tholburn, editors, *ISCOPE*, volume 1505 of *Lecture Notes in Computer Science*, pages 183–190. Springer, 1998.

SP Gary, H Li, S O'Rourke, and D Winske. Proton resonant firehose instability: Temperature anisotropy and fluctuating field constraints. *J Geophys Res*, 103(A7):14,567–74, 1998.

ABSTRACT OF THE DISSERTATION

Information-Theoretic and Hypothesis-Based Clustering in Bioinformatics

by

Sean Michael O'Rourke
Doctor of Philosophy in Computer Science and Engineering

University of California San Diego, 2009

Professor Sanjoy Dasgupta, Chair
Professor Eleazar Eskin, Co-Chair

Many machine learning problems in biology involve clustering data generated in complex or incompletely understood ways. Processes such as protein and viral evolution are difficult to model, involving complex mechanisms and constraints at multiple levels. This thesis presents a family of clustering algorithms, based on the Information Bottleneck method, to cluster such datasets by imposing constraints related to statistical tests of their known properties. The first algorithm clusters continuous data; we apply it to amino acid profiles to derive a compact discrete representation that preserves much of their information. This discretization yields an easily interpretable textual representation of amino acid profiles. It also greatly improves the speed of profile-profile alignment, and makes it possible to index large profile databases. The second algorithm clusters discrete sequences while constraining mutual information between sequence positions within each cluster. We apply it to the problem of finding population substructure in viral and human SNP data, showing it to be competitive with or superior to current approaches.

Biological datasets often strain the limits of modern computers, and advances in biotechnology promise to generate even more data in the future as computational power increases. We therefore present a randomized clustering algorithm for discrete sequences that is similar to the previous algorithm but scalable to much larger datasets. This clustering algorithm relies on statistical tests to perform structure learning, an approach that has the added benefit of naturally limiting model complexity. We use this algorithm to produce detailed phylogenies of large DNA mobile element families. Our results provide a more detailed picture of their history, and their important role in genomic evolution.

Chapter 1

Introduction

Clustering, the classification of unlabeled data into one or more groups, is a long-studied problem for which many algorithms have been developed [Har75,JMF99]. The input data may be images, documents, DNA strings, or vectors of discrete or continuous features extracted from the original data representation. The output is a simplification of the input data suitable for human perusal in exploratory data analysis, or to accept or reject a postulated hypothesis. A vast number of algorithms have been developed over the years to deal with the nature and scale of the input and with the purpose of the output.

Many computational questions in biology can be stated as clustering problems. The most commonly-clustered objects are sequences of amino or nucleic acids, typically represented as strings. Example problems include phylogeny [Fel04], the construction of a hierarchical clustering representing a set of individuals' shared ancestry; population substructure identification [PSD00], the division of a set of individuals into one or more subgroups sharing some common variation; DNA and protein homolog search [AMS+97,YL02], the identification of nucleic and amino acid sequences similar to one's sequence of interest, and therefore likely to have a related function or shared ancestry; and protein domain identification [NY04], the identification of a protein's structural subunits from previous structural knowledge about other proteins.

Other problems require clustering probability distributions or individuals represented by sets of binary or real-valued traits. Examples here include gene expression analysis [JTZ04], and protein tertiary structure prediction [GH03] (the prediction of a protein sequence's 3-dimensional folding structure from a database of known structures).

Models for biological sequences are often complex for two kinds of reasons. First, the processes generating the sequences create complex correlation between sequences and sequence positions. These processes may involve a combination of random single-point variation, copying of parts from one or more parent sequences, and duplication or deletion of subsequences. The rates of these different forms of variation are a product of complex underlying physical interactions. The process is also constrained by evolutionary history, as our own DNA is a function of our ancestors'. This results in useless relics, strange co-options, and suboptimal side-effects — the molecular equivalents of the vestigial coccyx, the proteins forming the eye's lens, and the awkwardly "backward" direction of the optic nerve [TC02, Daw04].

Second, the sequences themselves are often subject to complex functional constraints. For example, since parts of DNA sequences may encode proteins and regulatory signals crucial to a cell's or an organism's survival, the observed distribution of DNA sequences will be highly non-uniform. Further, sequence variations can confer a small selective advantage to an organism, causing the advantageous variants to eventually become dominant. These patterns may remain in DNA long after the conditions in which they conferred a selective advantage are gone.

A generative model for biological sequences must therefore reflect or at least be tolerant of these constraints, and a clustering algorithm must produce results consistent with such a model. In particular, a model of the underlying generative process is often either incomplete or far too complex to be learned in full. Furthermore, modern sequencing technology presents us with enormous datasets to be clustered under these models. A biological clustering algorithm should therefore provide useful results even in the face of incomplete models and vast quantities of data.

1.1 Types of clustering

Different clustering algorithms learn models varying along several dimensions, notably the following: *flat* models, in which clusters contain only datapoints, versus *hierarchical* ones, in which clusters may contain other clusters; *hard* assignment, in which each datapoint is associated with one cluster, versus *soft* assignment, in which each is associated with multiple clusters to a greater or lesser degree; *parametric* models, in which the features of datapoints in each cluster follow some probability distribution, versus *nonparametric* models, in which they do not.

Clustering algorithms also vary in whether they infer the assignment of datapoints to a fixed number of clusters, or whether they also infer the number of clusters. These two tasks are usually called parameter learning and structure learning, respectively. Because it is a discrete optimization problem, structure learning is usually more difficult, often requiring model- or problem-specific heuristics. Finally, structure learning clustering algorithms can vary in their approach: *Top-down* algorithms begin by assigning all individuals to a single cluster, and increase the number of clusters according to some heuristic. *Bottom-up* algorithms, on the other hand, start by assigning each individual to its own cluster, then merging these clusters.

1.2 Overview of this thesis

This thesis describes three related methods for clustering several different types of biological data, both continuous and discrete. Algorithmically, it can be divided in two parts: The first part, Chapters 2 and 3, describes two information-theoretic clustering algorithms based on the Slonim and Tishby's Information Bottleneck algorithm [TPB99]. The second part, Chapter 4, discusses an algorithm related to that in Chapter 3, but based on randomized hypothesis-testing rather than constrained minimization of mutual information. In terms of problems addressed, the thesis can also be divided into two, overlapping parts: Chapter 2 addresses the clustering of discrete probability distributions over amino acids, while Chapters 3 and 4 describe algorithms for clustering strings, typically of DNA.

In Chapter 2, we describe a flat clustering algorithm for continuous data based on Information Bottleneck (IB) that incorporates specified constraints between clusters and priors on cluster distributions. By varying the strength of the constraints and priors, we can control a trade-off between human interpretability and information loss. We apply this algorithm to the problem of finding an informationally optimal clustering of distributions over amino acids, encoded as 20-dimensional vectors.

A protein *profile* is a sequence of probability distributions over amino acids at each position of a multiple sequence alignment of that protein with other, similar proteins. Profile comparison is useful in the study of proteins because the additional information available in profiles allows the detection of more distant protein relationships than comparison of individual sequences. Proteins that fold into similar shapes, and thus have similar functions, are almost as dissimilar as unrelated sequences. However, aligning profiles is much more computationally expensive than aligning pairs of sequences.

We show that pairwise alignments between IB sequences are similar to those between the profiles which they represent, but can be computed in a fraction of the time. We also show that IB alignment scores can, like profile alignment scores, be used to find distant homologs in a protein database. Therefore, by encoding profiles using an IB discretization before aligning them, we achieve much of the sensitivity of profile alignment at the computational cost of simple sequence alignment.

The IB discretization also creates an informative, compact textual representation of protein multiple alignments. Finally, we discuss the implementation of a fast, indexed IB sequence database. The approach is based on the classic BLAST algorithm [AGM$^+$90], but the different statistical properties of IB alignment require a number of significant changes.

Chapter 3 describes sscc, a flat clustering algorithm for discrete sequence data with a known number of clusters, based again on Information Bottleneck. In addition to minimizing within-cluster entropy, sscc constrains the clusters' distributions' shapes by minimizing the multi-information among sequence positions within each cluster. The minimization is performed by sequentially moving sequences between clusters to improve a global score.

We then extend this algorithm with a top-down structure learning heuristic to automatically infer the number of clusters. The algorithm recursively looks within each cluster for pairs of positions with significant mutual information. When it finds such a pair, it splits the cluster to reduce mutual information between them. This split is effectively making a heuristic improvement to the objective function's multi-information term.

We apply this algorithm to the problem of *population substructure*, that is, recovering subpopulations of related DNA sequences from a mixture of several such subpopulations. Identifying population substructure is a crucial step in disease association studies because, if it is present, it can mask the effect of interest. The correlation of specific point mutations to higher disease risk may be obscured by other, unrelated genetic differences between subpopulations. For example, wild mice are generally both smarter and thinner than pet mice, and these differences have some genetic basis. However, if we search for fat genes among a mixture of wild and pet mice, we will identify both the fat and the smart genes, since the two traits are correlated with the two subpopulations. However, if we first separate the two subpopulations of mice, we can use the remaining variation within each subpopulation to identify only the fat genes.

Identifying population substructure is especially difficult in humans, where the genetic variation within subpopulations is greater than that between them. However, our algorithm's shape constraints make it possible to identify population substructure even when subpopulations are highly overlapping. On a variety of simulated data, our algorithm significantly outperforms state-of-the-art methods when the populations are highly similar. On human haplotype data, it matches the state-of-the-art. We also show that it can be used to find meaningful substructure in HIV sequence data.

Chapter 4 describes RATS, an algorithm related to SSCC with structure learning, but applicable to much larger datasets. The algorithm is based on a more aggressive structure learning heuristic guided by randomized statistical tests on clusters' class conditional distributions, combined with greedy local improvement. By performing fewer tests and making larger steps in search space, it reaches a solution much more quickly than SSCC. It optimizes an objective function similar to that of expectation maximization, with its test's statistical significance automatically limiting model complexity.

Mobile elements are short sequences of DNA that have copied and reinserted themselves into our genome millions of times, and now make up almost 45% of our DNA [BK04]. Because they are so prevalent and active, a more detailed picture of mobile element history can yield valuable genetic insights. For example, a comparison of the distribution of subfamily ages to known evolutionary history suggests that mobile element replication may coincide with periods of rapid speciation.

We apply RATS to several large collections of mobile elements, the largest consisting of over 2 million sequences, and demonstrate biologically meaningful results. A phylogeny constructed from all SINE elements in the ENCODE project database [TTB+03] of DNA samples from 64 species closely follows the known species phylogeny, and recovers several features of known SINE evolution. Clusterings of all human and chimpanzee L1 and *Alu* elements find 32 and 1484 new mobile element subfamilies, respectively, compared to RepBase [Jur00], a standard classification of subfamilies. We show that our finer clustering of *Alu* elements identifies recent insertions, and also provides a high-resolution picture of the rate of insertion over time suitable for comparison with other evolutionary events.

We conclude in Chapter 5 by reviewing the work presented here. We compare and contrast the information-theoretic and hypothesis testing clustering approaches, and describe our biological contributions. We also relate our methods to other problems in learning sparse correlation structures, noting similarities to current approaches to learning causal structure.

Chapter 2

Amino acid profile clustering using Information Bottleneck

In this chapter, we describe a method to map amino acid profiles onto a discrete alphabet that preserves most of their information. This method finds an informationally optimal discretization using an extension of the Information Bottleneck approach described in Section 2.2.

Sequence homologs are an important source of information about proteins. Amino acid profiles, which represent the position-specific substitution probabilities found in alignments of homologous proteins, are a richer encoding of biological sequences than the individual sequences themselves. The substitution probabilities show which amino acids can be replaced without changing the protein's structure, and suggest which of its parts are functional.

However, profiles have a number of disadvantages as a sequence representation. First, profile comparisons are an order of magnitude slower than sequence comparisons, making them impractical for large datasets. Second, because they are such a rich representation, profiles are difficult to visualize. Finally, because profiles are continuous rather than discrete, they cannot be indexed to create large sequence databases.

Using our IB discretization, we observe that an 80-character alphabet captures nearly 90% of the amino acid occurrence information found in profiles, compared to the consensus sequence's 78%. Distant homolog search with IB sequences is 88% as sensitive as with profiles compared to 61% with consensus sequences (AUC scores 0.73, 0.83, and 0.51, respectively), but like simple sequence comparison, is 30 times faster. Discrete IB encoding can therefore expand the range of sequence problems to which profile information can be applied to include batch queries over large databases like SwissProt, which were previously computationally infeasible.

2.1 Motivation

One of the most powerful techniques in protein analysis is the comparison of a target amino acid sequence with phylogenetically related or *homologous* proteins. Such comparisons can give insight into which portions of the protein are important by revealing the parts that were conserved through natural selection. While substitutions in non-functional regions may be harmless, those in functional regions may change the protein's structure, often making them lethal to the organism. For this reason, functional regions of a protein tend to be conserved between organisms while non-functional regions diverge through random substitution.

Many of the state-of-the-art protein analysis techniques incorporate homologous sequences by representing a set of homologous sequences as a probabilistic *profile*, a sequence of the marginal distributions of amino acids at each position in the sequence. For example, PSI-BLAST [AGM+90] uses profiles to refine database searches. The PHD algorithm [RS93] uses them purely for structure prediction. Yona and Levitt [YL02] use profiles to align distant homologs from the SCOP database [MBHC95]; the resulting alignments are similar to results from structural alignments, and tend to reflect both secondary and tertiary protein structure.

Although profiles provide a lot of information about the sequence, their use comes at a steep price. While efficient algorithms exist for aligning protein sequences and performing database queries (e.g. BLAST [AGM+90]), these algorithms operate on strings and are not applicable to profile alignment or profile database queries. Profile-based comparisons can be substantially more accurate than sequence-based ones, but are about 30 times slower, since substitution penalties must be calculated by computing distances between probability distributions rather than simply looked up in a table. This makes probabilistic profiles impractical for use with large bioinformatics databases like SwissProt, which contained 160 000 sequences comprised of 64 million amino acids in 2005 [BAW+05], and has since grown to over 400 000 sequences and almost 150 million amino acids [The08].

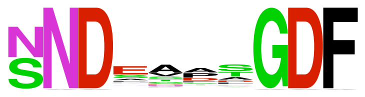
We propose a new discrete representation of proteins that incorporates information from homologs in a textual form we call *IB (Information Bottleneck) sequences*. Once a profile is represented using this discrete alphabet, alignment and database search can be performed using the efficient string algorithms developed for amino acid sequences, making profile information applicable to a greater range of problems. For example, the runtime for full pairwise Smith-Waterman [SW81] alignment between this sequence and all of SwissProt decreases from 250 hours to less than 8; a query for high-scoring alignments to 100 sequences of interest would take nearly three CPU-years with profiles, but just over a month with IB sequences. Either the resulting IB sequence alignments can be used directly, or a small set of high-scoring matches from this initial query can be realigned using profiles for greater precision. Therefore with IB sequences, profile information may be applied to a greater range of sequence problems with no loss in precision and minimal loss in recall.

IB sequences have another incidental benefit: By representing each class as a letter, discretized profiles can be presented in plain text, conveying more profile information than the original sequences in the same amount of space. These IB sequences are more accurate than consensus sequences and denser than profile matrices or sequence logos (see Figure 2.1). While sequence logos are likely a better representation for examining individual alignments, terse IB sequences are useful for presenting many alignments at once, such as when interpreting database query results. For example, Figure 2.1(c) shows that while logos more accurately reflect the first profile column, information about lower-conservation regions is completely lost at ordinary text size.

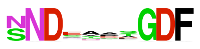
The main idea behind our approach is to compress profiles in a data-dependent manner by clustering the actual profiles and representing them by a small alphabet of distributions. Since this discretization removes some of the information carried by the full profiles, we cluster the distribution in a way that is directly targeted at minimizing the information loss. This is achieved using a variant of the Information Bottleneck (IB) method [TPB99], a distributional clustering approach for informationally optimal discretization. To preserve a clear textual representation, we want

A	0.0	0.0	0.0	0.09	0.34	0.23	0.12	0	0	0
C	0.0	0.0	0.0	0.04	0.01	0.01	0.03	0	0	0
D	0.0	0.0	1.0	0.01	0.05	0.14	0.09	0	1	0
E	0.0	0.0	0.0	0.38	0.04	0.00	0.04	0	0	0
F	0.0	0.0	0.0	0.06	0.00	0.08	0.04	0	0	1
G	0.0	0.0	0.0	0.00	0.06	0.01	0.03	1	0	0
H	0.0	0.0	0.0	0.02	0.00	0.04	0.00	0	0	0
I	0.0	0.0	0.0	0.00	0.00	0.03	0.00	0	0	0
K	0.0	0.0	0.0	0.04	0.01	0.01	0.00	0	0	0
L	0.0	0.0	0.0	0.01	0.01	0.00	0.09	0	0	0
M	0.0	0.0	0.0	0.00	0.00	0.03	0.00	0	0	0
N	0.5	1.0	0.0	0.05	0.05	0.01	0.01	0	0	0
P	0.0	0.0	0.0	0.02	0.00	0.23	0.00	0	0	0
Q	0.0	0.0	0.0	0.04	0.05	0.00	0.00	0	0	0
R	0.0	0.0	0.0	0.04	0.01	0.00	0.00	0	0	0
S	0.5	0.0	0.0	0.16	0.10	0.06	0.29	0	0	0
T	0.0	0.0	0.0	0.02	0.10	0.05	0.20	0	0	0
V	0.0	0.0	0.0	0.00	0.14	0.03	0.04	0	0	0
W	0.0	0.0	0.0	0.00	0.00	0.00	0.00	0	0	0
Y	0.0	0.0	0.0	0.01	0.00	0.04	0.04	0	0	0

(a)



(b)

P00790	---EAPT---
Consensus	NNDEAASGDF
IB	NNDeaptGDF
Logo	

(c)

Figure 2.1: Five representations of a part of an alignment of Pepsin A precursor P00790: (a) probabilistic profile; (b) sequence logo [CHCB04]; (c) four textual representations. The IB sequence is more compact than profiles or logos, but retains much of the conservation information lost by other textual formats. In the IB sequence, uppercase letter `X` represents strong conservation ($\sim$ 80%) of amino acid `X`, while lowercase `x` represents low conservation ($\sim$ 50%) of `X`.

this discretization to also reflect biologically meaningful categories by forming a superset of the standard 20-character amino acid alphabet. For example, we use “A” and “a” for strongly- and weakly-conserved Alanine. This formulation demands two types of constraints: similarities of the clusters’ conditional amino acid distributions to predefined values, and specific structural similarities between strongly- and weakly-conserved variants. We show below how the original IB formalism can be extended to naturally account for such constraints.

We apply our algorithm to SCOP [MBHC95], a database of proteins grouped hierarchically by structural similarity, and analyze the results in terms of both information loss and alignment quality. We show that IB discretization preserves much of the information in the original profiles using a small number of classes. We then show that like profile alignments, high-scoring IB alignments reflect distant homology, but that IB alignments can be computed 30 times faster than profile ones. IB discretization is therefore an attractive way to gain some of the additional sensitivity of profiles on tasks for which profile-profile comparison is not computationally feasible.

2.2 Information Bottleneck

Information Bottleneck [TPB99] (IB) is an information theoretic approach for distributional clustering. Given a joint distribution $p(X, Y)$ of two random variables X and Y, the goal is to obtain a compressed representation C of X, while preserving the information about Y. The two goals of compression and information preservation are quantified by the same measure of mutual information,

$$\begin{aligned} I(X;Y) &\stackrel{\text{def}}{=} \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

where $H(X)$ is the entropy of $p(X)$, and $H(X, Y)$ of $p(X, Y)$. $I(X; Y)$ is symmetric and non-negative, and is zero only when X and Y are conditionally independent. The IB problem is defined as the constrained optimization problem

$$\min_{p(c|x):I(C;Y)>K} I(C;X) \tag{2.1}$$

where K is a constraint on the level of information preserved about Y. The solution must also obey the following constraints

$$\begin{aligned} p(y|c) &= \sum_x p(y|x)p(x|c) \\ p(y) &= \sum_x p(y|x)p(x) \end{aligned}$$

This constrained optimization problem can be reformulated using Lagrange multipliers, and turned into a tradeoff optimization function with Lagrange multiplier β:

$$\min_{p(c|x)} \mathcal{L} \overset{\text{def}}{=} I(C;X) - \beta I(C;Y) \tag{2.2}$$

As an unsupervised learning technique, IB aims to characterize the set of solutions for the complete spectrum of constraint values K. This set of solutions is identical to the set of solutions of the tradeoff optimization problem obtained for the spectrum of β values.

When X is discrete, its natural compression is fuzzy clustering. In this case, the problem is not convex and cannot be guaranteed to contain a single global minimum. Fortunately, its solutions can be characterized analytically by the following set of self consistent equations:

$$\begin{aligned} p(c|x) &= \frac{p(c)}{Z(x,\beta)} \exp\left(-\beta D_{KL}[p(y|x)||p(y|c)]\right) && (2.3) \\ p(y|c) &= \sum_x p(y|x)p(x|c) && (2.4) \\ p(c) &= \sum_x p(c|x)p(x) \end{aligned}$$

where

$$Z(x,\beta) = \sum_c p(c) \exp\left(-\beta D_{KL}[p(y|x)||p(y|c)]\right)$$

By first computing $p(c|x)$ using Eq. (2.3), then recomputing the other distributions via Eqs. (2.4), these equations yield an iterative algorithm that is guaranteed to converge to a local minimum [TPB99]. While the optimal solutions of the IB functional are in general soft clusters, hard clusters are often more easily interpreted in practice. A series of algorithms was developed for hard IB clustering, including an algorithm that can be viewed as a one-step look-ahead sequential version of k-means [Slo02].

```
Randomly initialize $p(c \mid x)$
Find the corresponding $p(c),\ p(y \mid c)$ through Eqs. (2.4)
repeat
    $p_{i+1}(c|x) \leftarrow \frac{p_i(c)}{Z_{i+1}(x,\beta)} \exp\left(-\beta D_{KL}[p_i(y|x)||p_i(y|c)]\right)\ ,\ \forall\, c \in C,\ \forall\, x \in X$

    if hard clustering,

        $p_{i+1}(c|x) = \begin{cases} 1 & \text{if } c = \text{argmax}_c\, p(c|x) \\ 0 & \text{otherwise} \end{cases}$
    endif
    $p_{i+1}(c) \leftarrow \sum_x p(x) p_{i+1}(c|x)\ ,\ \forall\, t \in C$

    $p_{i+1}(y|c) = \frac{1}{p_{i+1}(c)} \sum_x p_{i+1}(c|x) p(x,y)\ ,\ \forall\, c \in C,\ \forall\, y \in Y$
until (stopping criterion)
```

Figure 2.2: Pseudocode for the iterative IB algorithm.

Section 2.2.1 describes and compares the performance of the iterative and sequential IB algorithms.

Friedman *et al.* [FMST01] describe *multivariate information bottleneck* (mIB), an extension of information bottleneck to joint distributions over several correlated input and cluster variables. Instead of a single observed variable X and a single cluster variable C, mIB incorporates sets of observed variables $\mathcal{X}$ and compression variables $\mathcal{C}$ with a specific conditional dependency structure. Intuitively, mIB's goal is to find distributions $p(\mathcal{C}|X)$ and $p(Y|\mathcal{C})$ such that $\sum_{\mathbf{c}\in\mathcal{C}} p(Y|\mathbf{c})p(\mathbf{c}|X)$ approximates $p(Y|\mathcal{X})$. For further details, including a formulation of the problem for arbitrary compression structures and a derivation of an analogous loss function, see Friedman *et al.* [FMST01].

```
$C \leftarrow$ random partition of $X$ into $K$ clusters
while not $done$
    $done \leftarrow TRUE$
    for every $x \in X$ :
        Remove $x$ from current cluster $c(x)$
        $c'(x) \leftarrow \text{argmin}_{c \in C} \Delta L(\{x\}, c)$
        if $c'(x) \neq c(x)$
            $done \leftarrow FALSE$ .
        Merge $x$ into $c'(x)$
    end for
end while
```

Figure 2.3: Pseudocode for the sequential IB algorithm.

2.2.1 Iterative vs. sequential IB

Slonim [Slo02] compares the performance and runtime of several IB algorithms. The first, iterative IB (iIB) (Figure 2.2), alternately updates the cluster assignment $p(c|x)$ and the resulting cluster distributions $p(y|c)$ and weights $p(c)$ via Eqs. (2.3) and (2.4). If hard clusters are desired, hard assignments are made in the first step. Since this algorithm only guarantees convergence to a local extremum, we repeated our experiments with five random initializations. In the current implementation, iteration was stopped when the current and previous distributions were sufficiently close together, as measured by $\sum_{y,s,c} |p_{t+1}(y|s,c) - p_t(y|s,c)|$.

The second IB algorithm, sequential IB (sIB) (Figure 2.3), first assigns elements to a fixed number of clusters, then individually moves them from cluster to cluster while calculating a 1-step lookahead score, until the score converges. Like iIB, sIB only guarantees convergence to a local extremum, and was therefore initialized with the results of five separate iIB runs.

Slonim [Slo02] found that hard-clustering sIB outperformed soft-clustering iIB on a document clustering task with 5,000 to 500,000 documents, finding fewer and better solutions on 100 random restarts. However, while sIB is more efficient than exhaustive bottom-up clustering methods like agglomerative clustering, sIB is still more expensive than iIB, since each reassignment of an instance requires recomputing the class conditional distributions. Therefore we used iIB with hard clustering, which only recomputes the conditional distributions after performing all updates. This reduces the convergence time by over an order of magnitude, from several hours to around ten minutes on a modest machine.

Slonim argued that sIB outperforms soft iIB in part because sIB's discrete steps allow it to escape local optima. We expect hard iIB to have similar behavior. Using the setup described in Section 2.3 with 40 clusters, we tested this by applying three complete sIB iterations to clusters obtained by multivariate iIB. sIB decreased the loss $\mathcal{L}$ by only about 3 percent (from 0.380 to 0.368), with most of this gain occurring in the first iteration. Up to exchanging labels, the 20 tightest clusters (the strongly-conserved categories) were nearly unchanged, while about half of the remaining clusters (the weakly-conserved categories) changed only slightly. This suggests that hard iIB and sIB find similar regularities in our data.

2.3 Method

Applying the Information Bottleneck approach to our profile discretization problem, X ranges over the set of single-position probabilistic profiles obtained from a set of aligned sequences and Y ranges over the set of 20 amino acids. In other words, $p(Y = y|X = x)$ is the probability of observing amino acid y at profile position x. Our goal is to find a set of profile categories C such that every position x's observed amino acid distribution $p(Y|X = x)$ is well-approximated by some $p(Y|C = c)$ where $p(C = c|X = x) = 1$.

The application studied in this chapter differs from standard IB in that we are interested in obtaining a representation that is both efficient and biologically meaningful. This leads us to place priors on clusters' conditional distributions (Section 2.3.1), and to enforce relations between certain clusters' distributions (Section 2.3.2).

2.3.1 Constraints on cluster conditional distributions

First, some clusters' identities are naturally determined by limiting them to correspond to the common 20-letter alphabet used to describe amino acids. From the point of view of distributions over amino acids, each of these symbols is used today as the delta function distribution which is fully concentrated on a single amino acid. For the goal of finding an efficient representation, we require the conditional distributions $p(Y|C=c)$ to be close to these delta distributions. More generally, we require $p(Y|C=c)$ for a specific cluster c to be close to predefined value $p(Y|C=\hat{c})$, thus adding constraints to the IB target function of the form

$$D_{KL}[p(y|\hat{c})||p(y|c)] < K(c)$$

for each such constraint. While solving the constrained optimization problem is difficult, the corresponding tradeoff optimization problem can be made very similar to standard IB. With the additional constraints, the IB trade-off optimization problem becomes

$$\min_{p(c|x)} \mathcal{L}' \equiv I(C;X) - \beta I(C;Y) + \beta \sum_{c \in C} \beta(c) D_{KL}[p(y|\hat{c})||p(y|c)] \quad . \tag{2.5}$$

We now use the following identity

$$\begin{aligned} & \sum_{x,c} p(x,c) D_{KL}[p(y|x)||p(y|c)] \\ = \; & \sum_x p(x) \sum_y p(y|x) \log p(y|x) - \sum_c p(c) \sum_y \log p(y|c) \sum_x p(y|x) p(x|c) \\ = \; & -H(Y|X) + H(Y|C) = I(X;Y) - I(Y;C) \end{aligned}$$

to rewrite the IB functional of Eq. (2.2) as

$$\mathcal{L} = I(C;X) + \beta \sum_{c \in C} \sum_{x \in X} p(x,c) D_{KL}[p(y|x)||p(y|c)] - \beta I(X;Y)$$

When $\sum_{c \in C} \beta(c) \leq 1$, we can similarly rewrite Eq. (2.5) as

$$\begin{aligned} \mathcal{L}' \; = \; & I(C;X) + \beta \sum_{x \in X} p(x) \sum_{c \in C} p(c|x) D_{KL}[p(y|x)||p(y|c)] \qquad (2.6) \\ & + \beta \sum_{c \in C} \beta(c) D_{KL}[p(y|\hat{c})||p(y|c)] - \beta I(X;Y) \\ = \; & I(C;X) + \beta \sum_{x' \in X'} p(x') \sum_{c \in C} p(c|x') D_{KL}[p(y|x')||p(y|c)] - \beta I(X;Y) \end{aligned}$$

The optimization problem therefore becomes equivalent to the original IB problem, but with a modified set of samples $x \in X'$, containing X plus additional pseudo-counts x' with prior probability $p(x') = \beta(c)$ (hence the requirement that $\sum_{c \in C} \beta(c) \leq 1$). This is similar to the inclusion of priors in Bayesian estimation.

Formulated this way, the biases can be easily incorporated in standard IB algorithms as additional pseudo-data. From an initial dataset defined by $p(Y|X)$ and $p(X)$ (typically $\frac{1}{|X|}$ for profiles) and biases $\hat{C} = \{\hat{c}\}$ with values $p_\beta(Y|\hat{C})$, we construct a new dataset $X' = X \cup \hat{C}$ defined by

$$p'(y|x') = \begin{cases} p(y|x') & \text{if } x' \in X \\ p_\beta(y|x') & \text{if } x' \in \hat{C} \end{cases}$$

and

$$p'(x') = \begin{cases} (1 - \sum_c \beta(c))\, p(x') & \text{if } x' \in X \\ \beta(x') & \text{if } x' \in \hat{C} \end{cases}$$

Finally, Eq. (2.5) is augmented to assign each pseudo-datapoint $\hat{c}$ to its cluster c, with $p(c|\hat{c}) = 1$, thereby fixing the biases to their clusters.

2.3.2 Constraining relations between cluster distributions

We want our discretization to capture both strongly- and weakly-conserved variants of the same symbol. While this can be done with standard IB using separate classes for the alternatives, the strong and weak variants' distributions are likely to be correlated. It is therefore preferable to define a single shared prior for both variants, and to learn a model capturing their correlation. Here we derive a multivariate IB model (see Section 2.2) to capture this dependency.

For profile discretization, we define two compression variables connected as in Friedman *et al.*'s [FMST01] "parallel IB": an amino acid class $C \in \{\mathtt{A}, \mathtt{C}, \ldots\}$ with an associated prior, and a conservation strength $S \in \{0, 1\}$. Our goal is to maximize the information about amino acid distribution Y contained in C and S together, while independently minimizing the information about position X contained in C and S. The IB loss function therefore becomes

$$\mathcal{L}_m \overset{\text{def}}{=} I(C;X) + I(S;X) - \beta I(Y;S,C) \tag{2.7}$$

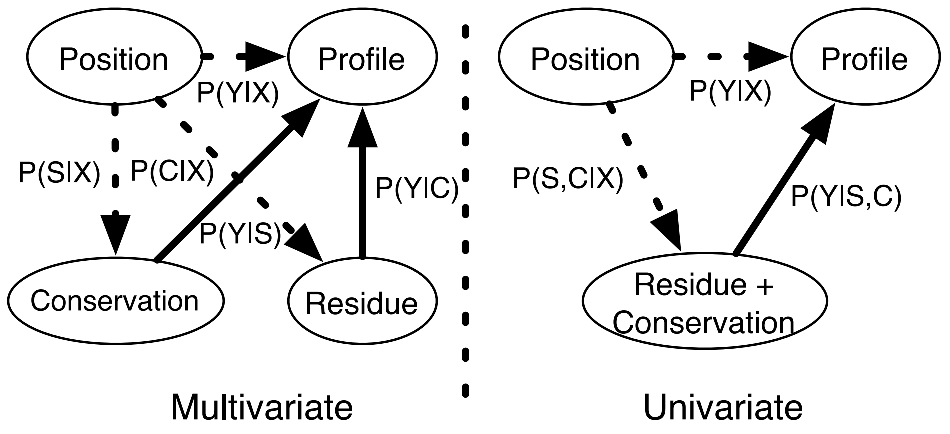


Figure 2.4: Graphical model representations of multivariate and univariate information bottleneck showing input (dashed) and output (solid) conditional dependencies.

Figure 2.4 illustrates our two models' dependency structures and parameterizations. Since the multivariate model correlates strong and weak variants of each category, it requires fewer priors than simple IB. It also has fewer parameters: a multivariate model with n_s strengths and n_c classes has as many categories as a univariate one with $n_{c'} = n_s n_c$ classes, but has only $n_s + n_c - 2$ free parameters for each x, instead of $n_s n_c - 1$.

2.4 Results

We evaluate IB alignment's ability to detect distant homologs by comparing the orders of profile, IB, and consensus alignment scores for a set of proteins with known evolutionary and structural relations. We also compare the pattern of gaps in individual profile alignments to those in the equivalent IB and consensus alignments. IB scores, like profile scores, capture a significant number of relations missed by consensus scores, and individual IB alignments more closely reflect the pattern of insertions in the original profile alignments.

Our data come from SCOP [MBHC95], a manually-constructed database of proteins grouped hierarchically by structural similarity and evolutionary relatedness. We expect proteins within the same SCOP family, which have clear evolutionary relationships and $\sim 30\%$ sequence identity, to have high-scoring profile and sequence alignments. We also expect proteins from different families in the same superfamily, which have probable evolutionary relationships but low sequence identity, to have significant but lower-scoring profile alignments but no significant sequence alignments.

For each protein, we first generate a profile from a CLUSTALW [THG94] multiple alignment with other proteins in its family, yielding 425,150 individual profile positions, then compute probabilistic profiles ignoring gap characters. We then compute IB classes from these profiles using iIB and the priors described below. Finally, we discretize the profiles into the resulting classes, using the Jensen-Shannon (JS) distance with mixing coefficient 0.5 rather than the KL distance optimized in encoding profiles to be consistent with Yona and Levitt [YL02].

In the following sections, we first examine how the amount of information from the original profiles encoded by IB categories varies with the number of clusters. We then consider how model structure, i.e. priors and relations between clusters, affects this information. Next, we compare individual IB, consensus, and profile alignments, and compare the order of alignment scores between distantly-related and unrelated proteins. Finally, we show how running time for profile and IB alignment varies with sequence length.

2.4.1 Information loss from discretization

One measure of the quality of IB clusters is the amount of information about Y (the amino acid distribution) lost through discretization, $I(Y;X) - I(Y;C)$. This represents the total information distance between profiles and the centers of their assigned clusters, and is a task-independent measure of the quality of a discretization. The change in $I(Y;X) - I(Y;C)$ between successive values of $|C|$ represents the amount of information gained by adding more categories, and thus the number of actual clusters of a particular scale in the data. Figure 2.1 shows the cluster information $I(Y;C)$ and position information $I(C;X)$ for consensus sequences, profiles, and (iterative) IB with no priors for $|C| = 40, \ldots, 500$. With $|C| \geq 80$ the IB alphabet captures over 90% of the available information.

Figure 2.5 shows the sequence logos for discretizations with $|C| = 20, 40, 80$ illustrating compression's effects. First, when the number of labels equals the number

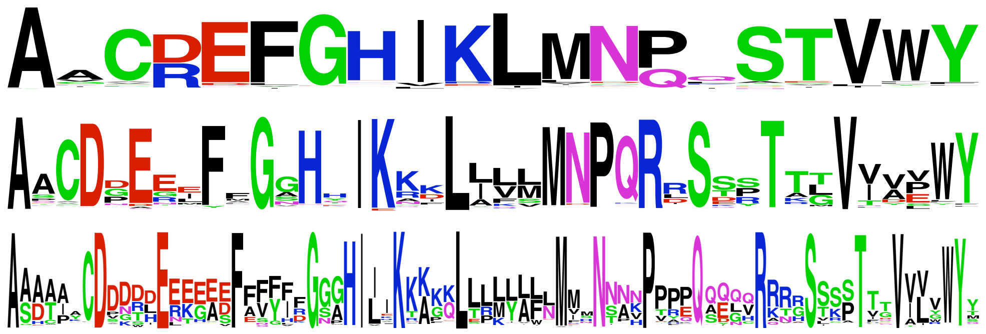


Figure 2.5: Sequence logos for $|C| = 20, 40, 80$, showing several features of IB discretization. First, variable numbers of clusters are assigned to different amino acids according to their overall frequencies: `A` and `L` are more common, while `C` is least common. Second, clusters capture strongly- and weakly-conserved variants, as well as some chemical similarities: `I`, `V`, `L`, and `M` are all hydrophobic.

Table 2.1: Information versus sequence type for consensus sequence, profiles, and IB without priors.

Seq. type	$I(Y;C)$	$I(C;X)$
Consensus	2.8503	
$\|C\| = 40$	3.0596	5.0793
$\|C\| = 80$	3.2083	5.7160
$\|C\| = 160$	3.3248	6.2442
$\|C\| = 320$	3.3986	6.6517
$\|C\| = 500$	3.4267	6.8297
Profiles	3.643	

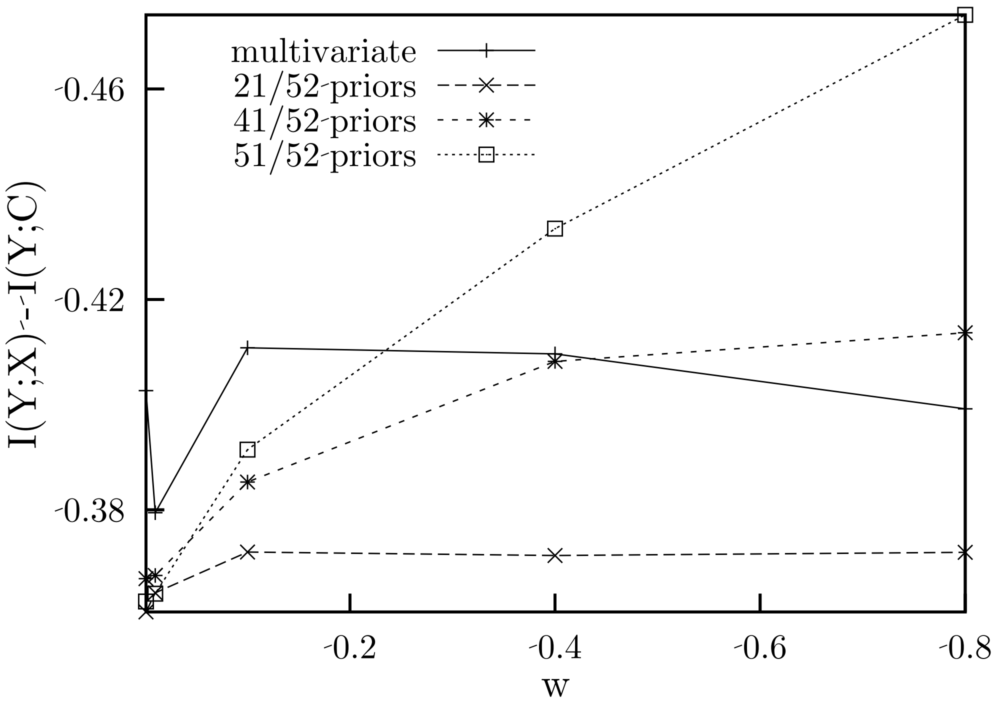


Figure 2.6: $I(Y;X) - I(Y;C)$ as a function of w for different groups of priors. The information loss for 52 categories without priors is 0.359, for 10, 0.474.

of amino acids ($|C| = 20$), the frequently-occurring amino acid `A` is allocated two labels, forcing `D` and `R` share a label. Second, as the number of labels increases, the least common amino acid `C` is allocated only a single label, while the number of labels assigned to the more common `A` and `L` consistently increases. This shows the data-dependence of our discretization compared to the simpler approach of allocating one or more clusters to each amino acid with varying levels of sequence conservation.

2.4.2 Effect of category constraints

For univariate IB, we use four types of priors reflecting biases on stability, physical properties, and observed substitution frequencies: (1) *Strongly conserved* classes, in which a single symbol is seen with $S\%$ probability. These are the only priors used for multivariate IB. (2) *Weakly conserved* classes, in which a single symbol occurs with $W\%$ probability; $(S - W)\%$ of the remaining probability mass is distributed among symbols with non-negative log-odds of substitution. (3) *Physical trait* classes, in which all symbols with the same hydrophobicity, charge, polarity, or aromaticity occur uniformly $S\%$ of the time. (4) A *uniform* class, in which all symbols occur with their background probabilities.

The choice of S and W depends upon both the data and one's prior notions of "strong" and "weak" conservation. Unbiased IB on a large subset of SCOP with several different numbers of unbiased categories yielded a mean frequency approaching 0.7 for the most common symbol in the 20 most sharply-distributed classes (0.59 ± 0.13 for $|C| = 52$; 0.66 ± 0.12 for $|C| = 80$; 0.70 ± 0.09 for $|C| = 100$). Similarly, the next 20 classes have a mean most-likely-symbol frequency around 0.4. These numbers can be seen as lower bounds on S and W. We therefore chose $S = 0.8$ and $W = 0.5$, reflecting a bias toward stronger definitions of conservation than those inferred from the data.

Figure 2.6 shows the effect on information loss of varying the prior weight $\sum_c \beta(c)$ with three sets of priors: 20 strongly conserved symbols and one background; these plus 20 weakly conserved symbols; and these plus 10 categories for physical characteristics. As expected, increasing the number or weight of priors increases information loss. However, with a small additional pool of unbiased categories information loss is nearly independent of prior strength. This suggests that our priors correspond to actual regularities in the data. Finally, note that despite having fewer free parameters than the univariate models, the mIB model achieves comparable performance, suggesting that our decomposition into conserved class and degree of conservation is reasonable.

2.4.3 Alignment similarity and distant homolog search

Since we are ultimately using the resulting IB classes in alignments, the true cost of discretization is best measured by the amount of change between profile and IB alignments, and the significance of this change. The latter is important because the best alignment can be very sensitive to small changes in the sequences or scoring matrix; if two radically different alignments have similar scores, neither is clearly "correct".

We can represent an alignment as a pair of index-insertion sequences, one for each profile sequence to be aligned (e.g. "1,2,_,_,3,..." versus "1,_,2,_,3,..."). The edit distance between these sequences for two alignments then measures how much they differ. However, even when this distance is large, the difference between two alignments may not be significant if both choices' scores are nearly the same. That is, if the optimal profile alignment's score is only slightly lower than the optimal IB class alignment's score *as computed with the original profiles*, either might be correct. We therefore report both the edit distance between alignments and this change in profile alignment score.

The score for aligning two IB symbols c_1 and c_2 is

$$\frac{1}{2}\left(1 - D_{JS}[p(y|c_1)||p(y|c_2)]\right)\left(1 + D_{JS}[q(y)||\bar{p}(y)]\right) - k_s \tag{2.8}$$

where $q(y) = \frac{1}{2}(p(y|c_1) + p(y|c_2))$, $\bar{p}(y)$ is the average probability of amino acid y across all profiles, and k_s is a constant chosen so that the average alignment score between pairs of randomly-chosen symbols is negative. We use $k_s = 0.45$ and gap open and extension penalties of $k_o = 2$ and $k_e = 0.2$, where k_s, k_o, and k_e have been chosen by Yona and Levitt [YL02] so that local alignment scores between random sequences follow the expected extreme value distribution.

Table 2.2: Alignment differences for alignments with IB models and sequence alignment, within and between superfamilies.

	Edit distance	Score change
	Same Superfamily	
mIB	0.154 ± 0.182	0.086 ± 0.166
IB	0.170 ± 0.189	0.107 ± 0.198
BLOSUM	0.390 ± 0.065	
	Same Clan	
mIB	0.124 ± 0.209	0.019 ± 0.029
IB	0.147 ± 0.232	0.022 ± 0.037
BLOSUM	0.360 ± 0.062	

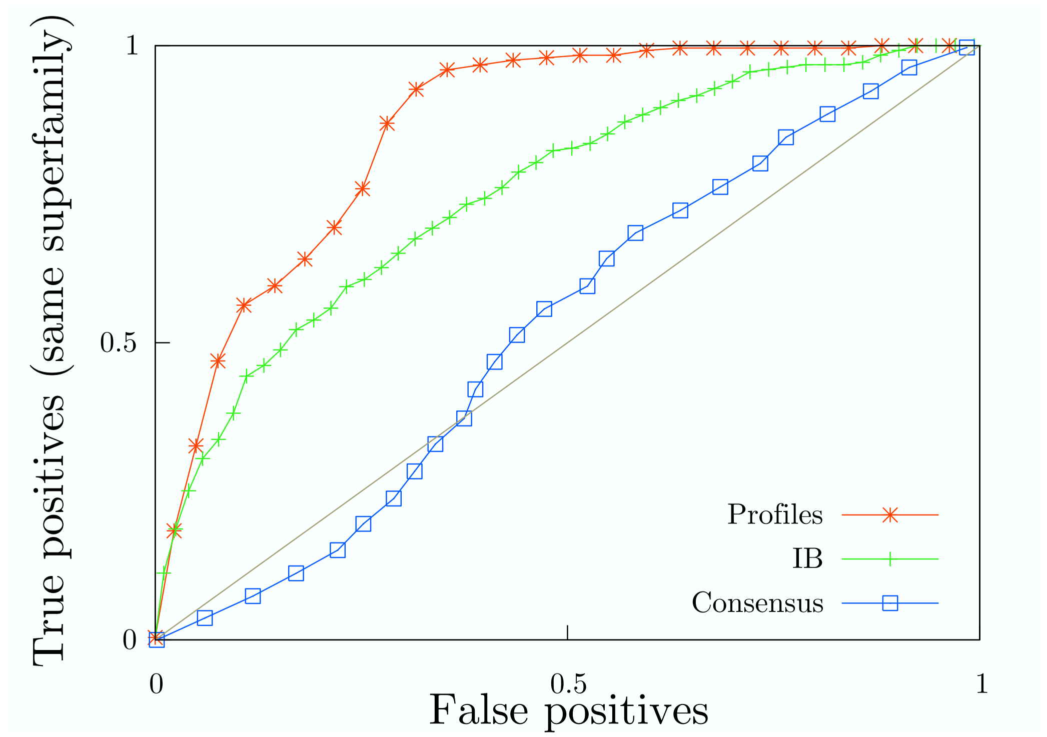


Figure 2.7: ROC curve for same vs. different superfamily classification by alignment score. 52 IB categories are used throughout.

Figure 2.2 shows both the edit distance and score change per length between profile alignments and those using IB classes, mIB classes, and the original sequences with the BLOSUM62 scoring matrix. Unless otherwise noted, IB alignments use $|C| = 52$ clusters, a number chosen to be conveniently represented by the 26 upper- and lower-case letters. To compare the profile and sequence alignments, profiles corresponding to gaps in the original sequences are replaced by gaps, and resulting pairs of aligned gaps in the profile-profile alignment are removed. We consider both sequences from the same family and those from other families in the same clan, the former being more similar than the latter, and therefore having better alignments. Assuming the profile-profile alignment is closest to the "true" alignment, IB alignment significantly outperforms sequence alignment in both cases, with mIB showing a slight additional improvement.

Since alignment scores predict structural relatedness, sequences with distant structural relationships, defined as those in the same SCOP superfamily, should have positive-scoring alignments. Yona and Levitt [YL02] compare the ranking of high-scoring profile-profile alignments to that of PSI-BLAST e-values, and show that profiles consistently assign high scores to more distant homologs. We perform this same test to compare profile, IB, and consensus sequence alignment scores. Figure 2.7 shows the ROC curve for detecting superfamily relationships between 117 families contained in 10 randomly-chosen SCOP superfamilies with between 3 and 35 members. While IB fares worse than profiles, consensus sequences perform essentially at chance.

2.4.4 Alignment running time

Most of the cost of aligning two profile sequences comes from computing JS distances between pairs of profiles. Encoding unencoded profile sequences before alignment, by significantly reducing the number of JS distance computations, yields a 4- to 20-fold improvement in alignment running time. Furthermore, sequences can be pre-encoded to perform repeated comparisons, yielding a 30-fold improvement.

Encoding two sequences of lengths n and m for IB alignment requires computing the $|C|(n + m)$ JS distances between each profile and each category, a sig-

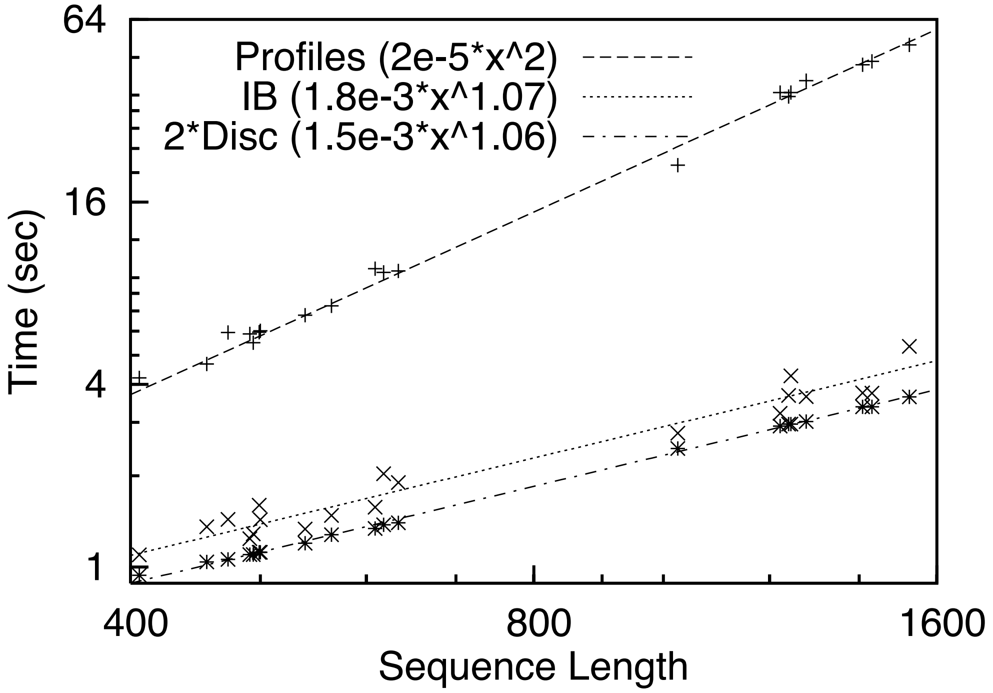


Figure 2.8: Running times for profile-profile and IB-profile alignment, and (twice) running time for IB discretization. Alignments were performed using the Smith-Waterman algorithm and computing the complete dynamic programming matrix. For IB, each sequence was first discretized using 50 categories. For profiles, distances were precomputed between every pair of sequence positions.

nificant improvement over the mn distance computations required for profile-profile alignment when $|C| \ll \frac{\min(m,n)}{2}$. Once the sequences are encoded or the pairwise distances computed, both methods take essentially the same amount of time to perform Smith-Waterman alignment. Figure A.3 compares the running time of profile and IB alignment for different sequence lengths, showing best fit curves for both to $f(x) = ax^b$. The results show that the number of JS distance computations dominates running time for typical sequence lengths: despite both methods' performing $O(n^2)$ work in Smith-Waterman alignment, IB alignment time is essentially linear in sequence length, while profile alignment is quadratic. Figure A.3 also plots the time taken to encode both input profiles in a 40-character IB alphabet, showing that encoding accounts for most of the cost of alignment. Since useful values of $|C|$ are much smaller than the average sequence length, and since most database applications can use pre-encoded sequences, the effect of $|C|$ on real running times is negligible. In particular, the time taken to align pre-encoded sequences is independent of $|C|$ for the values presented here.

On average hardware, each profile distance computation takes about $16.5\mu s$. At this rate, just the distance computations for a full pairwise alignment of SwissProt's 160,000 sequences, comprising 60 million residues, require 3×10^{10} CPU-seconds or about 950 years. The distance computations required to encode the database in a 40-character alphabet take 11 hours. Similarly, the distance computations for aligning a single 200-element sequence with every element of SwissProt take about 260 hours, or nearly the entire observed running time for performing full profile alignments. To compare, our unoptimized implementation of Smith-Waterman takes around 8 hours to align a typical sequence against SwissProt. This agrees well with the figure obtained by assuming an average sequence length of 365 residues and the observed single alignment times shown in figure A.3 (minus encoding time).

2.5 IB profile indexing

Even with IB encoding, full pairwise alignment of SwissProt would take an impractical 60 years using standard pairwise Smith-Waterman alignment. Although database indexing and search algorithms like BLAST [AGM+90] have been developed to speed searches over large sets of amino acid sequences, a database of profile sequences cannot be similarly indexed, because the profiles are continuous. IB discretization is a natural way to incorporate profile information in a discrete sequence database. Although directly applying the BLAST algorithm to IB sequences fails for reasons discussed in Section 2.5.2, a related approach can still be used.

In this section we survey the issues involved in creating an IB sequence database, and describe one possible approach. Like BLAST, our algorithm quickly finds short "seed" matches between the query and the database, then only performs full sequence alignment near these seeds. Unlike BLAST, however, our algorithm looks for both exact and approximate seed matches.

2.5.1 Index construction

To speed the search for high-scoring match positions, we create an index that maps each possible k-mer to all of its high-scoring matches in the database. (The index actually consists of two parts, a map from each k-mer s to every t satisfying the above condition, and a map from each t to its exact occurrences in the database.) Specifically, for score function S, the sum of the per-position scores defined by Eq. (2.8), we map a query k-mer q to all database k-mers d such that for randomly-chosen k-mers x_1 and x_2 and threshold ϵ,

$$P(S(x_1, x_2) > S(q, d)) \leq \epsilon$$

Note that k-mers with high-scoring matches to many others (i.e. those representing strongly-peaked distributions) will have many neighbors, while those with few such matches may have none. This is desirable since the regions of interest, functional domains, will have high conservation rates.

ϵ controls the tradeoff between sensitivity and both index size and search time. If ϵ is too low, then some matches' seeds will be missed. If it is too high, then the index becomes prohibitively large, and queries will require full alignments around a large number of low-quality seeds.

2.5.2 Sequence search

The BLAST [AGM$^+$90] program scans through a large sequence database by looking for short exact seed matches between the query sequence and the database, and only performing full alignments at these positions. However, this approach is not effective with IB sequences, since interestingly-related sequences may not share any exact seeds with the query for two reasons: First, two different categories may be similar enough to have a high match score, but seeds containing them will not be identical. Second, since the point of profile-profile alignment is to search for distantly-related sequences, the sequences of interest are likely to share relatively few exactly-matching seeds, but may still contain high-scoring non-identical ones.

Instead of finding all exact seed matches to the query sequence, we look for high-scoring approximate matches, then score each database sequence according to its approximate seed matches to the query. We then perform full alignment for the highest-scoring database sequences. For global alignment, a sequence's score is the sum the scores of seeds positioned such that they could appear in an alignment with fewer than w total insertions or deletions. For local alignment, the score is simply the sum of all seeds' scores.

Since this search algorithm operates on already-encoded IB sequences, it naturally handles both profiles and IB sequences already in the database. However, we may also want to query the IB database for matches to an ordinary amino acid sequence. While we could do this by aligning the input sequence against a sequence database, encoding the resulting profile, and finally performing a standard search, both of the encoding steps by themselves would take more time than the actual search. The simplest alternative is to create an approximate IB sequence from the input amino acid sequence by choosing, for each amino acid y, the IB category c most likely to have generated that amino acid, i.e. $\operatorname{argmax}_c P(y|c)$.

2.6 Conclusion

We have described IB sequences, a discrete encoding of amino acid profiles that allows profile information to be used for alignment and search at essentially the same computational cost as simple sequence alignment. The encoding is based on minimizing information loss, and its classes can be constrained to correspond to the standard amino acid representation, thus yielding an intuitive, compact textual form for profile information. Alignments of IB sequences encoded with a modest number of classes correspond significantly better to the original profile alignments than do alignments of the consensus sequences (edit distance 0.15 versus 0.39). High-scoring IB alignments reflect distant homology detected with profiles but not with consensus sequences (AUC score 0.73 versus 0.51).

Our model is potentially advantageous in three ways: First, it models rich conditional distribution structures and class constraints. It can, for example, be extended to incorporate structural information in the input representation, and to assigning structural significance to the resulting categories. Second, it allows us to apply existing fast discrete algorithms to continuous profile sequences when either profile comparison is computationally impractical, or only discrete-sequence algorithms exist. While the application is nearly automatic for simple applications like text search, more complex algorithms may not directly transfer as we see in Section 2.5. However, even problems that require new discrete algorithms are likely to see improved performance over the brute-force continuous alternatives.

Third, our discretization avoids undersampling problems while being more broadly applicable than single-position profiles. Ordinary profile applications are limited by high dimensionality to considering only single positions of a multiple alignment. This ignores significant correlation both between adjacent profile positions and between adjacent symbols in individual aligned homologs. Single-position profiles thus represent a drastic simplification of the underlying data. For example, while the average entropy of a single profile in our dataset is 0.99, the average entropy of an adjacent pair is only 1.23, suggesting an information loss far greater than the 10% lost by IB discretization. Therefore profile pairs can be represented more compactly than the cross-product of the single-position alphabet. Instead of considering sequences of adjacent single-position profiles, our method can be extended to discretize distributions over pairs or k-tuples of symbols in a multiple alignment. By applying IB to the 20^k-dimensional space of k-tuple profiles, we can avoid undersampling and obtain a richer sequence representation incorporating previously-ignored local correlation. Extending this approach to variable-length substrings yields an algorithm similar to suffix trees, known to be some of today's most efficient text compression methods [RST94].

This chapter is based on work with Gal Chechik, Robin Friedman, and Eleazar Eskin, and published in *BMC Bioinformatics*, 7(Suppl 1):S8, March 2006.

Chapter 3

Separation of overlapping subpopulations using mutual information

In the previous chapter, we described an extension of the Information Bottleneck method to incorporate priors on cluster distributions. We then used this algorithm to cluster sets of probability distributions to produce an informationally-optimal, data-dependent discretization. In this chapter, we present an algorithm for identifying substructure in overlapping subpopulations. The algorithm is based on another extension of Information Bottleneck, this time to limit the multi-information within individual clusters while clustering sets of character strings.

Identifying subpopulations and their ancestral sequences is an important first step in understanding population history and dynamics. However, several interesting cases including HIV evolution and human genetic variation feature highly overlapping subpopulations, whose ancestral subpopulations differ by fewer than the average number of mutations between members of the same population. In such cases both flat and hierarchical distance-based clustering methods optimize the wrong objective function, and will therefore fail to recover the correct population structure.

We apply our algorithm to human DNA and viral RNA to divide individuals into subpopulations, demonstrating its effectiveness in recovering subpopulations. While motivated by a particular biological application, the algorithm can be applied to both discrete and continuous clustering problems for which distance-based measures are inappropriate.

3.1 Motivation

Identifying distinct strains of HIV is an important first step in understanding the virus's spread because pathways of transmission correspond to distinct viral strains. Each such strain or population cluster can be characterized by a common *ancestral sequence* and a rate of mutation from this sequence. While there exist a number of clustering algorithms for finding population structure, most have been developed for non-overlapping subpopulations, in which the distances between ancestral sequences are greater than those between subpopulation members. However, because of HIV's high mutation rate, distances between HIV ancestral sequences are similar to those within subpopulations, drastically limiting these methods' effectiveness.

Many biological processes can be characterized as forms of sequence evolution, in which a set of individual sequences is generated by a process of alternating replication and random mutation. When different sequences replicate at different rates, such data can be modeled as a number of ancestral sequences, each generating offspring by mutation. One well-studied problem, *population substructure*, aims to recover the underlying ancestral sequences from the entire population and to cluster individuals by ancestral sequence. Another problem, *phylogenetic inference*, aims to find the hierarchical relations between the ancestral sequences, summarizing the series of events by which a diverse population has arisen from its common ancestor. Examples of these problems include determining human population substructure [HSN+05], tracing HIV evolution [KFF+02], and reconstructing *Alu* repeat insertion history [PEP04].

If the sequences are long relative to their mutation rate, the probability of the same mutation occurring independently in unrelated individuals is vanishingly small. Viewing subpopulations as clusters of points in sequence space, the mutation distance between points within a single cluster will be small relative to that between cluster means, and the substructure problem will be amenable to distance-based clustering. However, if the sequences are short relative to the mutation rate, inter- and intra-population mutation distances are on the same order, and the populations largely overlap. In this situation distance-based clustering algorithms (such as agglomerative linkage clustering) perform poorly, since an individual may often be closer to another population's ancestral sequence than to its own, and the resulting clusters will reflect local irregularities instead of the actual generating process [PEP04][1]. But if mutation at one position is independent of mutation at others, then observing correlation between two or more positions in a supposed subpopulation suggests that it actually consists of two or more subpopulations differing at those positions.

Here we propose a top-down clustering algorithm for reconstructing population substructure for evolutionary processes with similar mutation and replication rates. Our algorithm, sequential shape-constrained clustering (SSCC), finds clusters that maximize information about sequence identity while minimizing mutual information between positions within each cluster. SSCC extends standard top-down clustering with a penalty for multi-information between positions within a subpopulation. While we present results for discrete sequence clustering with a specific penalty function, SSCC is more general, and can be applied with other penalty functions to both discrete and continuous multivariate data.

In Section 3.3 we describe the formulation of the problem and the objective function. Section 3.4 presents a sequential iterative algorithm to minimize this function and discusses efficient calculations of the score function. In Section 3.5 we illustrate the objective function's behavior on artificial data designed to show how it differs from distance-based clustering. We then apply SSCC to recover population substructures from HIV RNA, human single nucleotide polymorphisms (SNPs), and *Alu* repeat elements, and compare our results to those obtained both by k-means and by state-of-the-art population substructure methods. In particular, we show that our method is competitive with, and in some cases more accurate than, the STRUCTURE program [PSD00], one of the most widely-used tools for this problem. Finally in Sec-

[1]Consider two populations A and B of bitstrings of length n, where As are instances of $\bar{A} = 0X1$, Bs of $\bar{B} = 1X0$; X represents a sequence of $n-2$ random bits. The distance D_{AA} between an A and $\bar{A}$ follows a binomial distribution $\mathrm{B}(n-2, 0.5)$, while the distance D_{AB} between it and $\bar{B}$ is $2 + \mathrm{B}(n-2, 0.5)$. So the probability $p(D_{AA} \geq D_{AB})$ of its being at least as close to $\bar{B}$ as to $\bar{A}$ is approximately $p\left(D_{AA} \geq E[D_{AB}]\right) = I_{0.5}(\frac{n}{2}+1, \frac{n}{2}-2)$, which for $n = 100$ equals 0.38. Since a substantial minority of the elements from each population are closer to the other's consensus sequence, a distance-based phylogeny will misleadingly relate elements from different populations, obscuring the two actual subpopulations. However, testing for correlated mutation can correctly identify $\bar{A}$ and $\bar{B}$.

tion 3.6 we suggest several extensions and briefly describe other problems for which the algorithm is well-suited.

3.2 Related work

Evolutionary biologists often combine the substructure and phylogeny problems, directly reconstructing phylogenies using distance-based algorithms such as neighbor-joining. However, Pritchard *et al.* [PSD00] argue that despite their visual appeal, distance-based methods lack the statistical power of model-based ones. Pritchard instead proposes a sophisticated Bayesian model for population substructure. The STRUCTURE program optimizes this model for a fixed number of subpopulations by Markov chain Monte Carlo (MCMC). Another model-based approach, SEMPHY [FMST01], uses structural expectation maximization to address the phylogenetic inference problem.

Our method is related to models of clustering with side information [XNJR03, SBHHW04,CT02]. Xing *et al.* [XNJR03] discuss how to learn a metric from the family of Mahalanobis distances, given pairs of samples which are known to be "similar," and show how this can be used for clustering. Shental *et al.* [SBHHW04] show how to use equivalence (and non-equivalence) constraints, that is pairs of samples known to be in the same cluster (or not), in order to learn the components of the within-cluster covariance matrix, so these can be normalized out when learning a Gaussian mixture model. The problems discussed here differ from those discussed in these two papers in several aspects: First, the biological process that created the clusters in our case leads to independence between positions (spherical clusters). Therefore, rather than learning a whitening transformation from the data, we enforce spherical clusters. Second, we operate in a discrete space over a given alphabet rater than a continuous space. Finally, rather than focusing on second order relations (covariance matrices), we force high-order independencies using the mutual information measure.

Clustering categorical variables while preserving information about another variable was addressed in the Information Bottleneck (IB) framework [TPB99]. There, information preservation $I(X;C)$ is traded for a compression term that determines the loss of information due to clustering. Adapting their notation to the one used here, the IB functional is

$$\min_{p(c|z)} I(Z;C) - \beta I(C;X)$$

where Z is a random variable that holds the identity of the sequence, rather than its value. In practice, hard clustering for a fixed number of clusters is sometimes used, and $1/\beta$ is taken to zero. Chechik *et al.* [CT02] discuss clustering while preserving information about one variable X^+ but removing information about a second variable X^-. In the limit of infinite β, this is formalized as finding

$$\min I(C;X^-) - \gamma I(C;X^+)$$

The method described here aims to cluster while preserving information about the sequences $I(C;X)$, but at the same time removing conditional information per cluster, that is, finding

$$\min I(X|C) - \gamma I(C;X)$$

3.3 Problem formulation

Let $X = (X_1, \ldots, X_l)$ be a vector random variable over the m-letter alphabet $\Sigma = \{a_1, \ldots, a_m\}$. Each instance x of X is a vector of length l, and we model the process that generates it as follows. First, one of n subpopulations $C \in \{1, \ldots, n\}$ is drawn with probability $p(C = c)$, and x is assigned the consensus sequence of the subpopulation $\mu_c = (\mu_{c1}, \ldots, \mu_{cl}) \in \Sigma^l$. Then the sequence is mutated independently at each position according to a subpopulation specific mutation rate $\sigma_c = (\sigma_{c1}, \ldots, \sigma_{cl})$, which determines the probability of mutating from the consensus value μ_{cj} to $a_k \neq \mu_{cj}$. Given subpopulation-specific consensus sequences and mutation rates (μ_c, σ_c), one can calculate the probability $p(x|c)$ that a given sequence x comes from a subpopulation c. The pair (μ_c, σ_c) can be regarded as the discrete analogue of the mean and (spherical) covariance matrix in a Gaussian mixture model. The goal of subpopulation identification is to divide the input sequence set into subpopulations such that the sequences in each subpopulation are similar. Instead of estimating the parameters μ and σ directly, we choose here to use a nonparametric model and measure within-subpopulation similarity by the homogeneity of the conditional entropy $H(X|C)$ [CT91], as compared with the overall homogeneity $H(X)$. This reduces to using the mutual information

$I(X;C) = H(X) - H(X|C)$ as a measure for the goodness of dividing sequences into subpopulations.

Since mutations occur independently, sequences x drawn from the same subpopulation c have

$$p(x_i|\{x\}_{j\neq i}, C = c) = p(x_i|C = c)$$

where $\{x\}_{j\neq i}$ is any set of positions excluding the position i. For example, focusing on pairs we have

$$p(x_i|x_j, C = c) = p(x_i|C = c)$$

for all $j \neq i$. As a result of this independence, the conditional mutual information $I(X_i; X_j|C)$ for every pair of positions X_i, X_j within a cluster should vanish. For subsets of higher order, the independence of all l variables can be quantified by an extension of the mutual information, the *multi-information* [SV98],

$$MI(X_1; \ldots; X_l) \overset{\text{def}}{=} D_{KL}[p(x_1, \ldots, x_l)||p(x_1)\ldots p(x_l)] \tag{3.1}$$

which is again non-negative and vanishes if and only if all X_i are independent.

We propose using this additional knowledge about conditional independence within subpopulations to devise a better clustering procedure for finding subpopulations. This additional constraint on position independence is most useful when many clusters are overlapping. For such data, distance-based clustering methods like k-means and Gaussian mixtures tend to find wrong clusters that are more concentrated in space, but have non-independent positions. This happens because when the distances between consensus vectors μ_i, μ_j are small relative to the mutation rates σ_i, σ_j the probability of cluster c_i generating a sequence x such that $p(x|c_{j\neq i}) > p(x|c_i)$ becomes significant. We show here that additional constraints on independence between dimensions can eliminate this problem.

The goal of subpopulation identification is therefore to find subpopulations of the sequences, such that sequences in each subpopulation are similar, but positions are independent. This can be formally quantified by maximizing the mutual information $I(X;C)$ while removing the position dependence $MI(X_1; \ldots; X_l|C)$,

$$\min_{\mu,\sigma} S(C) = \beta \;\; MI(X_1; \ldots; X_l|C) - I(X;C) \tag{3.2}$$

where β controls the trade-off between the two objective factors.

Since with limited data it is often difficult to estimate the high-dimensional joint distribution $p(X_1, \ldots, X_l)$ needed for estimating the multi-information, one can replace the above independence condition with a weaker, pairwise one

$$\min_{\mu,\sigma} S(C) = \beta \sum_{i<j} I(X_i; X_j|C) - I(X; C) \tag{3.3}$$

The two conditions become equivalent up to constants for Bethe-type distributions, where no high-order correlations above second order exist.

3.4 Iterative algorithm

We find local optima of Eq. (3.3) using a two-phase iterative algorithm combining sequential updates and top-down splitting. The first phase splits each existing cluster c if doing so improves $S(C)$. Since the number of possible splits is exponential in cluster size, we perform a greedy step, and choose to split the cluster into two families that differ at a single position i of the sequence. We choose the position which reduces the dependencies the most, namely, the position

$$i = \operatorname*{argmax}_i \sum_{j \neq i} I(X_j; X_i|C = c)$$

We then divide c into those sequences with the most-frequent value at position i, and those with all other values. In practice, we find that this criterion yields good splits at a small computation time cost. The second phase iterates through the clusters, sequentially updating each of their sequences' cluster assignments until no more improvement is possible.

Each step in the algorithm decreases the objective function, and Eq. (3.3) is bounded below by $-I(X; C)$, so the two-phase updates must converge. Furthermore, since the cluster distributions are determined by hard assignments of a finite number of instances, it must converge in a finite number of steps.

Since the algorithm performs a large number of sequential updates, computing the score of a category when considering reassigning an instance becomes a major limiting factor. We show here how the value of $S(C)$ can be updated efficiently after adding or removing a single instance. Consider the case where we add a single instance $x = (x_1, \ldots, x_l)$ to a category c with a distribution $p(X|c)$. This creates a new category c' with probability $p(c') = (p(c) + 1/|X|)$ and sequence distribution

$$p(X|c') = (1-\alpha)p(X|c) + \alpha[X = x]$$

where $\alpha = 1/(|c|+1)$ and $[X = x]$ is an indicator function. Separating the $X_i = x_i$ terms from the rest and summing over sequence positions i yields

$$I(X;c') = I(X;c) - I(x;c) + I(x;c') + p(c)\log\bar{\alpha}\sum_{i=1}^{l}(1 - p(x_i|c)) \tag{3.4}$$

where $\bar{\alpha} = 1 - \alpha$,

$$I(x;c) \overset{\text{def}}{=} \sum_i p(x_i, c)\log\left(\frac{p(x_i|c)}{p(x_i)}\right)$$

is the symbol-specific information that the category c provides about the specific sequence x, and

$$I(X;c) \overset{\text{def}}{=} \sum_x p(x, c)\log\left(\frac{p(x|c)}{p(x)}\right)$$

is the category-specific information that the category c provides about the sequences X.

A similar transformation can be applied to the second term in Eq. (3.3). Using

$$I(X_i;X_j|C=c) = H(X_i|C=c) + H(X_j|C=c) - H(X_i,X_j|C=c)$$

the above substitution yields

$$\begin{aligned} H(X_i|C=c') &= \bar{\alpha}H(X_i|C=c) - (1-p(x_i|c))\bar{\alpha}\log\bar{\alpha} \\ &\quad +\bar{\alpha}p(x_i|c)\log p(x_i|c) - p(x_i|c')\log p(x_i|c') \qquad (3.5) \\ H(X_i,X_j|C=c') &= \bar{\alpha}H(X_i,X_j|C=c) - (1-p(x_i,x_j|c))\bar{\alpha}\log\bar{\alpha} \\ &\quad +\bar{\alpha}p(x_i,x_j|c)\log p(x_i,x_j|c) - p(x_i,x_j|c')\log p(x_i,x_j|c') \qquad (3.6) \end{aligned}$$

where $p(x_i,x_j|c') = \bar{\alpha}p(x_i,x_j|c) + \alpha$. Combining these and summing over all (x_i, x_j), $i < j$ yields

$$\begin{aligned} \sum_{i<j} I(X_i;X_j|C=c') &= \bar{\alpha}\left(\sum_{i<j} I(X_i;X_j|C=c) - f_c(x)\right) - f_{c'}(x) \\ &\quad -\bar{\alpha}\log\bar{\alpha}\left((l-1)\sum_i p(x_i|c') - \sum_{i<j} p(x_i,x_j|c') - \frac{l(l-1)}{2}\right) \qquad (3.7) \end{aligned}$$

where

$$f_c(x) = (l-1)\sum_i p(x_i|c)\log p(x_i|c) - \sum_{i<j} p(x_i, x_j|c)\log p(x_i, x_j|c)$$

and $f_{c'}(x)$ is the analogous quantity for c' instead of c. Note that since both of these terms depend only upon the previous values of $I(X_i; X_j|C = c)$ and $I(X; c)$, and upon the probabilities of the instance s being considered, the score update can be computed in $O(l^2)$ time rather than $O(l^2(|X| + |\Sigma|^2))$.

3.5 Results

To illustrate SSCC's behavior, we generated three 50-character bit-strings with relative Hamming distances of 2, 3, and 5, then generated 20 copies of each with mutation rate 0.05, yielding a 60-member population with expected Hamming distance of 2.5 between a mutant and its ancestral sequence. When the stopping criterion is chosen to recover three clusters, SSCC recovers the ancestral sequences exactly, and assigns 93% of sequences to their generating clusters. On the other hand, iterative K-means, even when initialized with the correct parent sequences, typically converges to a degenerate solution in four or fewer iterations, demonstrating the inappropriateness of distance-based clustering for overlapping populations.

Note that because the clusters overlap, the most probable assignment of points to clusters may not be the one by which the data were generated. Because of this, and because we want to recover population substructure, recovery of ancestral sequences or cluster means is often a more appropriate performance measure than correct sequence labeling. By this measure both STRUCTURE and SSCC perform perfectly on this generated dataset. However, when the mutation rate is increased to 0.16, STRUCTURE finds a degenerate solution, while SSCC still recovers the correct ancestral sequences despite only correctly classifying 71% of individuals.

In the following sections we demonstrate the superior operation of SSCC on three real biological problems: HIV strain identification, human population substructure, and *Alu* repeat structure.

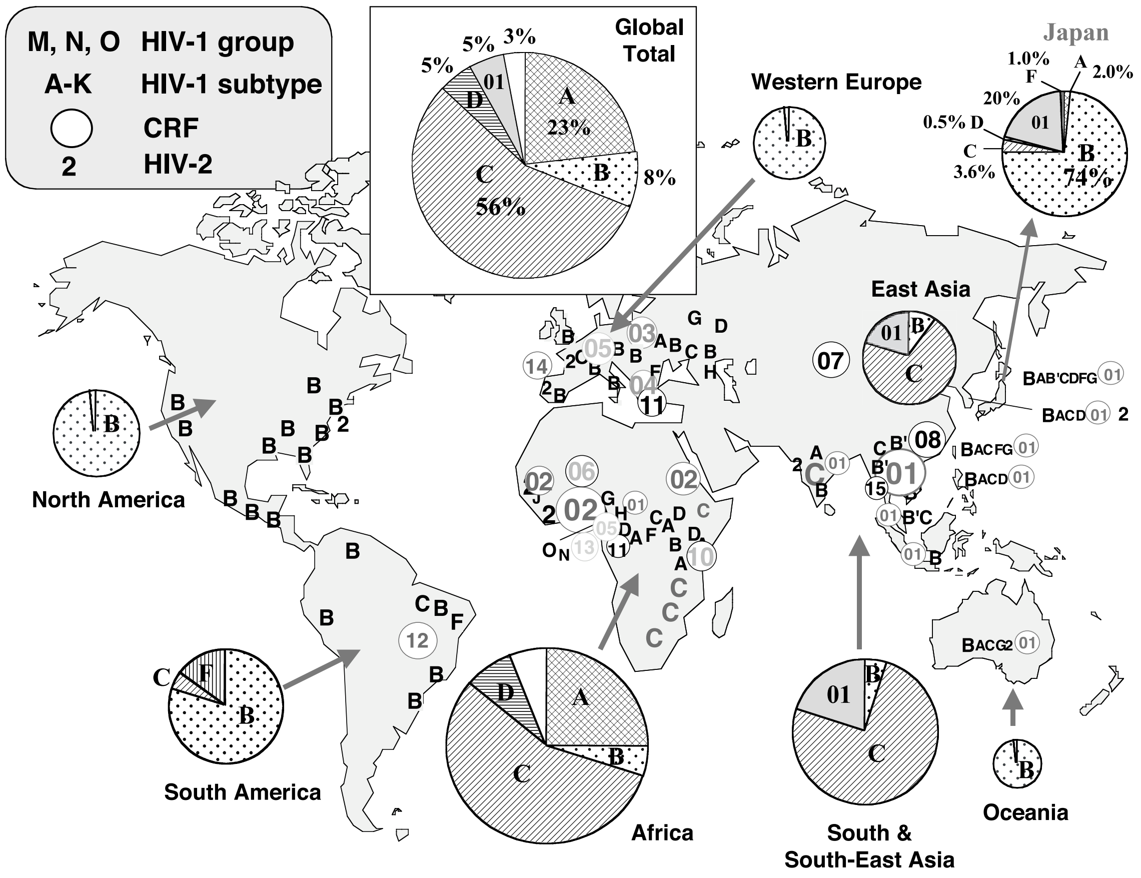


Figure 3.1: Worldwide distribution of HIV clades (from [TKM04]).

3.5.1 HIV-1 evolution

With the large number of HIV strains that have been already sequenced, HIV viral evolution presents a unique opportunity for studying population structure. Detecting this structure is extremely challenging for traditional phylogenetic methods because of a high recombination rate and a mutation rate up to six orders of magnitude greater than that of typical DNA.

The HIV-1 virus originated in East Africa, where most of its ten subtypes are found. Several subtypes have spread to other parts of Africa, while subtype B, a descendant of subtype D, is dominant in Europe and North America, but rare in Africa [SBR+99]. Figure 3.1 shows the worldwide prevalence of different HIV strains. Our dataset consists of 442 HIV-1 sequences from the Los Alamos HIV sequence database [KFF+02]. To avoid cross-strain recombination, which complicates popula-

Table 3.1: Regions and subtypes with frequency ≥ 0.1 in the predicted HIV-1 POL subpopulations, ordered alphabetically by primary region. The rows represent all predicted non-singleton clusters. See [RAB+00] for an explanation of HIV subfamily naming.

#	Regions	Subtypes
1	Asia (18)	C (8), 08_BC (4), 07_BC (3), BC (2)
2	Asia (9)	B (6), BC (2), 01B (2)
3	N.America/Europe (40), S.America (8)	B (51)
4	N.America/Europe (6), S.America (5)	B (5), BF (4), 03_AB (3)
5	S.America (20), N.America/Europe (6)	BF (12), F1 (6), 12_BF (5), F2 (3), 05_DF (3)
6	S/E.Africa (23)	C (22)
7	S/E.Africa (49)	D (35), A1D (12)
8	S/E.Africa (61)	C (57), A1C (8)
9	S/E.Africa (69), N.America/Europe (44), N/W.Africa (37), Asia (26)	A1 (39)

tion structure analysis, we follow standard practice and consider only the polymerase (POL) region instead of the whole genome.

Table 3.1 shows all non-singleton clusters predicted by SSCC ordered alphabetically by geographic region, with each row representing a separate cluster. SSCC clusters usually represent a single geographic region containing multiple strains (row 1), or a single strain that has spread to many regions (row 9). SSCC also well separates the European and American B strains from sub-Saharan African strains. These interesting relations between geographic origin of strains and SSCC clusters, obtained from sequences alone, show that the clusters reflect underlying population structure, and suggest that SSCC can be used for an even more refined classification of HIV strains.

3.5.2 Human population substructure

Genetic association studies are heralded as a powerful tool for discovering the genetic basis of human disease [CEKN04]. These studies analyze the genetic sequences of sets of healthy and diseased individuals to identify sequence variation associated with one or the other. Association studies assume that all subjects are from the same population and, when performed on populations with substructure, may produce many spurious associations. When one subpopulation has a higher incidence of the disease, any variation specific to that population will appear to cause the disease. Techniques for identifying mixed substructure within a sample are therefore an important part of genetic associations studies [PSD00].

In our setting, we have information on a set of single nucleotide polymorphisms (SNPs), or individual nucleotides that vary across a population. An individual's SNPs can be represented as an l-bit string encoding the variant at each position. We apply our method to a subset of a whole genome map of 1.5 million SNPs from 23 African Americans, 24 Asian Americans and 24 European Americans [HSN$^+$05]. To avoid linkage disequilibrium from proximity in the genome we use only every thousandth SNP, leaving a total of 1598 markers.

SSCC correctly labels all individuals using 1598 SNPs. We therefore compared SSCC to STRUCTURE on the harder problem of labeling individuals using only 80 of these SNPs. Our method correctly stopped at 3 clusters for 15 of 19 disjoint subsets of SNPs examined, achieving an average classification accuracy of 91.8% ($\sigma = 0.085$). We then ran STRUCTURE on each of these subsets with standard parameters and 3 clusters, achieved 90.1% average accuracy ($\sigma = 0.10$). This demonstrates that SSCC is competitive with current state-of-the-art methods for this problem.

3.5.3 *Alu* families

Nearly half our genome is the result of the activity of various classes of mobile elements [BK04]. *Alus* are a family of short interspersed nucleotide elements of about 280 nucleic acids that have, in the last 55 million years, been copied about 1 000 000 times, and now make up about 10% of our genome [BD02]. While it was once believed that a few "master *Alus*" gave birth to all *Alu* repeats, recent analyses have shown that a much larger number of *Alu* elements may be active [PEP04, CHB04]. Section 4.2 further describes *Alu* elements, and explains the motivation for better reconstructing their history.

Alu families are highly overlapping and therefore resistant to traditional phylogenetic methods. For example, the *Alu Sx*, *Alu Sz*, *Alu Sp* and *Alu Ya5* subfamilies we consider have consensus sequences differing by an average of 12.8 mutations (4.6%), while the average member of *Alu Sx* differs from its consensus sequence by an average of 34.8 mutations (12.4%, $\sigma = 4.88$).

We compare STRUCTURE to SSCC on a dataset from Price *et al.* [PEP04] of 4000 instances of the four above-mentioned families. Both methods very nearly recover the subfamily consensus sequences, finding ancestral sequences with Hamming distances $(2, 2, 1, 0)$ and $(3, 1, 1, 0)$, respectively. This small discrepancy may represent a real shortfall of both algorithms, but may also reflect the fact that our dataset contains a non-representative subset of the above-mentioned families. Despite finding nearly-identical ancestral sequences, however, STRUCTURE correctly labels only 78% of sequences to SSCC's 91%.

3.6 Conclusion

We have described a clustering objective function that combines information maximization with constraints on cluster shape, and a top-down sequential algorithm to optimize it. To address the problem of inferring population substructure in the presence of significant overlapping mutation, we define a version of our model which penalizes mutual information between sequence positions within each cluster. This constraint corresponds to the assumption that subpopulations are generated by random, uncorrelated mutations from an ancestral sequence. Our algorithm matches the popular STRUCTURE program in inferring human population structure from SNPs, and significantly outperforms it on the much harder problem of clustering *Alu* repeat elements.

There remain a number of avenues for further theoretical and algorithmic development. First, one could extend SSCC to the continuous case, using a parametric model for each class distribution. Second, an iterative Blahut-Arimoto [Bla72,TPB99] style update rule may provide better scalability. Third, it is possible to consider other forms of inter- and intra-cluster penalties. Finally, SSCC can be extended to perform hierarchical clustering by extending C in Eq. (3.3) to range over parent populations derived from the cluster splitting rule.

This chapter is based on work with Gal Chechik and Eleazar Eskin and published in the proceedings of the NIPS workshop on new methods and problems in computational biology, 2005.

Chapter 4

Reconstructing the phylogeny of mobile elements

The previous chapter presented a clustering algorithm for recovering population substructure from DNA sequences based on an extension of the Information Bottleneck algorithm. By enforcing constraints on the multi-information among positions within clusters, the algorithm was able to recover clusters even when the cluster centers were closer to each other than were many individuals within the same cluster. However, the algorithm is computationally intensive. While many interesting datasets contain hundreds of thousands or millions of sequences, the previous algorithm is not practically applicable to more than a few thousand.

The study of mobile element evolution yields valuable insights into the mechanism and history of genome rearrangement, and can help answer questions about our evolutionary history. However, because the mammalian genome contains millions of copies of mobile elements exhibiting a complex evolutionary history, traditional phylogenetic methods are ill-suited to reconstructing their history. New phylogenetic reconstruction algorithms which exploit the unique properties of mobile elements and handle large numbers of repeats are therefore necessary to better understand both mobile elements' evolution and our own.

In this chapter, we describe a randomized sequence clustering algorithm with a similar objective function, effectively minimizing cluster entropy and mutual information within clusters. We show that it scales to handle millions of sequences, and

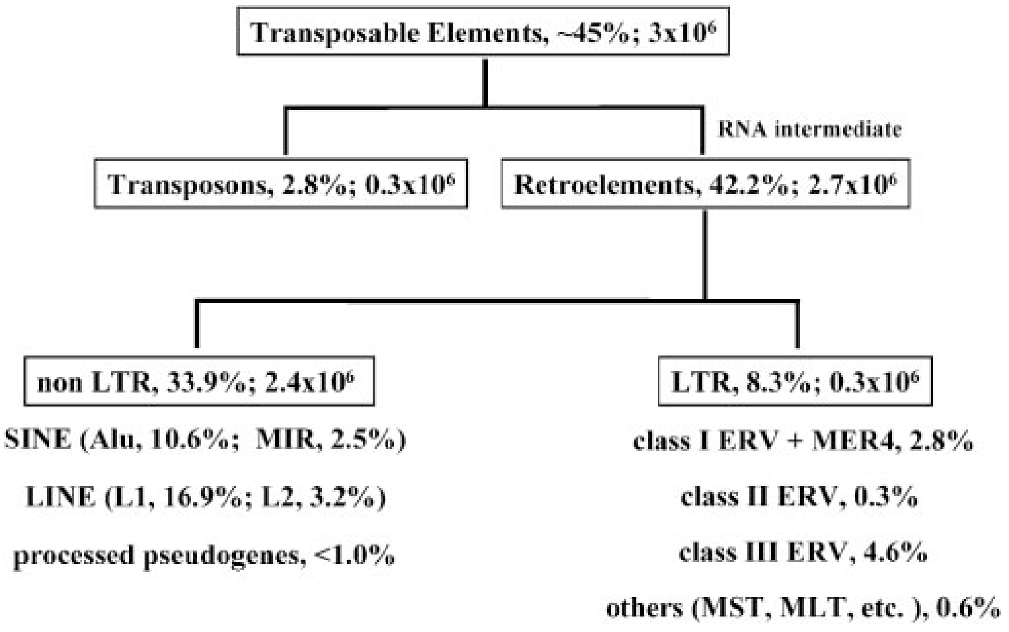

Transposable Elements, ~45%; 3x10⁶
RNA intermediate
Transposons, 2.8%; 0.3x10⁶
Retroelements, 42.2%; 2.7x10⁶
non LTR, 33.9%; 2.4x10⁶
SINE (Alu, 10.6%; MIR, 2.5%)
LINE (L1, 16.9%; L2, 3.2%)
processed pseudogenes, <1.0%
LTR, 8.3%; 0.3x10⁶
class I ERV + MER4, 2.8%
class II ERV, 0.3%
class III ERV, 4.6%
others (MST, MLT, etc.), 0.6%

Figure 4.1: Classes of mobile elements, and the percentages of the human genome they comprise (from [BK04]).

apply it to reconstruct the phylogenies of DNA mobile elements. Our results suggest that the history of mobile elements is significantly more complex than we currently understand, and that detailed understanding of mobile element history can enhance our understanding of species history.

4.1 Motivation

Nearly half our genome is the result of the activity of mobile elements, short sequences originating as exogenous viruses which copy and reinsert themselves into our genetic code (see Figure 4.1).[1] Of these mobile (or repeat) elements, a fifth are evolutionarily modern, found only in primates, and a few are still actively replicating today. Analysis of these mobile elements can help answer organism-level questions

[1] A fascinating recent paper by Malone and Hannon [MH09] shows that a simple immune system based on small RNA fragments (piRNA) has evolved to control mobile element propagation. Roughly speaking, the piRNA elements act as fingerprints for mobile elements. Piwi proteins bind to piRNA in certain genomic regions, which direct them to silence the mobile elements' expression. New piRNA are acquired when a mobile element inserts itself into a piRNA region.

of primate phylogeny and human population structure [WRO$^+$03]. Furthermore, a detailed picture of mobile element evolutionary history is crucial to understanding their role in gene expression regulation [CBC$^+$98], tissue-specificity [KJR04], genome rearrangement, protein evolution, and some genetic diseases [BD02, HB05]. Fundamental problems therefore include estimating the number, size, and age of mobile element families, the relations between them, and their distribution over species.

Reconstructing a mobile element phylogeny is difficult for at least three reasons: First, a single major repeat family can contain over 1 million elements, presenting a computational challenge for traditional phylogeny methods. Second, the families exhibit complex evolutionary history, with a large number of related and highly similar subfamilies. Third, because many of the mobile elements are ancient and are located in non-coding regions, they are highly degenerate due to mutation; it is not uncommon for a mobile element to have mutated in over 20% of its positions since insertion. Furthermore, truncation and partial insertion leave only fragments of most instances of the longer families. Although tools such as RepeatMasker [SHG06] exist to identify repeat elements in the genome, few computational tools exist to identify repeat element subfamilies and construct their phylogeny.

In this chapter, we describe a method for recovering the most likely phylogeny of observed repeats. Traditional phylogenetic methods are not appropriate for at least three reasons: First, since most are quadratic or worse in the number of sequences, they do not scale to the million or more repeat sequences we wish to analyze. Second, unlike in the case of species phylogeny, only a few repeat elements in the genome actively replicate, while the vast majority merely persist with gradual mutation. For this reason, repeat phylogenies are characterized by a few nodes in the tree each having a very large number of offspring and most nodes having none. Finally, since repeat subfamilies are highly overlapping in sequence space, the results of distance-based methods are uninformative (see Section 3.1).

Our method divides mobile elements into subfamilies via a novel clustering algorithm, Randomized Test and Split (RATS). The algorithm uses a statistical test of subfamily validity to recursively partition the set of elements, while ensuring that the final clustering is statistically well-motivated. A traditional approach to the subfamily identification problem would formulate a generative model for the data and apply the

Expectation Maximization (EM) algorithm. However, due to the tremendous number of mobile elements and the fact that the subfamilies are closely related to each other, this is impractical due to slow convergence and numerous poor local optima. Using simulated data, we demonstrate that RATS approximately recovers the EM partition in a fraction of the time.

Our method is based on the following three-phase algorithm from Price *et al.* [PEP04]:

1. Repeatedly compute the correlation between amino acids at every pair of positions in each subfamily and split it into two new subfamilies if any pair of positions fails a statistical test for independence.

2. When no such family exists, further split these initial subfamilies based on single-position deviations from a molecular clock estimated from the initial subfamilies.

3. Construct a minimum spanning tree over subfamilies' consensus sequences.

We extend and improve upon this approach in four ways. First, we test a random subset of pairs of positions for correlation rather than testing all of them (Section 4.3.2). This reduces the time and space complexity from quadratic to linear in repeat sequence length, allowing us to analyze significantly larger data sets and longer elements. Second, we incorporate more partial sequence fragments. Together, these advances allow us to extend our analysis to longer families of repeats such as L1. Third, we apply a stronger statistical test between pairs to recover more subfamilies (many of Price's novel subfamilies were found not by correlation tests, but by single-position splits). Finally, we relate our statistical test to the (approximate) optimization of an underlying generative model, formalizing the assumed repeat generation process.

We apply RATS to all mobile elements found in the ENCODE project database [TTB+03], and to repeats from the full genomes of human and chimpanzee, finding 32 new L1, 111 new SINE, and over 1000 new *Alu* subfamilies. We analyze the phylogeny of the SINE subfamilies, demonstrating its agreement with species phylogeny and with currently known repeat subfamilies.

4.2 *Alu* elements

The spread of repeat elements and their evolutionary and developmental roles have long aroused scientific interest, starting with McClintock's work on coloring differences in maize in the 1950s [OK05]. Much recent work has focused on the role and dynamics of the *Alu* repeat in humans and other primates. Eichler *et al.* [BLE03] investigate *Alu*'s potential role in producing the relatively abundant segmental duplications seen in the human genome. Zhou and Mishra [ZM05] propose that LINE-1 elements mediate some rearrangement in the rat genome, as *Alus* do in primates. Han *et al.* [HXW$^+$05] show that LINE elements may also play a role in genomic deletions in chimpanzee and human. Hedges and Batzer [HB05] review the putative role of several types of mobile element in genome growth, protein evolution, and human disease. Jurka [Jur04] reviews the dynamics of *Alu* insertion and deletion, and discusses L1-mediated retrotransposition as a mechanism for their duplication. Bourque *et al.* [BLV$^+$08] show that mobile elements act as "control elements" by embedding transcription factor binding sites, which are then spread throughout the genome.

Recent work has focused not just on the role of mobile elements in genomic evolution, but on the phylogeny of the elements themselves. Cordaux *et al.* [CHB04] reconstruct a phylogenetic network over the *Alu Y* subfamily, showing that human *Alu* elements came from multiple active sources. Price *et al.* [PEP04] reconstruct a phylogeny of all human *Alu* elements using a novel clustering method based on tests for correlated mutation. Salem *et al.* [SRX$^+$03] use *Alu Ye* elements to provide evidence about species relations among human, chimpanzee, and gorilla. Their repeat phylogeny, reconstructed by traditional methods [Fel04], gives evidence that hominids are monophyletic and more closely related to chimpanzee than to gorilla. Hedges and Batzer [HB05] argue that organism-level phylogenies should be used to provide further insight into mobile element population dynamics.

4.3 Methods

Our goal is to recover the most likely phylogeny of the observed instances. Rather than directly computing a phylogeny over all individuals, we approach the problem in two independent steps, first identifying the most likely subfamilies and their (implicit) source elements, then constructing the best phylogeny of these inferred source elements. The phylogeny of individual elements is then specified by this subfamily phylogeny and the individuals' subfamily memberships, with all individuals in a subfamily being offspring of the subfamily source element. We present our generative model of mobile element replication in Section 4.3.1. We then discuss the subfamily identification problem and an efficient algorithm to solve it in Section 4.3.2, and finally discuss the simpler problem of subsequently constructing the phylogeny Section 4.3.3.

4.3.1 Subfamily generation model

In our model, mobile elements replicate by asexual reproduction and we assume a neutral mutation rate. Each individual sequence z generates a copy of itself at time t with probability $p_c(z,t)$, and each site mutates with some probability $p_m(z,t)$. However very few individuals create copies, and these copies are all created over a relatively short period of time. In other words, $p_c(z,t) = 0$ almost everywhere, and $p_c(z,t) \gg p_m(z,t)$ where it is not. We can therefore make two simplifications: First, since almost no mutation occurs between copies of a single active element, we assume that all copies are initially identical. Second, since extant (inactive) individuals are simply the result of neutral mutation from these identical copies, we assume that a subfamily's elements are uniformly distributed (in sequence space) around its source element.

These considerations lead to the following view of mobile element generation: Each subfamily consists of a large number of inactive copies of a single active source element. (Source elements which themselves mutate are considered new distinct source elements if they are still active.) Each copy or instance undergoes independent point mutation over time, diverging from the sequence of the original source element. Each instance can, with some small probability, itself become a source element and generate its own distinct subfamily. Viewed as clusters of points in sequence space, the subfamilies form a highly-overlapping set of spheres whose radii reflect their ages.

More precisely, let $\Sigma = \{a_1, \ldots, a_m\}$ be an m-letter alphabet, and let $X =$

$(X_1, \ldots, X_l)$ be a vector random variable over Σ^l. Let C be a scalar random variable over cluster source element indices $\{1, \ldots, k\}$, with $\mu(C) \in \Sigma^l$ being source element C's sequence. The k source elements are used to generate n sequences $Z = \{z\}$ under the following model: First select a cluster $c \in C$ with probability $p(C = c)$ and make sequence z a copy of $\mu(c)$. Then independently mutate each letter z_i to some symbol $s \neq \mu_i(c)$ with a small cluster- and position-dependent probability $r_{is}(c)$. Equivalently, let $p(X_i|C{=}c)$ define the conditional probability distribution of position i of a sequence in c, and let $\mu_i(c)$ be its consensus value, $\mu_i(c) = \mathrm{argmax}_x\, p(X_i{=}x|C{=}c)$. Then $p(X_i{=}s|C{=}c) = r_{is}(c)$ for $s \neq \mu_i(c)$ and $p(X_i{=}\mu_i(c)|C{=}c) = 1 - \sum_s r_{is}(c)$. The distribution over sequences X can then be modeled as a mixture of per-cluster distributions:

$$p(X) = \sum_{c \in C} p(C = c) \prod_{i=1}^{l} p(X_i|C = c)$$

4.3.2 Subfamily identification

Given an n-element sample Z from $p(X)$, our goal is to recover the number of clusters k and, given k, to find the assignment $p(C|Z)$ of individuals to clusters maximizing the likelihood of the sample. The model's free parameters, representing the cluster probabilities $p(C)$ and discrete distributions of symbols occurring at each sequence position in each cluster $p(X|C)$, assume their maximum likelihood values.

This is an instance of hard clustering, a well-studied problem for which EM with soft assignment has been shown to perform well in many cases [MH98]. However, EM is not applicable to our problem for two reasons. First, our dataset is enormous: there are over one million elements in just the single largest repeat family in the human genome (*Alu*), representing a thousand or more distinct clusters. EM clustering, which requires $O\left(kl(m + n)\right)$ operations per iteration (or $\approx 10^{11}$ for $k = 10^3$, $n = 10^6$, and $l = 10^2$), is computationally impractical at this scale. Second, since the objective function has many poor local optima, EM often requires many random restarts to find a good solution (see Section 4.4.1), further increasing runtime.

Limiting model complexity

Since data likelihood will always increase as the number of clusters increases, we need to penalize or limit model complexity. One common solution to this well-known problem is to add a model complexity penalty to the objective function, with the Bayesian Information Criterion (BIC) being a popular choice [CH97]. The BIC for M model parameters and N data elements is $\frac{M}{2}\log N$.

However, we find on simulated data that the BIC penalty dominates our model score long before we have recovered the correct number of clusters (Section 4.4.1). For example, even on the small simulated dataset in Figure 4.8 with $k = 11$, the penalty is approximately 3×10^4, or almost ten times the improvement in log-likelihood over the random model. This happens because the BIC for our model, $\frac{k}{2}(1 + l(m-1))\log n$, depends linearly on the number of free parameters, and hence in our case on the sequence length. However, the number of parameters in our model is artificially high because we learn a set of independent clusters for data we assume are generated by a hierarchical model in which the clusters are highly dependent.

Our assumptions from Section 4.3.1 suggest an alternative approach. Since we assume that mutations are independent within a subfamily, if the distribution of mutations at a pair of positions within a single cluster fails a statistical test for independence, we have evidence of further substructure. When no such pair exists, we have evidence of a candidate solution. Therefore instead of directly maximizing the likelihood as we would with EM, we instead search for a solution that satisfies this statistical test.

Specifically, we consider the estimated mutual information between pairs of positions [CT91]:

$$I(X_i; X_j|c) = \sum_{s_i, s_j \in \Sigma} p(s_i, s_j|c) \log \frac{p(s_i, s_j|c)}{p(s_i|c)p(s_j|c)}$$

Goebel *et al.* [GDHM05] show that the mutual information between two uncorrelated discrete random variables X_1 and X_2 can be approximated by a gamma distribution

$$\Gamma\left(\frac{1}{2}(|X_1|-1)(|X_2|-1),\ \frac{1}{N\log 2}\right)$$

where N is the sample size and $|X_i|$ is the number of possible values of X_i. From this approximation we can derive a p-value for independence between two positions $1 \leq i < j \leq l$ and a corresponding threshold α for $I(X_i; X_j|c)$.

Rather than testing all $O(l^2)$ pairs of positions, we randomly sample enough pairs to detect correlation with high confidence. Assume that a single supposed cluster contains members of two true clusters differing in λ of l positions drawn from a binary alphabet, and that we want to test at least one pair of these λ positions with probability $1-p$. Then the probability of choosing a pair of correlated positions in a single draw is $\frac{\lambda(\lambda-1)}{l(l-1)}$ and probability of not detecting correlation after t trials is $p = \left(1 - \frac{\lambda(\lambda-1)}{l(l-1)}\right)^t$ Therefore choosing t to achieve our desired p yields

$$t = \frac{\log p}{\log\left(1 - \frac{\lambda(\lambda-1)}{l(l-1)}\right)}$$

For example, to detect a difference in 4 out of 300 positions with $p = 0.99$, similar to the *Alu* problem, we must sample 343 pairs. Mutation will increase the number of tests required by causing some of the $\lambda(\lambda - 1)$ pairs of correlated positions to no longer be significantly correlated.

Fast randomized clustering

Conveniently, this same random pair sampling suggests an efficient top-down clustering algorithm. When positions i and j are correlated in cluster c, splitting c into two clusters c_1 and c_2 such that c_1 contains all individuals z with $(z_i, z_j) = \operatorname{argmax} p(X_i = z_i, X_j = z_j|c)$, and c_2 the rest, will tend to improve $p(Z|C)$. When no such position exists, further cluster splits cannot significantly decrease the entropy of a cluster at multiple sequence positions at once, so we are near a local maximum of the likelihood function.

While these tests ensure that each pair of split clusters is statistically justified, multiple splits and poor initial splits may create clusters which could be merged without creating correlated pairs. Such problematic pairs can be found by trying to merge sets of clusters and repeating the above correlation sampling. Because testing all subsets of clusters is computationally impractical, we instead test only pairs of clusters with similar consensus sequences.

```
function RATS ($Z$, $n$)
    $\mathcal{C}_0 \leftarrow \{Z\}$
    for $i = 1$ to $n - 1$
        $\mathcal{C}_{i+1} \leftarrow \text{join}\left(\bigcup_{C \in \mathcal{C}_i} \text{split}(C)\right)$
    end for
    return $\mathcal{C}_n$
end function RATS

function split($C$)
    repeat $t$ times
        $(i, j) \leftarrow \text{random}([1, l])$
        if $I(X_i; X_j | C = c) \geq \alpha$
            $(s_i, s_j) \leftarrow \text{argmax}\, p(s_i, s_j | c)$
            $C' \leftarrow \{z | z_i = s_i, z_j = s_j, z \in C\}$
            return $\text{split}(C') \cup \text{split}(C - C')$
        end if
    end repeat
    return $\{C\}$
end function split

function join($\mathcal{C}$)
    $\mathcal{C}' \leftarrow \{\}$
    while $|\mathcal{C}| > 1$
        $(C_1, C_2) \leftarrow \text{argmin}\, d_H\left(\mu(C_1), \mu(C_2)\right)$
        $\mathcal{C} \leftarrow \mathcal{C} - \{C_1, C_2\}$
        if $\text{split}(C_1 \cup C_2) = \{C_1 \cup C_2\}$
            $\mathcal{C}' \leftarrow \mathcal{C}' \cup \{C_1 \cup C_2\}$
        else
            $\mathcal{C}' \leftarrow \mathcal{C}' \cup \{C_1, C_2\}$
        end if
    end while
    return $\mathcal{C}' \cup \mathcal{C}$
end function join
```

Figure 4.2: The RATS algorithm. The optional greedy updates are performed between `split` and `join`, and again after the final iteration.

Recursively applying this cluster splitting criterion yields the iterative algorithm in Figure 4.2. Starting with a single cluster, we iteratively apply the following two steps: First, recursively test and split the current set of clusters until no cluster fails the correlation test. Second, merge neighboring pairs of clusters when doing so does not create detectable correlation. However, the split heuristic will poorly assign some elements, and splitting can only make large-scale changes to the current clustering. We therefore apply an additional fine-grained greedy improvement between the splitting and joining steps, assigning each individual z to the cluster with the closest consensus sequence $\mu(c)$, i.e. $c = \text{argmin}_c \quad d_H(z, \mu(c))$ where d_H is the Hamming distance.

4.3.3 Subfamily phylogeny

Given a set of subfamilies, we next reconstruct the most likely phylogeny. In the standard formulation, one assumes that the observed sequences are the tree's leaves, then infers a maximum likelihood binary tree over them according to some mutation model (see e.g. [FNPP01]). Our problem is different in three ways: First, our subfamilies represent not just the leaves of the tree, but also its internal nodes, obviating the usual optimization over unobserved ancestral nodes. Second, the inferred $p(X|C)$ defines the mutation model for each cluster. Third, the average divergence within a cluster defines its age, allowing us to constrain our tree so that a node must be younger than its parent.

We assume that the consensus sequence of each subfamily is generated from its parent family, and is therefore drawn from the parent family's distribution, though it does not have to be one of the actual observed members of either. For each cluster $c \in C$, we choose as its parent the most probable ancestor for its consensus sequence $\mu(c)$,

$$\underset{\{c' | \text{age}(c') > \text{age}(c)\}}{\text{argmax}} \; p(c')p(\mu(c)|c')$$

where age(c) is the estimated age of cluster c.[2] By assuming that mutation rate is constant over the genome at each point in time, we can approximate a subfamily's age by its average mutation rate or (equivalently) its entropy. Although this assumption is clearly not valid for estimating a subfamily's absolute age, our algorithm depends only on the ordering of the age estimates across different subfamilies. Since a genome containing elements of one subfamily will contain elements of all of its ancestors, it is highly unlikely that molecular clock variation will cause a repeat subfamily to appear younger than its ancestors.

[2]In principle, phylogeny construction should account for uncertainty in age(c), $p(c)$, and $p(\mu(c)|c')$. This is becomes more important as the number of subfamilies increases, since many will be close in sequence space and/or age. In practice, large trees were not useful for further analysis, so we used the basic approach described here, which seemed sufficient for modest numbers of subfamilies. However, a discussion of alternative tree construction approaches, see Section 4.6.1.

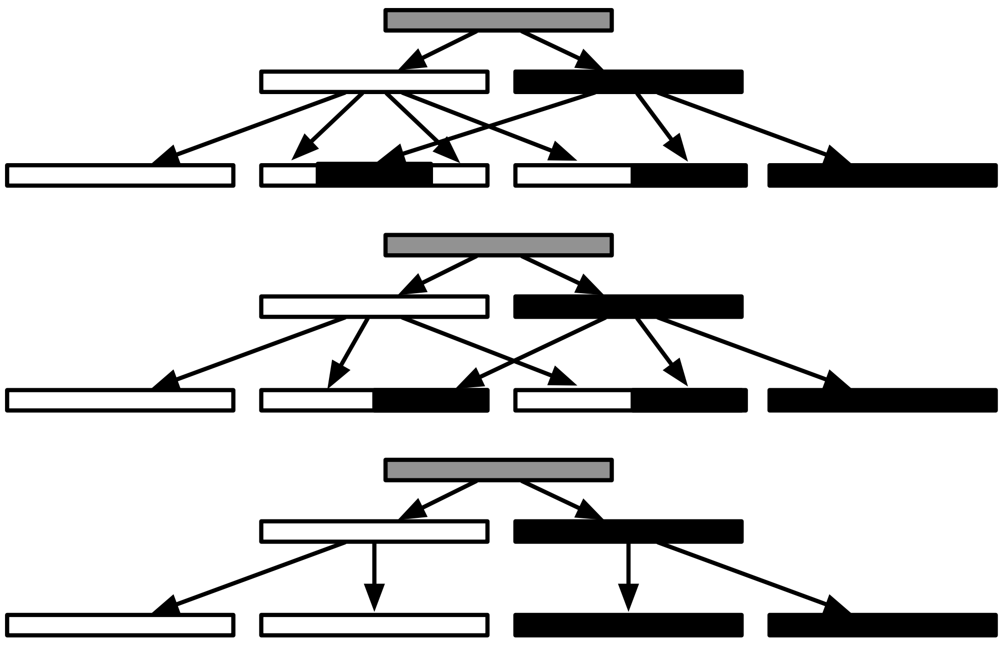


Figure 4.3: Population structures for simulated recombinant data.

4.3.4 Extension for recombination

We can extend the above approach to recombinant phylogenies in an analogous way. We want to find a recombinant phylogeny with relatively few recombination events. Rather than defining a prior over recombination rates or a trade-off between the number of recombinations and log-probability, we can select only statistically significant recombinations. In other words, we choose a model with recombination between parents c_1 and c_2 at position r only if

$$\max_{c_1,c_2,r} P\left(\log\left(\frac{P(\mu_{[1,r]}(c)|c_1)P(\mu_{[r+1,l]}(c)|c_2)}{P(\mu(c)|c')}\right) > k\right) < \alpha \tag{4.1}$$

for some significance level α, where $\mu_{[a,b]}(c)$ is the subsequence of cluster c's consensus sequence between positions a and b, and c' is its current, non-recombinant parent. A similar equation can be derived for two-point recombination. After finding a non-recombinant phylogeny as described in Section 4.3.3, we can assign recombinant parents to clusters satisfying Eq. (4.1).

We validate this algorithm on simulated data with the population substructures shown in Figure 4.3. The populations are designed to test the algorithm's

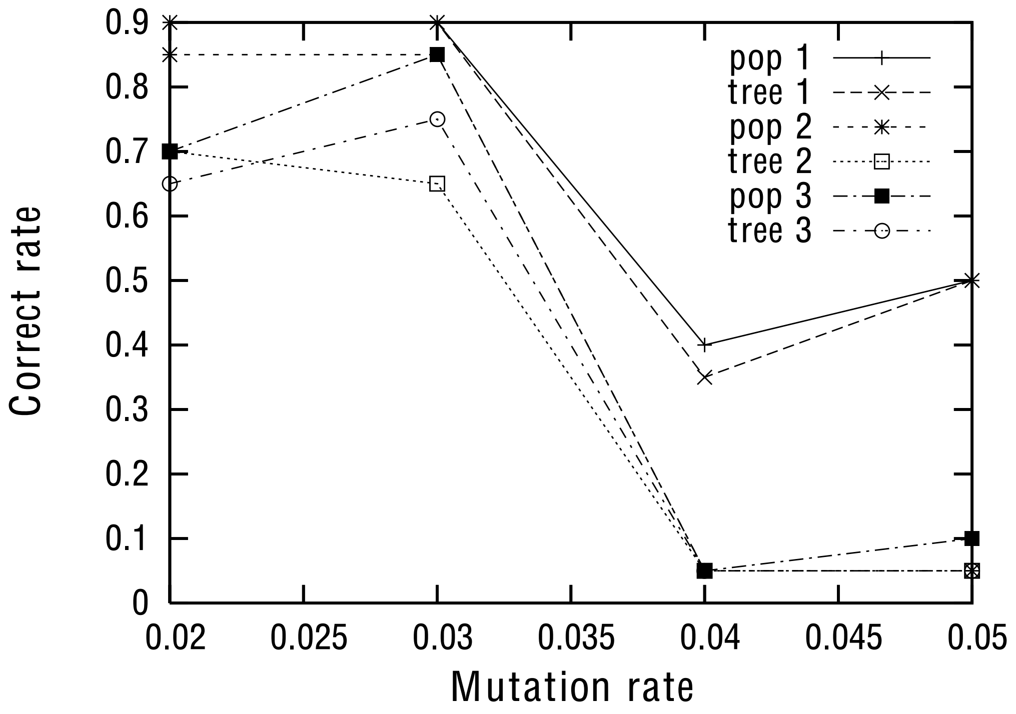


Figure 4.4: Subpopulation and phylogeny reconstruction accuracy *vs.* per-generation mutation rate.

sensitivity in detecting: (1) multiple recombinations from the same parents; (2) recombinant versus non-recombinant child populations; (3) two-point crossover events; and (4) distinct non-recombinant children of the same parent.

We explore performance over a range of average per-generation mutation rates, sequence lengths, and mutation counts of the subpopulation consensus sequences. We assume a constant subpopulation size of 20 binary sequences of length 50. Figure 4.4 shows the rate of inferring the correct number of subpopulations and correct phylogeny versus the per-generation mutation rate for each simulated population structure. As expected, accuracy decreases more quickly for recombinant than for non-recombinant populations, with similar performance for one- and two-point crossover. While it is not reflected in the figures, many of the incorrect substructures and phylogenies are correct for the first two generations, only failing to correctly identify or parent the third-generation populations.

Figure 4.5 shows the same information as Figure 4.4, but instead varies the distance between cluster consensus sequences. As expected, accuracy falls off dra-

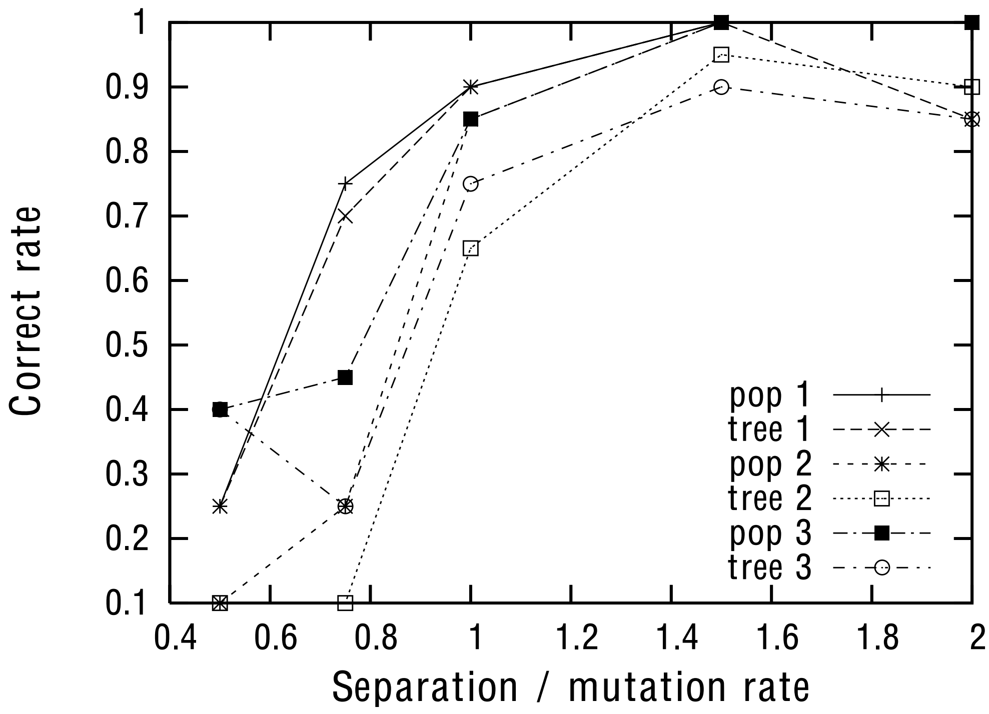


Figure 4.5: Subpopulation and phylogeny reconstruction accuracy *vs.* consensus sequence divergence relative to per-generation mutation rate.

matically as the rate of random mutation within a cluster exceeds its divergence from its parent. Note, however, that all populations are highly overlapping: in all cases the mean sequence divergence in the parent population is at least as great as that between parent and child consensus sequences.

4.4 Results

We first demonstrate that our method performs well relative to EM on simulated data similar to actual repeat elements. We then apply our method to repeat elements collected from over sixty organisms, finding subfamilies and phylogenies of the *Alu*, SINE (an evolutionary superset of *Alu*), and L1 families. We discuss a phylogeny of SINE elements across all ENCODE project species. Our repeat phylogeny's close agreement with known repeat family and species phylogenies validates our approach.

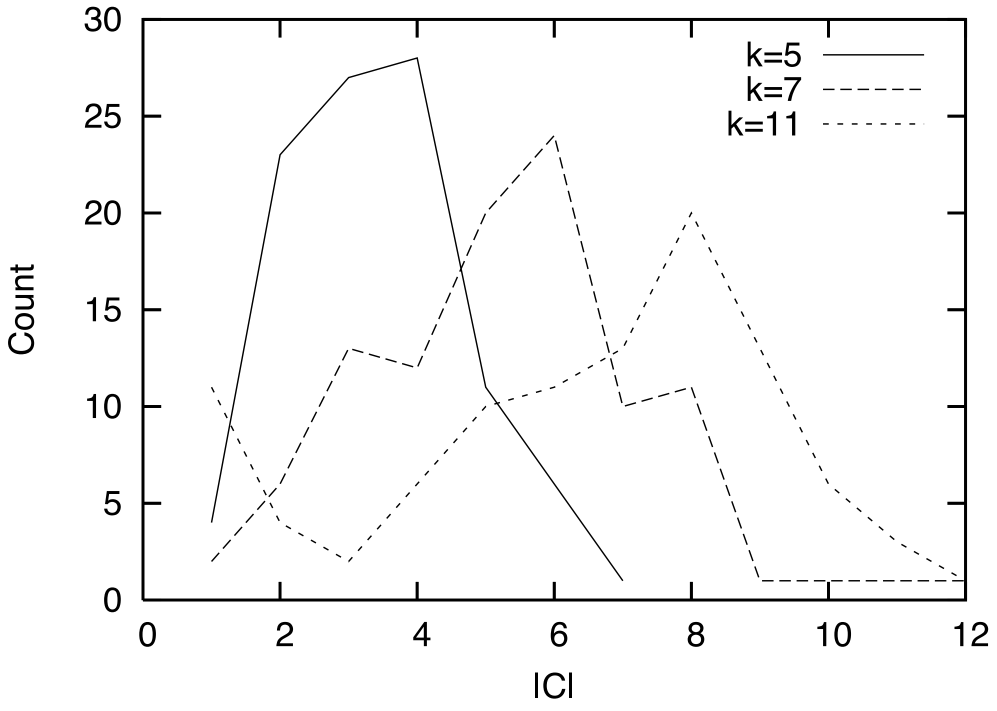


Figure 4.6: Frequency of number of subfamilies found for different true numbers of subfamilies.

4.4.1 Simulated data

A repeat phylogeny algorithm should obey two correctness properties: First, it should be approximately optimal: for any cardinality, the likelihood of the data given the model it finds is close to the optimum likelihood. Second, it should be conservative: the probability of the algorithm finding more than the true number of subfamilies should be acceptably small. Here we show that RATS exhibits these properties on simulated data similar to actual repeat subfamilies.

Each dataset consists of sequences in $\{0, 1\}^{300}$ drawn from one of k subfamilies separated by d mutations with a uniform per-position mutation rate u. For each (d, k, u), we generate 5 sets of repeats with these parameters, then run RATS 20 times. We approximate the optimal likelihood for each number of subfamilies found by taking the best of 10 EM solutions.

Figure 4.6 shows the distribution of the number of subfamilies found for $d = 6$ and $u = 0.1$ while varying k. As desired, RATS is conservative, finding more than the true number of subfamilies 1–12% of the time. Figure 4.8 compares the likelihood of

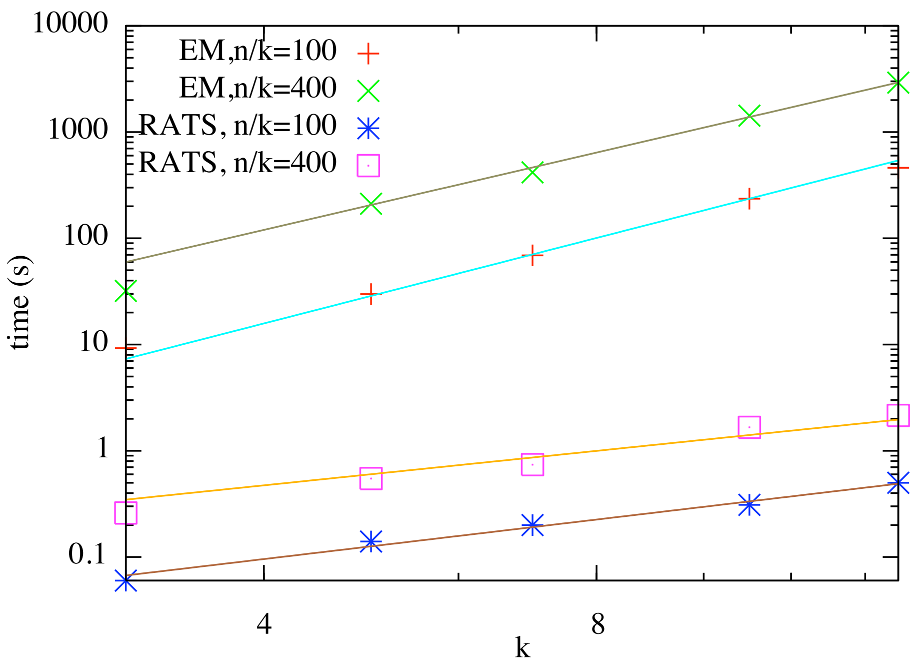


Figure 4.7: Running time of the EM algorithm and RATS for different numbers of subfamilies k and different numbers of instances per subfamily n/k.

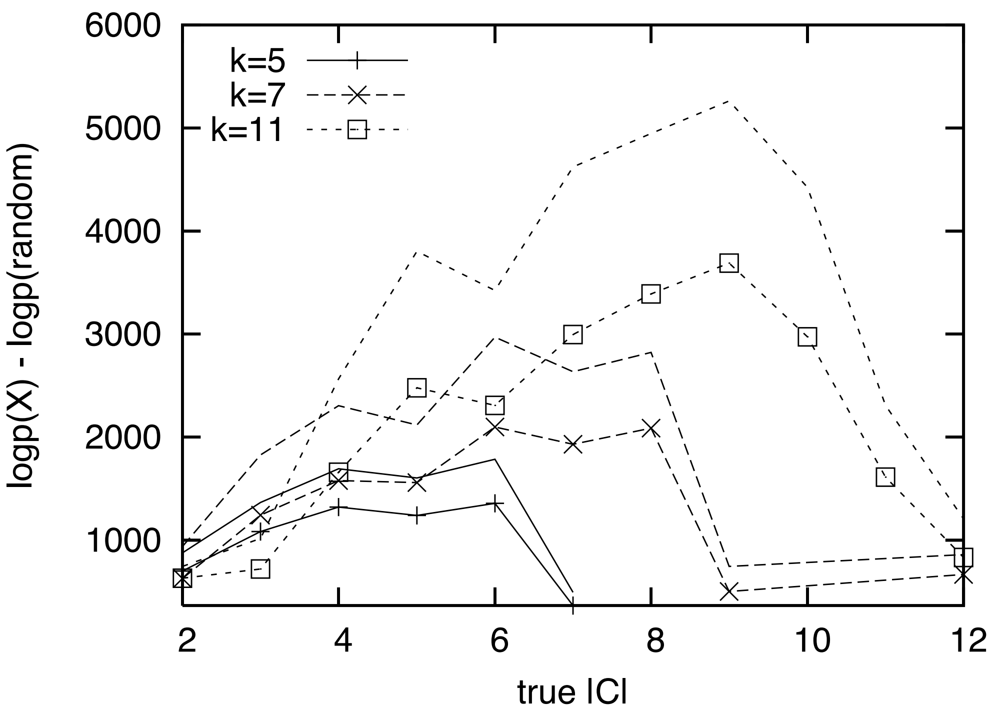


Figure 4.8: Log-probability versus random clustering for RATS (lines with marks) and EM (lines without).

the best solutions found by RATS and EM (with 10 restarts) to the average likelihood of a random clustering. (The random model likelihood is used instead of the generating model likelihood because, particularly for larger numbers of subfamilies with fewer members, the generating model can yield significantly lower likelihood than a learned model.) As expected, RATS consistently performs slightly worse than EM, with both approaching the random score as the number of subfamilies diverges from the true number.

Figure 4.7 shows the time for a single run of each algorithm as a function of the subfamily arity k and subfamily size n/k. Since the two algorithms are implemented in different languages, we cannot directly compare runtimes. However, EM's runtime grows at a faster rate, largely because the number of iterations to convergence grows with k. A least-squares fit of Ak^B to both curves shows EM is at least quadratic in k ($B = 2.7, 2.4$), where RATS is nearly linear ($B = 1.2, 1.1$).

To show that clusterings are consistent between runs, we ran RATS 10 times on the ENCODE data, using the pairwise adjusted Rand index to measure consistency between runs [HA85]. The number of subfamilies ranged from 274 to 300 with a mean of 286, and the Rand index ranged from 0.154 to 0.193 with a mean of 0.173, corresponding to a permutation-based p-value less than 10^{-8}.

4.4.2 Data preparation

We obtained "full" genome sequences for *P. troglodytes* and *H. sapiens* from the most recent sequence builds available at UCSC as of March 1, 2006. We also downloaded accession numbers for orthologous regions of 64 vertebrate species from the NIH Intramural Sequencing Center (NISC) Comparative Sequencing Program (ENCODE) [TTB$^+$03], obtaining the corresponding sequence data from GenBank. Sequences were joined together according to the position information specified within the GenBank files.

A data set of repeat elements was created for the complete set of sequences via RepeatMasker version 3.1.3 [SHG06] using a library of repeat elements from RepBase [Jur00], using the lowest sensitivity for the "full" genomes and standard sensitivity for the ENCODE regions. We generated multiple alignments for single repeat

Table 4.1: Numbers of repeats in various families found in RepBase and by our method. *Alu* and L1 repeats are from the full Human and Chimpanzee genomes, while SINE repeats are from the ENCODE database of orthologous regions.

Family	RepBase	RATS	Elements	Source
Alu	35	1519	2 309 150	primate
L1	102	134	24 249	primate
SINE	197	308	381 248	ENCODE

classes (e.g. *Alu*, L1) from pairwise alignments of each repeat instance to a single RepBase consensus sequence (e.g *Alu Sx*, HAL1#LINE/L1) as follows: Consensus sequences for repeat classes were created by Clustal W [THG94] multiple alignment of all constituent RepBase consensus sequences. Eleven SINE consensus sequences that aligned poorly to the others were excluded. Positions corresponding to gaps in the pairwise instance alignments were removed, and the aligned instances were threaded into the repeat class multiple alignment to yield a multiple alignment of instances. Interior gaps were treated as a separate symbol, leading and trailing gaps as missing data.

4.4.3 Novel repeat subfamilies

Table 4.1 compares the number of subfamilies we find to the number identified in RepBase. To ensure the correlation test's validity, RATS was constrained to test only subfamilies of more than 200 elements for L1, and more than 1000 for Alu and SINE. All runs used a p-value of 10^{-3} with a Bonferroni correction for testing multiple positions. The relatively small number of L1 subfamilies discovered may be due to the small number of available elements, and further L1 subfamilies may exist that cannot be statistically validated. The results, especially those for *Alu*, suggest that repeat phylogeny may exhibit much more fine-grained structure than is currently known.

Figure 4.9 shows the *Alu* phylogenies for subfamilies identified by RepBase and by Price *et al*. Our phylogeny has a similar shape, including the new branch reported by Price (label 7). However, a phylogeny with over one thousand is difficult to interpret and not especially useful. Instead, we first look in detail at members of young *Alu* families in Section 4.4.5. Then in Section 4.4.6, we look at the overall distribution of family ages.

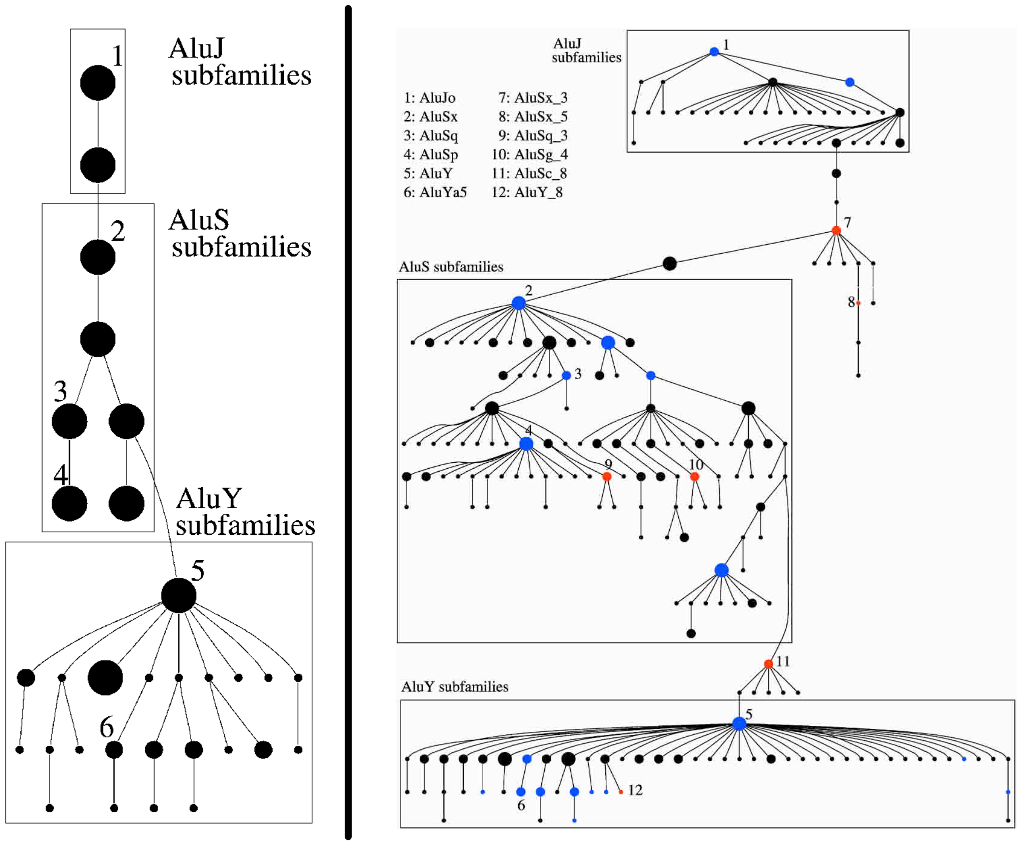


Figure 4.9: Left: *Alu* phylogeny identified in RepBase. Right: *Alu* phylogeny found by Price *et al.*

4.4.4 SINE phylogeny

Figure 4.10 presents a phylogeny of 381,248 SINE elements matching RepBase SINE elements found in 61 ENCODE species. Subfamilies are assigned a repeat subfamily (color) if at least 70% of their elements belong to that RepBase subfamily. Subfamilies are also assigned a species (border) if at least 70% come from a single species or a small group of related species. These labels may partly reflect variation in ENCODE's coverage between species. However a phylogeny constructed from more complete data, while it may have higher resolution, will still be consistent with the one presented here. Representative subfamilies are labeled with the average percent divergence of their elements from the subfamily consensus. The phylogeny is taken from a single run of RATS.

Although our predicted phylogeny contains 111 novel repeat subfamilies, it still reflects many known aspects of SINE phylogeny. We recover the basic relationship between the oldest *Alu J*, intermediate *Alu S*, and recent *Alu Y* clades. Ages estimated from subfamily divergences are roughly in agreement with estimated family ages [BLE03]. The *Alu Jo* subfamily is an ancestor of *Alu Jb* (Fig. 4.10, label 7). The 11%-diverged *Alu Sx* branch reported by Price [PEP04] appears at (Fig. 4.10, label 6).

Our results are also consistent with primate clades: for example, the *Alu Jo* branch at (Fig. 4.10, label 1) separates strepsirrhins from the new- and old-world monkeys. Additionally, the large group of Galago-specific subfamilies we observe is consistent with the sequence analysis of Zietkiewicz *et al.* [ZRSL98]. The *Alu Y* subfamily is (as expected) confined to old-world primates (Fig. 4.10, label 2), though without human sequence data only a subset of the currently-known *Alu Y* families are found. Finally, the divergence of the baboon-specific families dates them to after the divergence of old-world monkeys.

Further up the tree we find additional validation of our repeat phylogeny. Repeats from platypus and echidna, both monotremes, are concentrated in a subtree of MIRm elements at (Fig. 4.10, label 3). Similarly, the marsupial-specific MIR_Mars elements are correctly identified at (Fig. 4.10, label 4). Our analysis does not find a clade among marsupial, monotremic, and placental repeat elements. Monotremes diverged before marsupials, and Gilbert *et al.* [GL00] argue that the Ther-2 repeat

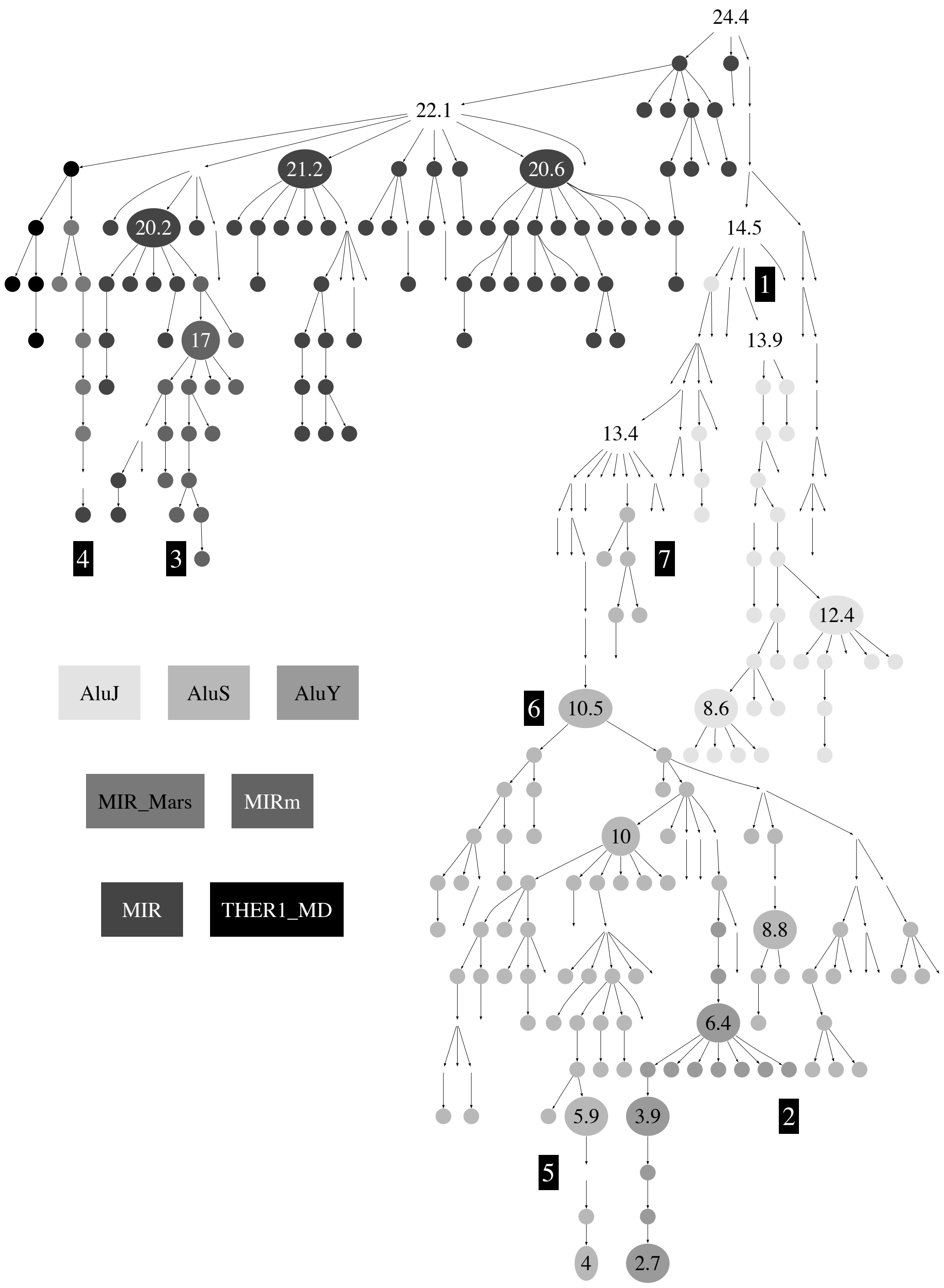


Figure 4.10: SINE phylogeny from one run of RATS on all ENCODE species, showing sequences from (1) strepsirrhins, (2) old-world primates, (3) monotremes, (4) marsupials, and (5) new-world primates. Edges represent putative relations between families. Node colors represent known RepBase families. Numbers inside nodes indicate average percent sequence divergence from the subfamily consensus sequence. Unattached, numbered black squares correspond to labels in the text.

Table 4.2: Comparison of human-specific *Alu* clusters found by Mills *et al.* [MBI+06] to clusters found by running RATS over all human *Alu* elements. Column 2 shows the most common RepBase subfamily of elements in a cluster. Columns 3 and 4 show the number of elements in Mills' and our clusters, respectively. Column 5 shows the number and percent of elements in a RATS cluster that are also found in the corresponding Mills cluster.

Cluster	RepBase	Mills	Us	Overlap	(%)
1	Ya5	1709	1795	1706	(95)
2	Yb8	1290	1098	1005	(92)
3	Yb8	–	149	136	(91)
4	Yc1	356	286	240	(84)
5	Yg6	261	234	232	(99)
6	Y	484	68	58	(85)
7	Ya4	–	162	158	(98)
8	Ye5	–	111	110	(99)
9	Yi6	–	97	96	(99)
10	Yd8	–	97	97	(100)
Total		4100	4097	3838	(94)

family is found only in marsupials and placental mammals. However, while the species phylogeny is well-understood, there remains some disagreement about MIR repeat phylogeny, and in some ways the marsupial genome is more similar to the monotreme than to the placental genome [MC04].

4.4.5 Young *Alu* families

Some *Alu* elements remain active in the human genome even today [DEH03]. Availing themselves of the recently-released complete chimpanzee genome, Mills *et al.* [MBI+06] take a comparative genomics approach to identifying these modern elements. First, they align the human and chimpanzee genomes to locate insertions and deletions. They then compare each inserted sequence to known mobile elements; if the insertion consists entirely of a complete mobile element, they conclude that it is a member of a family active since humans and chimpanzees diverged. Finally, they group these elements by RepBase subfamily to generate a list of recently-active mobile element families. Mills identifies five recent families of *Alu Y* elements.

If our method is sufficiently powerful, it should be able to identify these recent elements by placing them together in one or more recent families. Table 4.2 compares

Mills' recent families to clusters that were found by applying RATS to all human *Alu* elements. The corresponding ten RATS clusters significantly overlapping Mills' recent families contain 93.6% of their elements while only 6.4% of their members lie outside these families. This result suggests that RATS is able to accurately detect recently-active *Alu* elements.

4.4.6 *Alu* age distribution

Previous studies have shown that mobile element radiation correlates with periods of rapid speciation [Sve00, MH09]. Given their potential role in copying regulatory motifs [KJR04, BLV$^+$08] and facilitating genome rearrangement [BD02, HB05], mobile element duplication may an important means of rapid genomic and hence phenotypic change.

Kim *et al.* [KHR04] investigate the distribution of ages of mobile elements from five families (*Alu*, L1, LTR, MIR, L2) based on the human RepBase catalog. They estimate the age of each element based on its divergence from its RepBase consensus sequence a standard molecular clock [Int01]. The resulting histogram of repeat ages shows a single major peak of expansion for each family except L1, which has two. These peaks correspond roughly to known evolutionary branches between human ancestors and marsupials (MIR, L2), early eutherians (L1), strepsirrhins (LTR), new-world monkeys (*Alu*), and old-world monkeys (L1).

Our method's higher-resolution repeat clustering should produce a more fine-grained view of the historical rate of mobile element insertion. We clustered all human *Alu* elements, then plotted a histogram of cluster ages as estimated from average element divergence from its cluster mean and an *Alu*-specific molecular clock [Int01, XHH$^+$04]. The oldest distinguishable clusters, containing the oldest *Alu J* elements, are approximately 80 million years old. The youngest clusters our method can reliably detect, whose mean elements differ in approximately 4 positions, are about 4 million years old.

Figure 4.11 shows the age histogram, annotated with the approximate times of some evolutionary events. Our higher-resolution clustering clearly shows three peaks corresponding, from oldest to youngest, to the major expansion periods of the *Alu J*,

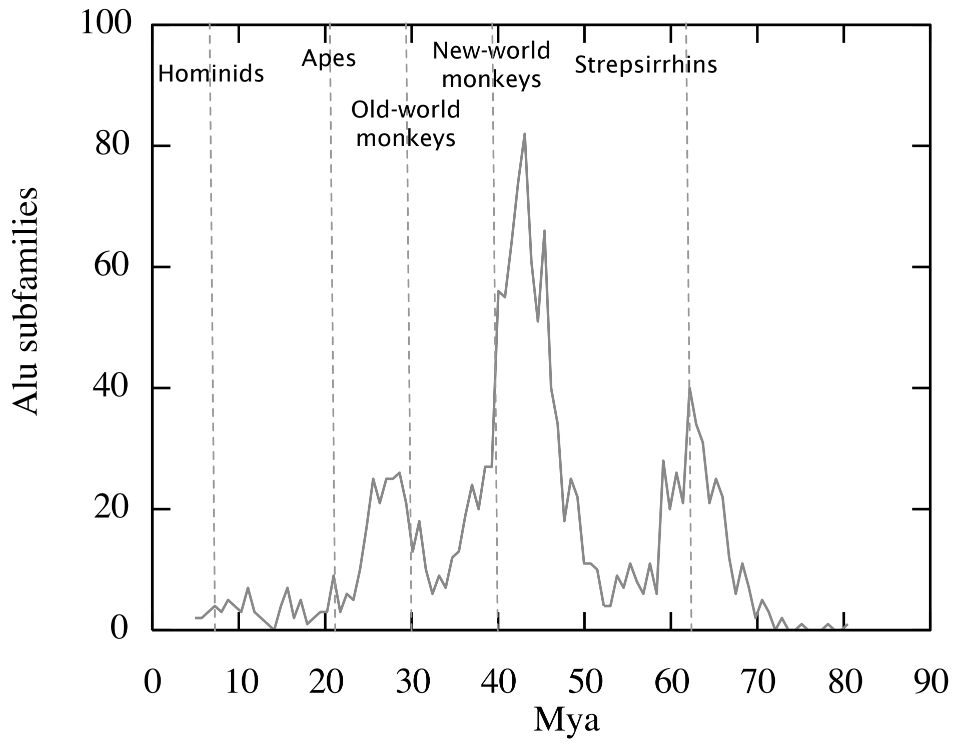


Figure 4.11: Histogram of *Alu* cluster ages versus evolutionary events, using an *Alu*-specific molecular clock [XHH$^+$04].

Alu S, and *Alu Y* families. These three periods of rapid *Alu* insertion correspond roughly to the divergences of three major primate clades: the strepsirrhins, old-world monkeys, and new-world monkeys. Our results do not show a corresponding burst of *Alu* activity and the divergence of chimpanzees or apes from hominid ancestors. The former event took place recently enough that our method may not able to distinguish the relevant clusters, while the latter may be correlated with activity in some other repeat family, such as L1. While these results are preliminary, they suggest that a more detailed view of mobile element activity over time may yield valuable insight into their evolutionary role.

4.5 Conclusion

This chapter has described RATS, a randomized clustering algorithm for rapidly finding repeat subfamilies and a procedure for reconstructing phylogenies from sets of subfamilies. Because the divisions between subfamilies RATS finds are statistically validated, and because simulations show that it estimates the number of subfamilies conservatively, we can be confident that real substructure is being detected. We have applied our approach to SINE repeat data, yielding a phylogeny consistent with known repeat and species relationships both among and beyond primates. We have also applied RATS to LINE and *Alu* elements from multiple species, showing that mobile elements display complex family substructure and history. Our results suggest a number of areas for further biological exploration.

4.6 Algorithmic Extensions

There remain a number of directions for algorithmic refinement. First, as noted above, inferring the phylogeny and clustering simultaneously would yield a more powerful model; an analogous randomized approach could again yield an efficient approximation. Second, including species phylogeny and repeat orthology information would improve sensitivity. Finally, large- and small-scale phylogeny could be handled simultaneously using an iterative approach combining clustering on inferred subfamilies with refinement of the guiding multiple alignment. This could substantially increase the algorithm's running time, but it would reduce any bias introduced by the original RepBase-guided alignment, and could also improve cluster resolution. A more sophisticated approach, which would construct the alignment and clustering simultaneously is outlined in Section 4.6.2, below.

These methodological improvements, particularly the last, open up a number of experimental avenues. Multi-scale clustering would make it possible to distinguish clades among primates, while fine-grained distinctions between families would enable the detection of repeat homoplasy and the analysis of repeat insertion hotspots. Large-scale repeat phylogeny therefore has the potential to contribute in a number of ways to our understanding of evolutionary processes and history.

4.6.1 Alternative phylogeny construction methods

Section 4.3.3 describes a simple algorithm to construct a phylogeny (i.e. a directed tree) from a set of clusters with estimated ages. The algorithm is not guaranteed to find the optimal tree, and the tree it does find can be sensitive to small changes in the clustering output. Although these problems were not a significant problem for our analysis, in this chapter, we first extend our simple approach to be robust to this uncertainty. We then extend an existing optimal directed spanning tree algorithm to construct a "minimum-weight partial ordering," defined as the partial ordering consistent with all possible minimum spanning trees over a directed graph with uncertain edge weights. These two algorithms can be combined to yield a robust, optimal ancestry tree for clusters with estimated ages.

Uncertain age and likelihood

First, when constructing a tree over many clusters, several may have similar ages, and it may be preferable to choose c's parent to be the most likely c' such that $\text{age}(c') > \text{age}(c) + \delta$ for tolerance δ. δ can either be chosen based on underlying biological processes, or estimated from the data itself. For example, δ could reflect the variation in cluster ages across multiple bootstrapped clustering runs, limiting the probability that two clusters' ages could be reversed.

Second, there may be several potential parents of c with near-equal likelihood:

$$P = \{\ c'' \mid \text{age}(c'') > \text{age}(c),\ p(c'')p(\mu(c)|c'') + \epsilon > p\ (c')p(\mu(c)|c')\}$$

where c' is the most-likely parent and ϵ is a small constant. In this case it may be preferable to first construct a directed acyclic graph (DAG) such that $c' \to c$ if and only if $c' \in P$. In other words, each cluster c is linked from all clusters almost as likely to be its parent as the most likely parent.

These two extensions can be combined either by choosing a hard age tolerance δ, or by choosing an appropriate probability distribution $p(c' \to c|\text{age}(c') - \text{age}(c))$. Given the resulting DAG O, we can construct a conservative tree T as follows:

1. For each node $v \in O$, find the least common ancestor v_0 of its parents.
2. Remove all incident edges to v, and add a single edge $v_0 \to v$ to T.

T can be thought of as a "most specific ancestry" rather than a phylogeny: If $c' \to c$ in T, then c' is the most recent certain ancestor of c, given our uncertainty about true parentage.

Optimal DAGs with uncertain weights

The minimum-weight arboration (MWA) problem is the directed analog of the minimum spanning tree for undirected graphs. Given a directed graph $G = (V, E)$ and weight function $w : E \to \Re$, we want to find a set T of $|V| - 1$ edges that spans V with minimum weight. In other words, T should be the minimum-weight graph containing exactly one edge incident to every vertex except one, the root. Chu and Liu [CL65] and Edmonds [Edm67] describe an elegant algorithm for finding the MWA in $O(|V||E|)$ time that works roughly as follows:

1. Find the subgraph G' consisting of the minimum-weight edge incident to each $v \in V$.
2. If no cycles exist, then G' is an MWA, since each node has a single incident edge and G' has $|V| - 1$ edges.

3. Otherwise, choose a cycle with minimum-weight edge e_m and *contract* it into a single node, adjusting the weights of all edges incident on the cycle. Specifically, if $u \to v$ is an incident edge, and $x \to v$ is a cycle edge, then

$$w(u \to v) \leftarrow w(u \to v) - (w(x \to v) - w(e_m))$$

4. Recursively find a MWA on the contracted graph.

5. Expand the contracted cycle using the edge chosen in this MWA, and remove the "upstream" edge in the cycle, i.e. $x \to v$ in step 3.

Since each search for cycles requires $O(|E|)$ time and there can be at most $|V| - 1$ contractions, the total running time is $O(|V||E|)$. Georgiadis [Geo03] shows that the algorithm finds an MWA for any w with suitably-defined less-than and addition operations on weights.

Here we extend this algorithm to the case of uncertain edge weights, i.e. weight functions $W : E \to (a, b) \in \Re^2$ in which each edge's weight is drawn from the range $[a, b]$. We say a weight function w is *consistent* with range-valued weight function W if $w(e) \in W(e)$ for all $e \in E$. Given a graph and a range-valued weight function W, we show how Edmonds' algorithm can find a partial ordering consistent with the MWAs for all weight functions w consistent with W (written $w \in W$).

A number of authors have previously addressed similar problems. Hutter and Zaffalon [ZH05] address the similar problem of finding "robust edges" in undirected, tree-structured graphical models with range-valued mutual information between nodes, where an edge is robust if it is common to all possible minimum spanning trees. They present $O(n^4)$ exact and $O(n^3)$ approximate algorithms that yield a forest of the common node relations. However, the set of robust edges may be a forest that does not capture all the partial ordering information available in either the original graph or the strongest consistent partial ordering. Because edges represent transitive "parent" relations in our case, we can use this additional ordering information to create a tree including edges not present in the original graph.

Beerenwinkel *et al.* [BRD+04] iteratively find mixtures of distinct trees best capturing a distribution's dependencies. Their problem is a form of clustering with structured labels, where individuals are drawn from one of several, compact classes of directed trees. On the other hand, our data are assumed to originate from a single cluster, and our goal is to recover ordering information consistent with all likely originating trees for the family.

Bagchi *et al.* [BBS06], in a context where computing edge weights is expensive, find an approximate MWA by considering only the k cheapest edges incident to each node. The resulting approximate solution is within a factor of $\frac{k}{k+1}$ of optimal, and is computed in $O(kV^2)$ time. Since in our case edge weights are given, reducing running time is the only reason to restrict the number of input edges considered. Also, it is not clear how their approximate algorithm extends to uncertain weights.

We are interested in solving the following two related problems: **Minimum weight partial ordering:**

Inputs: A directed graph $G = (V, E)$ and a range-valued weight function $W : E \rightarrow (a, b) \in \Re^2$.

Output: A subset $O \subseteq E$ such that O spans V with no cycles, and $u \rightsquigarrow v$ in O iff $u \rightsquigarrow v$ in the MWA for some $w \in W$, and $v \rightsquigarrow u$ for the MWA of no $w \in W$.

Our algorithm for strongest consistent partial ordering is identical to the original Chu-Liu-Edmonds algorithm except for modifications to steps 1 and 3, and a minor change to step 5. First, instead of selecting a single minimum-weight incident edge in step 1, we select all edges with minimum weight less than or equal to the smallest maximum incident edge weight. That is, we choose all incident edges $e = u \rightarrow v$ such that

$$\min W(e) \leq \operatorname*{argmax}_{e'=u'\rightarrow v} \max(W(e'))$$

Similarly, when re-expanding a node in step 5, we choose all analogous edges, and delete *all* of their upstream cycle edges.

For the weight adjustment in step 3, define arithmetic on weights as

$$\begin{aligned}(a, b) - (c, d) &= (a - d, b - c)\\ (a, b) + (c, d) &= (a + c, b + d)\end{aligned}$$

and let $w_{min} = (l_{min}, u_{min})$ where l_{min} and u_{min} are the minimum lower and upper edge weights in the cycle. Then the update is the same as that of the original algorithm, except using w_{min} instead of the weight of the minimum edge.

4.6.2 Incremental Alignment

Section 4.5 discussed the possibility of improving the final tree by realigning sequences in smaller clusters. Since multiple alignments can improve dramatically when aligning fewer, more closely-related sequences, this could improve the final clustering, though perhaps at substantial computational cost. Here we describe a similar approach which combines multiple alignment with clustering, skipping the initial RepBase-guided multiple alignment entirely.

Having to start with a full multiple alignment presents a number of problems. First, large multiple alignments between diverse sequences are often quite poor, with many unnecessary gaps and mistakes. Second, aligning a large number of sequences using a guiding alignment, as we did, may introduce bias into the initial multiple alignment that will color the final results. Third, for *de novo* elements found by algorithms such as RepeatScout [PJP05], no guiding alignment may be available. This would force us to perform a full multiple alignment on all the mobile elements of interest, which would be computationally infeasible for the hundreds of thousands or millions of elements we are trying to cluster.

The intuition behind this approach is that if the input sequences are simply stacked on top of each other naively, some similar sequences will be identical at a number of positions, increasing the mutual information between those positions. Therefore RATS can be used to chop the full input into pieces small enough to be consumed by a real multiple alignment algorithm, such as CLUSTAL W [THG94] or MAFFT [KMKM02], in an acceptable period of time. As more clusters are realigned, the overall multiple alignment and clustering will iteratively improve, converging to a set of clusters, each admitting a good multiple alignment and satisfying RATS's mutual information test.

To extend RATS to handle incremental multiple alignment, we must define the following operations, where a "small cluster" is one with that can be efficiently multiply aligned, a "large cluster" is not, and a "cluster" can be either:

1. To create the initial cluster, the simplest approach is to have all sequences start

at the same position. A slightly more ambitious heuristic, which might create a better starting point, would be to look for one or more seed patterns and place sequences containing those patterns so that one or more of the patterns match.

2. To create a new large cluster after a split is the same as above, except for possible implementation shortcuts.

3. To create a new large cluster by joining two clusters, we globally align their consensus sequences with the Smith-Waterman algorithm [SW81], then use that alignment to merge their sequences.

4. To create a new small cluster, whether from a join or a split, we perform a full multiple alignment.

5. To choose a candidate pair of clusters to join, we choose the two remaining clusters whose consensus sequences have the best global alignment score.

6. To perform a greedy update, the analogous approach would be to move each sequence to the cluster whose consensus sequence had the best global alignment score. However, since this is computationally infeasible, we instead first construct an index over the sequences, then search this index using the cluster consensus sequences. Each sequence is assigned to the cluster whose previous consensus sequence had the highest-scoring match.

This chapter is based on work with Nebojsa Jojic, Noah Zaitlen, and Eleazar Eskin and published in the proceedings of RECOMB 2007.

Chapter 5

Conclusions

In the previous chapters, we have first described two algorithms, IB sequences and SSCC, which demonstrate the flexibility of an information-theoretic approach to complex biological clustering problems. We have shown several ways of incorporating different kinds of prior knowledge and constraints. We have also described RATS, a randomized algorithm which enforces constraints specified by hypothesis tests, and is able to efficiently cluster millions of sequences. We have applied these algorithms to a number of biological datasets, showing that they can be valuable bioinformatics tools for comparing distantly-related proteins, identifying population substructure, and showing the history of mobile elements replication in organisms' genomes. Here we review our work in Sections 5.1, 5.2, and 5.3, then conclude by summarizing our algorithmic and biological contributions.

5.1 IB sequence alignment

Comparing an amino acid sequence with other phylogenetically-related proteins, or *homologs*, is a powerful tool in bioinformatics. Since evolutionarily related have similar biological functions, such a comparison gives insight into which segments of the protein are functionally important, and therefore must be conserved, and which are not, and can mutate without harming the organism. The most common representation for a set of homologous proteins is its *profile*, the sequence of the marginal distributions of amino acids at each position. Algorithms such as BLAST find homologs by aligning an amino acid sequence against a database of other sequences.

However, comparison of crystallized structures has shown that two proteins can be structurally homologous even when their sequences are as different as those of two randomly-chosen proteins. Since crystallizing proteins to find their structure is costly and time-consuming, we need a more sensitive test for detecting distant homologs. To address this problem, Yona and Levitt [YL02] proposed a method of aligning pairs of profiles generated by a sequence-based homology search. They showed that their approach reliably identified homologous proteins too different in sequence to be detected by sequence-based methods. However, representing amino acid sequences as profiles has a number of disadvantages. First, as a sequence of 20-dimensional probability distributions, a profile is difficult to visually interpret. Second, profile alignment is much more computationally expensive than sequence alignment, since it must compute the distance between amino acid distributions at each point rather than simply look up a score. Finally, because profiles are continuous, a profile database cannot take advantage of fast indexing approaches used for discrete data.

We overcome these problems by representing profiles by a small alphabet of representative distributions. To choose these representative distributions with minimal information loss, we find an informationally-optimal clustering created by an algorithm based on Information Bottleneck; the cluster means then form our alphabet of representative distributions. The IB profiles produced by this discretization can be aligned and indexed like ordinary amino acid sequences, using fast table lookups rather than costly distance computations. At the same time, they retain most of the information found in the original profiles, so unlike amino acid sequences, they can be used to detect distant homologs.

To aid human readability, we want our clustering to include the standard amino acid alphabet, so that for example clusters “A” and “a” represent strongly- and weakly-conserved alanine, respectively. To achieve this, we impose two types of constraints on the cluster distributions: First, some cluster conditional distributions should be similar to predefined values, so for example the cluster denoted “A” should represent strongly-conserved alanine. Second, clusters representing strong and weak conservation of the same amino acid should have similar shapes; since they have similar chemical roles, both clusters should show similar amino acid substitution patterns.

We first show that our discretization preserves most of the information in the original profiles, ranging from 84% with 40 clusters to 94% with 500. By comparison, representing profiles by the most-common amino acid at each position preserves 78% of their information. Next, we show that alignments between pairs of IB sequences are similar to those between the profiles they represent. We then show that IB comparison identifies distant homologs nearly as well as profile comparison, a task for which comparing the original amino acid sequences performs no better than chance. Finally, we note that the running time of IB sequence alignment is near-linear in sequence length, where profile alignment is quadratic, reflecting the fact that distance computations dominate alignment time.

We conclude by examining some of the issues involved in creating a BLAST-like seed-indexed database for IB sequences. We find that, because we are searching for such distant matches, a direct translation of that approach is ineffective. We instead propose a related alternative using a more sophisticated heuristic that uses approximate seed matches, and looks for matches in regions of high seed density. This shows that while IB encoding can greatly improve the performance of tasks once requiring full sequence comparison, more work is needed to realize its full potential.

5.2 Shape-constrained clustering

The population substructure problem is the problem of recovering distinct subpopulations of related DNA sequences from a mixture of several subpopulations. Finding population substructure is important to the study of a number of biological problems. One example is the separation of viral strains, where each subpopulation corresponds to a different transmission vector. Another example is genetic association studies, which try to determine the genetic basis of individual diseases or traits, where each subpopulation corresponds to a group of individuals with common ancestry. Identifying population substructure is crucial because subpopulations differ in multiple traits. If one looks for the genetic basis of a single trait in a population with substructure, it will appear that the genetic bases of *all* traits differing between the two subpopulations affect the trait of interest.

We assume a model in which subpopulations, or clusters, are created by first choosing a number of cluster consensus sequences, then generating sequences within each cluster by random mutation from these consensus sequences. When the consensus sequences are far apart in sequence space relative to the mutation rate, a sequence will be closer to sequences in its own cluster than to those in other clusters. In this case, clusters can be easily identified by distance-based clustering algorithms like agglomerative linkage clustering or k-means. However, if the consensus sequences are close relative to the rate of mutation, many sequences will be closer to those in other clusters than to those in their own. In this case distance-based methods will recover spurious clusters reflecting local irregularities rather than the true clusters.

To address this problem we present Sequential Shape-Constrained Clustering (SSCC) an algorithm to recover highly overlapping clusters by constraining their shapes. If each mutation at each position is independent, then all sequence positions should be conditionally independent given the cluster, and any correlation is evidence of substructure. We eliminate this substructure by penalizing this correlation. Specifically, our score function minimizes both cluster entropy and the pairwise mutual information between positions within each cluster, with a parameter controlling the trade-off between the two.

We find local a local optimum for a given number of clusters by sequential updates, moving individual sequences between clusters as long as doing so improves the score. Since each move improves the global score, this process is guaranteed to converge. Performing these updates requires recomputing the score that would result from moving each individual to each other cluster. We avoid this expensive computation by efficiently computing the change in score for each move without recomputing the overall score.

Sscc infers the number of clusters by top-down splitting guided by the score's mutual information term. First, we choose the position i with the greatest total mutual information with all other positions. We then split the cluster in two, with all sequences having the most common sequence value at i in one cluster, and all the rest in the other. Because these splits create unbalanced clusters, we optimize via sequential updates after every split. When no position in any cluster has a large enough mutual information with other positions, or when no cluster is above the minimum size, the algorithm terminates.

Comparing SSCC to STRUCTURE [PSD00], a state-of-the-art algorithm for population substructure, on simulated data, we find that SSCC outperforms STRUCTURE as the mutation rate increases relative to the distance between cluster consensus sequences. On human SNP data, we find that STRUCTURE and SSCC perform equally well. On human *Alu* sequences, both algorithms recover the cluster consensus sequences, but SSCC correctly assigns more individuals. Finally, we apply SSCC to a collection of HIV polymerase sequences, and show that it often recovers both viral clade and continent of origin. Sscc thus represents a flexible and powerful approach for population substructure identification.

5.3 Randomized clustering for mobile elements

Mobile elements are small, repeated sequences that make up nearly 50% of our DNA, conventionally divided into families based on their length and method of replication. While most individual elements remain inactive, active elements copy and reinsert themselves in our genome through one of several processes. Mobile elements remain active even today: there are active *Alu* elements in the modern human genome whose effect can be observed in population-specific *Alu* insertions. Past research has suggested that mobile elements play a role in a variety of genetic process, including segmental duplication, chromosome rearrangement, genome growth, protein evolution, and relocation of transcription factor binding sites. They have also been implicated as factors in a number of human diseases. Both their prevalence and their many genetic effects make them a natural object of study.

One important component of that study is determining the ancestry and age of insertion of mobile elements. Current phylogenetic algorithms are ill-suited to the task for three reasons: First, as in the population substructure problem described in the previous section and in Chapter 3, reconstructing mobile element history requires separating overlapping clusters subject to certain constraints. Second, neither SSCC nor traditional phylogenetic algorithms, most of which are quadratic or worse in the number of sequences, scale to the millions of mobile elements present in the human genome. Finally, mobile element phylogeny has the unusual characteristic that only a few elements ever become active, producing phylogenies where only a few nodes have offspring.

We present RATS, a novel clustering algorithm to recover mobile element phylogeny. Like SSCC, RATS finds a clustering obeying certain constraints on the correlation structure of the cluster conditional distributions. However, rather than minimizing a trade-off between cluster entropy and mutual information between pairs of positions, we ensure that clusters satisfy a statistical test based on this mutual information. Specifically, for each pair of positions (i, j), the mutual information between positions i and j within a cluster must occur between uncorrelated random variables with at least a certain probability.

This random sampling leads naturally to a top-down clustering algorithm. If a cluster fails the statistical test above at some pair of positions (i, j), we split it to reduce mutual information between positions i and j by placing all sequences with the most common values at i and j in one cluster, and the rest in another. Rather than testing all pairs of positions, we randomly sample enough pairs to detect correlation with high probability. If no such position exists, then the cluster is statistically justified or valid.

Recursively applying this splitting criterion yields a set of valid clusters. However, a poor sequence of splits may create sets of clusters that, when merged, would create a larger valid cluster. Such sets can be found by merging sets of clusters and performing our statistical test on the result. However, testing all sets of clusters is computationally impractical, we only test pairs of clusters with similar consensus sequences. By alternating recursive splits with tests for clusters that can be joined, we obtain a set of valid clusters in which no neighboring pair can be merged to form a valid cluster. To create a phylogeny from these clusters, we first order them by cluster entropy, which increases with age. A cluster's parent is then the older cluster whose conditional distribution is most likely to have generated that cluster's consensus sequence.

We first apply RATS to 2.3 million *Alu* and 24 000 complete L1 elements found in the available complete primate genomes. While RepBase [Jur04] lists only 35 *Alu* and 102 L1 subfamilies, we are able to identify 1519 and 134. Our result supports Price *et al.*'s [PEP04] earlier finding that there have been many more active *Alu* elements than were previously identified, and suggests that the same may be true for other families of mobile elements.

We validate our *Alu* clustering by showing that it identifies known young *Alu* families. Mills *et al.* [MBI$^+$06] identified *Alu* families that have been active since human and chimpanzee diverged through comparative genomics. We find that in any cluster we identify containing a significant number of elements from one of Mills' families, over 90% of the elements in that cluster are from that family. Thus, our method is sensitive enough to accurately identify even these young clusters, which should be hardest for it to distinguish due to their low sequence divergence from related young families.

Since a 1500-node phylogeny is difficult to interpret, we instead look at our clusters' age distribution, where ages are inferred from average sequence divergence using a standard molecular clock. Our results agree with those of Kim *et al.* [KHR04], who find a peak of *Alu* activity near the divergence of old- and new-world monkeys. Thanks to our higher-resolution phylogeny, we are able to detect three major peaks in *Alu* activity, corresponding to activity of the *Alu J*, *Alu S*, and *Alu Y* families. These three peaks coincide roughly with the divergences of strepsirrhins, new-world, and old-world monkeys from the hominid ancestor. These bursts of activity in successive *Alu* families may reflect repeated suppression of particular *Alu* families' replication by a mechanism like that described by Malone and Hannon [MH09], followed by resurgence when a new family becomes active.

We also apply RATS to 380 000 SINE elements from 61 species in the ENCODE database, finding 308 mobile element subfamilies where RepBase contains 197. We validate the phylogeny constructed from these subfamilies by showing that it reflects known aspects of SINE phylogeny and evolutionary events (see Figure 4.10 and Section 4.4.4). First, the known *Alu* phylogeny is correctly identified as a subtree, with the sequence divergence within clusters consistent with current estimates of *Alu* family ages. Second, the *Alu* subtree reflects known phylogenetic relationships between primate clades. Third, MIR elements specific to marsupials and monotremes are placed in their own subtrees, both of which are separate from the mammalian elements.

Our results are promising, but there remains much more analysis to be done of this new, high-resolution view of mobile element history. For example, mobile element insertion can be related to other genomic processes like segmental duplication, and mobile element insertion hotspots can identified. RATS can also be improved in a number of ways, including more robust methods of constructing phylogenies and better ways of interpreting them. The requirement that input sequences be aligned is also a weakness, which can be overcome by combining clustering with incremental multiple alignment.

5.4 Algorithmic contributions

Our first contribution was an extension of Information Bottleneck for clustering amino acid distributions with minimal information loss. This approach allows the incorporation of distributional constraints, including priors on class conditional distributions and correlation between these distributions. The algorithm, based on inference in an underlying graphical model, is flexible enough to incorporate both new information and other types of constraints. For example, our model can be extended to incorporate structural information as input, or to account for local sequence correlation by computing a discretization over k-tuples of adjacent sequence positions.

We have presented two related sequence clustering algorithms to address population substructure. The first, SSCC, iteratively minimizes an objective function that trades off between maximum entropy and a set of information-theoretic constraints. This objective extends that of Information Bottleneck to flexibly allow constraints on the class conditional distributions. It does so by including additional penalty terms, each related to a likelihood ratio test. For example, the added penalty in Eq. (3.3), added to reduce correlation between positions within each cluster, is

$$I(X_i; X_j|C = c) = E\left[\log p(X_i, X_j)\right] - E\left[\log p(X_i)p(X_j)\right]$$

for a single cluster c and pair of positions i and j. Given an appropriate threshold, this is simply a (log) likelihood ratio test (LRT) between two nested hypotheses differing in whether X_i and X_j are correlated. Thus the Lagrangian form of the objective replaces a LRT threshold k with a tradeoff parameter β. The structure learning extension adds a step to heuristically reduce this penalty by adding more clusters, effectively reducing the value of the associated LRT.

The second algorithm, RATS, directly eliminates rejections of the null hypothesis of uncorrelated sequence positions by splitting clusters to reduce correlation between positions. This has the added effect of reducing within-cluster entropy. To make this process more efficient, RATS performs only enough of the possible statistical tests to find correlations with high probability.

The two algorithms are similar in that both use statistical tests to constrain the posterior distribution inferred from nonparametric clustering. Both algorithms also perform structure learning in a top-down manner, based on heuristics to directly reduce the value of the same statistical test. However, they differ in three ways. First, while SSCC is a forward-only algorithm, RATS must perform both forward (split) and backward (join) steps because its large forward steps will not always find good solutions. Second, SSCC finds a maximum-entropy solution subject to the given constraints. RATS, on the other hand, approximates a maximum likelihood solution that does not violate the given statistical test, but that will do so if locally modified. Finally, SSCC enforces a smooth penalty term based on the test, while RATS probabilistically imposes a hard threshold of statistical significance.

The comparison between SSCC and RATS is somewhat analogous to that between model-scoring and constraint-based algorithms for causal inference. Model-scoring algorithms like Suzuki's MDL-based algorithm [Suz93] assign high scores to models which assign high likelihood to the observed data and have low complexity, and search for the highest-scoring model. While SSCC does not use a complexity penalty term, adding one would be the natural way to control model size. In contrast, constraint-based algorithms like IC [VP90] infer the generative process by searching for constraints, such as conditional independence. Rats performs a similar series of tests, but since we assume that the true generative model has no conditional dependencies within clusters, we are searching for dependencies rather than independencies.

We also see some of the same trade-offs between model-scoring and constraint-based causal algorithms as we saw between SSCC and RATS. Constraint-based algorithms are subject to cascading errors from performing many sequential independence tests. Rats is similarly subject to poor early splits, whose effects the join tests are intended to combat. Similarly, defining an appropriate complexity penalty in model-scoring algorithms is difficult. As we saw in Section 4.3.2, standard complexity penalties can be inappropriate when the family of models being explored does not exactly match the true, complex generative model. While significance tests made a complexity penalty unnecessary for RATS, the choice of such a penalty for SSCC would be difficult and problem-dependent.

5.5 Biological contributions

IB sequences provide an encoding of amino acid profiles that minimizes information loss, corresponds to currently-used discrete representation of proteins, and incorporates known chemical similarities between amino acids. The resulting representation is useful both as a compact visual representation of profile data, and as encoding for efficient approximate profile-profile alignment and database indexing. We have shown that such alignment is almost as sensitive as true profile-profile alignment in finding distant homologs, but is much more computationally efficient.

Sscc find population substructure where genetic variation within each population obeys certain constraints, even in cases where variation within the groups is much greater than variation between them. Such cases include human SNPs, viral RNA, and DNA mobile elements. In the last two cases, SSCC significantly outperforms current population substructure inference algorithms.

Rats makes it possible to create clusterings and phylogenies of entire mobile element families composed of millions of elements found in the complete genomes of one or more species. These clusters identify many more mobile element subfamilies than the currently-accepted classification. These clusters are consistent with known large-scale repeat activity: a phylogeny of SINE elements across multiple eutherian species constructed by our method is consistent in many respects with the species phylogeny. The results are also consistent with known small-scale repeat activity: the subfamilies of *Alu* identified by RATS in the human and chimpanzee genomes reflect known recent *Alu* activity.

Examination of the ages of mobile element subfamilies shows a correlation with periods of rapid speciation. As more species' complete genomes become available, clustering of large numbers of mobile elements will be useful in two ways: First, by looking at the ages of mobile element families shared by and unique to two related species, it is possible to estimate speciation times. Second, since more precise clustering will become possible as more elements are available, estimates of each element's insertion age will improve. These ages can in turn be used to estimate the times of other chromosomal events such as deletions, inversions, and duplications, as well as to relate the mobile element replication process itself to these other processes.

Appendix A

Other contributions: transducers for haplotype phasing

Recent high-throughput genotyping technologies promise a wealth of new information, but the data they supply provide an ambiguous and incomplete view of the true genetic state. In particular, the problems of inferring haplotypes, single-strand deletions, and population-wide recombination patterns from genotype data are both challenging and well-studied. We describe a straightforward maximum likelihood model addressing all of these questions. While solving the model directly takes exponential time, we show that an equivalent weighted finite state transducer model can be solved efficiently for large samples over long genetic regions.

A.1 Introduction and related work

Recent advances in high throughput single nucleotide polymorphism (SNP) genotyping techniques [MMea04, GSL$^+$05] have made possible large scale surveys of human genetic variation. Projects such as the HapMap project [AC05] and numerous large scale association studies [RM96] have revolutionized our approach to understanding human evolution and the genetic basis of disease. Analysis of genotype data presents many computational challenges including haplotype phasing [SSD01, ES95, MDea06], estimating the haplotype frequency distribution in the population [HH06], estimating the recombination rates of haplotypes [MMH$^+$04], and identifying single-chromosome deletions [CRE07].

The phasing problem is the following: Genotyping reports the allele at each SNP on both copies of the chromosome (the *genotype*), but not which allele came from which chromosome. Consider a SNP where there are two common alleles, A and G. There are four possible cases for the haplotype: If the two chromosomes are *homozygous* (i.e. they contain the same allele), the genotype is either A or G and we can infer that the base appears in both chromosomes. If, however, they are *heterozygous* (i.e. they contain either (A, G) or (G, A)), the genotype is H, and we cannot tell which allele contains which haplotype. This ambiguity is significant when examining haplotypes across multiple sites. For example, an individual with genotype $AGHH$ has either haplotypes $AGAA$ and $AGGG$, or $AGAG$ and $AGGA$.

Two landmark 2001 studies on human haplotype structure [DRS+01, PCea01] observed that in any local region of the genome, a small number of haplotypes account for a majority the haplotypes observed in the population. Both of these studies explained this behavior by positing that the human genome contains recombination hotspots which divide it into low-recombination "blocks of limited diversity." However, several newer studies have questioned this assumption and proposed instead that recombination rates are variable in the genome and that the apparent block structure is simply an artifact of ancient recombination events [MMH+04]. These studies point to a model of variable recombination rates as the right model to capture human haplotype structure.

Despite the numerous methods proposed for analyzing haplotype and genotype data, it is an open question which model best captures the structure of human haplotypes. Most state of the art haplotype phasing methods, including HAP [HE04], PHASE [SSD01], PL-EM [NQXL02] and GERBIL [KS05], assume relatively frequent recombination to predict haplotypes: they infer sparse haplotype probability distributions in putative haplotype blocks, then combine these local predictions to generate haplotype predictions for longer regions. Conversely, methods for inferring recombination rates usually assume that we have access to the true haplotypes, and infer from them the recombination rates and sites [MMH+04]. Several recent methods assume that each individual's haplotypes are a recombination of a small number of "ancestral" haplotypes [RKMU05].

We propose a new model for analysis of haplotype and genotype data which jointly estimates haplotype phase and recombination rate. Our model applies techniques long studied in the speech decoding and natural language processing communities [MPR02]. We revisit one of the first and most natural maximum likelihood models presented for haplotype estimation, and extend the traditional Expectation-Maximization (EM) algorithm to estimate the haplotype distribution in every local region. While the model is straightforward to derive, estimating its parameters is prohibitively expensive: analyzing a region of L SNPs, the standard EM algorithm has complexity of $O(2^{2L})$ without recombinations and $O(2^{4L})$ with recombinations. We show that the model can be realized using weighted finite state transducers, and that inference on these transducers is tractable even for large samples and long genomic regions.

A.2 Methods

We first describe the likelihood function for the basic haplotype phasing problem, and outline a straightforward EM optimization method. We then describe how to reformulate this method in terms of efficient operations on finite state transducers. To illustrate how the basic transducer model can be extended to handle more complex models, we briefly describe transducers for genotype errors, genotype information from father-child-mother trios, and recombination between ancestral haplotypes.

A.2.1 The haplotype phasing problem

Let $H = \{0, 1\}^L$ be the set of possible haplotypes of length L, with 0 and 1 representing common and rare alleles. Let $G = \{g_1, \ldots, g_N\} \in \{0, 1, 2\}^L$ be a set of N genotypes corresponding to pairs of haplotypes from H, where 0, 1, and 2 correspond to homozygous common $((0, 0))$, homozygous rare $((1, 1))$, and heterozygous $((0, 1)$ or $(1, 0))$ genotypes at each position. Our goal is to find the haplotype distribution

$p(H)$ maximizing

$$\begin{aligned}\mathcal{L}(H|G) &= \prod_i \sum_{h_1,h_2\in H} p(h_1,h_2|g_i) \qquad (A.1)\\ &\sim \prod_i \sum_{h_1,h_2\in H} p(h_1)p(h_2)\prod_j C(g_i^j,h_1^j,h_2^j)\end{aligned}$$

where $C(g^j,h_1^j,h_2^j)$ is an indicator function that is 1 when h_1^j and h_2^j are compatible with g^j according to the definition of G above, and 0 otherwise. As a convenient shorthand, $C(g,h_1,h_2)=1$ when $C(g^j,h_1^j,h_2^j)=1$ for all j.

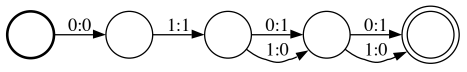
A straightforward EM solution to this problem would maximize $\mathcal{L}(H|G)$ by alternately solving the following equations:

$$\begin{aligned}p^{t+1}(h_1,h_2|g_i) &= \frac{C(g_i,h_1,h_2)p^t(h_1)p^t(h_2)}{\sum_{h_1',h_2'\in H} C(g_i,h_1',h_2')p^t(h_1')p^t(h_2')} \qquad (A.2)\\ p^{t+1}(h) &= \frac{1}{2N}\sum_{g_i}\sum_{h'\in H}\left(p^{t+1}(h,h'|g_i)+p^{t+1}(h',h|g_i)\right) \qquad (A.3)\end{aligned}$$

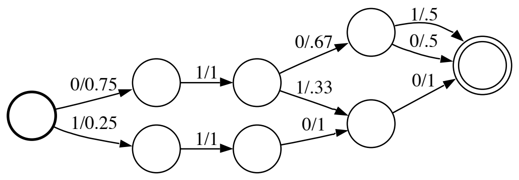
Since there are 2^L haplotypes of length L, this approach requires $O(2^{2L})$ time per iteration, making it impractical for large L.

A.2.2 Basic haplotype transducers

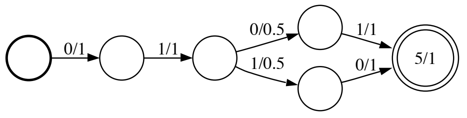
We propose an alternative approach using weighted finite state transducers (WFSTs) to efficiently represent the entire 2^L-haplotype distribution. A WFST is a finite state automaton where each transition is given a weight and a pair of symbols, one input and one output. It represents a weighted mapping from its inputs to its outputs, and its domain is the set of input strings corresponding to series of transitions leading from the initial state to an accepting state. A weighted finite state acceptor (WFSA) is a WFST representing the identity mapping. New WFSTs can be formed by composition and union of existing ones, and equivalent ones can be created by various "optimizations" such as determinization and elimination of null transitions and redundant edges. WFSTs have long been used in speech and language processing to compose models operating at different levels (e.g. phoneme, word, and phrase) into efficient recognition systems [MPR02]. For a fuller treatment, please see Lothaire [Lot05].



(a) The genotype transducer representing `0122`.



(c) The haplotype acceptor representing the distribution $\{\texttt{1100}, \texttt{0100}, \texttt{0101}, \texttt{0110}\}$



(b) The haplotype acceptor resulting from the composition $H \circ G \circ H$ of transducers in Figures A.1(a) and A.1(c), corresponding to the distribution $\{\texttt{0101}, \texttt{0110}\}$.

Figure A.1: Example genotype and haplotype transducers. "`X/w`" represents input/output `X` with weight `w`.

WFSTs are particularly well-suited to distributions over sequences in which most of the probability mass is concentrated on a polynomial-size subset, because they represent only the non-zero-probability subset, and represent many repeated substructures only once. Haplotype distributions have highly concentrated probability mass for two reasons: First, a population heterozygous at k of L loci can have at most 2^k distinct haplotypes, and in many samples the individuals are homozygous at most locations. Second, any population of N individuals can have at most $2N$ distinct haplotypes, and usually $2N \ll 2^L$.

While many transducer operations can be exponential in the worst case, they are linear in many common cases, including those encountered in haplotype phasing. In particular, since a transducer representing $2N$ haplotypes of length L cannot have more than $2N(L+1)$ states and each state besides the first cannot have more than 2 outgoing edges, operations on the distribution over L-position haplotypes of N individuals can be performed in $O(NL)$. This makes it possible to directly optimize equation (A.1) over thousands of positions. Furthermore, transducer composition provides an elegant way to, from independently-derived base models, build complex models incorporating phylogeny, measurement error, and recombination among an unknown number of ancestral haplotypes.

We represent a genotype's compatibility function as a transducer from haplotypes to haplotypes (from $\{0,1\}$ to $\{0,1\}$); figure A.1(a) shows the transducer for the genotype `012`. A haplotype is represented as an acceptor over $\{0,1\}$, and the haplotype distribution $p(H)$ as union of weighted acceptors. Figure A.1(c) shows the acceptor representing a uniform mix of the haplotypes `1100`, `0100`, `0101`, and `0110`. To compute the unnormalized distribution $C(g_i, h_1, h_2)p(h_1)p(h_2)$ for all h_1 and h_2 and some particular g_i, we compose g_i's transducer G_i with the haplotype distribution transducer H on either side. We would ordinarily compute $p(h_1, h_2|g_i)$ by dividing the unnormalized distribution by its sum over all 2^{2L} values of h_1 and h_2. With the transducer representation we can perform this normalization (denoted $\mathcal{N}$) much more efficiently by "pushing" the edge weights so the outgoing edge weights at each state sum to 1, then remove any excess weight pushed to the final state.

With this representation, the iteration steps described in equations (A.2) and (A.3) are represented as the following transducer operations:

$$P_i^{t+1} = \mathcal{N}(H^t \circ G_i \circ H^t) \tag{A.4}$$

$$H^{t+1} = \mathcal{N}\left(\mathcal{O}\left(\bigcup_i P_i^{t+1}\right)\right) \tag{A.5}$$

Note that the left and right haplotype distributions are equivalent, so only the output ($\mathcal{O}$) or input ($\mathcal{I}$) projection is needed.

A.2.3 Genotype errors

The simplest extension we consider is handling genotype errors. These usually appear as positions with two true but unknown alleles, rather than positions with incorrectly-identified, truly missing, or partially-known alleles. Genotype errors are relatively common in most haplotype data: about 10% of positions in the HapMap CEPH trios contain missing data. Here we assume that the probability of error is independent of the genotype. To permit these errors, we extend G to range over $\{0,1,2,3\}^L$, where 3 represents a genotyping error. We also extend C so that $C(3, h_1, h_2) = 1$ for all h_1 and h_2, since any pair of haplotypes is compatible with a genotype error. The genotype transducer for 3 has weight-one transitions for all input/output pairs.

A.2.4 Genotype trios

Because genotypes collected from father-mother-child trios are redundant – the parents each transmit a haplotype to the child – it is easier to infer haplotype characteristics such as phase and heterozygous deletion from trio data than from the genotypes of a population of unrelated individuals. Many haplotyping experiments therefore collect trio data to support such inference. To model trio data, we define a compatibility function C_3 reflecting inheritance constraints among the four haplotypes. Let f_i, m_i, and c_i be the father's, mother's, and child's genotypes, and h_{ft}, h_{fu}, h_{mt}, and h_{mu} be the father's and mother's transmitted and untransmitted haplotypes. Then

$$C_3\left(f_i, m_i, c_i, h_{ft}, h_{fu}, h_{mt}, h_{mu}\right) = C(f_i, h_{ft}, h_{fu}) \wedge C(c_i, h_{ft}, h_{mt}) \wedge C(m_i, h_{mt}, h_{mu})$$

and equations (A.1–A.3) must be extended over the joint distribution of h_{ft}, h_{fu}, h_{mt}, and h_{mu}. Since the sums now range over 2^{4L} possible haplotype quartets, a straightforward solution becomes impractical even for modest L.

When the transducer model is extended to haplotype trios, equations (A.4) and (A.5) become

$$P_i^{t+1} = \mathcal{N}\left(H^t \circ F_i \circ H^t \circ C_i \circ H^t \circ M_i \circ H^t\right) \tag{A.6}$$

$$H^{t+1} = \mathcal{N}\left(\bigcup_i \left(\mathcal{O}(P_i^{t+1}) \cup \mathcal{I}(P_i^{t+1}) \cup \mathcal{O}(\mathcal{I}(P_i^{t+1}) \circ F_i) \cup \mathcal{I}(M_i \circ \mathcal{O}(P_i^{t+1}))\right)\right) \tag{A.7}$$

Note that since the posterior distributions of the transmitted and untransmitted haplotypes are not the same, both input and output projections must be included in equation (A.7).

A.2.5 Ancestral recombination

As mentioned in section A.2.1, recombination between haplotypes reduces long-range correlation between alleles, a property exploited by many haplotype phasing methods to avoid exponential running times. Recombination also creates dependencies in the haplotype distribution: For example, the probability of observing a haplotype `ABC` is increased by observing the blockwise-similar haplotypes `AXC` and

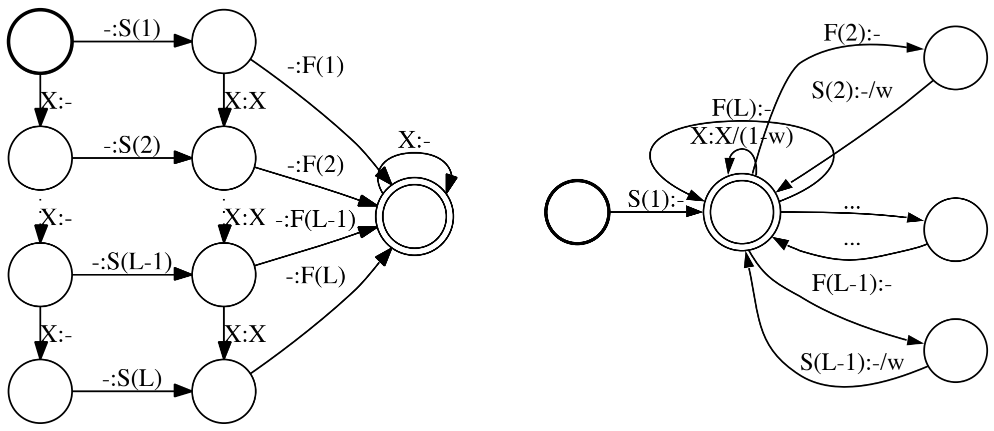


Figure A.2: Transducers for haplotype recombination. "`X`" is any haplotype symbol; "`S(i)`" and "`F(i)`" are unique recombination marks; "`-`" is an ϵ-transition; "`w`" is the probability of recombination. Left: A transducer producing all substrings labeled by start- and end-points. Right: A transducer tiling substrings.

`XBC`. Since ancient recombinations are more likely to be shared by a large fraction of the present population, the effect of haplotype recombination in a modern sample can be approximated as recombination between an unknown number of *ancestral haplotypes*.

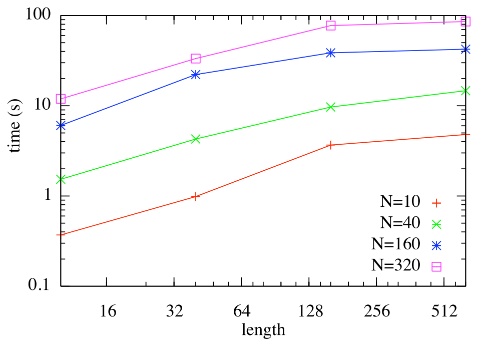
With transducers we can model the effect of recombination directly, without any assumptions about the number of ancestral haplotypes, the locations of recombination breakpoints, or the correlation of breakpoint locations between individuals. To do so, we represent recombination using two transducers: T_{sub} (figure A.2(a)), producing all subsequences $H^{i:j}$ of the full haplotype distribution H, and T_{tile} (figure A.2(b)), forming tilings of these haplotype distributions covering $H = H^{1:L}$. Equation (A.5) then becomes

$$H^{t+1} = \mathcal{N}\left(\bigcup_i \mathcal{O}\left(\mathcal{K}\left(P_i^{t+1} \circ T_{sub}\right) \circ T_{tile}\right)\right) \qquad \text{(A.8)}$$

where $\mathcal{K}$ denotes Kleene closure. Note that updates to the haplotype distribution reflecting recombination and observed genotypes are independent.

T_{sub} and T_{tile} correspond to the following likelihood function:

$$\mathcal{L}(H|G) = \prod_i \sum_{r_1,r_2 \in R} p(r_1)p(r_2) \sum_{h_1,h_2 \in H} p(h_1|r_1)p(h_2|r_2)C(g_i,h_1,h_2) \qquad \text{(A.9)}$$



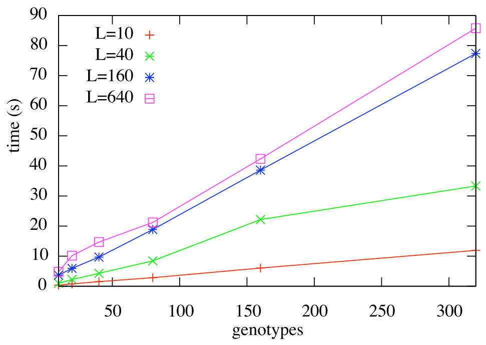


Figure A.3: Runtime versus L (left) and N (right) for the basic transducer.

where $R = \{r | r \in \mathcal{P}([1, L]), 1 \in r, L \in r\}$, $p(r) = (1 - w)^{|r|}$ for recombination probability $0 < w \leq 1$, and

$$p(h|r) = \prod_{i=1}^{|r|-1} \sum_{h_2 \in H} \left[h_2^{1:r_i - 1} = h^{1:r_i - 1} \wedge h_2^{r_{i+1}:L} = h^{r_{i+1}:L} \right] p(h_2)$$

A.3 Experiments

We implemented our models using the AT&T FSM Library [MPR98], which supplies command-line tools performing the fundamental transducer operations. We first assessed the model's accuracy and performance on synthetic data, then compared it to representative haplotyping programs on real trio data from HapMap. All experiments were run on a 1.86 GHz Intel Xeon with 8GB memory.

Figure A.3 shows runtime versus N and L for the basic (section A.2.2) transducer over genotypes created from 4 random haplotypes without missing data. The runtime appears linear in N but sublinear in L because, for this particular dataset and implementation, the most expensive operation is the union and subsequent simplification of the N individual haplotype distributions.

We also applied the basic model, this time with missing data, to the CEPH trio data from HapMap. We assessed its accuracy by comparing our most likely haplotypes to those inferred by HAP [HE04], a state-of-the-art haplotype phasing program, within each HAP-reported recombination block. For all blocks of length greater than four, our method's 4 most probable haplotypes matched those reported by HAP with at most one non-matching position among them.

A.4 Conclusion and future work

We have shown how finite state transducers are useful for haplotype phasing, and that they scale efficiently to long sequences and large numbers of individuals. Also, because independently-developed submodels can be composed automatically and efficiently, transducers can be used to model many effects simultaneously. Genotype data contains many such interacting effects. Some for which transducer models seem appropriate are phylogenies with multiple siblings, copy number variation, haplotype inversion, and raw copy number data. To handle these new factors, the models discussed so far must be extended to handle continuous inputs, tied weights, and dependency structures with complex n-ary transducers.

Bibliography

[AC05] D Altshuler and International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437(7063):1299–320, 2005.

[AGM+90] SF Altschul, W Gish, W Miller, EW Myers, and DJ Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–10, October 1990.

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI–BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25:3389–3402, 1997.

[BAW+05] A Bairoch, R Apweiler, CH Wu, WC Barker, BB, S Ferro, E Gasteiger, H Huang, R Lopez, M Magrane, MJ Martin, DA Natale, C O'Donovan, N Redaschi, and LL Yeh. The universal protein resource (UniProt). *Nucl. Acids Res.*, 33(suppl. 1):D154–159, 2005.

[BBS06] A Bagchi, A Bhargava, and T Suel. Approximate maximum weighted branchings. *Information Processing Letters*, February 2006.

[BD02] MA Batzer and PL Deininger. Alu repeats and the human genomic diversity. *Nature rev. genet.*, 3:370–9, May 2002.

[BK04] N Bannert and R Kurth. Retroelements and the human genome: new perspectives on an old relation. *PNAS*, 101(suppl. 2):14572–9, 2004.

[Bla72] RE Blahut. Computation of channel capacity and rate distortion function. *IEEE Trans Inform Theory*, IT-18:460–73, 1972.

[BLE03] JA Bailey, G Liu, and EE Eichler. An Alu transposition model for the origin and expansion of human segmental duplications. *Am J Hum Genet*, 73:823–34, 2003.

[BLV+08] G Bourque, B Leong, VB Vega, X Chen, YL Lee, KG Srinivasan, JL Chew, Y Ruan, CL Wei, HH Ng, and ET Liu. Evolution of the mammalian transcription factor binding repertoire via transposable elements. *Genome Res*, 18(11):1752–1762, 2008.

[BRD$^+$04] N Beerenwinkel, J Rahnenf uhrer, M D aumer, D Homann, R Kaiser, J Selbig, and T Lengauer. Learning multiple evolutionary pathways from cross-sectional data. In *Proc. RECOMB*, pages 36–44, 2004.

[CBC$^+$98] WM Chu, R Ballard, BW Carpick, BR Williams, and CW Schmid. Potential Alu function: regulation of the activity ofdouble-stranded RNA-activated kinase PKR. *Mol Cell Biol*, 17:804–12, 1998.

[CEKN04] CS Carlson, MA Eberle, L Kruglyak, and DA Nickerson. Mapping complex disease loci in whole-genome association studies. *Nature*, 429(6990):446–52, May 2004.

[CH97] DM Chickering and D Heckerman. Efficient approximations for the marginal likelihood of bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212, 1997.

[CHB04] R Cordaux, DJ Hedges, and MA Batzer. Retrotransposition of Alu elements: how many sources? *Trends Genet.*, 20(10):464–7, October 2004.

[CHCB04] GE Crooks, G Hon, JM Chandonia, and SE Brenner. WebLogo: a sequence logo generator. *Genome Research*, 14(6):1188–90, 2004.

[CL65] YJ Chu and TH Liu. On the shortest arborescence of a directed graph. *Sci. Sinica*, 14:1396–1400, 1965.

[CRE07] E Corona, B Raphael, and E Eskin. Identification of deletion polymorphisms from haplotypes. In *Proc. RECOMB*, volume 4453 of *LNBI*, pages 354–365, 2007., 2007.

[CT91] TM Cover and JA Thomas. *The elements of information theory.* Plenum Press, New York, 1991.

[CT02] G Chechik and N Tishby. Extracting relevant structures with side information. In *Proc. NIPS*, Cambridge, MA, 2002. MIT Press.

[Daw04] R Dawkins. *The Ancestor's Tale.* Houghton Mifflin, Boston, 2004.

[DEH03] M Dewannieux, C Esnault, and T Heidemann. LINE-mediated retrotransposition of marked Alu sequences. *Nat Genet*, 35:41–8, 2003.

[DRS$^+$01] MJ Daly, JD Rioux, SF Schaffner, TJ Hudson, and ES Lander. High-resolution haplotype structure in the human genome. *Nature Genet.*, 29(2):229–32, 2001.

[Edm67] J Edmonds. Optimum branchings. *J Res. NBS*, 71B:233–40, 1967.

[ES95] L Excoffier and M Slatkin. Maximum-likelihood estimation of molecular haplotype frequenci es in a diploid population. *Mol Biol Evol*, 12(5):921–7, Sept 1995.

[Fel04] J Felsenstein. PHYLIP (phylogeny inference package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle, 2004.

[FMST01] N Friedman, O Mosenzon, N Slonim, and N Tishby. Multivariate information bottleneck. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001)*, pages 152–161, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[FNPP01] N Friedman, M Ninio, I Pe'er, and T Pupko. A structural EM algorithm for phylogentic inference. *J. Comp. Biol.*, 2001.

[GDHM05] B Goebel, Z Dawy, J Hagenauer, and JC Mueller. An approximation to the distribution of finite sample size mutual information estimates. In *IEEE International Conference on Communications (ICC)*, Seoul, South Korea, May 2005.

[Geo03] L Georgiadis. Arborescence optimization problems solvable by edmondsalgorithm. *Theor. Comp. Sci.*, 1–3(301):427–37, 2003.

[GH03] S Goldsmith-Fischman and B Honig. Structural genomics: computational methods for structure analysis. *Protein Sci*, 12(9):1813–21, 2003.

[GL00] N Gilbert and D Labuda. Evolutionary inventions and continuity of CORE-SINEs in mammals. *J. Mol. Biol.*, 298:365–77, 2000.

[GSL$^+$05] K Gunderson, F Steemers, G Lee, L Mendoza, and A Chee. A genome-wide scalable SNP genotyping assay using microarray technology. *Nat. Genet.*, 37:549–54, 2005.

[HA85] L Hubert and P Arabie. Comparing partitions. *J Classification*, pages 193–218, 1985.

[Har75] JA Hartigan. *Clustering*. Wiley, 1975.

[HB05] DJ Hedges and MA Batzer. From the margins of the genome: mobile elements shape primate evolution. *BioEssays*, 27(8):785–94, 2005.

[HE04] E Halperin and E Eskin. Haplotype reconstruction from genotype data using imperfect phylogeny. *Bioinformatics*, 20(12):1842–9, 2004.

[HH06] E Halperin and E Hazan. HAPLOFREQ–estimating haplotype frequencies efficiently. *J Comp Bio*, 13:481–500, Mar 2006. 10.1089/cmb.2006.13.481.

[HSN$^+$05] DA Hinds, LL Stuve, GB Nilsen, E Halperin, E Eskin, DG Ballinger, KA Frazer, and DR Cox. Whole genome patterns of common DNA variation in diverse human populations. *Science*, 307:1072–9, 2005.

[HXW$^+$05] K Han, J Xing, H Wang, DJ Hedges, RK Garber, R Cordaux, and MA Batzer. Under the genomic radar: the stealth model of Alu amplification. *Genome Res.*, 15:655–64, 2005.

[Int01] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[JMF99] AK Jain, MN Murty, and PJ Flynn. Data clustering: a review. *ACM computing surveys*, 31(3):265–323, 1999.

[JTZ04] D Jiang, C Tang, and A Zhang. Cluster analysis for gene expression data: A survey. *IEEE Trans. Knowledge Data Eng.*, 16(11):1370–86, 2004.

[Jur00] J Jurka. Repbase update: A database and an electronic journal of repetitive elements. *Trends Genet*, 9:418–20, 2000.

[Jur04] J Jurka. Evolutionary impact of human Alu repetitive elements. *Current Opinion in Genetics & Development*, 14(6):603–8, December 2004.

[KFF$^+$02] CL Kuiken, B Foley, E Freed, B Hahn, B Korber, P A Marx, F McCutchan, JW Mellors, and S Wolinksy. HIV sequence compendium. Technical Report LA-UR 03-3564, Theoretical Biology and Biophysics Group, Los Alamos National Laboratory, Los Alamos, NM, 2002.

[KHR04] TM Kim, SJ Hong, and MG Rhyu. Periodic explosive expansion of human retroelements associated with the evolution of the hominoid primate. *J Korean Med Sci*, 19:177–85, 2004.

[KJR04] TM Kim, YC Jung, and MG Rhyu. Alu and l1 retroelements are correlated with the tissue extent and peak rate of gene expression, respectively. *J Korean Med Sci*, pages 783–92, 2004.

[KMKM02] K Katoh, K Misawa, K Kuma, and T Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucl. Acids Res.*, 30(14):3059–66, 2002.

[KS05] G Kimmel and R Shamir. GERBIL: Genotype resolution and block identification using likelihood. *PNAS*, 102(1):158–62, 2005.

[Lot05] M Lothaire, editor. *Applied Combinatorics on Words*, volume 105 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2005.

[MBHC95] AG Murzin, SE Brenner, T Hubbard, and C Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J Mol Biol*, 247:536–40, 1995.

[MBI$^+$06] RE Mills, EA Bennett, RC Iskow, CT Luttig, C Tsui, WS Pittard, and SE Devine. Recently mobilized transposons in the human and chimpanzee genomes. *Am J Hum Genet*, 78(4):671–80, 2006.

[MC04] WJ Miller and P Capy, editors. *Mobile genetic elements*, chapter Retrotransposon mapping in molecular systematics. Humana, 2004.

[MDea06] J Marchini, P Donnelly, and et al. A comparison of phasing algorithms for trios and unrelated individuals. *Am J Hum Genet*, 78:437–50, Mar 2006. 10.1086/500808.

[MH98] M Meila and D Heckerman. An experimental comparison of several clustering and initialization methods. Technical Report MSR-TR-98-06, Microsoft Research, 1998.

[MH09] CD Malone and GJ Hannon. Small RNAs as guardians of the genome. *Cell*, 136:656–68, 2009.

[MMea04] H Matsuzaki, R Mei, and et al. Genotyping over 100,000 SNPs on a pair of oligonucleotide arrays. *Nat. Methods.*, 1:109–11, 2004.

[MMH$^+$04] GA McVean, SR Myers, S Hunt, P Deloukas, DR Bentley, and P Donnelly. The fine-scale structure of recombination rate variation in the human genome. *Science*, 304(5670):581–4, 2004.

[MPR98] M Mohri, FCN Pereira, and M Riley. A rational design for a weighted finite-state transducer library. In *LNCS*, volume 1436, 1998.

[MPR02] M Mohri, FCN Pereira, and M Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

[NQXL02] T Niu, S Qin, X Xu, and J Liu. Bayesian haplotype inference for multiple linked single nucleotide polymorphisms. *Am. J. Hum. Genet.*, 70:157–169, 2002.

[NY04] N Nagarajan and G Yona. Automatic prediction of protein domains from sequence information using a hybrid learning system. *Bioinformatics*, 20(9):1335–60, Jun 2004.

[OK05] EM Ostertag and HH Kazazian. LINEs in mind. *Nature*, 435:890–1, 2005.

[PCea01] N Patil, DR Cox, and et al. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(5547):1719–23, Nov 23 2001.

[PEP04] AL Price, E Eskin, and PA Pevzner. Whole genome analysis of Alu repeat elements reveals complex evolutionary history. *Genome Res.*, 14:2245–52, 2004.

[PJP05] AL Price, NC Jones, and PA Pevzner. De novo identication of repeat families in large genomes. *Bioinformatics*, 23 Suppl. 1:i351–8, 2005.

[PSD00] JK Pritchard, M Stephens, and PJ Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155:945–59, 2000.

[RAB$^+$00] DL Robertson, JP Anderson, JA Bradac, JK Carr, B Foley, RK Funkhouser, F Gao, and Hahn BH. HIV-1 nomenclature proposal. *Science*, 288(5463):55–6, 2000.

[RKMU05] P Rastas, M Koivisto, H Mannila, and E Ukkonen. A hidden Markov technique for haplotype reconstruction. In R Casadio and G Myers, editors, *WABI*, volume 3692 of *LNCS*, pages 140–51, 2005.

[RM96] N Risch and K Merikangas. The future of genetic studies of complex human diseases. *Science.*, 273:1516–7, 1996.

[RS93] B Rost and C Sander. Prediction of protein secondary structure at better than 70% accuracy. *J Mol Biol*, 232:584–99, 1993.

[RST94] D Ron, Y Singer, and N Tishby. The power of amnesia. In JD Cowan, G Tesauro, and J Alspector, editors, *NIPS*, volume 6, pages 176–83. Morgan Kaufmann Publishers, Inc., 1994.

[SBHHW04] N Shental, A Bar-Hillel, T Hertz, and D Weinshall. Computing gaussian mixture models with EM using equivalence constraints. In S Thrun, L Saul, and B Schölkopf, editors, *Proc. NIPS 16*. MIT Press, Cambridge, MA, 2004.

[SBR$^+$99] PM Sharp, E Bailes, DL Robertson, F Gao, and BH Hahn. Origins and evolution of AIDS viruses. *Biol Bull*, 196(3):338–42, 1999.

[SHG06] AFA Smit, R Hubley, and P Green. RepeatMasker, 2006.

[Slo02] N Slonim. *The Information Bottleneck: Theory and Applications*. PhD thesis, Hebrew University, Jerusalem, Israel, 2002.

[SRX$^+$03] AH Salem, DA Ray, Jinchuan Xing, Pauline A. Callinan, Jeremy S. Myers, Dale J. Hedges, Randall K. Garber, David J. Witherspoon, Lynn B. Jorde, and Mark A. Batzer. Alu elements and hominid phylogenetics. *PNAS*, 100(22):12787–12791, 2003.

[SSD01] M Stephens, N Smith, and P Donnelly. A new statistical method for haplotype reconstruction from population data. *Am J Hum Genet*, 68:978–89, 2001.

[Suz93] J Suzuki. A construction of Bayesian networks from databases based on an MDL scheme. In *Proc. UAI*, pages 266–73, 1993.

[SV98] M Studenty and J Vejnarova. The multi-information function as a tool for measuring stochastic dependence. In MI Jordan, editor, *Learning in Graphical Models*, pages 261–297. MIT Press, Cambridge, MA, 1998.

[Sve00] ED Sverdlov. Retroviruses and primate evolution. *Bioessays*, 22:161–71, 2000.

[SW81] TF Smith and MS Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147:195–197, 1981.

[TC02] JR True and SB Carroll. Gene co-option in physiological and morphological evolution. *Ann Rev Cell and Devel Biol*, 18:53–80, 2002.

[The08] The UniProt Consortium. The universal protein resource (UniProt). *Nucleic Acids Res*, 36:D190–5, 2008.

[THG94] JD Thompson, DG Higgins, and TJ Gibson. CLUSTAL W: improving the sensitivity of progressivemultiple sequence alignment through sequence weighting,position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22:4673–80, 1994.

[TKM04] Y Takebe, S Kusagawa, and K Motomura. Molecular epidemiology of HIV: Tracking AIDS pandemic. *Pediatrics International*, 46(2):236–244, 2004.

[TPB99] N Tishby, FC Pereira, and W Bialek. The information bottleneck method. In *Proc. of the 37th Annual Allerton Conference on Communication, Control and Computing*, pages 368–77, 1999.

[TTB$^+$03] JW Thomas, JW Touchman, RW Blakesley, GG Bouffard, SM Beckstrom-Sternberg, EH Margulies, M Blanchette, AC Siepel, PJ Thomas, and JC McDowell. Comparative analyses of multi-species sequences from targeted genomic regions. *Nature*, 424:788–93, 2003.

[VP90] TS Verma and J Pearl. Equivalence and synthesis of causal models. Technical Report R-150, UC Los Angeles, Dept. of Comp. Sci., 1990.

[WRO$^+$03] WS Watkins, AR Rogers, CT Ostler, S Wooding, MJ Bamshad, Brassington, E Anna-Marie, and ML Carroll. Genetic Variation Among World Populations: Inferences From 100 Alu Insertion Polymorphisms. *Genome Res.*, 13(7):1607–18, 2003.

[XHH$^+$04] J Xing, DJ Hedges, K Han, H Wang, R Cordaux, and MA Batzer. Alu element mutation spectra: Molecular clocks and the effect of DNA methylation. *J Mol Biol*, 344:675–82, 2004.

[XNJR03] EP Xing, AY Ng, MI Jordan, and S Russell. Distance metric learning with application to clustering with side-information. In S Becker, S Thrun, and K Obermayer, editors, *Proc. NIPS 15*, pages 505–512. MIT Press, Cambridge, MA, 2003.

[YL02] G Yona and M Levitt. Within the twilight zone: A sensitive profile-profile comparison tool based on information theory. *J Mol Biol*, 315:1257–75, 2002.

[ZH05] M Zaffalon and M Hutter. Robust inference of trees. *Ann. Math. and Artificial Intelligence*, 45(1–2):215–39, 2005.

[ZM05] Y Zhou and B Mishra. Quantifying the mechanisms for segmental duplications in mammalian genomes by statistical analysis and modeling. *PNAS*, 102(11):4051–6, 2005.

[ZRSL98] E Zietkiewicz, C Richer, D Sinnett, and D Labuda. Monophyletic origin of Alu elements in primates. *J. Mol. Evol.*, 47(2):172–82, 1998.