# Probabilistic Hierarchical Clustering for Biological Data

Eran Segal*
Computer Science Dept.
Stanford University
Stanford, CA 94305-9010
*eran@cs.stanford.edu*

Daphne Koller
Computer Science Dept.
Stanford University
Stanford, CA 94305-9010
*koller@cs.stanford.edu*

## ABSTRACT

Biological data, such as gene expression profiles or protein sequences, is often organized in a hierarchy of classes, where the instances assigned to "nearby" classes in the tree are similar. Most approaches for constructing a hierarchy use simple local operations, that are very sensitive to noise or variation in the data. In this paper, we describe *probabilistic abstraction hierarchies (PAH)* [11], a general probabilistic framework for clustering data into a hierarchy, and show how it can be applied to a wide variety of biological data sets. In a PAH, each class is associated with a probabilistic generative model for the data in the class. The PAH clustering algorithm simultaneously optimizes three things: the assignment of data instances to clusters, the models associated with the clusters, and the structure of the abstraction hierarchy. A unique feature of the PAH approach is that it utilizes global optimization algorithms for the last two steps, substantially reducing the sensitivity to noise and the propensity to local maxima. We show how to apply this framework to gene expression data, protein sequence data, and HIV protease sequence data. We also show how our framework supports hierarchies involving more than one type of data. We demonstrate that our method extracts useful biological knowledge and is substantially more robust than hierarchical agglomerative clustering.

## 1. Introduction

Recent developments in genomics have allowed the accumulation of large amounts of biological data: gene and protein sequences, gene expression data, and more. Important biological knowledge is implicit in this data, but extracting it is a difficult task. One very useful approach for providing insight into the data is to organize them in a "similarity" hierarchy; data instances that are "nearby" are often functionally related to each other, so that the hierarchy provides insight about cellular mechanisms. One example is the analysis of genomic expression data, where the level of mRNA transcript of every gene in the cell can be measured simultaneously using DNA microarrays. The most commonly used method for analyzing this data is clustering, a process which identifies clusters of genes that have similar expression profiles (e.g., [3, 14]). Genes that are similarly expressed are often involved in the same cellular processes, so that clustering suggests functional relationships between clustered genes. Another important example is the classification of proteins into a taxonomy of subfamilies based on their sequence or structure (e.g., [15]). Proteins whose sequence or structure are similar often play similar roles in the cell, so that proteins whose function is unknown may be assigned putative functions by measuring their similarity to the different subclasses in the hierarchy.

Discovering the hierarchy is a key part of the analysis. The algorithms most often applied to biological data use an agglomerative bottom-up approach [3, 15]. Although these algorithms have been shown to provide much insight into the biological processes, they suffer from several limitations. First, they proceed via a series of local improvements, making them particularly prone to local maxima. Second, the clusters are often constructed based directly on the raw data; models (if any), are constructed as a post-processing step. Thus, domain knowledge about the type of distribution from which data instances are sampled is rarely used in the formation of the hierarchy. These limitations imply that the hierarchies constructed are often quite brittle: their structure can vary substantially as a result of small perturbations in the data. Clearly, a high sensitivity of the learned hierarchy to the data raises doubts about the validity of biological conclusions drawn from the hierarchy. The fact that tree-construction algorithms can be sensitive to data was made by Felsenstein [4] in the context of phylogenetic trees, leading to the use of bootstrap techniques in order to increase the biological validity of the results.

Although one could apply bootstrap techniques to hierarchical clustering techniques as well, in this paper we propose an alternative technique for clustering biological data, which also has several other advantages. Our approach is based on *probabilistic abstraction hierarchies (PAH)* [11], a probabilistically principled general framework for learning abstraction hierarchies from data. It uses a Bayesian approach, where the different models correspond to different abstraction hierarchies. Each leaf in the tree corresponds to a class, and is associated with a *class-specific probabilistic model (CPM)* from which the data is generated. The prior over models (hierarchies) is designed to enforce our intuition that classes that are near to each other in the tree have similar data distributions. Segal *et al.* [11] present a general algorithm for learning the PAH model parameters and the tree structure in the PAH framework. Unlike many other clustering approaches, their al-

*Corresponding author

gorithm uses "global" optimization steps for learning the best possible hierarchy and CPM parameters.

There are several important advantages to the PAH framework, that make it particularly suitable for biological data sets. The global optimization steps help avoid local maxima. Furthermore, the abstraction hierarchy tends to pull the parameters of one CPM closer to those of nearby ones, which naturally leads to a form of parameter smoothing or *shrinkage* [9]. Both of these increase the robustness of the model, making it less sensitive both to noise in the data and to the particular choice of data set used to construct the hierarchy. The robustness and reproducibility of the hierarchy make conclusions derived from the hierarchy more valid from a biological perspective. Finally, the model-based approach allows the exploitation of domain structural knowledge more easily.

In this paper, we extend the general PAH framework to handle biological data. We focus on the two types of data where clustering has been used most often: gene expression and sequence data. For each data type, we describe both an appropriate class of CPMs and an appropriate prior over models. We also show how to combine models for different data types, allowing us to learn from heterogeneous sources simultaneously. This analysis may reveal patterns which are not apparent in either source alone. We describe efficient implementations of the generic learning algorithms for these instantiations of the PAH framework.

We provide experimental results for our algorithm on a variety of synthetic and real biological data sets, involving both sequence data, expression data, and a combination of both. We show that our algorithm is substantially more robust to sampling variation than hierarchical agglomerative clustering. We also show that the hierarchies produced by PAH are more biologically plausible, in that genes with similar functional categorization are much more likely to be placed close together.

## 2. Probabilistic Abstraction Hierarchies

In this section, we review the *probabilistic abstraction hierarchy (PAH)* framework of Segal *et al.* [11]. The PAH framework resembles certain "flat" clustering algorithms such as *Autoclass* [1] or $k$-means, where each data instance belongs to one of $k$ classes, each of which is associated with a different *class-specific probabilistic model (CPM)*. Each data instance is sampled independently by first selecting one of the $k$ classes according to a multinomial distribution, and then randomly selecting the data instance from the CPM of the chosen class. In standard clustering models, there is no relation between the individual CPMs. In a probabilistic abstraction hierarchy, the classes are organized in an abstraction hierarchy, or taxonomy, so that the models of the different classes respect the structure of the hierarchy: classes that are "nearby" in the hierarchy have similar probabilistic models.

More precisely, let $\mathcal{S}$ be a state space, from which our data instances are drawn. For example, instances in $\mathcal{S}$ can be protein sequences; or they can be tuples of expression measurements for genes in some set of arrays.

DEFINITION 2.1. *A* probabilistic abstraction hierarchy (PAH) *$\mathcal{A}$ is an undirected tree $T$ with nodes $V = \{v_1, \ldots, v_m\}$ and edges $E$, such that $T$ has exactly $k$ leaves $v_1, \ldots, v_k$. Each node $v_i$, $i = 1, \ldots, m$, is associated with a CPM $M_i$, which defines a distribution over $\mathcal{S}$; $\boldsymbol{M}$ is used to denote $M_1, \ldots, M_m$. There is also a multinomial distribution, parameterized by $\boldsymbol{\theta}$, over the leaves $v_1, \ldots, v_k$.*

The PAH framework is very general, and places no restrictions on the form of the CPMs: any probabilistic model that defines a probability distribution over $S$ can be used.

In the PAH model, data is generated only from CPMs at the leaves of the tree. We sample a class variable $C$ from the space $1, \ldots, k$, via a multinomial distribution $P(C \mid \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$, and then sample a data instance $S$ in $\mathcal{S}$ from the CPM of the appropriate leaf $P(S \mid M_c)$. Thus, the PAH model defines a joint distribution over the pair of random variables $C, S$: The joint probability of an element $s \in \mathcal{S}$, and a value $c$ for $C$, given a PAH $\mathcal{A}$, is defined as: $P(s, c \mid \mathcal{A}) = P(C = c \mid \boldsymbol{\theta}) P(s \mid M_c)$. The distribution of $s$ given $\mathcal{A}$, from which the data are generated, is $P(s \mid \mathcal{A})$, where $c$ is summed out.

The internal nodes in the tree are used to define a natural class hierarchy by enforcing similarity between the CPMs at nearby leaves. This goal is achieved by our choice of $\mathcal{A}$. The PAH approach is based on a Bayesian framework: we define a prior distribution over hierarchies, that prefers hierarchies that enforce our intuition. The prior is defined via a distance function $\rho(M_i, M_j)$ that measures the distance between CPMs $M_i, M_j$ that are neighbors in the tree. The function $\rho$ is not required to be a distance in the mathematical sense; instead, it only needs to be symmetric, non-negative, and satisfy that $\rho(M_i, M_j) = 0$ iff $M_i = M_j$. As all CPMs define a distribution over the same space $\mathcal{S}$, a natural choice for a distance measure is the *Kullback-Leibler (KL) distance* [2], also called relative entropy. The KL-distance defines the distance between two distributions $p$ and $q$ over a sample space $\mathcal{S}$ as:

$$\boldsymbol{D}_{KL}(p; q) = \int_{s \in \mathcal{S}} p(s) \log \frac{p(s)}{q(s)} ds,$$

where the integral is replaced by a summation in the discrete case. This distance measure has the advantage that it can be applied to any two CPMs over $\mathcal{S}$, regardless of their parameterization. As KL-distance is, in general, not symmetric, we can also use a symmetric version: $\rho(M_i, M_j) = (\boldsymbol{D}_{KL}(M_i; M_j) + \boldsymbol{D}_{KL}(M_j; M_i))/2$. Given a distance $\rho$, the prior over PAHs is defined as

$$P(\mathcal{A}) \propto \prod_{(i,j) \in E} \exp\left(-\lambda \rho(M_i, M_j)\right),$$

where $\lambda$ represents the extent to which differences in distances are penalized (larger $\lambda$ represents a larger penalty).

We can now put together our prior over models $\mathcal{A}$ with our likelihood function to define a posterior over $\mathcal{A}$'s. Given a set of data instances $D$ sampled from $P(S \mid \mathcal{A})$, we have that $P(\mathcal{A} \mid D) \propto P(\mathcal{A}) P(D \mid \mathcal{A})$. We can evaluate the quality of a PAH $\mathcal{A}$ using the log-posterior $\log P(\mathcal{A}) + \log P(D \mid \mathcal{A})$. The log-posterior serves as a score for a model $\mathcal{A}$, and our goal is to find a PAH $\mathcal{A}$ that maximizes this score. By maximizing this expression, we are trading off the fit of the mixture model over the leaves to the data $D$, and the desire to generate a hierarchy in which nearby models are similar.

## 3. PAH for Biological Models

In this section, we show how the generic PAH framework can be applied to biological data. We focus on gene expression and sequence data and define a PAH model for each using existing biological models as the main building blocks. We then define a PAH model over the joint domain of gene expression and sequence data, which allows us to combine both types of data into the PAH framework. In each case, we define both the form of the CPMs $M_i$ and the distance function $\rho$.

### 3.1 PAH for Gene Expression

In the case of gene expression data, each data instance corresponds to a gene, and encodes a set of expression level measurements of that gene in a suite of $n$ different experiments, each encoded as the log-ratio to some reference measurement. In this case, our sample space $\mathcal{S}$ is a subset of $\mathbb{R}^n$. A natural choice for a distribution over $\mathcal{S}$ is a multivariate Gaussian with an independent Gaussian component for each experiment. We assume that the variances of the Gaussian components are all equal to a known value $\sigma^2$. Thus, a *PAH for gene expression* associates with each node $v_i$ a CPM $M_i = \mathcal{N}(\vec{\mu}^i; \sigma^2 I)$, where $\vec{\mu}^i \in \mathbb{R}^n$ is the mean and $I$ is the identity matrix.

For the distance measure $\rho$, we use KL-distance, which for our Gaussian CPMs is particularly simple. For two distributions $M_i = \mathcal{N}(\vec{\mu}^i; \sigma^2 I)$ and $M_j = \mathcal{N}(\vec{\mu}^j; \sigma^2 I)$, we have that $\boldsymbol{D}_{KL}(M_i; M_j) = \frac{1}{\sigma^2}\sum_{\ell=1}^{n}(\mu^i_\ell - \mu^j_\ell)^2$, which is simply the squared Euclidean distance between the means of the Gaussians, normalized by the variance. We define $\rho(M_i, M_j) = \boldsymbol{D}_{KL}(M_i; M_j)$.

### 3.2 PAH for Sequences

We now turn our attention to sequence data, whether protein sequences or DNA sequences. In this case, each data instance in $\mathcal{S}$ is a sequence of variable length, with each position consisting of a *residue* — a letter in a finite alphabet (A,C,G,T in the case of DNA sequences). To capture the characteristics of a class of sequences, we follow the approach of Haussler *et al.* [8] and represent the distribution over a set of multiply-aligned sequences using a *profile hidden Markov model*: For each concensus column of the multiple alignment, a "match" state models the distribution of the residues allowed in the column. An "insert" state and "delete" state at each column allow for insertion of one or more residues between that column and the next, or for deleting the concensus residue. Position specific transition probabilities between states are also specified. Figure 1(a) shows a schematic diagram of a profile HMM.

Thus, we use profile HMMs as our CPMs in the PAH. More precisely a *PAH for sequences* associates with each node $v_i$ a profile HMM given by: $M_i = (P_{M_i}(H_{start}), \{P_{M_i}(H_\ell \mid H_{\ell-1})\}_{\ell=1}^{L}, \{P_{M_i}(O_\ell \mid H_\ell)\}_{\ell=1}^{L})$, where $H_\ell$ and $O_\ell$ are the state and residue variables, respectively, of the $\ell$-th column, and $P_{M_i}$ represents the distribution induced by the profile $M_i$,

We note that a PAH with profile HMM CPMs resembles the mixture profile HMM approach of Krogh *et al.* [8]. Their approach consists of $k$ different profile HMMs, each intended to capture a different subclass of proteins. However, there is no relation between the $k$ profile HMMs, which can be arbitrarily different. By contrast, the PAH framework imposes an abstraction hierarchy over the different profile HMMs, forcing similarity between ones that are nearby in the hierarchy. For a protein family, there is an intuitive desired relationship between subclasses in the hierarchy. As illustrated in the visualization of Figure 1(b), a CPM close to the leaves of the tree should be more specialized and thus have fairly peaked distributions, particularly for residues in the domains associated with the subfamily. Conversely, CPMs closer to the root of the tree, acting to bridge between their neighboring families, are expected to have less peaked distributions, e.g., peak only around domains which are common to an entire subtree.

To complete the specification of the PAH model for sequences, we need to define the distance function $\rho$. Once again, a natural choice is the KL-distance. However, there are considerable computational burdens incurred in using KL for profile HMMs. Let $\boldsymbol{H}$ and $\boldsymbol{O}$ denote the set of hidden state and observed residue variables, respectively, with one state and one residue variable for each column of the $L$ columns in the profile HMM. For two profile HMMs $M_i$ and $M_j$ with the same number of columns, it is well-known that:

$$\begin{aligned}\boldsymbol{D}_{KL}(M_i; M_j) \quad = \quad & \boldsymbol{D}_{KL}(P_{M_i}(H_{start}); P_{M_j}(H_{start})) + \\ & \sum_{\ell=1}^{L}\sum_{H_\ell} P_{M_i}(H_\ell)\boldsymbol{D}_{KL}(P_{M_i}(O_\ell \mid H_\ell); P_{M_j}(O_\ell \mid H_\ell)) + \\ & \sum_{\ell=1}^{L}\sum_{H_{\ell-1}} P_{M_i}(H_{\ell-1})\boldsymbol{D}_{KL}(P_{M_i}(H_\ell \mid H_{\ell-1}); P_{M_j}(H_\ell \mid H_{\ell-1})),\end{aligned}$$

where $P_{M_i}$ and $P_{M_j}$ denote the distributions induced by $M_i$ and $M_j$, respectively. Thus, each computation of the KL-distance requires that we compute the marginal distributions $P_{M_i}(H_\ell)$ for all $L$ columns in the profile. While this computation can be carried out efficiently using dynamic programming, its cost is still be quadratic in the number of hidden states and linear in the number of columns in the profile. As we discuss in Section 4, computing the distance between two neighboring models is a basic operation carried out many times in the process of learning the hierarchy from data, so even a quadratic computation is too expensive.

We therefore propose a closely related distance measure, which avoids the need for this computation. This distance function, which we call *Independent Kullback Leibler* (IKL), is a generalization of the *Total Relative Entropy* measure defined in [15].

$$\begin{aligned}\boldsymbol{D}_{IKL}(M_i; M_j) \quad = \quad & \boldsymbol{D}_{KL}(P_{M_i}(H_{start}); P_{M_j}(H_{start})) + \\ & \sum_{\ell=1}^{L}\sum_{H_{\ell-1}} \boldsymbol{D}_{KL}(P_{M_i}(H_\ell \mid H_{\ell-1}); P_{M_j}(H_\ell \mid H_{\ell-1})) + \\ & \sum_{\ell=1}^{L}\sum_{H_\ell} \boldsymbol{D}_{KL}(P_{M_i}(O_\ell \mid H_\ell); P_{M_j}(O_\ell \mid H_\ell)).\end{aligned}$$

Note that the IKL measure is very easy to compute: it is simply the sum of the KL distances between each of the multinomial distributions parameterizing the profiles $M_i$ and $M_j$. However, unlike KL-distance, IKL does require that the two models have the same set of parameters, i.e., that the two profile HMMs have exactly the same number of match states. We define $\rho(M_i, M_j) = (\boldsymbol{D}_{IKL}(M_i; M_j) + \boldsymbol{D}_{IKL}(M_j; M_i))/2$.

### 3.3 Joint PAH Model

We have now presented PAH models for two important types of biolgical data: gene expression profiles and gene or protein sequences. However, each of these models can learn a hierarchy based only on one type of data. The information provided by the sequence of the protein product of a gene and the information provided by its expression pattern are often complementary. By combining them, we may be able to construct a model that reveals patterns and provides insights that separate models over each type of data cannot.

Formally, in this case, each data instance in $\mathcal{S}$ is associated with a gene, and consists of both the mRNA expression level of that gene in a set of experiments, and the sequence of the protein product of the gene. We then define each CPM $M_i$ to be a pair, consisting of a multivariate Gaussian over the expression profiles and a profile HMM over the protein sequences.

Using the PAH framework, it is easy to define hierarchies over heterogenous data by independently combining CPMs

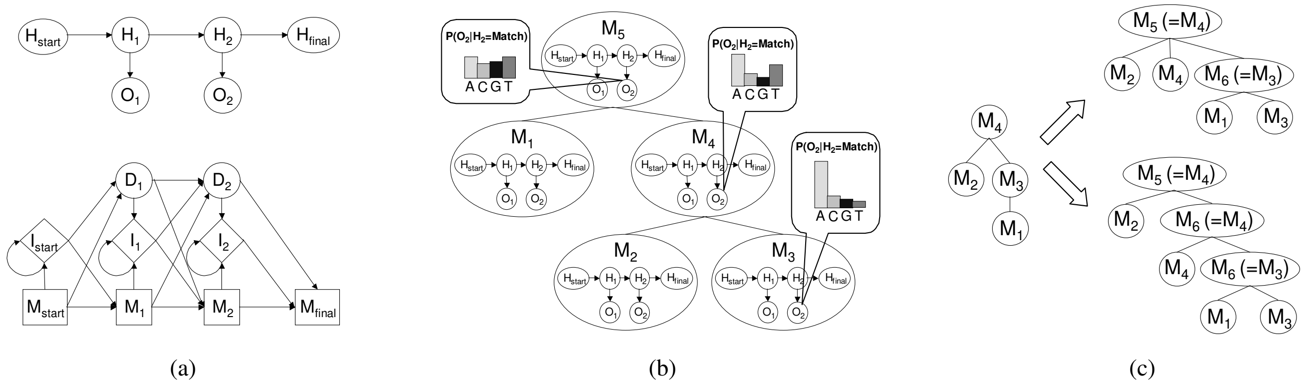


**Figure 1: (a) A profile HMM with 2 match columns (top): $S$ nodes correspond to (hidden) states and $O$ nodes correspond to (observed) residues. For each $O_i$, a distribution over the residues in column $i$ is specified. For each $S_i$, a distribution over state transitions, given the states of $S_{i-1}$, is specified. The allowed state transitions are also shown (bottom) (b) a PAH with 3 leaves using profile HMM CPMs along with the associated residue distributions at column 2 of the profile. (c) Two different weight-preserving transformations for a tree with 4 leaves $M_1, \ldots, M_4$.**

of different types, to define a product distribution over the product space. To compute the distance function between two CCPMs, we once again use the KL-distance function, or its IKL variant. For two cross-product distributions, the KL-distance is simply the sum of the KL-distances between the different components. Thus, a joint PAH model for sequence and expression will be a PAH with compound CPMs $M_i$, each defined over two CPMs: an independent mulivariate Gaussian CPM $M_i^g$ and a profile HMM $M_i^p$. The compound distance function $\rho(M_i, M_j)$ is the sum $\rho(M_i^g, M_j^g) + \rho(M_i^p, M_j^p)$.

# 4. Learning the Models

In this section, we briefly review the algorithm of [11] for learning a PAH $\mathcal{A}$ from an input data set $D = \{d[1], \ldots, d[N]\}$. This learning task is fairly complex, as many aspects are unknown: We are uncertain about the structure of the tree $T$, the parameters of all the CPMs $M_1, \ldots, M_m$ at the nodes of $T$ (leaves and internal nodes), the multinomial parameters $\boldsymbol{\theta}$, and the assignment of the instances in $D$ to leaves of $T$.

We start by describing the case of complete data, where for each data instance $d[j] \in D$, we are given the leaf from which it was generated. For this case, we show how to learn the structure of the tree $T$ and the setting of the parameters $\boldsymbol{\theta}$ and $\boldsymbol{M}$. We note that this assumption of complete data is, in fact, the assumption made in hierarchical agglomerative clustering techniques, as applied both to gene expression [3] and to sequence data [15]. There, the hierarchy is built by initially assigning each data instance to its own node and then proceeding by local operations that merge, at each step, the nodes that are closest relative to a predefined distance function. Our approach for complete data can also be applied to the situation where each data case has its own leaf node. It offers an interesting alternative to hierarchical agglomerative clustering, due to the advantages of global optimization steps. However, the probabilistic framework also allows us to deal with cases where we are uncertain about the assignments of data instances to the leaves of $T$.

### 4.1 Learning CPMs from Complete Data

Assume that the structure of the tree $T$ and the assignment of each data instance $d[m] \in D$ to one of the $k$ leaves, denoted $C[m]$, are known. We are thus left with the task of parameter estimation, that is, finding the setting of the parameters which maximize $P(\mathcal{A} \mid D)$. More specifically, we want to find $\boldsymbol{\theta}_{min}, \boldsymbol{M}_{min}$ that minimize $J = -\log P(D \mid \mathcal{A}) - \log P(\mathcal{A})$. Substituting the definitions into $J$, we get that

$$J = -\sum_{m=1}^{|D|} \log P(C[m] \mid \boldsymbol{\theta}) \quad (1)$$

$$-\sum_i \sum_{m: C[m]=i} \log P(d[m] \mid M_i) + \sum_{(i,j)\in E} \lambda \rho(M_i, M_j).$$

The first two terms arise from the negative log-likelihood of the data (each data is evaluated relative to the model at the leaf from which it was generated). The third term arises from our choice of prior.

The first term, involving the multinomial parameters $\boldsymbol{\theta}$, separates from the rest, so that the optimization of $J$ relative to $\boldsymbol{\theta}$ reduces to straightforward maximum likelihood estimation. To optimize the CPM parameters, the key property turns out to be the convexity of the $J$ function, which holds in a wide variety of choices of CPMs and $\rho$; in particular, it holds for all the models described in Section 3. The convexity property allows us to find the global minimum of $J$ using a simple iterative procedure. In each iteration, we optimize the parameters of one of the $M_i$'s, fixing the parameters of the remaining CPMs $M_j$ ($j \neq i$). In other words, we set

$$M_i := \text{argmin}_{M_i} J(D, \{M_j\}_{j\neq i}), \quad (2)$$

where the value of $J$ is computed relative to the current values of the other $M_j$'s. This procedure is repeated for each of the $M_i$'s in a round robin fashion, until convergence. By the joint convexity of $J$, this iterative procedure is guaranteed to converge to the global minimum of $J$. An examination of (1) shows that the optimization of each CPM $M_i$ involves only the data cases assigned to $M_i$ (if $i$ is a leaf) and the parameters of the CPMs $M_j$ that are neighbors of $M_i$ in the tree, thereby simplifying the computation substantially.

A particularly satisfying case occurs in our PAH for gene expression. Here, the optimizinig parameters in (2) can be computed easily in closed form:

$$\mu_\ell^i = \frac{\lambda \sum_{j:(i,j)\in E} \mu_\ell^j + \delta_{i\leq k} \sum_{m:C[m]=i} d[m]_\ell}{\lambda N(i) + \delta_{i\leq k} \sum_{j:C[m]=i} 1}$$

where $\delta_{i\leq k}$ in an indicator variable which is 1 iff $i$ is less than $k$ (and thus $M_i$ is a leaf model), and $N(i)$ is the number of neighbors of node $i$ in the tree. This formula is very intuitive, as each mean is simply a weighted average of the means of its neighbors, and, if it is a leaf, the actual values of the data points assigned to it. The weight depends on the value of $\lambda$, which determines the tradeoff between fitting the data and moving closer to classes nearby in the tree.

In other cases, such as our PAH model for sequences, a closed form solution for (2) does not necessarily exist. However, the optimization problem is still convex, and we thus use the conjugate gradient method to perform the optimization for each $M_i$. Convexity guarantees that the conjugate gradient method finds the optimal setting of the parameters of $M_i$ in each optimization step.

For completeness, we provide the gradient for the sequence PAH. Let $\pi$ be some multinomial distribution in the profile HMM. The distribution $\pi$ represents $P(X \mid Y = y)$ for some variable $X$ in the profile HMM and some assignment $Y = y$ to its parent; for example, $X$ might be $H_\ell$ and $Y$ its parent $H_{\ell-1}$, so that $\pi$ represents the transition distribution from some state in column $\ell - 1$ to the different states in column $\ell$. Let $\pi_i$ be the value of $\pi$ in $M_i$ and $\pi_j$ its value in $M_j$. We can now simplify the two terms in Equation 1 that relate to $\pi$. The form of the IKL-distance implies that the only term in $\rho(M_i, M_j)$ that relates to $\pi$ is $\boldsymbol{D}_{KL}(\pi_i; \pi_j)$. The other, for $M_i$ at leaves, relates to the log-likelihood of the data given $M_i$. Standard analysis shows that the only part of the data that is relevant to $\pi$ are data cases $m$ where $Y[m] = y$. Let $C_i[x, y]$ denote the number of data cases such that $C[m] = i$, and where $X[m] = x$ and $Y[m] = y$.

Using these ideas to simplify the function $J$, and taking the derivative, we obtain that the derivative of $\pi_{ix}$ for any value $x$ of $X$ is:

$$\begin{aligned}\frac{\partial J}{\partial \pi_{ix}} &= N(i)(\log \pi_{ix} + 1) - \sum_{j:(i,j)\in E} \log \pi_{jx} \\ &\quad - \sum_{j:(i,j)\in E} \frac{\pi_{jx}}{\pi_{ix}} - \delta_{i\leq k} C_i[x, y] \frac{1}{\pi_{ix}}\end{aligned}$$

This formula represents the unconstrained gradient, ignoring the constraint that $\sum_x \pi_{ix} = 1$. To ensure that this constraint is maintained in the gradient process, we simply subtracting $\frac{1}{|X|} \sum_x \frac{\partial J}{\partial \pi_{ix}}$ from each of the components of the gradient vector.

### 4.2 Learning Structure from Complete Data

We now turn our attention to the problem of learning the structure of the tree. We first consider an empty tree containing only the (unconnected) leaf nodes $v_1, \ldots, v_k$, and find the optimal parameter settings for each leaf CPM using the algorithm of Section 4.1. Note that these CPMs are unrelated, and the parameters of each one are computed independently of other CPMs.

Given this initial set of CPMs for the leaf nodes $v_1, \ldots, v_k$, the algorithm tries to learn a good tree structure $T$ relative to these CPMs. Due to the decomposability of $\log P(\mathcal{A})$, the quality of the tree can be measured via the sum of the edge weights $\sum_{(i,j)\in E} \rho(M_i, M_j)$. The goal is to find the lowest weight tree, subject to the restriction that the tree structure must keep the same set of leaves $v_1, \ldots, v_k$. Segal *et al.* propose an algorithm (inspired by the work of [6]) that uses a minimum spanning tree (MST) algorithm for constructing low-cost trees.

At each iteration, the algorithm starts out with a tree over nodes $v_1, \ldots, v_m$. It takes the leaves $v_1, \ldots, v_k$ of this tree, and constructs an MST over them. Of course, in the resulting tree, some of the $M_i$ are no longer leaves. This problem is corrected by a transformation that "pushes" a leaf down the tree, duplicating its model; this transformation preserves the weight (score) of the tree. By using only $v_1, \ldots, v_k$, the algorithm simply "throws away" the entire structure of the previous tree. However, we can also construct new MSTs built from all nodes $v_1, \ldots, v_m$ of the previous tree. For all nodes $v_i$ for $1 \leq i \leq k$, which end up as internal nodes, we perform the same transformation described above. In both cases, this transformation is not unique, as it depends on the order in which the steps are executed; see Figure 1(c). The algorithm therefore generates an entire pool of candidate trees (from both $v_1, \ldots, v_k$ and $v_1, \ldots, v_m$), generated using different random resolutions of ambiguities in the weight-preserving transformation. For each such tree, the algorithm of Section 4.1 can be used to find an optimal setting of the parameters. The trees are evaluated relative to our score ($\log P(\mathcal{A} \mid D)$), and the highest scoring tree is kept.

The tree just constructed has a new set of CPMs, so we can repeat this process. To detect termination, the algorithm also keeps the tree from the previous iteration, and terminates when the score of all trees in the newly constructed pool is lower than the score of the best tree from previous iteration.

### 4.3 Incomplete Data

Finally, we discuss the case where we do not have a fixed assignment of data cases to leaves in the tree. In this case, we have unobserved variables that specify the choice of class $C[m] \in \{1, \ldots, k\}$ for each data instance $d[m]$. The algorithm handles the issue of missing data using the *structural EM* algorithm of Friedman [5] which extends the EM algorithm to the problem of model selection. The algorithm repeatedly alternates between two steps. In the E-step, the distribution over the unobserved variables is computed by evaluating $P(C[m] = i \mid d[m], \mathcal{A})$ for all $1 \leq m \leq |D|$. In the M-step, the model $\mathcal{A}$ — its parameters as well as its structure — is updated so as to increase the score, relative to the distribution computed in the E-step. This process uses the algorithms of Sections 4.1 and 4.2, but where the assignment of data cases to leaves is now soft rather than hard. The effect of this is simply that different data cases are now taken into consideration with varying weights in the parameter estimation step in Section 4.1 and in the scoring of different trees in Section 4.2.

## 5. Experimental Results

In this section, we present the results of running the PAH algorithm on a variety of biological data sets. In all of our experiments, we initialized PAH such that each data instance has its own node. This assignment of data cases to leaves is only an initialization, and could be modified by the algorithm as described in Section 4.3.

Many of our results are compared to hierarchical agglomerative clustering (HAC), which is most commonly used for biological data. To perform a direct comparison between PAH and HAC, we often need to obtain a probabilistic model from HAC. To do so, we create CPMs from the genes that HAC assigned to each internal node. In both PAH and HAC, we then assign each gene (in the training set or the test set) to the hierarchy by choosing the best (highest likelihood) CPM among all the nodes in the tree (including internal nodes).

**Synthetic Data.** A good algorithm for learning abstraction hierarchies should recover the true hierarchy as well as possible. We tested this by running our algorithm on synthetic data, allowing us to compare its performance to a "ground truth". We generated a synthetic data set by sampling from the leaves of a PAH; the generating PAH was learned from a real gene expression data set. To allow a comparison with HAC, we generated one data instance from each leaf. We generated five training sets, each with 80 (imaginary) genes and 100 experiments, and PAH and HAC for each data set. For PAH, we used the model described in Section 3.1. For HAC, we used the same sum of squared errors as the distance function. For HAC, we also tried the Pearson correlation coefficient [3] as a distance metric, but its performance was consistently worse.

We tested the recovery of the hierarchy by defining the distance between two genes as the length of the tree-path between the nodes to which they were assigned, and compared the distances between the generating and learned model. The correlation coefficient was $0.72 \pm 0.08$ for PAH, compared to a much worse $0.27 \pm 0.09$ for HAC. The average $L_1$ error was $4.78 \pm 1.29$ for PAH and $12.46 \pm 1.09$ for HAC. Thus, PAH recovers an abstraction hierarchy much better than HAC.

**The Globin Protein Family.** We next tested the ability of our model to capture the characteristics of globins, a large family of heme-containing proteins of various architectures. We selected the entire set of 829 globins from the SWISS-PROT database by selecting all proteins with the keyword "globin". To test our algorithm's ability to "recognize" globins, we randomly selected a subset of 500 globins from the full set of 829, and used ClustalW [16] to multiply align them so that the trace of states (match, insert and delete) of each data instance through the profile HMM is given. We used this aligned data set to train a sequence PAH. We tried different settings for $\lambda$ (the coefficient of the penalty term in $P(\mathcal{A})$), thereby exploring the range of only fitting the data ($\lambda = 0$) and greatly favoring hierarchies in which nearby models are similar (large $\lambda$). We then measured the performance of each PAH on the training and test set, by recording the log-probability of each gene relative to the CPM at the node to which it was assignmed. The probability was computed using the forward algorithm, without assuming multiple alignment. We then deducted the log-probability of the random sequence from each such log-probability, obtaining the log-odds ratio. Figure 2(a) shows, for each $\lambda$, the average log odds ratio that PAH assigns to globins in the training set and in the test set. The graph is a typical learning curve, showing that the training set prediction is highest when $\lambda = 0$, but that test set accuracy is highest at $\lambda = 0.6$, where we have some tradeoff between fitting the data and favoring hierarchies in which nearby models are similar. To test the discriminative power of PAH, we also compared the log-odds ratio that PAH assigned to held-out globin sequences and to $10,000$ non-globin sequences selected randomly from SWISS-PROT. Not surprisingly, PAH achieved perfect discrimination between globins and non-globins. (We note that Krogh *et al.* [8] achieved similar results using profile HMMs.)

Finally, we checked whether the hierarchy built was also able to discover the different subfamilies of the globin family.

| Model | p | Training set avg. L1 difference | Test set avg. L1 difference |
|---|---|---|---|
| PAH | 90% | $2.57 \pm 1.57$ | $2.61 \pm 1.68$ |
| HAC | | $7.43 \pm 4.78$ | $11.34 \pm 7.2$ |
| PAH | 80% | $2.95 \pm 1.81$ | $2.76 \pm 1.61$ |
| HAC | | $9.44 \pm 5.68$ | $20.87 \pm 10.68$ |
| PAH | 70% | $3.11 \pm 1.84$ | $3.17 \pm 1.85$ |
| HAC | | $9.79 \pm 5.4$ | $21.72 \pm 13.8$ |

**Table 1: Robustness of PAH and HAC to different subsets of training instances.**

A node in the hierarchy has a high correspondence to a globin subfamily if all or most of the globins assigned to its subtree are of the same globin subfamily. We found several such nodes that corresponded to different subfamilies of globins including myoglobin, two different hemoglobin subfamilies (alpha chain and beta chain) and leghemoglobin.

**Robustness.** Our goal in constructing a hierarchy is to extract meaningful biological conclusions from the data. The available data invariably reflects only a subset of the experiments that we could have performed. If our analysis produces very different results for slightly different training data, the biological conclusions are unlikely to be meaningful. Thus, we want data instances that are assigned to nearby nodes in the tree to be assigned to nearby nodes even if the hierarchy was built from a slightly different training set.

To test the robustness of our algorithm, we used the Yeast Compendium data of [7], which measures the expression profiles triggered by specific gene mutations. We selected 450 genes and all 298 arrays, focusing on genes that changed significantly. For each of three values of $p$ ranging from $70\%$ to $90\%$, we generated ten different training sets by sampling (without replacement) $p$ percent of the 450 genes, the rest of which form a test set.

We then placed both training and test genes within the hierarchy. For each data set, every pair of genes either appear together in the training set, the test set, or do not appear together (i.e., one appears in the training set and the other in the test set). We compared, for each pair of genes, their distances in training sets in which they appear together and their distances in test sets in which they appear together. The results are summarized in Table 1. Our results on the training data show that PAH consistently constructs very similar hierarchies, even from very different subsets of the data. By contrast, the hierarchies constructed by HAC are much less consistent. The results on the test data are even more striking. PAH is very consistent about its classification into the hierachy even of test instances — ones not used to construct the hierarchy. In fact, there is no significant difference between its performance on the training data and the test data. By contrast, HAC places test instances in very different configurations in different trees, reducing our confidence in the biological validity of the learned structure.

**Discovering Gene Functions.** One of the main goals when building gene hierarchies based on sequence or expression data is to derive hypotheses about gene functions. If, in the learned hierarchy, genes with similar function cluster together, then we can hypothesize that genes whose function is unknown that cluster to the same area in the hierarchy also share a similar functional role.

To test whether PAH produces hierarchies with functional properties, we used the Yeast Compendium data of [7]. We selected 457 genes and the full set of 298 arrays, focusing

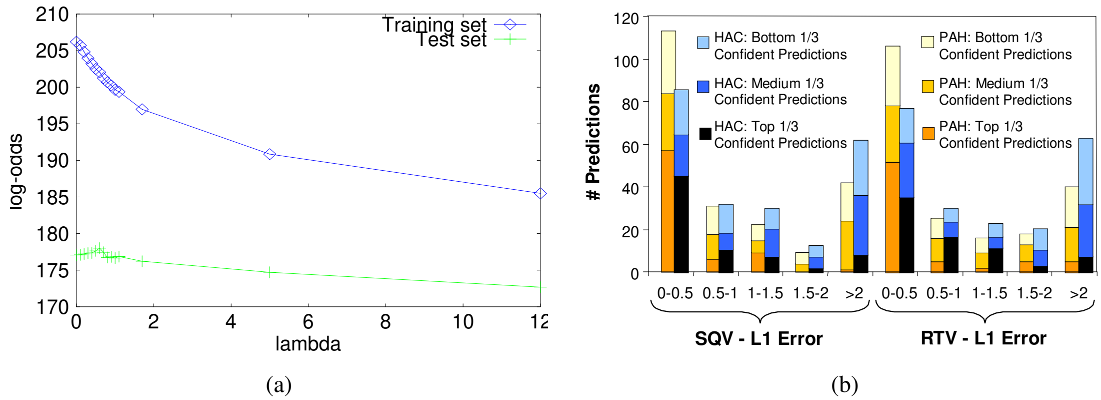


**Figure 2: (a) The average log-odds ratio for globin sequences in the training and test set. (b) Prediction of HIV drug-resistance from sequences.**

on genes that had some functional annotations in the MIPS database [10]. We learned hierarchies in three different settings: from expression data alone; from the protein sequences of the genes alone; and from a combined data set of sequence and expression. In each setting, we learned both a PAH and a HAC model. When learning HAC models for sequences, we used the agglomerative clustering approach similar to that of Sjolander *et al.* [15]: We construct a profile HMM for each data instance, then agglomerate the ones that are closest using our IKL distance (which generalizes the measure used in [15] for PSSMs). Newly created classes are also associated with profile HMMs, and participate in the agglomerative clustering in the standard way. For the joint expression/sequence models, we used the same distance measure as in our joint PAH model for the clustering.

For each of the learned hierarchies, we tested the extent to which the learned hierarchy had correspondence to gene functions. We chose to focus only on annotated genes, as the existence of a functional annotation in the database is substantially more indicative than a lack of one. We measured, for each functional category in MIPS, the distances between genes annotated with that function, and comparing these distances to those that would be measured if we were to randomly select a set having the same number of genes from the hierarchy. Note that each comparison is done within the same hierarchy. Table 2 lists the significant functions that were found in each hierarchy along with a p-value obtained from a student's t-test, where the null hypothesis is that the observed distances occurs by chance.

As can be seen, there are several functional categories which the PAH models put together in the hierarchy with a very high significance level, while the HAC models group together very few functions and with a much lower significance level. In particular, PAH automatically discovers several important pathways. We also see that the sequence models and expression models discover an almost disjoint set of functional categories. More surprisingly, the joint model does not rediscover the union of these categories. The information contained in the two different data sets seems to be sufficiently different that one cannot construct a coherent hierarchy capturing both. However, the joint data allows it to discover categories that were revealed by neither data set alone.

**Predicting Resistance to HIV Drugs.** Current drugs for the treatment of HIV infection include several agents, all aimed at inhibiting viral replication. When an HIV infected patient fails to respond to treatment, viral drug resistance development is often the main reason. Drug resistance arises from mutations in the viral genome, specifically in the regions that encode the molecular targets of therapy, HIV protease and RT enzymes. HIV RT and protease mutations alter the viral enzymes in such a way that the enzyme's function is no longer inhibited by the drug. An understanding of the genetic changes that render a particular drug ineffective is important to the development of new drugs, and to the design of effective drug combinations. Certain mutations confer resistance on their own, while others produce measurable resistance only when present in combination with others. We therefore decided to test the ability of hierarchical clustering algorithms to build hierarchies targeted at understanding drug resistance mutation patterns.

We used the database of Shafer *et al.* [13], which contains many of the published HIV RT and protease sequences, as well as drug susceptibility tests on many of the sequences. The drug susceptibility for a particular drug measures the *fold resistance* — the quantity of the drug, relative to susceptible wildtype isolates, required to achieve a certain level of inhibitory concentration (IC) of the virus. We focused on the protease sequences since they almost all have the same length of 99 amino acids. For each type of drug, we learned a PAH based both on the sequence and its associated (log) fold resistance at IC50. We used the joint model of Section 3.3, replacing the expression level with the fold resistance. We also learned a HAC based on the sequence data assigned the genes to nodes, and computed the average log-resistance level for each node.

To test the extent to which the hierarchy learned corresponded with drug resistance related mutations, we tried to predict the fold resistance of a held out sequence, by assigning the sequence to the best node in the hierarchy (as described above), and using the mean log-resistance at that node as our prediction. The results, based on five-fold cross-validation, are shown in Figure 2(b). The results indicate that the PAH hierarchy provides better correspondence to the drug resistance results. By recording the probability that the best CPM assigns to each sequence, we also obtain a confidence level. Reassuringly, most of the high quality predictions were obtained at the higher confidence levels. Furthermore, if we restrict attention to high-confidence predictions, the difference in quality between PAH and HAC is substantially larger.

| MIPS functional category | PAH (e) | HAC (e) | PAH (s) | HAC (s) | PAH (e+s) | HAC (e+s) |
|---|---|---|---|---|---|---|
| Cell growth, division, DNA synthesis (100) | $3.9e-9$ | | | | | |
| Mating type determination (41) | $1.8e-8$ | | | | $7.6e-7$ | |
| Nuclear organization (49) | $1.4e-4$ | | | | | |
| Transcription (61) | $8.8e-4$ | | | | | |
| Protein destination (47) | $1.5e-3$ | | | | | |
| Cellular organization (237) | $2.3e-3$ | $2.3e-2$ | | | $3.2e-3$ | |
| mRNA transcription (48) | $2.9e-3$ | | | | | |
| Metabolism (196) | | $2.5e-2$ | | | | |
| Drug transporters (10) | | | $8.8e-11$ | | | |
| C-compound & carbohydrate transporters (15) | | | $1.6e-5$ | $2.3e-4$ | $1.4e-7$ | |
| Transport facilitation (57) | | | $2.8e-4$ | | $3.1e-3$ | |
| Organization plasma membrane (41) | | | $6.3e-4$ | | | |
| Cellular import (57) | | | $2.5e-3$ | | | |
| Amino acid metabolism (50) | $2.5e-2$ | | | | | |
| Protein modification (24) | $3.6e-2$ | | | | | |
| Cell cycle control and mitosis (34) | $4.9e-2$ | | | | | |
| Protein synthesis (14) | $4.1e-2$ | | | | | |
| Organization of endoplasmatic reticulum (13) | $2.8e-2$ | | | | | |
| Stress response (70) | | | | | $2.8e-2$ | |
| Extracellular / secretion proteins (15) | | | | | $4.1e-4$ | |

**Table 2: Significant gene functions found in hierarchies; (e) stands for expression data and (s) for sequences; number in parentheses represents number of genes labeled with that function.**

## 6. Discussion

In this paper, we propose the use of probabilistic abstraction hierarchies [11] for constructing hierarchies over biological data. A unique feature of PAH is the use of global optimization steps for constructing the hierarchy and for finding the optimal setting of the entire set of parameters. This feature helps in avoiding local maxima and in reducing sensitivity to noise. We showed how the general PAH framework can be applied effectively to two important types of biological data: expression profiles and sequences. We presented results on a wide variety of data sets, showing that the PAH approach generates robust and biologically meaningful hierarchies.

We can extend our approach in several directions. For example, the generality of the CPM models could allow us to construct hierarchies where different CPMs have different structures, e.g., where CPMs have different profile HMM lengths. We are also exploring the use of this approach as part of a rich probabilistic framework such as that of [12], replacing a "flat" set of clusters with a more biologically plausible hierachy.

**Acknowledgements.** We thank Nir Friedman and Dirk Ormoneit for helpful discussions, and Dr. Bob Shafer for making the HIV dataset available to us. This work was supported by NSF Grant ACI-0082554 under the NSF ITR program, and by the Sloan Foundation. Eran Segal was also supported by a Stanford Graduate Fellowship (SGF).

## 7. REFERENCES

[1] P. Cheeseman and J. Stutz. *Bayesian Classification (AutoClass): Theory and Results*. AAAI Press, 1995.

[2] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[3] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–68, 1998.

[4] J. Felsenstein. Confidence limits on phylogenies: An approach using the bootstrap. *Evolution*, 39:783–791, 1985.

[5] N. Friedman. The Bayesian structural EM algorithm. In *Proc. UAI*, 1998.

[6] N. Friedman, M. Ninio, I. Pe'er, and T. Pupko. A structural EM algorithm for phylogentic inference. In *Proc. RECOMB*, 2001.

[7] T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–26, 2000.

[8] A. Krogh, M. Brown, S. Mian, K. Sjolander, and D. Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Mol. Biology*, 235:1501–1531, 1994.

[9] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proc. ICML*, 1998.

[10] HW. Mewes, K. Heumann, A. Kaps, K. Mayer, F. Pfeiffer, S. Stocker, and D. Frishman. MIPS: a database for protein sequences and complete genomes. *Nucleic Acids Research*, 27:44:48, 1999.

[11] E. Segal, D. Koller, and D. Ormoneit. Probabilistic abstraction hierarchies. In *Proc. NIPS*, 2001.

[12] E. Segal, B. Taskar, A.P. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. In *Proc. ISMB*. 2001.

[13] R.W. Shafer, D. Stevenson, and B. Chan. Human immunodeficiency virus reverse transcriptase and protease sequence database. *Nucleic Acids Research*, 27(1):348–352, 1999.

[14] R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analisys. In *Proc. ISMB*, 2000.

[15] K. Sjolander. Phylogenetic inference in protein superfamilies: Analysis of sh2 domains. In *Proc. ISMB*. 1998.

[16] J.D. Thompson, D.G. Higgins, and T.J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acid Research*, 22:4673–4680, 1994.