Enzyme function prediction with interpretable models

In *Computational Systems Biology*, Humana Press. (R. Samudrala, J. McDermott, R. Bumgarner, Editors)

Umar Syed[1] and Golan Yona[2*]

[1] Department of Computer Science, Princeton University

[2] Department of Computer Science, Technion - Israel Institute of Technology *and*

Department of Biological Statistics and Computational Biology, Cornell University

[*] Corresponding Author. Email: golan@cs.technion.ac.il golan.yona@cornell.edu

Abstract

Enzymes play central roles in metabolic pathways and the prediction of metabolic pathways in newly sequenced genomes usually starts with the assignment of genes to enzymatic reactions. However, genes with similar catalytic activity are not necessarily similar in sequence, and therefore the traditional sequence similarity-based approach often fails to identify the relevant enzymes, thus hindering efforts to map the metabolome of an organism.

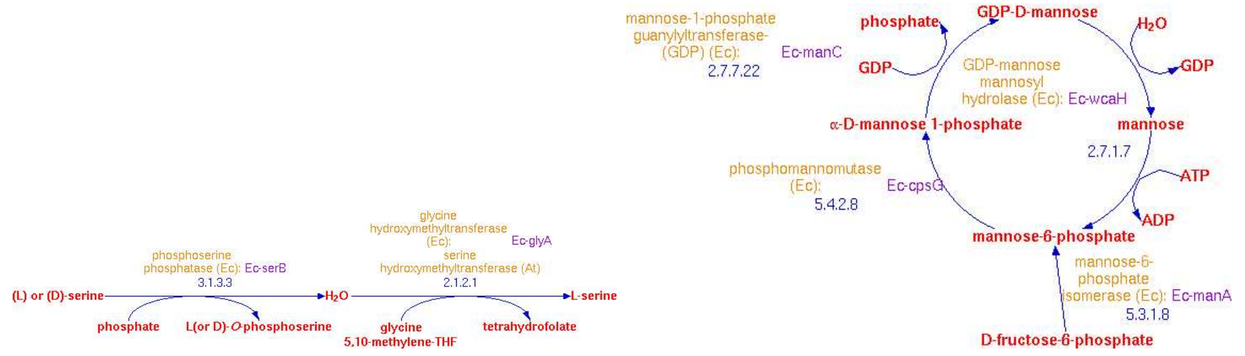
Here we study the direct relationship between basic protein properties and their function. Our goal is to develop a new tool for functional prediction (e.g. prediction of Enzyme Commission number) that can be used to complement and support other techniques based on sequence or structure information. In order to define this mapping we collected a set of 453 features and properties that characterize proteins and are believed to be related to structural and functional aspects of proteins. We introduce a **mixture model of stochastic decision trees** to learn the set of potentially complex relationships between features and function. To study these correlations, trees are created and tested on the Pfam classification of proteins, which is based on sequence, and the EC classification, which is based on enzymatic function. The model is very effective in learning highly diverged protein families or families that are not defined on the basis of sequence. The resulting tree structures highlight the properties that are strongly correlated with structural and functional aspects of protein families, and can be used to suggest a concise definition of a protein family.

Keywords: sequence-function relationships, functional prediction, decision trees, enzyme classification

1 Introduction

To understand why predicting the Enzyme Commission number (EC number) of an enzyme is important, consider the related problems of pathway prediction and of filling "pathway holes". With recent advances in sequencing technologies, the number of genomes that are completely sequenced is increasing steadily, opening new challenges to researchers. One of the main challenges is to decipher the intricate network of cellular pathways that exist in each genome. Understanding this network is the key to understanding the functional role of individual genes, and the genetic variation across organisms with respect to key processes and mechanisms that are essential to sustain life. Of special interest are metabolic pathways that consist mostly of enzymatic reactions and are responsible for nucleotide and amino acid synthesis and degradation, energy metabolism, and other functions. Many metabolic pathways have been studied and documented in the literature, usually in specific model organisms such as Yeast or E.coli.

Since experimental verification of pathways is a time consuming and expensive process, there is a great interest in computational methods that can extend the existing knowledge about pathways to newly sequenced genomes. It is often the case that the same metabolic pathway exists in multiple organisms. The different pathways are "homologous" to each other in the sense that they have the same overall structure. In each organism, different enzymes fill the various roles in the pathway. At the same time, different enzymes from different organisms that appear in the same location in a pathway have the same or similar functions. Hence, a pathway can be viewed as a generic graph diagram, where the nodes of the graph are the locations in the pathway, and are annotated with just the function associated with that location. For example, this is the way pathways are represented in the pathway databases KEGG [Kanehisa & Goto 2000] and MetaCyc [Caspi et al. 2006], where EC numbers are used to denote the function of each node in the graph (see Figure 1).



phosphoserine phosphatase (Ec): Ec-serB
3.1.3.3
glycine hydroxymethyltransferase (Ec): Ec-glyA
serine hydroxymethyltransferase (At)
2.1.2.1
(L) or (D)-serine
H₂O
L-serine
phosphate
L(or D)-O-phosphoserine
glycine
5,10-methylene-THF
tetrahydrofolate
mannose-1-phosphate guanylyltransferase-(GDP) (Ec): Ec-manC
2.7.7.22
phosphate
GDP-D-mannose
H₂O
GDP
GDP
GDP-mannose mannosyl hydrolase (Ec): Ec-wcaH
α-D-mannose 1-phosphate
mannose
2.7.1.7
phosphomannomutase (Ec): Ec-cpsG
5.4.2.8
ATP
ADP
mannose-6-phosphate
mannose-6-phosphate isomerase (Ec): Ec-manA
5.3.1.8
D-fructose-6-phosphate

Figure 1: **Left: Glycine biosynthesis pathway. Right: I/GDP-mannose metabolism.** (Pathway layouts were retrieved from the MetaCyc database [Caspi et al. 2006]). Metabolic pathways are sequences of consecutive enzymatic reactions. Each reaction starts with a certain metabolite and produces another metabolite through synthesis, break down or other transformations. The Glycine biosynthesis pathway produces the amino acid Glycine from the amino acid Serine. The GDP-mannose metabolism is one of the pathways that produce the compounds necessary to build cellular structures and organelles, such as cell walls.

The most popular approach for *pathway prediction* is to use pathway diagrams that were experimentally determined in one organism and map genes from a newly sequenced genome onto these diagrams [Paley & Karp 2002, Bono et al. 1998]. This approach requires a functional association of genes (and their protein products) with enzymatic reactions or enzyme families. If, for a particular organism, the enzyme corresponding to a node in the pathway diagram is unknown, this is called a *"pathway hole"* for that organism. Only proteins from that organism with the proper enzymatic activity can be candidates for filling the hole.

To identify the set of candidate enzymes various approaches have been employed. For example, [Green & Karp 2004] used BLAST to identify the best enzyme from among all candidate genes. Specifically, for each pathway hole, all proteins in the genome of interest were BLAST-ed against a set of "query" proteins, which consisted of proteins with the correct EC number taken from other genomes. Those proteins most similar to the query set were viewed as the most promising candidates. [Chen & Vitkup 2006] fill pathway holes by choosing an enzyme for the hole that has a similar phylogenetic profile as nearby enzymes in the pathway. [Kharchenko et al. 2006] took an integrative approach to filling holes. They used phylogenetic profiles, expression profiles, physical interaction data, gene fusion data and chromosomal clustering data to compare each candidate with the hole's neighbors in the pathway. A different integrative approach is used in [Popescu & Yona 2005] where genes were selectively assigned to pathways so as to maximize the co-expression of genes that were assigned to the same pathway while minimizing conflicts and shared assignments across pathways. This work was later extended in [Popescu & Yona 2006] where the deterministic assignments were replaced with probabilistic assignments that reflect the affinity of the different enzymes with different cellular processes.

All these methods require a set of candidates for each hole. And while they can use the trivial candidate set consisting of all of the proteins in the subject organism, their performance is likely to significantly improve if the candidate sets are small and focused. For example, [Yaminishi et al. 2005] found in experiments with their pathway inference algorithm that constraining pathways so that they are consistent with EC annotations "improves the [performance] in all cases". Similar results were reported in [Popescu & Yona 2006]. Hence, accurate EC annotation of proteins is an important step towards pathway prediction and functional annotation of genes in general. Producing effective mappings from genes to enzyme families is the focus of this chapter.

1.1 Enzyme class prediction: survey

1.1.1 The EC hierarchy

Traditionally, enzymes have been organized into a a systematic, hierarchical classification scheme devised by the IUB Enzyme Commission [EC]. The EC number of an enzyme has the form A.B.C.D, where the first digit A indicates the enzyme's major functional family: oxidoreductase, transferase, hydrolase, lyasase, isomerase, or ligase. The second digit B places the enzyme in a functional subfamily, the third into a sub-subfamily, and so on. For example, the EC number 1.2.3.4 designates enzymes which are oxidoreductase (the first digit) that act on the aldehyde or oxo group of donors (the second digit) and have oxygen as an acceptor (the third digit). In this case, the last digit specifies the particular reaction that is catalyzed by this family of enzymes. Some enzymes whose functions are poorly understood have incomplete EC numbers, i.e. they have only been assigned the first few digits of an EC number. Also, some multifunction enzymes have more than one EC number.

1.1.2 Prediction based on sequence similarity

Given a new protein sequence, the first step toward predicting its function is usually through sequence analysis. The most basic form of sequence analysis is sequence comparison, in search of sequence similarity. This analysis is still the most common way of functional prediction today, and the majority of sequences in the protein databases are annotated using just sequence comparison.

However, in many cases sequences have diverged to the extent that their common ancestry cannot be detected even with the most powerful sequence comparison algorithms. This is true for many enzyme families; although the proteins that share a particular EC number all possess the same function, they can nevertheless have very different sequences. As a result, predicting the EC number of a protein, even the first few digits, can be very challenging, and using only sequence alignment techniques is often insufficient.

For example, [Shah & Hunter 1997] used BLAST and FASTA to examine how well sequence alignment can predict the EC number of an enzyme. They surmised that if sequence alignment was enough to completely characterize an EC family, then every enzyme should have a higher alignment score with members of its own family than with members of other families. However, they found that this is the case for only 40% of EC families; for some families, the connection between sequence similarity and functional similarity was very weak. They noted several reasons for this, including the prevalence of multidomain proteins among enzymes, as well as convergent and divergent evolution.

1.1.3 Structure based approaches

When the structure of a protein is known, sequence analysis can be followed by structure comparison with the known structures in the PDB database. Since structure is more conserved than sequence, structural similarity can suggest functional similarity even when sequences have diverged beyond detection. However, enzyme families do not necessarily correlate with groups of structurally similar proteins. Moreover, proteins can have similar functions even if their structures differ [Wilson & Irwin 1999, Stawiski et al. 2000].

For example, [Todd et al. 2001] showed that the same enzyme family can be mapped to different structural folds. A clear mapping in the opposite direction also cannot be established. The authors showed that 25% of CATH superfamilies have members of different enzyme types and above 30% sequence identity is necessary to deduce (with 90% accuracy) the first three digits of the EC number. Similar results are reported in [Devos & Valencia 2000], who examined pairs of enzymes from FSSP [Holm & Sander 1994], a database of structural alignments. They found that among structurally similar proteins, above 50% sequence identity usually implied that the two enzymes shared the first three digits of their EC number, and above 80% sequence identity was enough to preserve all four digits. However, they also found that it was difficult to classify enzymes correctly below 50% identity, while below 30% was "problematic".

[Wilson et al. 2000] performed a similar analysis of the SCOP database [Murzin et al. 1995]. Among enzymes sharing a particular SCOP fold, they found that the first three digits of the EC number were not preserved below 40% identity, and even the first digit was not preserved below 25% sequence identity.

While these results were not very promising, [Rost 2002] argued that they were actually too optimistic, since the distribution of enzymes in curated databases is not representative of the true distribution of enzymes in nature. He proposed to correct for this bias by reducing the redundancy in those databases, and after doing so, found that even a very low BLAST e-value between two proteins did not always imply that they had identical EC numbers. He did show, however, that structural similarity predicts EC number much better than sequence similarity, though still not especially well. But since structural data is sparse, and is not available for newly sequenced genes, structure-based methods are not very useful for genome-wide enzyme prediction.

1.1.4 Approaches based on alternate representation of proteins

Instead of traditional sequence or structural alignment techniques, several groups have proposed comparison algorithms that rely on an alternative representation of proteins. These representations are usually derived from sequence motifs (i.e. short patterns), simple physio-chemical properties that can easily be computed from sequence, or annotations that are independent of sequence but can be found in databases for many proteins, e.g. subcellular location.

For example, [desJardins et al. 1997] represented each protein as a feature vector where the features were the composition of amino acids and the composition of secondary structure elements like helix, loop, etc. Note that this representation completely ignores the order of the amino acids in the sequence. However, a large number of generic machine learning algorithms require that training data instances be described as feature vectors, so the major advantage of this type of representation is that it allows these algorithms to be straightforwardly applied. DesJardins et al experimented with the naive Bayes classifier, decision trees, and nearest neighbor approaches. Although their results were inferior to sequence alignment methods, they were surprisingly good considering the simplicity of the features they used.

It is also possible to extract features from the molecular 3D structure of proteins. These can include per-residue properties like the contact energy or the distance from the center of mass of the protein. Obviously, this can only be applied to proteins whose structure is already known. Recently, [Borro et al. 2006] applied a Naive Bayes algorithm to this kind of representation, and were able to predict the first digit of the EC number with 45% accuracy.

[Cai & Chou] used a feature vector consisting of the amino acid composition, as well as the correlation of amino acid properties between residues near each other (but not necessarily adjacent) on the sequence. Their feature vectors also incorporated the GO [GO Consortium] annotations associated with the motifs contained in the sequence. In order to focus their efforts on difficult cases, they restricted their dataset to proteins that had less than 20% sequence identity to each other. Still, they were able to predict the first digit of the EC number of an enzyme with 85% accuracy. However, as there are only six possible values for the first EC digit, their method can uncover only a broad indication of the function of the enzyme.

A related approach is described in [Clare & King 2003]. This work uses an elaborate algorithm for predicting protein function that integrates several types of data. Given a query protein, they computed several sequence-derived and database-derived properties for it. They also performed a PSI-BLAST [Altschul et al. 1997] search to find homologous proteins, and computed the same properties for those homologs as well. Next, they used inductive logic programming to find frequently-occurring patterns among all these properties, and let each pattern be a binary feature. Finally, they used those features to train decision trees. There are several similarities between their algorithm and ours, but a major difference is that their decision trees are deterministic, whereas ours are stochastic (as described in section 4.1). Although they did not attempt to predict EC numbers, they were able to infer annotations from the MIPS classification scheme [Mewes et al. 1999] with accuracies ranging from 38% to 75%.

An important subgenre within these methods consists of those that use Support Vector Machines (SVMs) [Burges 1998]. Alternate protein representations often reside in very high-dimensional spaces, and SVMs are particularly well-suited for learning in these kinds of spaces, provided an appropriate kernel can be devised. [Jaakola et al. 1999] introduced the idea of using SVMs in biosequence analysis. They focused on detecting remote protein homologs, and devised SVM classifiers for SCOP families using an HMM-based kernel, where the feature vectors were generated by computing the derivatives of the sequence's likelihood with respect to the HMM parameters. Their study was followed by many others who applied SVMs to various classification tasks, including EC prediction. For example, [Han et al. 2004] formed a feature vector from several basic physio-chemical properties of proteins, and used a Gaussian kernel in the resulting feature space. They applied their algorithm to a set of 50 enzymes that have no known sequence homologs (the set was constructed using PSI-BLAST). On this difficult set, they were able to predict the first two digits of the EC number with 72% accuracy.

A novel kernel function, the "mismatch" kernel, was introduced in [Leslie et al. 2004] and tested on the SCOP classification. They defined their feature space such that the inner product of two vectors in the space is large only when the two sequences corresponding to those vectors contain many of the same k-mers (i.e. amino acid sequences of length k). [Ben-Hur & Brutlag 2006] explored a very similar approach as [Leslie et al. 2004], but used sequence motifs instead of k-mers in their mismatch kernel. They tried to predict EC numbers, and reported better results than simpler methods such as the nearest neighbor algorithm.

1.1.5 Other Approaches

Some authors have proposed novel data sources to predict EC numbers. For example, in prokaryotic organisms, it is well known that functionally related genes are often grouped together on the genome. [Kolesov et al. 2001] extended this concept with the following hypothesis: if the orthologs of two genes tend to be near each other on many genomes, then the two genes themselves are likely to have a similar function.

Other ideas can be viewed as advanced sequence similarity techniques. For instance, [Tian et al. 2004] used HMMs to determine the "functionally discriminating residues" of each EC family. According to their definition, these are the minimal set of residues needed to distinguish members of an EC family from non-members. [Emmanuel et al. 2005] proposed a Bayesian method to combine BLAST scores for the purpose of EC classification. They extend the simple approach of assigning each enzyme to the same EC class as its BLAST best-hit. Instead, for each enzyme, they estimate its membership probability in each EC class based on the EC classes of other enzymes with similar BLAST best-hit scores.

1.2 Our approach

In view of the previous sections, it is clear that multiple aspects of protein similarity should be considered, beyond sequence and structure, when functional similarity is sought. Features such as the domain content, subcellular location, tissue specificity, pairwise interactions and expression profiles may indicate common context and may suggest functional similarity even in the absence of clear sequence or structure similarity. These protein attributes and others are usually ignored, either because data is not available or because it is not clear how to use them to quantify functional similarity.

In this chapter we describe a method to predict the function of a protein based on basic biochemical properties augmented with (partial) data available from database records of biological properties that may hint at the biological context of the protein. Our goal is to identify the most relevant and informative features and combination of features that best characterize protein families (e.g. enzyme families), and to create a model for each protein family that can be used for classification and function prediction. Our approach hinges on algorithms for decision tree building [Duda et al. 2000, Mitchell 1997, Breiman et al. 1993] but further expands the traditional work on decision trees, by introducing a **mixture model of stochastic decision trees** (SDT). The trees handle missing attributes, ambiguities and fuzzyness — which are to be expected when analyzing database records of proteins — by using a probabilistic framework, and are optimized to ensure high generalization power by testing several validation techniques.

The resulting tree structure indicates the properties that are strongly correlated with the class of proteins modeled. This set of properties depends on the classification task. For example, the set of properties most strongly correlated with structural aspects are not necessarily the same as the properties that are most relevant for functional aspects. Clearly, this set can also change from one protein family to another. To study these correlations, trees were created and tested on two different types of known protein classifications: a sequence-based classification of proteins (Pfam [Bateman et al. 1999]) and a functional classification of proteins (the enzyme classification system [EC]).

The learning system consists of two main parts: the first is the feature extraction system. Much thought was given to selecting an extensive set of attributes that would create reliable representations of protein families. The second part is the model used to identify regularities in the patterns of these attributes for each family — the set of attributes that best correlate with functional aspects of proteins and can most accurately predict membership in the family. We first turn to describing the feature extraction system, and then to the decision tree learning model and our new SDT model.

2 Methods I - Data preparation and feature extraction

Our basic data set is the SWISSPROT database release 39.3 and the TrEMBL database [Bairoch & Apweiler 1999] release 14.4 as of July 15 2000, with 464744 proteins (the dataset is available at `http://biozon.org/ftp/data/papers/ec/`). A total of 453 features are used to describe each protein. The features are divided into three sets of features: features that can be calculated directly from the sequence, features that are predicted from the sequence and features that are extracted from database records.

2.1 Sequence features

These features include composition percentages for the 20 individual amino acids, as well as for 16 amino acid groups adopted from [Hobohm & Sander 1995] and [Ferran et al. 1994]. The groups are: charged (DEHIKLRV), positively charged (HKR), negatively charged (DE), polar (DEHKNQRSTWY), aliphatic (ILV), aromatic (FHWY), small (AGST), tiny (AG), bulky (FHRWY), hydrophobic (ILMV), hydrophobic aromatic (FWY), neutral and weakly hydrophobic (AGPST), hydrophilic acidic (EDNQ), hydrophilic basic (KRH), acidic (ED), and polar and uncharged (NQ).

Despite the overlap between these amino acid groups, each one has a unique characteristic. It is this characteristic that may play a role in defining and distinguishing family members from non-members, and therefore all the groups were considered to be distinct features.

We also calculated composition percentages for each of the 400 possible dipeptides. However, to save computation time, only the most informative dipeptides were used during learning. The selection of dipeptides was done dynamically, during the training phase of the algorithm, on a per-family basis. We discuss this refinement further in the Appendix, section 9.1.1.

In addition to sequence composition, we also computed the average hydrophobicity, average isoelectric point, and average molecular weight of the protein sequence, as well as the overall sequence length. The hydrophobicities used are the scaled values, as described in [Black & Mould 1991]. The values for isoelectric point can be found at [IonSource].

2.2 Predicted features

These represent the percentage of each protein that is predicted to be a coil, helix, or strand, as computed by PSIPRED [McGuffin et al. 2000], a secondary structure prediction program. We ran PSIPRED in 'single' mode for each sequence, meaning that the program did not search for sequence homologs to assist in its prediction.

2.3 Database features

We extracted nine database features from SWISSPROT: three binary, five nominal, and one numerical (integer). The three binary features respectively indicate the presence of alternative products, enzyme cofactors, and catalytic activity in each protein, as determined from the optional fields of the CC section of each SWISSPROT protein record (see SWISSPROT user manual [Swiss-Prot]). A lack of annotation was interpreted as a '0' value; otherwise, the value was set to '1'.

The nominal features (tissue specificity, subcellular location, organism classification and species) are more involved. Each protein is defined over zero or more values for each nominal feature: if $Values(A)$ is the set of all possible values for nominal attribute A, then each protein is defined over a subset of $Values(A)$. In all cases, a complete lack of information for A in a protein record was interpreted as an 'unknown' value for that attribute, and was treated specially by the decision tree algorithm (described in the Appendix, section 9.1.5). Two definitions of tissue specificity were included, one derived from the RC:TISSUE field, and one from the CC:TISSUE SPECIFICITY field. The subcellular location of the protein was derived from the

CC:SUBCELLULAR LOCATION field, while the organism classification and species were taken from the OC and OS fields, respectively.

Lastly, we also computed the number of PROSITE patterns exhibited by each protein by consulting the appropriate DR line of each record. Specifically, for each protein, we summed the number of certain and uncertain hits over all PROSITE patterns. This attribute is an indication of the domain/motif "complexity" of the protein[1].

It should be noted that we intentionally skipped the keywords in the KW field of SWISSPROT records. These keywords are strongly linked with protein function. However, as opposed to the other database attributes, these keywords explicitly describe protein functionality and are based on human annotation and interpretation. If ours was purely a classification task, the performance clearly could have improved by integrating SWISSPROT keywords. However, since we are interested in learning the mapping between proteins' properties and their function, we decided to exclude these keywords. We selected only database attributes that are based on experimentally verified information.

3 Methods II - The decision tree learning model

To model protein families, we developed a novel learning algorithm, which we call a **mixture model of stochastic decision trees** (SDT). Each family is modeled by a collection of decision trees that capture the salient features that are common to the members of the family. Our choice of the model was motivated by the nature of the data. Decision trees are useful when the data is nominal, i.e. when there is no natural notion of similarity or even ordering between objects. Such is the case for some of the attributes we associate with proteins, for example, tissue specificity and subcellular location. These attributes rule out the use of many other popular machine learning models, such as Support Vector Machines. Other attractive aspects of decision trees are robustness to errors both in classification and attribute values. Moreover, decision trees can be used when some of the data is missing, and this is often the case with database attributes. In the next sections, we first describe the traditional decision tree model, and then introduce the mixture model and other variations. The reader who is not interested in the details of the learning model can skip to section 6. However, we recommend reading sections 3.1 through 5 to get an idea of the main components of the model.

3.1 The basic decision tree model

Decision trees classify instances by sorting them down the tree from the root to a leaf node, which provides the classification of the instance. Each leaf node is either associated with a category label or probabilities over different categories. Each internal node specifies a test of some attribute of the instance and each branch descending from that node corresponds to a subset or range of the possible values of this attribute. An example decision tree is given in Figure 2. An instance is classified by testing the attributes of the instance, one at a time, starting with the one defined by the root node, and moving down the tree along the branch corresponding to the specific value of the attribute in that instance, until it reaches a leaf node. Note that this kind of scenario does not require a metric over instances.

3.2 The traditional training procedure for decision trees

When training decision trees from sample data S, the goal is to create a concise model that is consistent with the training data. Most learning algorithms use variations on a core algorithm that employs a top-down, greedy search through the space of all possible decision trees. Trees are grown from the root to the leaves, and the algorithm progressively splits the set of training examples into smaller and purer subsets by introducing additional nodes to the bottom of the growing tree with tests over new properties.

Usually, the criterion according to which the next attribute is selected is based on the reduction in *impurity* (class mixup) when testing on the attribute: the difference between the impurity before and after

[1]Alternatively, one could use an attribute for each possible motif, or a single motif attribute that indicates the set of motifs that is present in the protein. However, with more than 1300 PROSITE patterns the set of possible values (or combinations of values) is too large to handle efficiently during learning.

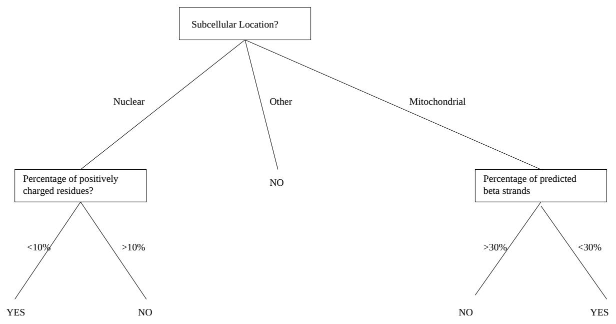

Subcellular Location?
Nuclear
Other
Mitochondrial
Percentage of positively charged residues?
NO
Percentage of predicted beta strands
<10%
>10%
>30%
<30%
YES
NO
NO
YES

Figure 2: **A sample decision tree for a specific class of proteins**. An instance is classified by testing its attributes, starting from the test at the root node. If the final answer is yes then the instance is classified to the class.

the split. One possible measure of the impurity or uncertainty is the entropy of the sample set at that node. The entropy impurity is also called the information impurity and is defined as

$$i(S) = Entropy(S) \equiv -\sum_{j=1}^{c} p_j \log_2 p_j$$

where p_j is the fraction of examples in the set S that are in category j. The common strategy is to select the attribute that decreases the impurity the most, where the drop in impurity is defined as:

$$\delta i(S) = i(S) - \sum_{v \in Values(A)} \frac{\mid S_v \mid}{\mid S \mid} i(S_v)$$

where $Values(A)$ is the set of all possible values for attribute A. When the measure used is the entropy impurity, the drop in impurity is called information gain and is denoted by $Gain(S, A)$

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{\mid S_v \mid}{\mid S \mid} Entropy(S_v)$$

The ID3 learning algorithm [Quinlan 1986] uses this criterion to decide on which attribute to test next. The outline of the algorithm is given below:

1. Initialize a root node with the set of all samples S.
2. If no test reduces the node impurity
   (a) Then node is a leaf.
3. Else
   (a) Select best test for decision node.
   (b) Partition instances by test outcome.
   (c) Construct one branch for each possible outcome.
   (d) Build subtrees recursively.

After the learning procedure has terminated, the learned tree can be used to classify new instances, as depicted in Figure 2. During classification, an example is repeatedly subjected to tests at decision nodes until it reaches a leaf, where it is assigned a probability equal to the percent of training examples at the leaf that are class members.

3.3 The extended training procedure for decision trees

The ID3 algorithm may result in trees that over-fit the training data, capturing coincidental regularities that are not necessarily characteristic of the class being modeled. Such trees will perform poorly on new unseen examples. The C4.5 algorithm [Quinlan 1993] is an improvement over ID3 by using rule **post-pruning** to prevent overfitting, and several other modifications to handle missing attributes and to account for bias in information gain introduced by multi-value attributes (using the *GainRatio* and $Mantaras$ functions described in section 9.1.6 of the Appendix).

Our basic learning procedure (which is one component of our new learning system) is similar to the C4.5 algorithm. We make intensive use of validation-based post-pruning in our algorithm. We refer to the set of samples available for learning as the **learning set** (the remaining samples are compiled into the **test set** used later for performance evaluation). The proteins in the learning set are divided into 10 equal parts. Let n be the size of the learning set. Ten different trees are generated for each family, and each one uses $\frac{9}{10}n$ as its **training set** and $\frac{1}{10}n$ as its **validation set**. Each tree is built using the training set and pruned using the validation set. Pruning is done by running an exhaustive search over all tree nodes. Each node is temporarily pruned (i.e. converted to a leaf) and the performance of the new tree is evaluated over the validation set. The node that produces the largest increase in performance is selected and the corresponding subtree is pruned. The process continues as long as it does not cause a decrease in performance over the validation set (we evaluate performance using a variety of different *evaluation functions*, which are discussed in section 4.2).

Cross-validation pruning improves consistency and provides a more balanced model that learns the regularities that are common to different subsets of the data rather than those that are coincidental and occur only in a specific data set.

Eventually, the 10 trees are combined to obtain a single prediction. Each tree is weighted according to its performance over the learning set, where the performance is measured in terms of the evaluation function, denoted by Q (see section 4.2). The final output is a performance-weighted average of the probabilities returned by each of the 10 trees. Denote by $P_i(+/x)$ the class probability of the sample x according to the i-th tree, then the total probability assigned by the set of trees is given by

$$P_{Trees}(+/x) = \frac{\sum_{i \in Trees} Q(i) P_i(+/x)}{\sum_{i \in Trees} Q(i)} \tag{1}$$

where $Q(i)$ is the quality (performance) of the i-th tree over the learning set (the higher, the better the tree separates the samples). It should be noted that the 10 trees that are learned from the 10 sets are generally very similar to each other, and their similar core structure is a reflection of the properties that truly characterize the protein family. However, because of the greedy nature of the search for the optimal tree, other, competing tree structures may be missed. This issue is addressed in the next section where we introduce the concept of a stochastic decision tree.

In addition to cross-validation based post-pruning, our algorithm includes other variations and enhancements that were adopted from different papers in this field, as well as several novel elements that we introduced into the learning procedure. Among the enhancements are:

- Dynamic attribute filtering.
- An effective procedure for discretizing numerical features.
- An efficient algorithm for finding binary partitions of multi-valued attributes.
- Methods to handle instances that have missing or multiple values for an attribute.
- Various impurity measures for attribute selection.
- Different weighting functions and adjusted impurity measures (such as **mixed entropy** and **interlaced entropy**) to handle skewed distributions in which negatives samples far outnumber positive samples, such as in our protein classification problem.

For more details on these and other elements of the extended learning algorithm, the interested reader is referred to section 9.1 of the Appendix.

4 Methods III - The mixture of stochastic decision trees

The next elements of our model address some of the fundamental problems with traditional decision tree learning algorithms. Specifically, we address three elements: optimization, evaluation and model selection. More precisely, we first propose an effective method of searching the hypothesis space that overcomes the pitfalls of the deterministic learning algorithms. Secondly, we introduce some novel criterion functions to evaluate decision tree performance. Thirdly, we propose an alternative method (MDL-based pruning) for deciding on the most probable model that is especially effective for small data sets. The first two elements are described next. The third is described in section 9.2 of the Appendix.

4.1 The stochastic framework

The basic procedures for building decision trees (such as those that are described in section 3) are deterministic. They employ a greedy approach that always selects the attribute the maximizes the information gain. However, this local maximization is not guaranteed to produce the "best" decision tree that describes the data. It happens quite often that several attributes have very similar information gain values. The choice of the one that marginally outperforms the others at some point in the training process may prove to be less advantageous later on. Even if the chosen attribute has a significantly better information gain, it still does not guarantee that in the optimal tree this attribute is indeed used at this point. To address this limitation imposed by the classical decision tree learning algorithm, we switched to a stochastic (probabilistic) framework, in which the attribute is selected with probability that depends on its information gain

$$Prob(A) = \frac{InfoGain(S,A)}{\sum_i InfoGain(S,A_i)}$$

Thus, attributes with higher information gain have higher probability to be selected, but even attributes with small information gain have a non-zero probability to get selected. To diversify the composite model of a protein family a total of 10 trees are learned for each training set; one deterministic (using the traditional approach) and 9 stochastic trees. With 10 training sets per family (see section 3.3), the final mixture model has a total of 100 trees per family. As before, denote by $P_i(+/x)$ the class probability of the sample x according to the i-th tree, then the total probability assigned by the hybrid mixture of trees is given by

$$P_{Mixture}(+/x) = \frac{\sum_{\text{training set } j} \sum_{i \in Trees(j)} Q(i) P_i(+/x)}{\sum_{\text{training set } j} \sum_{i \in Trees(j)} Q(i)} \tag{2}$$

where $Trees(j)$ is the set of trees learned from the j-th training set.

Our model was originally introduced in [Syed & Yona 2003]. It is similar to other methods that use ensembles of decision trees, such as [Dietterich 2000] and [Ho 1998]. In particular, [Breiman 2001] introduced the *random forests* model, in which a large mixture of decision trees is trained, and where the best split at each node is chosen from among a randomly selected subset of all the attributes. The parallels with our method are clear, but a major difference is that our method biases the selection towards higher quality attributes, instead of randomly constraining the selection to a small set. Moreover, some of the trees in our mixture are trained using the traditional deterministic method. Breiman's analysis suggests that these changes may yield improved performance, since he proves that the generalization error of a random forest decreases as the quality of the individual trees in the forest increase.

4.2 Evaluation of decision trees

The evaluation function Q is a key element of our learning model (see equations 1 and 2) as well as for performance assessment. A common measure of performance that is used in many machine learning applications (including decision trees learning algorithms such as C4.5) is the **accuracy**, i.e. the percentage of correctly classified examples

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} = \frac{tp + tn}{total}$$

where tp is the number of true positives, tn number of true negatives, fp is the number of false positives, and fn is the number of false negatives. However, with the majority of the samples being negative examples, the accuracy may not be a good indicator of the discriminating power of the model. When the task is to discern the members of a specific class (family) from a large collection of negative examples, better measures of performance are the **sensitivity** and **selectivity** (the positive predictive power), defined as

$$sensitivity = tp/(tp + fn) \qquad selectivity = tp/(tp + fp)$$

The categorization of samples as true (false) positives/negatives depends on the model and on the output it assigns to the samples. Since decision trees output probabilities, we need a way to interpret these real numbers as either 'positive' or 'negative'. Usually one sets a threshold score T, a probability above which samples are predicted to be positive. Thus, if a sample reaches a leaf node j with membership probability $P_j(+)$ (defined based on the relative fraction of positive samples in this node), then it will be classified as positive if $P_j(+) > T$ (see section 9.3 of the Appendix for discussion on methods for setting the threshold).

The ROC measure. An alternative that does not require defining a threshold first is the Receiver Operating Characteristic (ROC) score. To compute this score, one first has to sort all samples according to their probabilities (as assigned by the model), then plot the number of true positives as a function of the false positives, and finally measure the area under the curve. This measure will be maximized when all the true positives are assigned higher scores than the negative samples. A variation on this score is ROC50, which only measures the area under the curve up to the first 50 false positives. The idea behind this plot is that in scanning a database search results one may be willing to overlook few errors if additional meaningful similarities can be detected. The area under the curve can be used to compare the overall performance of different methods.

The Jensen-Shannon measure. We propose a new evaluation function that takes into account the complete distributions of positives and negatives induced by the decision tree model, and accounts for their *statistical distance*. Specifically we use the Jensen-Shannon (JS) divergence between probability distributions [Lin 1991]. Given two probability distributions $\mathbf{p}$ and $\mathbf{q}$, for every $0 \leq \lambda \leq 1$, the λ-JS **divergence** is defined as

$$D_\lambda^{JS}[\mathbf{p}||\mathbf{q}] = \lambda D^{KL}[\mathbf{p}||\mathbf{r}] + (1-\lambda) D^{KL}[\mathbf{q}||\mathbf{r}]$$

where $D^{KL}[\mathbf{p}||\mathbf{q}]$ is the relative entropy of $\mathbf{p}$ with respect to $\mathbf{q}$ (also called the Kullback-Leibler divergence [Kullback 1959]) and

$$\mathbf{r} = \lambda \mathbf{p} + (1-\lambda)\mathbf{q}$$

can be considered as the most likely common source distribution of both distributions $\mathbf{p}$ and $\mathbf{q}$, with λ as a prior weight (here we set $\lambda = 1/2$ since the weighted samples are divided equally between the two categories; see section 9.1.7 of the Appendix). In our case, $\mathbf{p}$ and $\mathbf{q}$ are the empirical distributions of the positive and negative examples with respect to the class probabilities assigned to them by the decision tree model. We call the corresponding measure simply the **divergence score** and denote it by D^{JS}. This measure is symmetric and ranges between 0 and 1, where the divergence for identical distributions is 0.

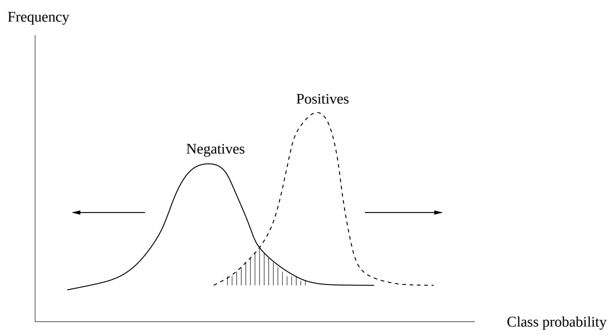


Figure 3: **Optimizing performance with the Jensen-Shannon measure**. Ideally, the model should assign high probability to class members and low probability to non-class members. The goal is to maximize the statistical distance between the distribution of positives and the distribution of negatives and minimize the overlap.

All the measures provide rough estimates of the expected accuracy. However, unlike the other measures, the divergence score is less sensitive to outliers and is more likely to reflect the true estimate when applied to future examples. Moreover, there is no need to define a threshold T. Our goal is to maximize the statistical distance between the distribution of positives and the distribution of negatives, as exemplified in Figure 3, and minimize the overlap. The smaller the overlap the better the expected accuracy.

All four performance measures were tested during post-pruning (a node is pruned if the probabilities induced by the pruned tree are better at separating the positives from the negatives in terms of the evaluation function selected; see **post-pruning** in section 3.3 for more details). Also, while testing a particular pruning technique, we always used that same evaluation function to compute $Q(i)$ in equations 1 and 2.

5 Methods IV - Optimization of the learning algorithm

There are many strategic decisions that one can take throughout the learning process. To find a good choice of parameters, we first optimized the learning strategy. Having incorporated the modifications described in sections 3.3 and 4 into our model and the tree-generating program, we converged to a locally optimal learning strategy by conducting a rough greedy search over the **space of decision tree learning algorithms**. All performance evaluations are done over the test set using the Pfam data set (see section 6.1) and averaged over all families. A step-by-step description of that search procedure follows:

The initial basic configuration is very similar to the C4.5 learning algorithm [Quinlan 1993]. It uses multiple (variable) branching factor, unweighted entropy-based gain ratio, 5 cross-validation sets, accuracy-based post-pruning (with the equivalence-point based threshold as described in section 9.3 of the Appendix), unweighted leaf nodes in prediction and a smaller subset of 53 features after removing all dipeptide information. The performance of this configuration was poor (sensitivity of 0.35). Introducing dipeptide information, and increasing the number of cross validation sets to 10 resulted in a significant improvement in performance (0.55). This setup was gradually improved by considering each one of the modifications discussed in the previous sections. The modifications were added one by one, and were accepted only if they improved performance over the previous configuration of the parameters. The results of this search are summarized in Table 1.

configuration	sensitivity (= selectivity)	accepted/rejected
basic (C4.5)	0.35	initial
dipeptides + 10 cross-validation sets	0.55	accepted
mantaras metric	0.56	accepted
binary splitting	0.66	accepted
weighted entropy	0.69	accepted
confidence-based threshold	0.63	rejected
sen/sel post-pruning	0.69	accepted
JS-based post-pruning	0.7	accepted
roc50 post-pruning	0.7	rejected
20 cross-validation sets	0.69	rejected
MDL post-pruning (entropy based)	0.71	accepted
MDL post-pruning (probability based)	0.73	accepted
weighted leafs	0.72	rejected
mixed entropy	0.67	rejected
interlaced entropy	0.73	rejected
stochastic trees	0.81	accepted

Table 1: **Optimization of learning strategy.** We started with a basic learning algorithm that is very similar to the C4.5 learning algorithm. Then, one step at a time, we introduced another modification to the learning algorithm and tested its impact on the performance. Performance was measured in terms of the prediction sensitivity over the test set using the equivalence point criterion (see section 9.3 of the Appendix). At the equivalence point, the sensitivity equals the selectivity.

Switching to binary splitting, weighted entropy and JS-based post-pruning, and the introduction of the stochastic decision trees improved performance and were accepted. Especially noticeable is the improvement due to the replacement of a single deterministic tree with a mixture of stochastic trees. Increasing the number of cross-validation sets did improve the performance but only until a certain point (10 sets), beyond which no further improvement was observed. Some of the weighting procedures (such as mixed-entropy and leaf weighting) were rejected, as well as other modifications that were later outperformed by alternative strategies (e.g. sensitivity/selectivity based pruning was selected first during the search but was then outperformed by Jensen-Shannon based pruning). It should be noted that a very good performance was also obtained with MDL-based pruning (see section 9.2 of the Appendix). The MDL approach is especially effective for small families (see section 6.1). However, because of its dependency on an external parameter that requires additional tuning we focus here on the simpler Jensen-Shannon based pruning[2].

The final configuration was set to the mixture of stochastic decision trees, including information on dipeptides (dynamically selected on a per-family basis, during training as described in section 9.1.1 of the Appendix), with binary splitting, weighted entropy, 10 sets of cross-validation, and JS-based pruning and evaluation. The probability assigned by the mixture model to a sample x is defined as

$$P_{Mixture}(+/x) = \frac{\sum_{training\ set\ j} \sum_{i \in Trees(j)} Q_{JS}(i) P_i(+/x)}{\sum_{training\ set\ j} \sum_{i \in Trees(j)} Q_{JS}(i)}$$

where $Q^{JS}(i)$ is the divergence/separation score of the i-th tree over the validation set. Note that the divergence score does not require a method to choose a threshold.

6 Results

We used two types of known protein classifications, the Pfam database of protein families and the EC database of enzyme families. We tested our model over both databases by training models for all families using the optimal learning strategy, as described in the previous section, and testing them over unseen examples.

6.1 The Pfam classification test

Our first test was on the Pfam database of protein domains and families. Of the 464744 proteins in the SWISSPROT and TrEMBL databases, 237925 proteins are classified into 2128 families (Pfam release 5.2). Most of the families in the Pfam database are domain families that characterize only a small part of the member proteins (these domains usually appear in conjunction with other domains). To prevent ambiguity in the class labels we included in our data set only those proteins that were labeled as belonging to exactly one family. In particular, since sequence databases are constantly growing, the absence of a label does not necessarily mean that the protein does not belong to a functional family. Since the properties we use in our model are global (calculated based on the complete sequence), only protein families that cover most of the sequence were considered for learning and evaluation. Specifically, we used a subset of 233 families with more than 80% coverage and at least 50 members (the dataset is available at `http://biozon.org/ftp/data/papers/ec/`). Because of the high coverage we expect that for these families their functionality is indeed associated with the complete protein chain.

Each family was randomly partitioned into a learning set and a test set in the ratio of 3:1, so that 75% of the member proteins were used for learning, and 25% for testing, and a model was trained for each family using the optimal learning strategy as described in section 5. The mixture model was then used to assign probabilities to each sample in the test set and the performance was evaluated by measuring sensitivity at the equivalence point (i.e. the point where the selectivity equals the sensitivity; see section 9.3 of the Appendix for details).

In all cases, the mixture model performed better than the best individual tree. Interestingly, for almost half the families, the best stochastic tree performed better (by up to 10%) than the greedy, deterministic tree, which is evidence for the usefulness of the stochastic approach. In Figure 4 we show five trees for the Pfam family Apolipoprotein (proteins whose function is to carry cholesterol and other fats through the blood and help in maintaining normal levels of these molecules). The first one is deterministic while the other trees were built using the probabilistic algorithm. The performance of the five trees is 0.96, 0.85, 0.84, 0.89, and 0.99, with the fifth stochastic tree outperforming the deterministic tree.

To assess the power of our new model we compared it with BLAST [Altschul et al. 1997], the most popular database search algorithm. Since BLAST is a pairwise comparison algorithm we repeated the search for all query sequences in the family. Specifically, for each query, we sorted all proteins by their

[2]It also should be noted that the final average performance, after adding the stochastic trees, was the same with Jensen-Shannon and with MDL-based pruning (in both cases it was 0.81).

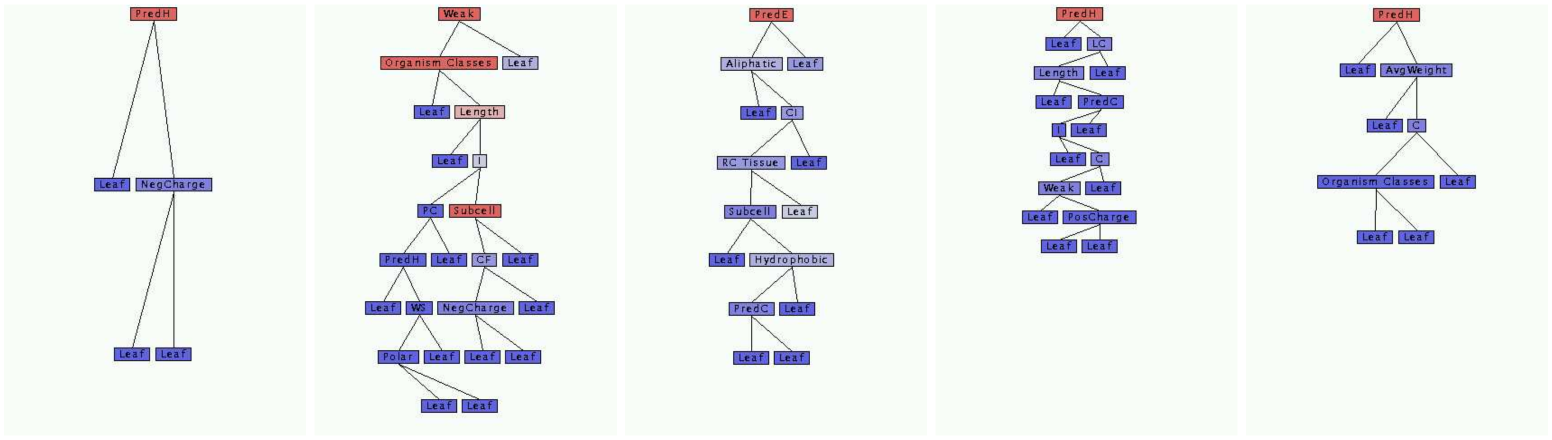


Figure 4: **Alternative decision trees for Pfam family Apolipoprotein**. The leftmost tree is a deterministic tree. The others are stochastic. The rightmost tree performed the best. The color of each decision node is proportional to the (weighted) entropy at that node, with deep red indicating high entropy (high class impurity) and deep blue indicating low entropy. Lighter colors indicate intermediate entropy values.

BLAST score against the query, and then measured sensitivity for detecting the family at the equivalence point (i.e. the threshold where the selectivity equals the sensitivity; see section 9.3 of the Appendix). For each family, we report the best performing query, as well as the performance of a typical BLAST search, by averaging over all queries. Table 2 gives the results for the 30 largest families. The average performance of the SDT model over all families was 81%. A typical BLAST search detected 86% of the member proteins, while the best BLAST search detected 94% of the proteins on average (in a blind prediction scenario one does not have a knowledge of which is the best query, as this is determined based on the performance over the test set, and therefore the average performance should be considered as the typical performance). Note that for many families, especially the larger ones, the SDT model outperformed a typical BLAST query, and in some cases the best BLAST query (e.g. picornavirus coat proteins [rhv], Hemagglutinin-neuraminidase [HN] and histones). This is very encouraging considering that our model does not use any information about the order of amino acids in the sequences beyond pair statistics.

Our analysis suggests three main reasons why our model is short of detecting all member proteins for some of these sequence-based protein families. First, since Pfam is not necessarily a functional classification but rather a domain classification, many of the features that we associate with complete protein chains may not be correlated with the specific domain. It is possible to "localize" some features and calculate them just along the domain region. However, some features, such as database attributes, cannot be localized.

The second reason is the use of weighted entropy. Although weighted entropy definitely improves performance (see Table 1), it can also stop the learning too early, leaving some of the leaf nodes impure, and thus affecting performance (see Figure 5). Specifically, because of the skewed background distributions, positives are assigned a much higher weight than negatives. Take for example the UbiA family. There are 58 family members and 237867 negative examples, of which 43 and 178400 are in the learning set, respectively. The weight of each one of the positive examples is more than 4000 times the weight of a negative example. Consider a node with 20 positive examples and 1000 negative examples. If the total weight of the initial set of positives and negatives is 100 (50 each), then the sum of the weighted positive examples in that node is 26 while the negative samples sum to 0.3, resulting in weighted entropy of 0.09. However, this node is far from being pure, and although we set a very low impurity threshold some nodes do end being somewhat impure. In the case of the UbiA family one of the nodes has an entropy of 0.15535 with 0.27 weighted negative examples and 11.8 weighted positive examples. However the number of actual negatives and positives is 876 and 9 respectively. This node clearly needs refinement. Employing other weighting protocols can resolve this problem (see section 9.1.7 of the Appendix). Indeed, switching to unweighted sample entropy after exhausting weighted sample entropy (interlaced entropy) improved performance over the 25 worst performing families by several percent.

But perhaps a more significant flaw is that validation-based pruning is overly aggressive for small families.

Family name	Family size	Test set size	Percent test set detected by SDT	BLAST
GP120	21644	5411	1.00	—
COX1	3397	850	0.97	1.00 (0.73)
MHC_II_beta	2348	587	0.97	1.00 (0.97)
F-protein	1637	410	0.99	0.99 (0.96)
Hemagglutinin	1360	340	0.99	1.00 (0.97)
p450	1328	332	0.78	0.97 (0.86)
globin	1064	266	0.89	0.87 (0.66)
adh_short	1012	253	0.63	0.97 (0.79)
rhv	895	224	0.95	0.84 (0.58)
vMSA	890	223	1.00	1.00 (0.91)
ras	793	199	0.86	0.99 (0.98)
NADHdh	786	197	0.94	0.99 (0.76)
Tat	707	177	0.99	1.00 (0.99)
adh_zinc	695	174	0.76	0.99 (0.84)
VPR	645	162	0.99	1.00 (0.98)
sugar_tr	624	156	0.65	0.91 (0.61)
Vif	604	151	0.99	1.00 (1.00)
fer4_NifH	561	141	0.85	0.96 (0.84)
actin	555	139	0.90	1.00 (0.97)
histone	536	134	0.95	0.54 (0.35)
cpn60_TCP1	516	129	0.85	0.95 (0.84)
HSP70	511	128	0.89	1.00 (0.99)
Gram-ve_porins	495	124	0.96	0.94 (0.82)
late_protein_L1	482	121	0.93	1.00 (0.76)
COX3	480	120	0.95	1.00 (0.94)
fusion_gly	442	111	0.99	1.00 (0.83)
HN	414	104	0.95	0.82 (0.51)
REV	375	94	0.95	0.97 (0.87)
serpin	353	89	0.78	0.95 (0.86)
COesterase	347	87	0.79	0.99 (0.92)

Table 2: **SDT performance for the 30 largest Pfam families.** Performance is evaluated using the equivalence point criterion (see text). For BLAST two numbers are given, for the best query and for a typical query (in parenthesis). The average performance of the SDT model over all 233 families was 0.81.

This is because, for these families, the validation set typically contains only a handful of family members. It is unlikely that such a small sample will be representative of the family overall. So during pruning, many branches of the tree (which was trained on a much larger sample) will be deemed irrelevant and pruned away. For these families, MDL-based pruning (see section 9.2) may be a better alternative, as it seeks to balance tree size with the performance of the tree over the entire learning set. We tried MDL-based pruning on a subset of 50 small families. Compared to JS pruning, the lowest performing families seem to make the biggest gains, with performance boosting over this subset from an average of 0.34 to 0.47 (Figure 6).

6.2 The information content of features

In our representation, each protein sequence is represented by a set of 453 attributes. However, not all of the attributes are equally important and not all of them are necessarily used in the model for each protein family. Those that are used are not exclusive in that they allow even proteins with unknown values for these attributes to be classified to the family. This is because the learning system can accommodate ambiguities by defining multiple rules for each protein family during the learning process.

To evaluate the quality of each feature as a protein family predictor, and to quantify its information content, we trained 453 trees for each family, one for each feature. The tree was trained using the optimal configuration as described in the previous section[3]. The results were averaged over all families and are given in Table 4. Results for just the dipeptide attributes are available in Table 5.

Clearly, no single attribute can serve as a good predictor in general. Performance varies between the different attributes and database attributes seem to have less information content than predicted and calculated attributes, perhaps because little information is currently available in database records. For most proteins these attributes are undefined and fractions of these proteins reach many leaf nodes (following the

[3]A simple way to estimate the information content of an attribute would be to calculate the class entropy for each feature, and average over all families. However, this calculation ignores some of the aspects of decision trees that are addressed in the Appendix.

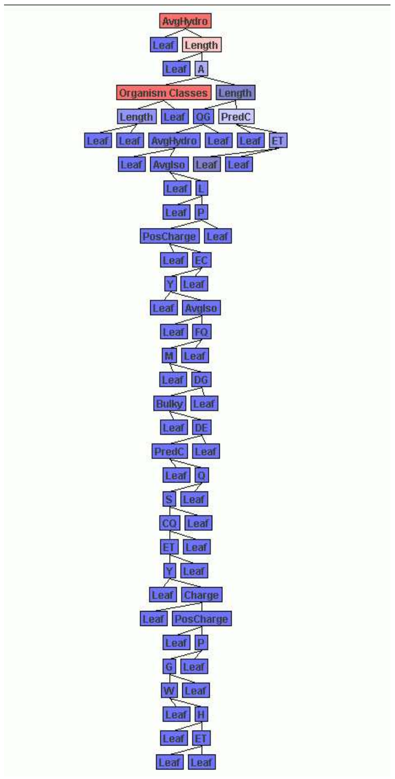


Figure 5: **An example tree for the UbiA family.** Our model did not perform well over this family. The main reason is demonstrated in this tree. One of the leaf nodes has an entropy of 0.15535 with 2727.95 weighted negative examples and 118421 weighted positive examples. However the number of actual negatives and positives is 876 and 9 respectively.

ratios for proteins with known values for these attributes, as described in section 9.1.5 of the Appendix), thus affecting the discriminative power of the attribute.

Surprisingly, the organism classification attribute is the best predictor (information gain of 0.390). Note also its depth (5.6), meaning that it is usually used higher in the decision trees. Indeed many protein families exist only in a single kingdom or subphyla. The success of this attribute suggests that other database attributes can play a more significant role, as more data becomes available. The organism class is followed by the average hydrophobicity and the length. Other informative attributes are the relative compositions of different groups of amino acids, such as acidic, negatively charged, aromatic and so on.

We also recorded the usage frequency of attributes. The usage frequency of an attribute (Table 6) is an indication of its role in classification at a more general level. Such attributes are the length, the organism class, predicted secondary structure, and the frequency of different amino acid groups.

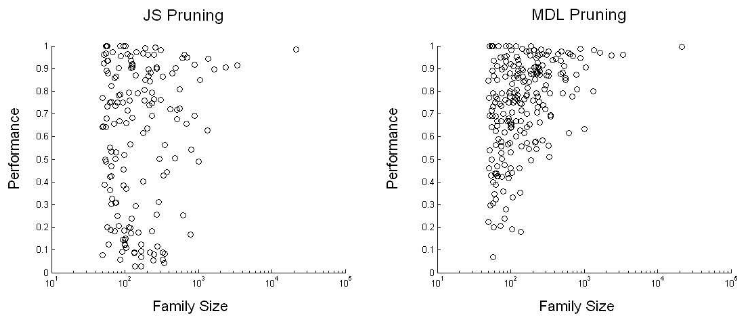


Figure 6: **Improvement in performance with MDL-based pruning**. The left plot shows performance (on the y-axis) vs. family size (on the x-axis) under JS-based pruning. Each circle represents a family. The right plot shows the improvement from MDL-pruning. Note that the mass of families clustered at the bottom-left of the JS plot shifted upwards in the MDL plot, while the rest of the distribution seems relatively unchanged.

6.3 The EC classification test

While the Pfam test establishes the validity and proves the potential of our method, it is the EC classification test (which was independent of the optimization of the learning strategy) that shows the true prediction power in cases where sequence based methods can fail.

The EC classification system is an extensive collection of functionally-based protein families. Enzyme families are defined based on the functional role of the enzymes in the cell rather than on common evolutionary ancestry, and enzymes that perform similar functions are not necessarily homologs and might not exhibit any sequence or structure similarity. Clearly, detecting this kind of similarity is much more difficult, as it involves inferring the functionality of the proteins in vivo.

Family	Family	Test set	Number	Percent testset	detected by	
name	size	size	subfamilies	SDT	BLAST	SAM
4.1.1.39	3879	970	2	0.99	0.95 (0.91)	0.99
1.9.3.1	1904	476	18	0.95	0.85 (0.47)	0.92
3.6.1.34	1612	403	26	0.87	0.42 (0.23)	0.69
2.7.7.48	488	122	21	0.95	0.37 (0.14)	0.78
2.7.7.6	473	119	26	0.64	0.42 (0.18)	0.76
2.7.7.7	461	116	13	0.67	0.45 (0.24)	0.90
3.1.3.48	395	99	3	0.85	0.83 (0.59)	0.97
1.14.14.1	382	96	1	0.76	0.70 (0.58)	0.77
2.7.1.112	367	92	2	0.79	0.92 (0.82)	0.89
1.1.1.1	360	90	2	0.91	0.77 (0.61)	0.92
5.99.1.3	320	80	2	0.81	0.92 (0.66)	0.92
1.2.1.12	318	80	1	0.88	0.98 (0.94)	0.98
1.6.99.3	315	79	37	0.68	0.40 (0.18)	0.69
2.7.7.49	284	71	1	0.77	0.80 (0.57)	0.82
5.2.1.8	245	62	2	0.76	0.72 (0.55)	0.95
2.5.1.18	240	60	5	0.77	0.77 (0.49)	0.91
3.2.1.14	222	56	2	0.73	0.51 (0.31)	0.90
6.3.1.2	213	54	1	0.87	0.97 (0.86)	0.98
3.1.3.16	211	53	9	0.77	0.67 (0.46)	0.95
3.2.1.4	207	52	2	0.40	0.43 (0.23)	0.89
1.11.1.7	179	45	5	0.76	0.82 (0.66)	0.83
3.1.1.4	175	44	3	0.93	0.91 (0.82)	0.94
1.11.1.6	170	43	3	0.91	0.92 (0.82)	0.99
3.2.1.1	168	42	2	0.79	0.76 (0.57)	0.85
1.18.6.1	168	42	2	0.74	0.57 (0.42)	0.92
3.4.99.46	161	41	1	0.88	0.98 (0.73)	1.00
2.7.2.3	157	40	2	0.80	0.97 (0.93)	0.95
3.1.26.4	154	39	2	0.69	0.50 (0.33)	0.32
4.1.1.31	150	38	2	0.92	0.99 (0.85)	0.96
3.5.2.6	150	38	3	0.74	0.73 (0.52)	0.96

Table 3: **SDT performance for the 30 largest EC families.** All columns but the forth are the same as in Table 2. Fourth column is the number of different sequence subfamilies (Chau and Yona, unpublished work). The average performance of the SDT model over all 229 families was 0.71.

To test our model on this functional classification we extracted all enzyme families and trained models for all families with more than 50 members, for a total 229 families (some of these were 'high-level' EC families, e.g. 1.2.3.-.). Models were trained as described in section 6.1. As with the Pfam classification test, we compared our results to BLAST. We also compared our model with another statistical model, the Hidden Markov based alignment program SAM [Hughey et al. 1999]. Each SAM model is trained first over the set of unaligned training sequences and then used to search the database and assign probabilities to each sample in the test set. The performance is evaluated as before using the equivalence point criterion. The results are listed in Table 3 for the 30 largest families. The average performance of the SDT model over all 229 families was 71%. Note that in this case the SDT model outperformed the best BLAST in many cases, especially when the enzyme family is composed of several sequence-based families. SAM performance over a subset of 122 families (all the families with full, 4 digit EC numbers, e.g. 1.2.3.4) was 89%. The performance of our model over the same set was 76%. As most members of the same EC family have similar sequences the success of this model is not surprising. However, for extremely diverse and large families our model was more effective in detecting family members (see 1.9.3.1, 3.6.1.34, and 2.7.7.48). Again, one has to remember that the SDT model was trained on features rather based on the sequences, and yet it was surprisingly successful in classification. Integrating this model with other models can be expected to boost the overall performance by detecting remote homologies. For example, the mixtures of trees from the SDT model might be used to refine sequence similarity-based models, combining the strengths of both approaches. One can also expect integration of higher-order significant sequence elements (conserved k-tuples) to improve performance as well. Indeed, k-tuples were successfully used to classify proteins into families using support vector machines with a string kernel [Leslie et al. 2004].

It should be noted that for the EC classification task we used the same final configuration of the learning system that was optimized over the Pfam data set (see Table 1). By optimizing the parameters of our learning system over a separate classification, we avoided the danger of overfitting the learning strategy to the irregularities of the EC families.

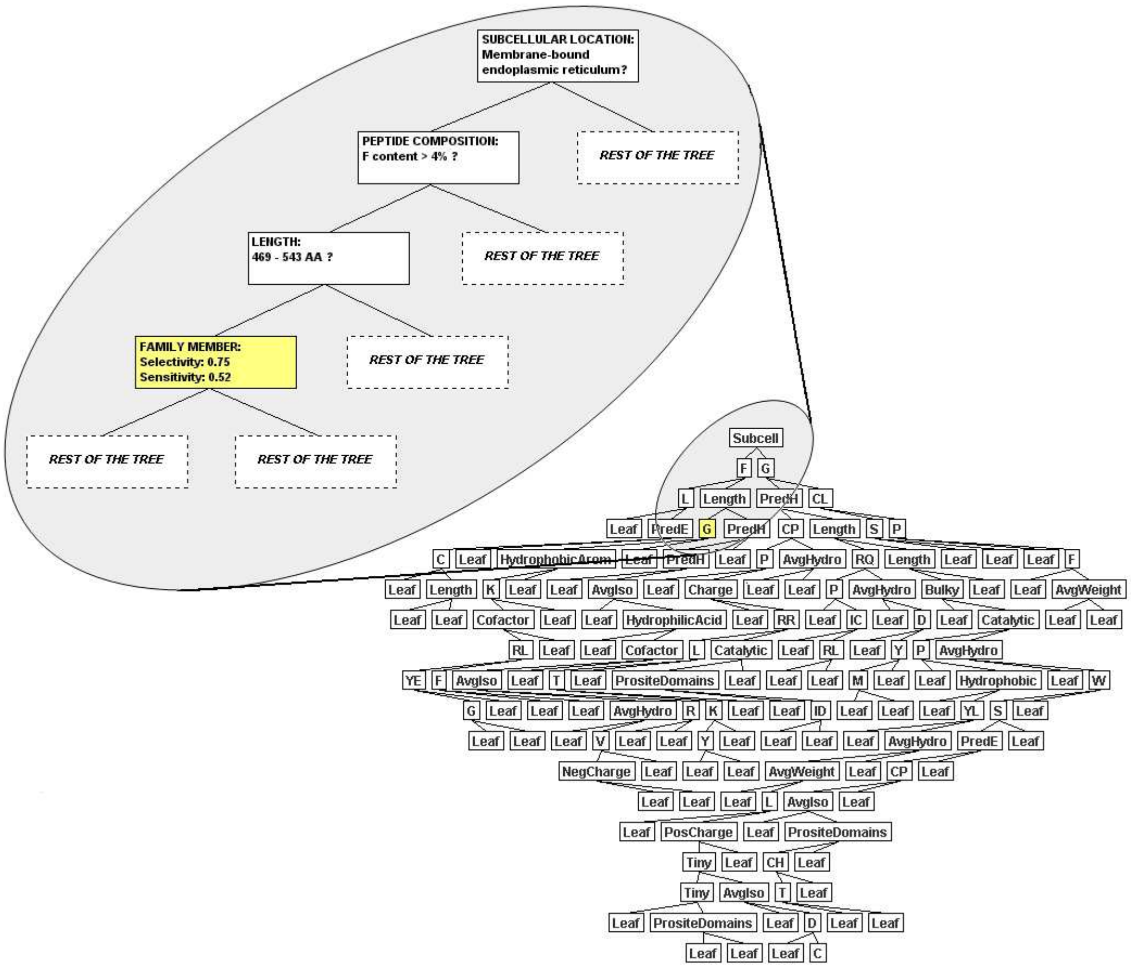

SUBCELLULAR LOCATION: Membrane-bound endoplasmic reticulum?
PEPTIDE COMPOSITION: F content > 4% ?
REST OF THE TREE
LENGTH: 469 - 543 AA ?
REST OF THE TREE
FAMILY MEMBER: Selectivity: 0.75 Sensitivity: 0.52
REST OF THE TREE
REST OF THE TREE
REST OF THE TREE
Subcell

Figure 7: **A decision tree for EC family 1.14.14.1**. The complete tree is shown at lower-right. The part of the tree discussed in the text has been enlarged. The highlighted rule detects members of 1.14.14.1 with 75% selectivity and 52% sensitivity.

6.4 Interpretability of decision trees - decision rules

A decision tree represents a disjunction of conjunctions of constraints on the attribute values of instances, where each path (each decision) from the tree root to a leaf corresponds to one such conjunction. Thus, a decision tree can be converted to an equivalent rule set that can be considered a logical description of a category. Each one of our decision trees was converted into a rule set, with rules sorted by their accuracy over the test set. Some of these rules are especially interesting as they might suggest a concise description of a family in terms of a few elements or principles. For example, one of the features that characterizes EC family 1.14.14.1 of monooxygenases (including cytochrome P450) is subcellular location. Many members of this family are located in the membrane-bound endoplasmic reticulum (see Figure 7). Note that after the initial split according to subcellular location, other features refine the definition of family members. One rule given by this tree indicates that proteins located in the membrane-bound endoplasmic reticulum, that have above-average content of F residues, and are approximately 500 residues long, have a 75% probability of belonging to family 1.14.14.1 (i.e. selectivity = 0.75), with 52% coverage of the family (i.e sensitivity = 0.52). For comparison, the probability to select a member of this family by chance (the background probability) is 0.01

Interestingly, one of the stochastic trees highlights another feature that characterizes the family: members are more abundant in several tissues (blastocyst, mycelium, prostate, liver, adrenal, lung, hepatopancreas, blood and ovary) while rare in others. The split by this attribute reduces the impurity by 30%. However, this is just one element of a more complex set of dependencies, and to understand the role of tissue specificity in characterizing family members it is necessary to look at the complete set of rules.

Another example where subcellular location is useful for prediction is family 1.10.2.2 (including the ubiquinol-cytochrome-c reductases). Family members tend to be located in the inner mitochondrial membrane, particularly the matrix side, and half of the deterministic trees in the mixture for the family use this subcellular location in the root split of the tree. Yet another example is enzyme family 2.7.7.6 (RNA polymerases). It turns out that, among proteins that have low content of A, G, W, and E residues, and have fewer than 1000 residues, proteins in the nucleus are disproportionately likely to be members of 2.7.7.6 (selectivity = 0.12, compared to background probability of 0.013, and sensitivity = 0.11). Tissue specificity plays a role in defining the trees of the enzyme family 1.6.99.3 (NADH dehydrogenases), as proteins in the thorax muscle disproportionately belong to the family (selectivity = 0.21, compared to background probability of 0.004, and sensitivity = 0.77). Twenty percent of deterministic root splits in the mixture for 1.6.99.3 are based on tissue specificity.

Some EC families are naturally annotated as having co-factors. For example, members of the 1.18.99.1 hydrogenases often have Iron-sulfur and Nickel as cofactors. These co-factors in themselves are not enough to distinguish family members from non-members[4], but inclusion of this information in decision rules can greatly increase their accuracy. For example, proteins that have Iron-sulfur as co-factors, have high CT dipeptide content, and are found in bacteria or viruses, are significantly enriched for membership in family 1.18.99.1 (see Figure 8). Another example is the co-factor Thiamine pyrophosphate. There are 30 enzyme families from four different (out of six) major EC groups that use Thiamine pyrophosphate as a co-factor (source: Biozon [Birkland & Yona 2006]). However, when this fact is combined with information about amino acid and Prosite domain content, it yields an effective classifier for family 1.2.4.1 (ROC50 score of 0.65 vs. 0.51 for best BLAST). Twenty percent of trees in the mixture for this family use the co-factor attribute in the root split of the tree, including 90% of the deterministic trees.

Another type of information that can be extracted from these trees is structural information. For example, family members of 1.10.3.2 (including laccases and urishiol oxidases) tend to have high predicted beta sheet content. Indeed, members of 1.10.3.2 with solved 3D structure all belong to "mainly beta" class in CATH and "all beta" class in SCOP. Again, by itself, this information is clearly inconclusive. However, proteins that have high predicted beta sheet content, above-average content of P and N residues, and are present in flowering plants, are significantly enriched for membership in 1.10.3.2 (selectivity = 0.30, compared to background probability of 0.001, and sensitivity = 0.97). A similar observation can be made about enzyme family 3.6.1.23. In this case, proteins low in predicted helix content, high in tryptophan, and present in flowering plants, are disproportionately likely to be members of 3.6.1.23 (selectivity = 0.10, compared to background probability of 0.002, and sensitivity = 0.85).

Many such rules can be extracted from the set of trees. It must be emphasized that only on rare occasions is a single rule sufficient to produce a highly-accurate classifier (such as with family 1.14.14.1, described above). It is the combination of many rules in the mixture of stochastic decision trees that yields superior performance over BLAST. However, the examples given in this section illustrate that the non-sequence attributes used in our algorithm do contribute substantially to the overall performance of the mixtures.

[4]In fact, there are more than 30 different enzyme families that use Nickel as co-factor (source: Biozon [Birkland & Yona 2006]).

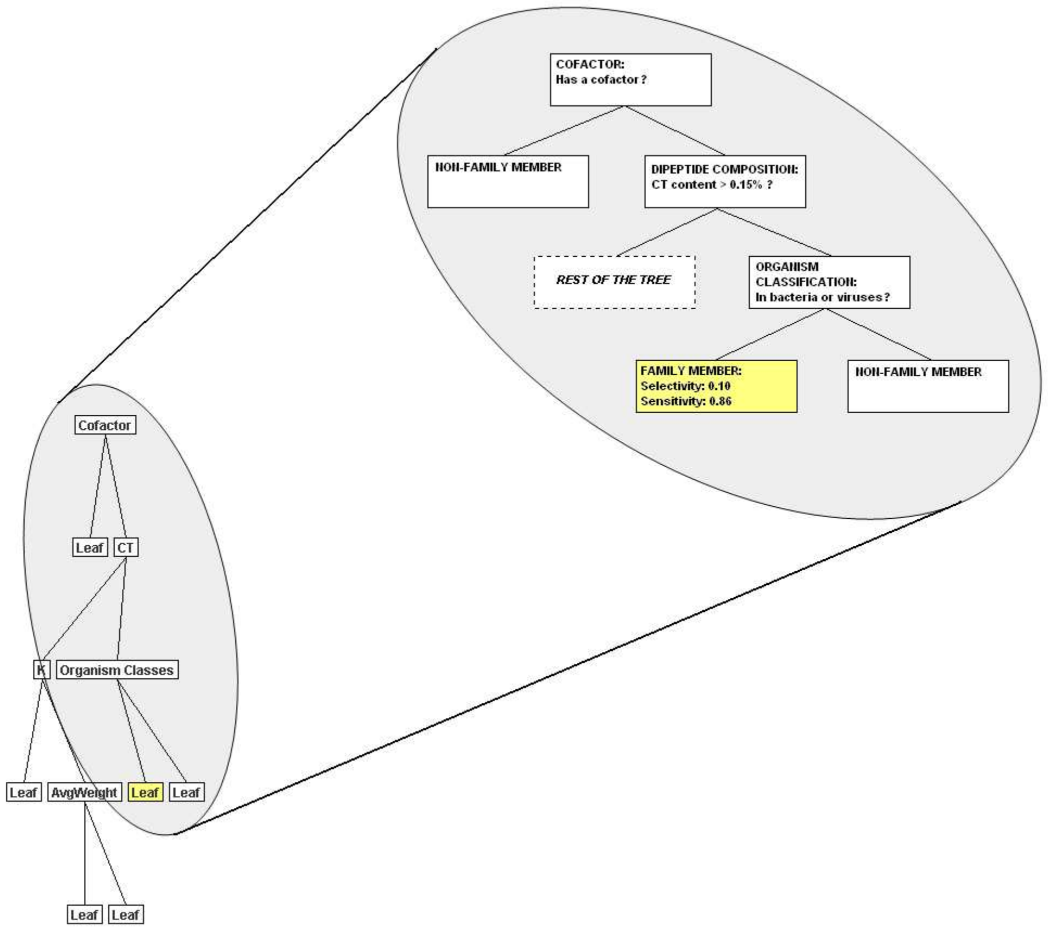


Figure 8: **A decision tree for EC family 1.18.99.1**. The complete tree is shown at lower-left. The part of the tree discussed in the text has been enlarged. The highlighted rule detects members of 1.18.99.1 with 10% selectivity (compared to background probability of 0.001) and 86% sensitivity.

7 Conclusions

The gene data that is available today presents us with a major challenge: how to decipher the rules that define protein functionality. Obviously the sequences of these proteins have a bearing on this question. According to the central dogma of molecular biology, it is the protein sequence that dictates the structure of the protein and this structure in turn prescribes the biological function of the protein. Moreover, sequence similarity can suggest common heritage and similar functions. Traditionally, sequence analysis has played a significant role in predicting gene function. However, there is a growing number of protein families that cannot be defined based on sequence similarity, and are linked and grouped based on other features. Some may argue that the tools that are available today for sequence analysis already exhaust sequence information. Indeed, in some cases it seems that new evidence must be presented, either in the form of structural similarity, expression data or protein interaction data before relationships can be established.

Here we argue that there is more information to utilize from sequence data. Our ability to analyze an entity and explore its relationships with other entities really depends on the representation we employ for that entity. Traditionally, proteins were analyzed as sequences of amino acids. This fairly simple representation was complex enough to detect many relationships. But once the order of amino acids was eliminated much of the information was lost. In this paper we present a new representation for proteins, through an extensive set of attributes, and introduce a novel model to decipher the regularities that distinguish family members from non-members. The model uses only sequence data and features extracted from database records, but can detect subtle principles that cannot always be established based on pure sequence similarity.

The set of attributes covers a broad range of protein features, from basic compositional properties to physio-chemical features, predicted features that hint at the topology and the shape of the protein structure, and database attributes that provide additional information about the subcellular location of the protein, its tissue preferences and other features that are likely to be linked to protein functionality. This set of features can be extended as more data becomes available. The combination of nominal attributes with numeric attributes in our model suggested the use of decision trees, one of the very few machine learning techniques that can handle mixed data. In addition to being suited for our learning task, the model can tolerate noise and missing data, as is often the case with database attributes. However, even the most recent learning algorithms for decision trees proved unsuccessful and the accuracy of the learned models was insufficient for effective prediction. Therefore we embarked on developing a new model, based on the decision tree model, that we call stochastic decision trees. In addition, we introduce several other components and options in the learning algorithm, such as different pruning methods and different weighting schemes. To find the optimal learning strategy we search through the space of decision tree learning algorithms until we converge to a locally optimal strategy. Four main elements characterize our final model: (1) dynamic feature and value selection for large feature and value sets, (2) probabilistic attribute selection protocol, (3) the use of a mixture of trees, and (4) the use of positive and negative sample divergence properties in all stages of tree learning, from pruning to tree weighting and evaluation. We use the Jensen-Shannon divergence measure to assess the trees in terms of the separation they induce over the data set between the positive and negative samples (members and non-members). Trees that maximize the separation are assigned a higher score (weight) and play a more dominant role in prediction. Thus the learning process essentially maximizes the buffer between the positive and negative samples. One might notice the similarity with kernel-based methods such as Support Vector Machines that try to maximize the margins around the separating hyper-planes. A good alternative to the JS-based pruning is the MDL-based approach that is described in the Appendix (section 9.2). This approach compensates for some of the deficiencies of validation-based post-pruning and is especially useful with small datasets, where not enough samples are available for training and validation.

To assess our model we evaluated it over two well known classifications; the Pfam sequence-based classification and the EC function-based enzyme classification. The results were compared to the popular BLAST algorithm and SAM, an HMM-based alignment program. Indeed our model compares favorably with BLAST and SAM over the Pfam data set, but the power of our model is more pronounced when the protein family is defined based on function, as in the EC database, rather than just based on sequence. Although for most EC families sequence similarity is still the most dominant factor in their definition, these families are less conserved than Pfam families, and in some cases are composed of multiple sequence subfamilies. Learning such families is a complex task, especially when the subfamilies do not seem to be related and cannot be reliably aligned. As was demonstrated here, our method can be applied successfully to predict the enzyme class of a protein, a task that sequence similarity-based methods often perform poorly on.

One of the advantages of our method is that the sequences need not be aligned. The model can learn the features common to a diverse set of proteins of shared function, sometimes without even evident sequence similarity. When these features are clearly a property of the protein family and are not found in other sequences then they serve as good predictors and are integrated into the model.

There are several modifications that may improve performance. One modification that we are considering is to modify the pruning algorithm and switch from a local approach to a global approach (where all nodes are considered before deciding on the node to be pruned). Another modification would be to assign probabilities to attribute values, since for some nominal attributes it is possible to quantify the likelihood of the different values. Clearly, integration of other features can refine the models, and finally, boosting techniques can also help to improve the performance.

8 Acknowledgments

This work is supported by the National Science Foundation under Grant No. 0133311 to Golan Yona.

References

[Altschul et al. 1997] Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W. & Lipman, D.J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.* **25**, 3389-3402.

[Bairoch & Apweiler 1999] Bairoch, A. & Apweiler, R. (1999). The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucl. Acids Res.* **27**, 49-54.

[Bateman et al. 1999] Bateman, A., Birney, E., Durbin, R., Eddy, S. R., Finn R. D., & Sonnhammer E. L. (1999). Pfam 3.1: 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucl. Acids Res.* **27**, 260-262.

[Ben-Hur & Brutlag 2006] Ben-Hur, A. and Brutlag, D. L. (2005). Protein sequence motifs: Highly predictive features of protein function. In: *Feature extraction, foundations and applications.* Guyon, I., Gunn, S., Nikravesh, M. & Zadeh, L. (eds.) Springer Verlag.

[Birkland & Yona 2006] Birkland, A. & Yona, G. (2006). The BIOZON Database: a Hub of Heterogeneous Biological Data. *Nucl. Acids Res.* **34** D235-D242.

[Black & Mould 1991] Black, S.D. & Mould, D.R. (1991). Development of Hydrophobicity Parameters to Analyze Proteins Which Bear Post or Cotranslational Modifications. *Anal. Biochem.* **193**, 72-82.

[Bono et al. 1998] Bono, H., Ogata, H., Goto, S. & Kanehisa, M. (1998). Reconstruction of Amino Acid Biosynthesis Pathways from the Complete Genome Sequence. *Genome Res.* **8** 203-210.

[Borro et al. 2006] Borro, L. C., Oliveira, S. R. M., Yamagishi, M. E. B., Mancini, A. L., Jardine, J. G., Mazoni, I., dos Santos, E. H., Higa, R. H., Kuser P. R. & Neshich G. (2006). Predicting enzyme class from protein structure using Bayesian classification. *Genet. Mol. Res.* **5**, 193-202.

[Breiman 2001] Breiman, L. (2001). Random forests. *Machine Learning* **45**, 5-32.

[Breiman et al. 1984] Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1993). "Classification and Regression Trees". Wadsworth Int. Group, Belmont, California,

[Breiman et al. 1993] Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1993). "Classification and Regression Trees". Chapman & Hall, New York.

[Burges 1998] Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* **2**, 121-167.

[Cai & Chou] Cai, Y-D. & Chou, K-C. (2004). Using functional domain composition to predict enzyme family classes. *J. Proteome Res.* **4**, 109-111.

[Casari et al. 1995] Casari, G., Sander, C. & Valencia, A. (1995). A method to predict functional residues in proteins. *Nat. Struct. Biol.* **2**, 171-178.

[Caspi et al. 2006] Caspi, R., Foerster, H., Fulcher, C. A., Hopkinson, R., Ingraham, J., Kaipa, P., Krummenacker, M., Paley, S., Pick, J., Rhee, S. Y., Tissier, C., Zhang, P. & Karp, P. D. (2006). MetaCyc: A multiorganism database of metabolic pathways and enzymes. *Nucl. Acids Res.* **34**, D511-D516.

[Chen & Vitkup 2006] Chen, L. & Vitkup, D. (2006). Predicting genes for orphan metabolic activities using phylogenetic profiles. *Genome Biology* **7**, R17.

[Clare & King 2003] Clare, A. & King R. D. (2003). Predicting gene function in Saccharomyces cerevisiae. *Bioinformatics* **19**, ii42-ii49

[desJardins et al. 1997] desJardins, M., Karp, P. D., Krummenacker, M., Lee, T. J., Ouzounis, C. A. (1997). Prediction of enzyme classification from protein sequence without the use of sequence similarity. *In the proceedings of ISMB 1997* 92-99.

[Devos & Valencia 2000] Devos, D. & Valencia, A. (2000). Practical limits of function prediction. *Proteins: Structure, Function, and Genetics* **41**, 98-107.

[Dietterich 2000] Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning* **40**, 139-157.

[Duda et al. 2000] Duda, R. O., Hart, P. E. & Stork, D. G. (2000). "Pattern classification". John Wiley and Sons, New York.

[Emmanuel et al. 2005] Levy, E. D., Ouzounis, C. A., Gilks, W. R. and Audit, B. (2005). Probabilistic annotation of protein sequences based on functional classifications. *BMC Bioinformatics* **6**, 302.

[EC] http://www.chem.qmw.ac.uk/iubmb/enzyme/

[Eskin et al. 2000] Eskin, E., Grundy, W. N. & Singer, Y. (2000). Protein Family Classification using Sparse Markov Transducers. *In the proceedings of ISMB 2000*, 20-23.

[Fayyad & Irani 1993] Fayyad, U. M. & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. of the 13th int. conf. on AI*, 1022-1027. Morgan Kaufmann, San Mateo, California.

[Ferran et al. 1994] Ferran, E. A., Pflugfelder, B. & Ferrara P. (1994). Self-Organized Neural Maps of Human Protein Sequences. *Protein Sci.* **3**, 507-521.

[GO Consortium] The Gene Ontology Consortium. (2000). Gene Ontology: tool for the unification of biology. *Nat. Genetics* **25**, 25-29.

[Green & Karp 2004] Green, M. & Karp, P. D. (2004). A Bayesian method for identifying missing enzymes in predicted metabolic pathway databases. *BMC Bioinformatics* **5**, 76.

[Han et al. 2004] Han, L. Y., Cai, C. Z., Ji, Z. L., Cao, Z. W., Cui, J. & Chen, Y. Z. (2004). Predicting functional family of novel enzymes irrespective of sequence similarity: a statistical learning approach. *Nucl. Acids Res.* **32**, 6437-6444.

[Hjorth 1994] Hjorth, J. S. U. (1994). "Computer intensive statistical methods validation, model selection, and bootstrap". Chapman & Hall, London.

[Ho 1998] Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **20**, 832-844.

[Hobohm & Sander 1995] Hobohm, U. & Sander, C. (1995). A sequence property approach to searching protein database. *J. Mol. Biol.* **251**, 390-399.

[Holm & Sander 1994] Holm, L. & Sander, C. (1994). The FSSP database of structurally aligned protein fold families. *Nucl. Acids Res.* **22**, 3600-3609.

[Hughey et al. 1999] Hughey, R., Karplus, K. & Krogh, A. (1999). SAM: Sequence alignment and modeling software system. Technical report UCSC-CRL-99-11. University of California, Santa Cruz, CA.

[IonSource] http://www.ionsource.com/virtit/VirtualIT/aainfo.htm

[Jaakola et al. 1999] Jaakola, T., Diekhans, M. & Haussler, D. (1999). Using the Fisher kernel method to detect remote protein homologies. *In the proceedings of ISMB 1999* 149-158.

[Jain et al. 1998] Jain, A. K. , Dubes, R. C. & Chen, C. (1998). Bootstrap techniques for error estimation. *IEEE Transactions on Pattern Analysis and Applications* **9**, 628-633.

[Kanehisa & Goto 2000] Kanehisa, M. & Goto, S. (2000). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucl. Acids Res.* **28**, 27-30.

[Kharchenko et al. 2006] Kharchenko, P., Chen, L., Freund, Y., Vitkup, D. & Church, G. M. (2006). Identifying metabolic enzymes with multiple types of association evidence. *BMC Bioinformatics* **7**, 177.

[Kohavi & Sahami 1996] Kohavi, R. & Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. *Second International Conference on Knowledge Discovery and Data Mining*, 114-119.

[Kolesov et al. 2001] Kolesov, G., Mewes, H. W. & Frishman, D. (2001). SNAPping up functionally related genes based on context information: a colinearity-free approach. *J. Mol. Biol.* **311**, 639-656.

[Kononenko 1995] Kononenko, I. (1995). On biases in estimating multi-valued attributes. In *Int. Conf. on AI.* 1034-1040.

[Kullback 1959] Kullback, S. (1959). "Information theory and statistics". John Wiley and Sons, New York.

[Leslie et al. 2004] Leslie, C., Eskin, E., Cohen, A., Weston, J. & Noble, W. S. (2004). Mismatch string kernels for discriminitive protein classification. *Bioinformatics* **1**, 1-10.

[Lin 1991] Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Trans. Info. Theory* **37:1**, 145-151.

[Mantaras 1991] Mantaras, R. L. (1991). A Distance-based Attribute Selection Measure for Decision Tree Induction. *Machine Learning* **6**, 81-92.

[McGuffin et al. 2000] McGuffin, L. J. , Bryson, K. & Jones, D. T. (2000). The PSIPRED protein structure prediction server. *Bioinformatics* **16**, 404-405.

[Mewes et al. 1999] Mewes, H. W., Heumann, K., Kaps, A., Mayer, K., Pfeiffer, F., Stocker, S. & Frishman, D. (1999). MIPS: a database for genomes and protein sequences. *Nucl. Acids Res.* **27**, 44-48.

[Mitchell 1997] Mitchell, T. M. (1997). "Machine Learning". McGraw-Hill.

[Murzin et al. 1995] Murzin A. G., Brenner S. E., Hubbard T., Chothia C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* **247**, 536-540.

[Paley & Karp 2002] Paley, S. M. & Karp, P.D. (2002). Evaluation of computational metabolic-pathway predictions for Helicobacter pylori. *Bioinformatics* **18**, 715-724.

[Pearson 1995] Pearson, W. R. (1995). Comparison of methods for searching protein sequence databases. *Protein Sci.* **4**, 1145-1160.

[Popescu & Yona 2005] Popescu, L. & Yona, G. (2005). Automation of gene assignments to metabolic pathways using high-throughput expression data. *BMC Bioinformatics* **6** 217.

[Popescu & Yona 2006] Popescu, L. & Yona, G. (2006). Expectation-Maximization algorithms for fuzzy assignment of genes to cellular pathways. In proceedings of the *Computational Systems Bioinformatics*.

[Quinlan 1986] Quinlan, J.R., (1986). Induction of decision trees. *Machine Learning.* **1**, 81-106.

[Quinlan 1993] Quinlan, J.R., (1993). "C4.5: Programs for Machine Learning". Morgan Kaufmann.

[Rissanen 1989] Rissanen, J. (1989). Stochastic Complexity in Statistical Inquiry. *World Scientific*.

[Rost 2002] Rost, B. (2002). Enzyme function less conserved than anticipated. *J. Mol. Biol.* **318**, 595-608.

[Shah & Hunter 1997] Shah, I. & Hunter, L. (1997). Predicting enzyme function from sequence: a systematic appraisal. *In the proceedings of ISMB 1997* 276-283.

[Shakhnarovich et al. 2001] Shakhnarovich, G. , El-Yaniv, R. & Baram, Y. (2001). Smoothed bootstrap and statistical data cloning for classifier evaluation. *ICML-2001*

[Stawiski et al. 2000] Stawiski, E. W. , Baucom, A. E. , Lohr, S. C. & Gregoret, L. M. (2000). Predicting protein function from structure: unique structural features of proteases. *Proc. Natl. Acad. Sci. USA* **97**, 3954-3958.

[Swiss-Prot] http://www.expasy.org/sprot/userman.html

[Syed & Yona 2003] Syed, U. & Yona, G. (2003). Using a mixture of probabilistic decision trees for direct prediction of protein function. *In the proceedings of RECOMB 2003* 289-300.

[Tian et al. 2004] Tian, W., Arakaki, A. K. & Skolnick, J. (2004). EFICAz: a comprehensive approach for accurate genome-scale enzyme function inference. *Nucl. Acids Res.* **32**, 6226-6239.

[Todd et al. 2001] Todd, A.E., Orengo, C.A. & Thornton, J.M. (2001). Evolution of function in protein superfamilies, from a structural perspective. *J. Mol. Biol.* **307**, 1113-1143.

[Wilson et al. 2000] Wilson, C. A., Kreychman, J. & Gerstein, M. (2000). Assessing annotation transfer for genomics: quantifying the relations between protein sequence, structure and function through traditional and probabilistic scores. *J. Mol. Biol.* **297**, 233-249.

[Wilson & Irwin 1999] Wilson, D. B. & Irwin, D. C. (1999). Genetics and properties of cellulases. *Adv. Biochem. Eng.* **65**, 2-21.

[Yaminishi et al. 2005] Yaminishi, Y., Vert, J. & Kanehisa, M. (2005). Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics* **21**, i468-i477.

9 Appendix

9.1 The extended training procedure for decision trees

Our extended decision tree learning algorithm includes the following elements:

9.1.1 Dynamic attribute filtering

In the case of dipeptide composition, the number of attributes is too large to be used exhaustively during training. To narrow the scope of data that will be input to the decision tree algorithm, we need a filtering criterion that can examine the training data and retain only those attributes that seem important. This importance is estimated by computing the ratio of the frequency of a dipeptide among family members to its frequency among all non-family members. For example, for an attribute like 'AA', this ratio is the percent composition of 'AA' in family member sequences divided by the percent composition of 'AA' in non-family member sequences. Note that these calculations are restricted to the training set only. We retain only those dipeptides for which this ratio is one standard deviation from the mean. However, to keep the computation feasible, we used at most the top 30 such dipeptides.

9.1.2 Discretizing numerical features

In a decision tree, numerical features must be divided into subranges before they can be used to split a node. Many researchers have approached this problem and the best solution is by no means clear. The Fayyad-Irani heuristic [Fayyad & Irani 1993] is an efficient and flexible method for discretizing numerical attributes. It is a greedy recursive protocol, inspired by the Minimum Description Length principle, that produces a partition of k-arity, where k is selected by the protocol itself. Given a k-ary partition, the algorithm splits it into a (k+1)-ary partition by examining every remaining cut point and selecting the one that maximizes an MDL-like criterion function [5]. If its value is less than zero for every remaining cut point, the protocol terminates and the current partition is used. Although others have used this protocol to globally partition numerical attributes [Kohavi & Sahami 1996], we use it at every decision node to induce local partitions of the numerical attributes based on the given subset of sample points at that node, thus adjusting the decision to the subset of data available. The protocol can also be constrained to produce only binary splits (see **Binary Splitting**, below).

9.1.3 Binary splitting

While many of our attributes are naturally suited to having multiple values, we wanted to test the impact of forcing all splits to be binary. This preserves training data for later splits, and consequently may improve accuracy, particularly since a very small fraction of our data set represents class (family) members. Moreover, binary splits are not susceptible to the biased entropy of many-way splits (see **Selection measures** below). With numerical attributes, the most straight-forward procedure is also the most efficient and exhaustive: simply restrict discretization to exactly 2 subranges (i.e. a single cut point). For a particular attribute, the optimal split can found in $O(N)$ time, where N is the number of possible cut points [Fayyad & Irani 1993]. However, for nominal attributes, since there is no ordering over the values, the exhaustive procedure of testing every possible binary partition of the values leads to a time complexity of $O(2^d)$ where d is the number of possible values for the attribute $d = |Values(A)|$. There are at least two ways to deal with this problem. The most naive way is to examine only those binary partitions of $Values(A)$ in which one value is

[5] Specifically, the following formula is used

$$Gain(S, A, T) - \frac{\log_2(N-1)}{N} - \frac{\Delta(S, A, T)}{N}$$

where N is the number of samples, T is the cut point and

$$\Delta(S, A, T) = \log_2(3^k - 2) - [k \times Entropy(S) - k_1 \times Entropy(S_1) - k_2 \times Entropy(S_2)]$$

with k, k_1 and k_2 being the number of classes in each set (2 in our case). For more details see [Fayyad & Irani 1993].

assigned to one branch and all other values are assigned to the other branch. This is functionally equivalent to converting each value of A into a separate, purely binary attribute, and it reduces the time complexity to $O(d)$.

A more sophisticated partitioning algorithm is found in CART [Breiman et al. 1984]. Our problem can be expressed as having to find a subset of $Values(A)$ to assign to one child, such that $Gain(S, A)$ is maximized. Let each value $v_i \in Values(A)$ correspond to a subset S_i of S, which will consist of the examples that have v_i as a value. Sort the v_i's in order of increasing class purity of the S_i's. In this new order, the optimal subset is the union of $S_0..S_k$, where k is some number in $0..d$. Therefore, one can find the optimal binary split in only $O(d)$, excluding sorting time. This optimal partitioning algorithm is the one we use during tree training.

9.1.4 Multiple values for attributes

The nominal database attributes that we use are somewhat different than those typically found in the literature on decision trees. Each example can take on several values for each nominal attribute, instead of just one. For example, a protein can be abundant in the liver as well as in heart tissues, and the number of possible combinations is huge. During training, when a node is split using a nominal attribute A, the set $Values(A)$ is partitioned across the node's children. With traditional nominal attributes, each example would be assigned to the unique child that corresponds to its value. In our model, however, each example may correspond to several children. To overcome this problem we divide the example itself across all children, weighting each child's portion by the number of values the example has in common with that child.

9.1.5 Missing attributes

Sometimes, an individual example will lack the attribute being tested by a decision node. We adopt the approach implemented in the C4.5 algorithm to handle these samples, and partition them across the child nodes maintaining the same proportions as the rest of the training examples (i.e. the proportions are calculated using only samples that have reached the splitting node and that have known values for the test attribute). These fractions can be partitioned again and again with every unknown attribute. Consequently, the ambiguous examples will reach several leaf nodes during classification. In these cases, the final probability is defined as the average of all leaf probabilities, weighted by the fraction of the example that reached each leaf.

9.1.6 Selection measures

The C4.5 algorithm improves upon ID3 by using $GainRatio(S, A)$ [Quinlan 1993], which is $Gain(S, A)$ normalized by the *split information*:

$$SplitInfo(S, A) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

Note that this expression is just the information content of the attribute itself. This modification is meant to compensate for the bias of $Gain(S, A)$, which tends to be higher for attributes with more values. However, $GainRatio(S, A)$ is itself less than ideal, since it may be biased toward attributes with very low $SplitInfo(S, A)$ instead high $Gain(S, A)$. To counter this, we also tested a closely-related distance metric introduced by [Mantaras 1991], who showed formally that his measure does not suffer from the limitations of $Gain(S, A)$ or $GainRatio(S, A)$ (though there is empirical evidence that it is not guaranteed to eliminate the bias altogether [Kononenko 1995]). To simplify notation, in the following sections we refer to both *GainRatio* and the $Mantaras$ selection functions as **information gain**, unless specified otherwise. Note that these measures are only used when computing multi-branch splits, since their purpose is to combat high arity splitting. The basic $Gain(S, A)$ is sufficient when dealing with binary splits.

9.1.7 Handling skewed distributions

When trying to discern protein families, the overwhelming majority of the training set is composed of negative examples (usually more than 90%). As a result, one may run into problems when trying to learn models of small families. Because of its greedy nature, most of the learning phase will concentrate on learning the regularities in the negative examples that result in significant information gain. Not only will this process cause extended learning times but it may also totally miss subtle regularities observed in the positive samples. The simple solution of presenting only a small set of negative samples equal in size to the set of positive samples is insufficient because some regularities observed in the positive set may be mistakenly considered as discriminating regularities, while if considered along with the whole set of negative examples they may prove to be uninformative.

To overcome this problem, we adopted a different approach, where all the negative examples are presented, but their sum is given the same weight as the sum of all the positives examples (as suggested in [Eskin et al. 2000]). Thus, every positive example contributes $1/\#positives$ and every negative example contributes $1/\#negatives$ to the total counts, and are weighted accordingly in the calculations of the information gain.

However, by doing so we introduce another problem. It may happen that we will end with a tree of high purity (when using the weighted samples), but of high impurity when considering the original unweighted samples, thus increasing significantly the rate of false positives. We tested two possible solutions that attempt to balance the two approaches. In the first, the criterion function for splitting and pruning is the average of the weighted-sample information gain with the unweighted-sample information gain (this protocol is called **mixed entropy**). The second solution starts by training the trees using the weighted samples. The usage of unweighted-sample entropy is delayed until the maximal information gain over all possible attributes at a given node is below a certain threshold. At that point each attribute is reconsidered using unweighted entropy and the training proceeds for as long as the information gain is below the threshold (we call this protocol **interlaced entropy**).

Note that the weighted samples are used only during the calculations of the information gain. For all other purposes, including pruning and prediction, the probabilities based on the unweighted samples are used.

9.1.8 Leaf weighting

Not all leaf nodes are good predictors and different nodes may have different accuracies, when tested on new samples. This is especially true if the number of samples in a node is small. To address that we tested also a variant where weights are associated with leaf nodes, and the final prediction is adjusted accordingly. The weights are trained using the perceptron learning algorithm [Duda et al. 2000], until a local minimum of the performance is achieved over the training set.

9.2 Post-pruning by the MDL principle

One of the major problems of validation-based post-pruning methods is that a sufficiently large number of positives in the validation set are required to ensure high generalization power. Otherwise, many true regularities learned from the positives in training set may be pruned, since they are not supported by the small validation set. The effect of post-pruning on decision trees in such cases can be drastic.

With very small data sets, where every single sample is important, one might want to consider alternative pruning methods that use all the given data. The chi-squared test, used in a pre-pruning context, is one possibility. However, it may suffer from what is called the *horizon effect*[6]. Here we use an alternative method based on the **minimum description length principle** [Rissanen 1989], abbreviated as MDL.

The MDL principle is a common heuristic for selecting the most probable or valid model. It is based on an argument that was postulated by Occam in the 14th century. According to this argument (called *Occam's razor*), given two models that explain an observation equally well, it is advised to select the simpler model, under the assumption that it will generalize better to new examples. Therefore, the most probable model (of all possible models) for a given data set is the one that minimizes the complexity of the model while maximizing its ability to explain the data. This concept is captured by the **description length**. The description length of a given **model (hypothesis)** *and* **data** is defined as the description length of **the model** plus the description length of the **data given the model**.

[6]This refers to the phenomenon where decision node splits that initially appear poor in fact lead to better performance later on during the training process. Any prepruning technique will be susceptible to this kind of short-sightedness [Duda et al. 2000].

One way to define the description length of a model is by its algorithmic complexity (Kolmogorov complexity). However, this approach is not very practical, since to define the algorithmic complexity of a model, one has to find the shortest program that encodes this model, something that is hardly ever possible.

A more practical method, that formulates the same principle in probabilistic terms, is the Bayesian approach. Under this approach, the most probable model is the one that maximizes the posterior probability of the model given the data

$$P(h/D) = \frac{P(h)P(D/h)}{P(D)}$$

The optimal hypothesis h^* is the one that maximizes $P(h/D)$ or, equivalently, minimizes $-P(h/D)$

$$\begin{aligned} h^* &= \arg\min_h [\frac{-P(h)P(D/h)}{P(D)}] \\ &= \arg\min_h [-P(h)P(D/h)] \\ &= \arg\min_h [-\log_2 P(h) - \log_2 P(D/h)] \end{aligned}$$

By Shannon's theorem, $-\log_2 P(x)$ is related to the shortest description of a string x (in our case the model h), and the likelihood $-\log_2 P(D/h)$ is a measure for the description of the data given the model. Therefore the MDL principle and the Bayesian approach as formulated above are practically the same.

To approximate the first term we calculate the binary encoding of a tree. For each node j we devote $\log N + \log k_j$ bits, where N is the number of attributes in our model and k_j is the number of possible values for the attribute associated with node j. In this way, the total description length of a tree is approximated by the sum of the descriptions of its nodes. Note that the number of possible trees increases exponentially with the description length, therefore the probability of each tree decreases exponentially. Thus, the logarithm as applied to the probability of the tree generates a term that is linear in the binary encoding of the tree, as desired.

To calculate the second term we used two different approaches. The first, called **probability based MDL post-pruning**, calculates $P(D/h)$ based on the probabilities the model assigns to each sample. Specifically,

$$P(D/h) = \prod_{i=1}^{n} P(+/x_i) = \prod_{i=1}^{n} \sum_{j \in leaves(T)} f_j(x_i) Prob_j(x_i)$$

where $f_j(x)$ is the fraction of x_i that reaches leaf j and $Prob_j(x_i)$ is the probability assigned to x_i by that leaf[7]. Intuitively, the likelihood term $\log P(D/h)$ accounts for the uncertainty that is left in the tree. With probability of 1 the uncertainty is zero, or in other words, the tree describes the data perfectly. Otherwise, some uncertainty exists and we must include in the description the variance or the deviation of the data from the tree, since only a combined description of the tree and the variance will enable complete recovery of the data set. Our second approach, called **entropy-based MDL post-pruning**, attempts to directly estimate the total uncertainty that remains in the tree by measuring the total entropy in all the leaf nodes. Our tests indicate that the probability-based approach outperforms the entropy-based approach.

Finally, the two terms are combined together into a single measure. Although the MDL heuristic relies on sound arguments, the exact formulation requires some adjustment and one usually needs to introduce a factor α to fix the scale such that

$$DescLen = ModelDesc + \alpha \cdot Likelihood \tag{3}$$

To determine α, we study the distribution of uncertainties in the trained trees versus the description lengths of the nodes, and set the range of possible values for α so that they scale similarly. The exact value is selected based on performance optimization (see Figure 9).

To find the optimal tree, using this new pruning method, we train a complete tree and then prune it in an exhaustive fashion, similar to the post-pruning method described in section 3.3, but using this new MDL criterion function. The elimination of the validation set provides us with more training samples at our

[7]The reason that a fraction of a sample can reach a node is related to the problem of missing values and is explained in section 9.1.5 of the Appendix

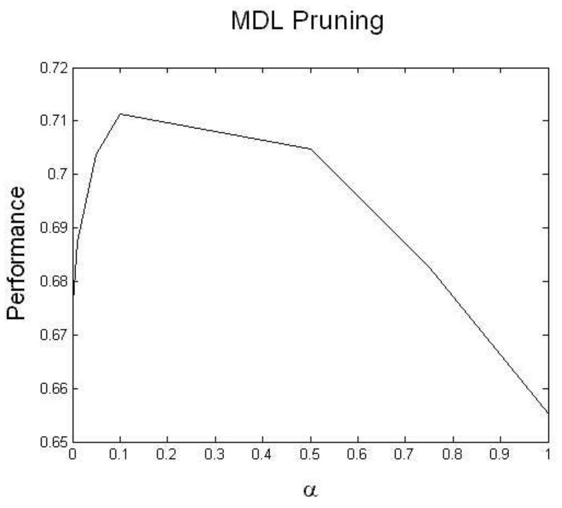


Figure 9: **The minimum description length**. The description length consists of two elements, and the relative contribution of each is determined by the weight α (see text). We tested a range of possible values for α over a random subset of 50 PFAM families, to determine the one that maximizes the performance. For each value of α we trained a new set of trees and measured the average performance over the subset. The graph plots the average performance as a function of the value of α.

disposal. However, it also eliminates a stochastic element that can increase robustness, as it unifies all 10 training sets used by the post-pruning procedure into one. To compensate for the missing validation set in that aspect, we re-introduce 10 training sets by using the bootstrapping sampling method [Hjorth 1994] of the learning set. Specifically, each training set of size n is generated by selecting instances from the learning set at random, with repetitions. Theoretical results [Jain et al. 1998] indicate that models estimated from bootstrap data sets will converge to the optimal model at the limit of infinite bootstrap data sets. Moreover, the bootstrap method was proved to be more robust in providing reliable estimates of performance in the presence of small data sets [Shakhnarovich et al. 2001].

With the MDL-based pruning and evaluation, the final output of the mixture model is given by

$$P_{Mixture}(+/x) = \frac{\sum_{training\ set\ j} \sum_{i \in Trees(j)} Q_{MDL}(i) P_i(+/x)}{\sum_{training\ set\ j} \sum_{i \in Trees(j)} Q_{MDL}(i)}$$

where $Q^{MDL}(i)$ is defined as $1/MDL(i)$ and $MDL(i)$ is the description length of the i-th tree.

9.3 Setting the threshold for evaluation and prediction

When evaluating the performance of a model that assigns probabilities to samples one needs a clear definition of positives and negatives. Usually a threshold T is defined, and if the sample probability exceeds this threshold, then the sample is defined as positive.

How should one define the threshold T? We tested two different approaches. The naive approach would be to take a majority vote, so that all samples with probability higher than 0.5 are defined as positive. However, this may be misleading, since the number of positives and negatives may differ significantly to begin with. A more sensible approach would be to take into account the prior probabilities of positives $P_0(+)$ and negatives $P_0(-)$, and set the threshold to $P_0(-)$. To account for random fluctuations due to the varying sample sizes at the nodes, a different significance threshold T_j is calculated for each node. Specifically, if the total number of samples in a leaf node j is N_j then the number of positives that will reach this node by chance (the null hypothesis) is expected to follow a binomial distribution with mean $N_j \times P_0(+)$ and variance $N_j \times P_0(+) \times P_0(-)$. Using the normal approximation for the binomial distribution, we set the threshold at the 95% percentile of the distribution, and if the node probability $P_j(+)$ exceeds this threshold then we define this instance to be positive. Thus, each example is classified unambiguously in a discrete manner, and the accuracy is computed thereby. If the sample reaches a single leaf node, then only the confidence interval for this node is used to assign a label. If it reaches more than one node, a "combined" node is created, where the datasets are collected from the different leaf nodes, weighted by the fractions of the sample that reach each leaf node, and the sample probabilities, the threshold and the label are calculated accordingly. We refer to this approach as the **confidence-based** approach.

The second approach uses the equivalence point criterion. All sequences in the database are sorted according to the output assigned by the model. The equivalence point is the point where the number of false positives equals the number of false negatives [Pearson 1995]. All proteins that are predicted with higher probability are labeled as positives. In other words, the equivalence point is the point that balances sensitivity and selectivity. We refer to this approach as the **equivalence point** approach.

Attribute	Information gain	Depth
OrganismClasses	0.390	5.6
AvgHydro	0.289	6.4
Length	0.259	6.3
NegCharge	0.253	6.5
Acid	0.253	6.7
Aromatic	0.245	6.9
HydrophobicArom	0.238	7.0
HydrophilicAcid	0.225	7.0
AvgWeight	0.223	6.9
Bulky	0.210	7.2
PosCharge	0.209	6.9
HydrophilicBase	0.209	6.9
E	0.209	7.2
Charge	0.207	6.8
Hydrophobic	0.202	6.9
PredH	0.200	6.7
AvgIso	0.196	7.2
Polar	0.191	6.7
W	0.190	7.4
R	0.190	7.4
Aliphatic	0.187	6.8
PrositeDomains	0.185	7.5
PredC	0.184	6.9
D	0.182	7.4
Y	0.179	7.6
Tiny	0.176	7.0
L	0.172	7.4
G	0.168	7.3
K	0.166	7.5
C	0.163	7.4
PredE	0.162	7.0
Weak	0.161	7.2
Small	0.160	7.3
F	0.155	7.5
Proline	0.147	7.4
P	0.147	7.4
A	0.142	7.3
V	0.141	7.6
T	0.135	7.7
PolarUn	0.127	7.5
N	0.121	7.7
H	0.121	7.8
I	0.110	7.7
S	0.099	7.9
Species	0.098	6.7
Q	0.098	7.9
M	0.098	7.9
Catalytic	0.089	8.4
Cofactor	0.039	9.7
Subcell	0.035	6.8
RCTissue	0.011	7.5
Alternate	0.006	10.7
TissueSpec	0.002	6.9

Table 4: **Attribute information - part 1 (excludes dipeptides).** Attributes are sorted by decreasing information gain with respect to that attribute only (second column). The results are averaged over all families. Since throughout the learning phase we use the weighted entropy (see section 9.1.7), the information gain in this case is given over the weighted samples. These numbers are also more effective for identifying informative attributes; because of the skewed distribution of positives and negatives the background unweighted entropy is very small to begin with and significant improvement in prediction power will translate only to slight improvements in the unweighted entropy (the non-linearity of the entropy function further obscures significant changes in classification accuracy). The third column is the average depth in the trees at which the attribute is used. The depth of the root node is 0; the average depth of the bottom of the trees is 16.83.

Attribute	Information gain	Depth
DP	0.192	7.9
GG	0.180	6.6
YG	0.171	4.8
EL	0.167	5.5
LD	0.166	—
GA	0.166	5.4
DG	0.164	8.4
GL	0.163	6.6
PL	0.162	—
IG	0.161	5.9
GR	0.160	6.9
LY	0.157	—
AG	0.157	6.9
LG	0.156	4.0
FL	0.155	7.1
FF	0.154	8.0
RR	0.149	6.6
EV	0.149	—
EG	0.149	9.2
EE	0.149	6.6
WL	0.148	7.7
QL	0.148	5.9
DV	0.148	—
VD	0.147	—
NG	0.147	3.8
LE	0.147	6.1
KK	0.147	7.0
EA	0.147	7.0
AV	0.147	—
NV	0.146	5.4
GK	0.146	7.9
DL	0.145	—
VR	0.144	5.2
IL	0.144	—
AI	0.144	7.1
AA	0.144	6.4
MP	0.143	7.1
GT	0.143	5.6
NP	0.142	8.6
LR	0.142	6.6

Table 5: **Attribute information - part 2 (dipeptides only).** See Table 4 for details. Missing depth information means that the dipeptide was never used in the mixture of stochastic decision trees. Only the top 40 dipeptide attributes are listed.

Attribute	Usage frequency (splits)	Usage frequency (trees)	Ratio
OrganismClasses	0.030	0.686	0.044
Length	0.033	0.646	0.051
PredH	0.023	0.547	0.042
AvgHydro	0.023	0.544	0.042
PredE	0.022	0.527	0.042
PredC	0.020	0.507	0.039
AvgIso	0.019	0.494	0.038
Hydrophobic	0.019	0.490	0.039
AvgWeight	0.019	0.490	0.039
Aromatic	0.019	0.489	0.039
Polar	0.019	0.478	0.040
HydrophobicArom	0.018	0.478	0.038
Charge	0.018	0.474	0.038
G	0.018	0.473	0.038
Aliphatic	0.018	0.467	0.039
NegCharge	0.017	0.462	0.037
L	0.017	0.462	0.037
Tiny	0.017	0.454	0.037
PosCharge	0.016	0.449	0.036
C	0.017	0.449	0.038
R	0.016	0.448	0.036
PrositeDomains	0.016	0.442	0.036
K	0.016	0.438	0.037
HydrophilicAcid	0.016	0.436	0.037
P	0.016	0.435	0.037
A	0.016	0.434	0.037
Acid	0.016	0.431	0.037
W	0.015	0.430	0.035
Small	0.016	0.429	0.037
Weak	0.015	0.423	0.035
E	0.015	0.422	0.036
HydrophilicBase	0.016	0.420	0.038
V	0.015	0.416	0.036
F	0.015	0.413	0.036
Y	0.015	0.412	0.036
I	0.015	0.411	0.036
D	0.015	0.409	0.037
Proline	0.015	0.406	0.037
S	0.014	0.404	0.035
PolarUn	0.014	0.404	0.035
T	0.014	0.401	0.035
H	0.014	0.399	0.035
N	0.014	0.393	0.036
Q	0.013	0.385	0.034
M	0.013	0.371	0.035
Species	0.009	0.291	0.030
Catalytic	0.007	0.256	0.027
Bulky	0.008	0.208	0.038
Cofactor	0.004	0.140	0.029
Subcell	0.002	0.075	0.027
RCTissue	0.001	0.057	0.018
Alternate	0.001	0.024	0.042
TissueSpec	0.0005	0.016	0.031

Table 6: **Attribute usage.** Percentage of *all splits* in the mixture of stochastic decision trees that use each attribute. The third column lists the percentage of *all trees* that used the attribute and the fourth is the ratio between the two. A higher ratio of percentage of splits to percentage of trees indicates multiple tests over the same attribute in the same tree (usually because of a range of values or multiple values where the information about family association is not limited to a single transition point).