Automata Learning and Stochastic Modeling for Biosequence Analysis

Thesis submitted for the degree
"Doctor of Philosophy"

Gill Bejerano

Submitted to the Senate of the Hebrew University in the year 2003

This work was carried out under the supervision of

Prof. Naftali Tishby and Prof. Hanah Margalit

Contents

Thesis Outline

Proteins play major and diverse roles in nearly all activities that take place within a living cell. A protein molecule is a linear polymer of amino acids joined by peptide bonds in a specific sequence. This sequence is encoded in the DNA of the cell in protein coding genes. The genes are transcribed from DNA to RNA, which is then translated to form the linear amino acid sequence. Proteins fold into specific three dimensional conformations. Most proteins perform their task, or tasks, through a combination of their specialized conformation and the physico-chemical properties of their constitutive amino acids.

Protein sequences can be easily determined these days, either directly using the protein molecule itself, or, more often, indirectly by sequencing the encoding genes. However, the structure of a protein is hard to observe experimentally, with current technology. Indeed, the structures of only a fraction of the known proteins has been solved to date. Given a solitary protein sequence, we also lack the capability, currently, of deducing *de novo* the conformation it adopts. And without a good structural model, it is difficult to speculate the functional role of the protein, which is our ultimate goal.

Different protein sequences both within a single organism, and between different organisms share noticeable similarities. Indeed the known proteins can be organized in a hierarchical manner, based on apparent sequence, structural and functional conservation. With many genome sequencing efforts spanning the globe, multitudes of novel and hypothetical genes are being discovered on a daily basis. Close to one million potential protein sequences are known to date, deeming careful manual analysis of this ever-growing set nearly impossible.

It is the goal of this thesis to present novel computational tools for the analysis of newly discovered protein sequences in several related aspects: Detection of the functional units, called domains, from which a protein is composed. Assignment of each such domain to a family of related proteins, all having instances of the same domain. And finally, assisting in fine-scale comparison of known protein sub-families that share very high sequence similarity, and yet perform somewhat different functions.

The body of this thesis focuses on the exploration and exploitation of Markovian dependencies in related protein sequences. Through the study and modeling of these dependencies,

- We show that a stationary variable memory Markov model (Ron et al., 1996) can capture the notion of a protein sequence family (Bejerano and Yona, 1999). This model is learned from a seed of whole sequences of family members. The input sequences are not aligned, nor does the algorithm attempt to align them. The resulting model can then accurately assign family membership for novel sequences (Bejerano, 2003a). A library of such models is shown to perform as well as the widely used, but more computationally demanding hidden Markov models, in parsing novel multi-domain sequences into their known domains, (Bejerano and Yona, 2001).

- The computational complexity of training and prediction using these model are optimized to linear time and space requirements using novel data structures and algorithms (Apostolico and Bejerano, 2000a). This effort is crucial to allow us to scan the ever-growing repositories of potential protein sequences for matches to a novel sequence. It also opens the way, in terms of computational complexity, to more global organizational efforts of the entire known protein space (Apostolico and Bejerano, 2000b).
- The variable memory Markov models are then used to segment a set of unlabeled, unaligned complete protein sequences into their constituent domains (Bejerano et al., 2001). In order to achieve this much harder goal, a novel information theoretic principled clustering algorithm is developed (Seldin et al., 2001). Again, no alignment of the input sequences is attempted during the process. As a result, instances of the same domains can be detected even when appearing in different combinations and ordering in the input set sequences.
- Finally, we devise a discriminative framework for multi-classification of protein sequences using these Markovian models (Slonim et al., 2002). This approach results in much smaller, specialized models, whose selected features also offer possible insights into functionally and structurally important residues in the context of protein sub-families (Slonim et al., 2003).

Also included in the form of an appendix to the thesis are two works which resulted from studying proteins and related molecular sequences,

- A novel branch and bound algorithm (Bejerano, 2003b) which is a first practical optimization of a very common exact statistical test for categorical data (such as amino acid composition). The proof technique we use there was later applied to improve computational complexity of other important tests (Bejerano et al., 2003).
- A study that relates the evolution of protein coding genes with information theoretic measures of transmission fidelity (Bejerano et al., 2000).

Chapter 1

Introduction

Computational molecular biology, generally identified with the potentially broader term Bioinformatics, is a relatively new and burgeoning scientific field. It has its roots in manual inspections of the first very short genomic regions and protein sequences, laboriously sequenced by individuals. Today the field is flooded by genomic and related molecular data churned at an ever increasing pace by world-wide large scale initiatives. As a result it has grown to become a field of intensive research, carried out by biologists, joined *en masse* by computer scientists, mathematicians, physicists and others. The confluence of these different scientific communities has produced fertile grounds for the exchange of exciting ideas. It has also led mathematicians to describe their models in prose, and biologists to invent mathematical notation.

A survey of the complete body of knowledge related to this thesis is beyond the scope of this work. Even so, it is very challenging to try and present it in a manner accessible to both the biologically and the mathematically inclined. I shall therefore resort to provide sufficient background to allow the evaluation of this work, altering at times between prose and rigor.

This first chapter provides a very brief introduction to the realm of proteins, and surveys the biological terms I shall use throughout the thesis. It is mostly based on the standard textbooks of the field, Branden and Tooze (1999); Alberts et al. (1997); Lewin (2000). I then go on to motivate and define the broad goals of this research. A more comprehensive discussion of each subject matter appears in the relevant chapters that follow.

1.1 Proteins

Protein molecules are involved in almost all activities that take place within every living cell. They carry out the transport and storage of small molecules. They play an important role in the transduction of molecular signals within the cell, as well as to and from it. They both help build and take part in the make up of the cell's skeleton. Certain proteins termed enzymes catalyze and regulate a broad range of biochemical processes taking place within the cell. Others termed antibodies defend the cell against invading molecules from the outside world. Many groups of proteins are known, taking part in different activities within the cell. Depending on cell speciation in higher organisms (such as in tissues), each type of cell contains several thousand different kinds of proteins which play a major role in determining the functional role and activity carried by the cell throughout its lifespan.

A protein is a complex macromolecule. And yet, it is built from simpler repeating units chosen

A	ALA	Alanine	R	ARG	Arginine
V	VAL	Valine	S	SER	Serine
L	LEU	Leucine	T	THR	Threonine
I	ILE	Isoleucine	C	CYS	Cysteine
F	PHE	Phenylalanine	N	ASN	Asparagine
P	PRO	Proline	Q	GLU	Glutamine
M	MET	Methionine	H	HIS	Histidine
D	ASP	Aspartic Acid	Y	TYR	Tyrosine
E	GLU	Glutamic Acid	W	TRP	Tryptophan
K	LYS	Lysine	G	GLY	Glycine

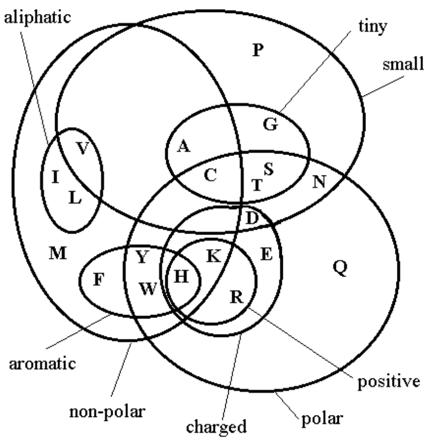


Figure 1.1: **The Twenty Amino Acids.** (left) One and three letter codes of the amino acids. (right) A Venn diagram conveying several properties of the different amino acids. (adapted from Taylor, 1968).

from a limited set of building blocks which are bonded, one after the other, in a linear order. These building blocks are the twenty **amino acids** (Figure 1.1). All amino acids share a common molecular structure, differing only at their side chains. The different side chains confer diverse biochemical properties to the various amino acids. Each amino acid along a protein sequence is also called a **residue**. The blueprints from which all new proteins are synthesized are stored in the **DNA**, the genetically inheritable material, which is also linearly built from simple building blocks, albeit into a much longer double stranded helix structure. These building blocks are called **bases** and only four different ones are used. A region of DNA coding for a protein sequence is called a **gene**. Translation from the four letter alphabet of bases into an amino acid sequence is performed in triplets of bases, known as **codons**. The **genetic code** used by the organism matches a single amino acid against each codon, except for designated **stop codons**, each capable of signalling the end of an encoded protein sequence.

Special machinery and intermediate molecules within the cell are responsible for the synthesis of new proteins from their genes. The linear amino acid sequence which is synthesized during this process is called the protein **primary structure**. The average protein sequence is about 300 amino acid long, with short proteins having only tens of amino acids, and long ones having several thousands. A fully synthesized protein sequence adopts a very specific three dimensional (3D) structure, known as its **tertiary structure**, where every amino acid falls into a specific location and orientation. Different levels of organization can be found within a protein structure. Several **secondary structure** elements can be discerned. These are short (3–40) consecutive amino acids stretches which adopt a well defined local shape, such as an alpha **helix** and a beta **strand**. Other secondary structure elements of less defined shape which usually serve to connect these elements are called **loops**, turns or coils. Figure 1.2 schematically shows the three descriptive levels.

A **structural motif** is a combination of several sequence-consecutive secondary structure elements which adopts a specific geometric arrangement (e.g., helix-turn-helix). A **sequence motif** is a closely related term that describes a sequence segment which occurs in a group of proteins. Some, but not all motifs are associated with a specific biological function. Several secondary structure elements may congregate together in space to form a **super secondary structure**, such as the Greek key motif. In general, helices and strands tend to form the **core** of globular protein structures, whereas loops are more likely to be found on the surface of the molecule.

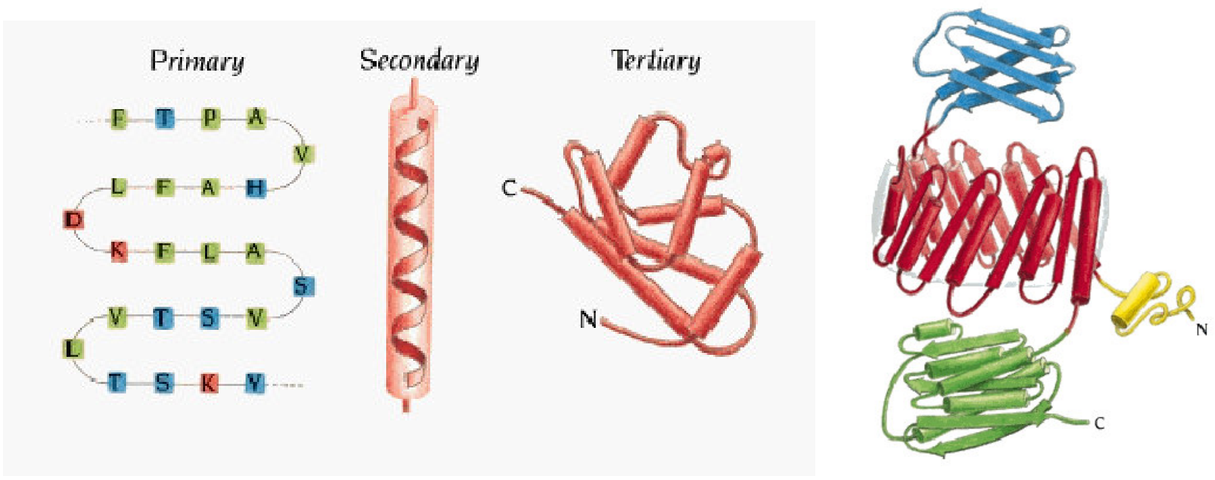


Figure 1.2: **Protein structure.** (left) Schematic depiction of the three descriptive levels of a protein: its sequence of amino acids; a helix shaped secondary structure; and the three dimensional organization of the entire protein. (right) An example of a multi domain protein. (adapted from Branden and Tooze, 1999).

A protein **domain** is the fundamental unit of structure. It combines several, not necessarily sequence-contiguous, secondary structure elements and motifs, which are packed into a compact globular structure. A domain can fold, independently of the complete sequence in which it is embedded, into a stable 3D structure and usually carries a specific function. Some proteins are **single domain** proteins while the sequence of others, called **multi domain** proteins, code for several domains, possibly including two or more repetitions of any given domain (see Figure 1.2). Finally, several proteins may form a complex, which is then called a protein **quaternary structure**.

1.2 Protein Classification

Opinions and evidence vary in recent years as to whether the necessary information for the folding process is always wholly encoded in the protein sequence itself, or may sometimes be media dependent; and as to whether a protein starts to fold into shape as it is being synthesized or collapses to its conformation only as a whole unit. The rigid view of the final conformation itself has also been challenged recently, in certain cases. What is clearly indisputable is the fact that the shape of a protein, together with the biochemical properties of different amino acids placed along its structure are the main factors which allow the protein to function in a specific and well defined manner.

While each protein sequence usually adopts a unique 3D structure, many different proteins adopt the same shape, or architecture, a combination of secondary structure elements or domains which are positioned in a certain manner in space relative to each other. This general structural shape is called a protein **fold**.

A common hierarchical approach to protein classification, assuming for the moment known structures and functions, first groups all proteins into common folds. It is a well established fact that similar protein sequences fold into similar structures. This is true in most cases, however its converse certainly is not. Very different protein sequences have already been found that fold in the same manner. A fold is thus separated into **super families** of proteins which are suspected to be **homologous**, evolutionary related through some distant common ancestor protein. A super family is made of one or more protein **families**. The sequence similarity within a family is high enough to infer evolutionary relatedness to a high degree of certainty. Usually at least one member

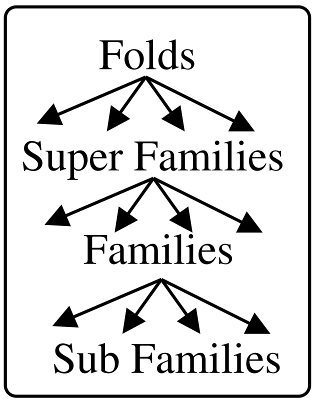


Figure 1.3: **Hierarchical view of the protein world.** The protein world can be organized in a hierarchical manner, starting from the typical fold a protein adopts, and descending through the evolutionary and functional relationships, branching to homologous super-families, families and sub-families.

of a family is similar to a member of another family, justifying the placing of the two in the same super-family. Other linking evidence is also taken into account, such as conspicuous conservation at functionally crucial regions. A family of proteins thus mostly contains sequences with relatively high sequence similarity which fold into similar spatial structures, and therefore presumably also perform the same function. Some families are further broken down into **sub-families**, based on functional evidence which suggests that despite the similarities we have just mentioned, different groups within the family perform somewhat different functions within their respective organisms.

Thus, at the top of this hierarchical view of the protein world we group proteins that adopt similar folds, assuming that in many cases similar function will entail. We then use two intermediate levels to capture evolutionary relatedness, again, assuming that most related proteins diverged from a common ancestor and are thus more likely to perform the same function. At the bottommost level we obtain relatively homogeneous groups of evolutionary related proteins, which need be further split only if different sub-groups have indeed adopted different functional traits, while maintaining very similar sequences and structures. From a computational point of view we have just performed a **hierarchical clustering** of the sequence space in a biologically meaningful manner that respects sequence, structure and function similarities. Each group of sequences put together using this scheme, at different **granularity levels** (i.e., fold, super-family, etc.) is termed a **cluster**. The resulting hierarchy is depicted in Figure 1.3.

1.3 Protein Sequence Analysis

Clearly, understanding the function of all proteins is a major goal of molecular biology. This knowledge is essential to our understanding of nearly all fundamental processes taking place within a living cell. Finding the structure of a given protein can serve as a key to infer its function. This information is valuable not only to advance basic science but also carries practical implications in important fields such as rational drug and protein design.

Experimental determination of the 3D structure of a given protein, however, is a hard and laborious task. Through the use of x-ray crystallography and nuclear magnetic resonance techniques, the structure of but a fraction of all known proteins have been established to date. It is also clear that for most non globular proteins current structure determination techniques will not do. Protein sequences, on the other hand are easily obtainable nowadays, through a mostly automated reading process. **Genomic sequencing initiatives** world-wide, such as the Human genome project, are proving vast amounts of putative protein sequences. These, often internationally co-ordinated efforts, yield ever increasing amounts of sequenced DNA, and derivatives (such as expressed sequence tags). The DNA segments are then searched computationally for protein coding genes in a process dubbed **gene hunting**. These searches have yielded to date close to a *million* novel putative protein sequences. These sequences which await analysis are deposited in huge public-domain databases, accessible to the scientific community through the internet.

However, despite our realization that in most cases the protein sequence itself seems to hold all necessary information to determine its 3D structure and function, we are far from being able to correctly infer this shape computationally. Efforts at reconstructing the folding process, which we are far from understanding, as well as more indirect *de novo* approaches (such as finding the minimal energy structure for a given sequence) are only starting to come of age.

To bridge this gap between the easily obtainable sequence, and the mostly unavailable structure and function information we return to the basic observation made previously. Namely, that proteins which are similar in sequence tend to fall into the same, or a very similar structure and perform equal or very similar function. Exceptions to this generally true rule have also been analyzed, yielding further insight into what constitutes a small or large departure from a known structure.

The basic computational task following this formulation is known in machine learning methodology as a "nearest neighbour" approach. Given a new protein sequence of which nothing more is known, one wishes to search the pool of annotated protein sequences to find the closest match. A simplest approach would then assign to the novel sequence the structure and function of its nearest neighbour. More elaborate analysis will rather super-impose the novel sequence on the known one and characterize the differences between the two into a prediction of conservation of structure and entailing function. This pairwise comparison of protein sequences is one of the first and best studied issues in early bioinformatics research. It has led to the development of well tailored search tools, which now serve as the first step of every biological analysis of any unannotated sequence.

As our knowledge of the protein world grew it became clear that many newly discovered sequences do not have well annotated close homologs. When a closely related sequence does not exist, one is naturally interested in finding more distant homologs of a novel protein sequence. Here, however, pairwise methods come into difficulties. Without a good understanding of the underlying structure encoded in a protein primary structure it is nearly impossible to differentiate, when comparing two sequences, between spurious similarities and scant yet meaningful indications of homology. Spurious hits between two unrelated sequences are the artifact of the limited amino acid alphabet combined with the algorithmic approach behind pairwise searches which tries to best match the two sequences, amino acid to amino acid. On the other hand, few similarities between two sequences, that concentrate on crucial structural and functional elements of the protein (such as a protein active site), can indicate distant homologs. This region of distant homology on the borderline of random similarity has been termed the **twilight zone** of protein homology, and is to date partially unresolved.

1.4 Motivation and Goals

For a novel protein sequence, assume that we have found its most significant match in a database of partially annotated proteins. In frequent twilight zone cases we are required to differentiate between a distantly related pair and an unrelated pair, without a clear understanding of the patterns of conservation in distant relatives. A further step in computational sequence analysis makes the following observation. Collect all sequences which are clear homologs of the template sequence, compare these against the template and deduce empirically which regions have been better conserved during evolution and which are more variable. Armed with this knowledge return to examine the homology with the novel sequence. If the pattern of conservation follows along the evolutionary conservation pattern conclude homology, otherwise reject as not related.

This task serves as the first goal of the thesis. We will utilize the fact that many proteins families have already been established, including anywhere from several up to thousands of well established members. Using these so called **family seed** sequences we will build a variable length Markovian model that captures underlying properties shared by all family members. We will then compare novel sequences, not to single sequences, but rather against a **protein family model**. We will show that our novel modeling technique outperforms pairwise sequence comparison, and is comparable to the state of the art hidden Markov models in use today.

In light of the magnitude and rapid increase in novel protein sequences awaiting annotation, we will put special emphasis in optimizing the run time of our algorithm. As a result we will obtain an approach to learn and predict using the novel models in linear time, surpassing the equivalent quadratic hidden Markov model algorithms.

We will then turn to the much harder task of unsupervised **protein sequence domain** detection from sequence information alone. These are regions of high conservation within groups of proteins, often corresponding each to a single **structural domain**. Due to the multi-domain nature of many proteins, novel sequences may resemble one protein family along one region, and other families along other regions. The segmentation of the novel proteins into their respective domains is commonly done based on prior structural knowledge, where it is available, extrapolating to other sequences sharing similar segments. As a result protein sequences which were not thoroughly investigated, and such is the vast majority in the post-genomic era, may contain various yet undiscovered protein domains. We will thus set out to capture protein domains from unaligned unannotated sequences, using the same Markovian model introduced above. We will first show that these models are capable of modeling two or more domains simultaneously from an unlabeled set of proteins exhibiting several instance of each domain of interest. We will then devise a novel algorithm whose aim is to segment a given set of unlabeled sequences into the underlying domains, based on the information in the given set alone. As this is a much harder task, we will analyze to what extent our new tool answers it.

Finally, we will also use the Markovian modeling technique for another task which cannot be easily achieved using existing tools. We analyze a set of protein sub-families, which all have very similar sequences and yet seem to perform different functions. Using discriminative analysis methods we try to explain the observed differences between the groups by highlighting the most informative sequence motifs differentiating between them. Such computational analysis may direct us to important structural elements (possibly as small as a single, strategically placed amino acid) which are well conserved within a group but differ between them.

Having given a prologue with the basic terminology and our main goals, the rest of the thesis is organized as follows. In Chapter 2 we perform a literature survey of the current bioinformatic tools and mathematical approaches to protein family classification, segmentation and discriminative analysis. In Chapter 3 we introduce Markovian sequence modeling. We motivate it in our context and proceed to show how we adapt the Markovian modeling approach to protein classification. We demonstrate the ability of this method to capture successfully the notion of a protein family from a given seed of examples, and generalize it in the task of classifying novel sequences correctly. In Chapter 4 we optimize the runtime of our Markovian modeling approach. We augment the data structures and replace all underlying algorithms to obtain an optimal linear run time and memory usage complexity. Such an optimization is important as the databases themselves grow at an almost exponential rate. Chapter 5 defines the mathematical approach behind the segmentation challenge, and attempts to solve it using a novel clustering algorithm for Markovian models. Chapter 6 turns to examine discriminative modeling in the context of protein sub-families, and conclusions of the main part follow in Chapter 7. In Appendix A we supplement two additional works. A novel algorithm for performing exact statistical tests for categorical data using a branch and bound approach, and an analysis relating the evolution of protein coding genes with measures of fidelity in an information theoretic context.

Chapter 2

Protein Family Related Methods and Tools

In this chapter we review mathematical models, established bioinformatic tools and research papers which relate to our work. We survey efforts to cluster protein sequences, to segment them into their constituent domains, and to discriminate sequence-wise between functional sub-families.

2.1 Preliminaries

Almost a million verified and putative protein sequences are currently known. This impressive number increases at an ever accelerating pace, primarily as a result of the many genomic sequencing initiatives operating world wide. Available technology has completely automated the process of reading genomic sequences, allowing the rapid sequencing of dozens of complete genomes from various organisms.

Major repositories for verified and putative protein sequences are the **Swiss-Prot** and **TrEMBL**[1] databases (Boeckmann et al., 2003). Swissprot currently holds more than $100,000$ annotated protein sequence entries, each of which manually inspected before insertion into the database. Its companion database Trembl, currently about six times as large, holds novel entries awaiting manual inspection, and subsequent transfer to the annotated database.

Protein structure determination, however, is still a manual and laborious task to a large extent. As a result, the number of proteins whose structure has been experimentally determined, several thousands, is but a fraction of all known sequences. All solved 3D structures are deposited in the **PDB** database (Westbrook et al., 2003). The structure of a protein is the key to understanding its function. Indeed, the function of most solved structures has been inferred, at least to some extent, following close manual inspection, and integration of experimental evidence. Three main databases classify known protein structures hierarchically: **SCOP** (Lo Conte et al., 2002), **CATH** (Pearl et al., 2003) and **FSSP** (Holm and Sander, 1998), ranging from a manual through a semi-automatic to a fully-automated scheme, respectively. Differences between the three clustering schemes exist (e.g., discussed in Hadley and Jones, 1999), but essentially, especially the first two manually guided databases, use the methodology discussed in Section 1.2, clustering under similar structural architecture, homologous super-families, and families.

While these databases are mostly unanimous in classifying the relatively few known structures,

[1]A Bioinformatics database or tool name is usually an abbreviation of a phrase that somewhat describes it. Instead of pursuing these abbreviations, we offer a full description of each such object. Following definition, we will mostly homogenize the use of capital letters, for ease of reading, and often capitalize only the initial letter in each such name.

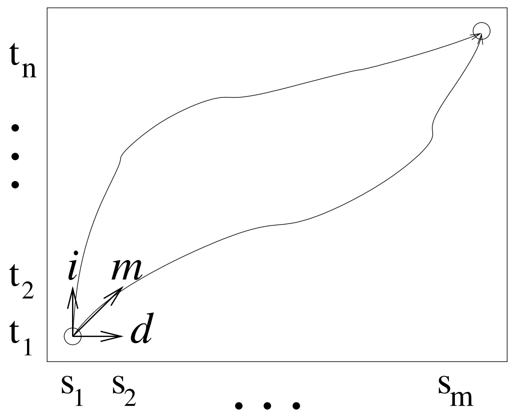


Figure 2.1: **Edit distance computation** between two protein sequences, s and t. We begin by comparing s_1 and t_1. We can either match the two and proceed to compare s_2 to t_2, delete s_1 and proceed to compare s_2 to t_1 or insert t_1 and proceed to compare s_1 to t_2. These three operations are denoted above by their initials m, d, i, respectively. From the point we have arrived at we again choose either of these three options, and continue to do so until we reach the upper right corner of the rectangle. The path we have chosen defines a series of operations that, when applied to sequence s, turns it into sequence t. We add costs to each match operation, depending on the two amino acids matched (low cost for identical or similar ones, high cost for different ones), and to the **indel** (insert or delete) operations to obtain a path cost. The Smith-Waterman algorithm then searches efficiently for the minimal cost path which is termed the **edit distance** between the two sequences. The $O(mn)$ search is performed using a **dynamic programming** (DP) technique, which relies on the fact that sub-paths of the optimal path are optimal themselves. The problem is recursively broken into smaller sub-problems whose solutions are combined to obtain the global minimum.

organizing the far larger world of known protein sequences is an intense research area, made difficult by our lack of deep understanding of the mapping between sequence and structure. As reviewed in Chapter 1, the first and most thoroughly studied tools in computational sequence analysis are the pairwise comparison algorithms. The seminal **Smith-Waterman** homology detection algorithm (Smith and Waterman, 1981) searches for the minimal cost of amino acid insertions, deletions and substitutions which, when applied to one sequence, turn it into the other (see Figure 2.1). The free parameters of this model, including amino acid substitution cost, gap opening and gap extension costs, have been optimized over the years. However, the complexity of this algorithm is quadratic, deeming it rather slow when scanning a single sequence against a large database of other sequences, or when trying to compute all pairwise similarities in a large set of protein sequences. As a result, two heuristic linear approximation methods for pairwise comparison have also become very popular. These are **BLAST** (Altschul et al., 1997) and **FASTA** (Pearson, 2000), which have also been refined over the last decade. These three methods are now used as a first step in the examination of every novel protein sequence. They also serve as the starting point to several tools aimed at capturing higher order structure in the protein world, which we review next.

2.2 Protein Sequence Classification

As we have seen sequences with known structures and functions can be classified hierarchically into folds, super-families, families, and sub-families (Section 1.2). Computational classification of protein sequences aims at extending the resulting hierarchy to include the majority of protein sequences for which neither structure not function have been verified experimentally. We survey three general approaches to this task.

		Model Assignment	
		P	N
Its Validity	T	**TP**	**TN**
	F	**FP**	**FN**

Figure 2.2: **The four categories of a binary classification test.** When a model is presented with a binary classification task, four types of calls are possible: the model may label an item as either positive (abbreviated P in the table above) or negative (N). All such calls are either true (T) or false (F), when compared to the real labels. Each of the four two-letter combinations defines a variable that holds the counts of items classified into that category. Thus, for example, the real positive items are the true positives (TP) together with the false negatives (FN), or simply TP+FN. Several measures of classification accuracy defined based on these variables are introduced throughout the text.

2.2.1 Supervised Approaches

Goal. Given a subset of sequences, or pre-cut subsequences, tagged as either belonging or not to a certain protein cluster, at either level of granularity (Section 1.2), devise a mathematical model which is capable when presented with novel protein sequences, to correctly conclude whether they belong to the cluster.

This is a supervised task because the given sequences have been tagged, and possibly excised, by an external source. Subsequent modeling will rely on this tagging for concept building. It is also a binary classification task, and as such each decision made can fall into one of four categories: Correctly labeled true positives (TP) and true negatives (TN), and incorrectly labeled false positives (FP) and false negatives (FN), as illustrated in Figure 2.2. The resulting model should perform well by two criteria: **Sensitivity**, which measures the ability to detect sequences that belong to the cluster, and is formally defined as $\frac{TP}{TP+FN}$. And **specificity**, which measures the ability to reject sequences that do not belong to it, defined as $\frac{TN}{TN+FP}$. Falling in with conventions we will use cluster and family interchangeably even when the intended level of granularity differs (i.e., when modeling a super-family).

Regular Expressions

Regular expressions have their roots in the Unix operating system, most commonly used there to specify a group of files or other objects without explicit enumeration (Friedl, 2002).

Mathematically, a regular expression (or regexp, or pattern) is a text string that describes some set of strings. Regexp R matches a string S if S is in the set of strings described by R. In the context of protein sequences we can thus define a regexp to describe a protein family if the sequences of family members match this regexp as plain strings. The alphabet of protein regular expressions is thus the accepted single character coding of the twenty amino acids (Figure 1.1). The common regexp operators are:

- The match-self operator. Each one of the twenty characters matching only itself. E.g., `V` matches only valine in the protein sequence.
- The match-any-amino-acid operator. This operator matches any single amino acid. Usually denoted by `X`.
- The concatenation operator which is implicit. E.g., `VXV` matches only a valine, followed by any amino acid, followed by another valine.
- Repetition operators of the forms match zero or more, once or more, and interval operators. E.g., `V-X(4,5)-V` will match two valines which are four or five residues apart.
- List operators. Either a matching list that matches a single character represented by one of the list items, or a non-matching list which matches a single character not represented by one of the list items. E.g., `[VAF]` matches either a valine, an alanine or a phenylalanine residue.
- The alternation operator. E.g., `VA|AV` will match a consecutive pair of valine and alanine residues, in either order.

Ideally we would want to come up with regular expressions that match all known and probable protein family members, and reject all others.

The **PROSITE** database (Sigrist et al., 2002) is primarily a repository of protein related regular expressions. Rather than trying to describe complete sequences, a Prosite regexp is said to (locally) match a protein sequence if it has at least one contiguous region that exactly matches the given regexp. Protein patterns are obtained from literature searches, as well as crafted from well-characterized families, from sequence searches against Swissprot and Trembl, and from sequence clustering (see below).

The first step in pattern construction is the generation of a reliable **multiple sequence alignment** (MSA). Since related sequences are the product of gene duplication and subsequent mutation events, it is possible, at least in principle, to align related protein sequences such that each column represents a single residue from the ancestral sequence (which may have been substituted or deleted in some) or one that was subsequently inserted in one or several family members (see Figure 2.3). The curator searches the alignment for a short conserved subsequence, typically 4–5 residues long, which is part of a region known to be important or which includes one or more biologically significant residues. This core pattern is then examined against Swissprot and Trembl and extended until a best match to the given protein family is achieved. Published patterns collected into Prosite may also be further optimized this way. For example, the chosen Prosite signature (numbered PS00193, Sigrist et al., 2002) for the Cytochrome b C-terminal domain family of Figure 2.3 is `P-[DE]-W-[FY]-[LFY](2)`, which focuses on a family invariant P-E-W triplet.

While intuitively appealing, regular expressions are rather limited in this context. First, being typically confined to small regions, regexps are relatively vulnerable to spurious hits along the many protein sequences available (resulting in low specificity). Another weakness lies in their all or nothing approach. Namely, either a protein sequence matches a pattern, or it does not. For example, if the sample set from which the motif was carved contained only leucine at a certain position, and a subsequent novel site is found, matching the entire pattern, apart from an isoleucine replacing the leucine, it will be rejected. On the other hand, if one tries to accommodate too much variability per column, by the time that enough positions are expanded, an unacceptably high number of false matches would have joined the identified set. As a result many protein families as well as functional and structural domains cannot be characterized satisfactorily using patterns.

```
PETD_SYNP2/65       PGNPFATPLEILPEWYLYPVFQILRVLPNKLLGIACQGAIPLGLMMVP...
PETD_NOSSP/65       PANPFATPLEILPEWYLYPVFQILRSLPNKLLGVLAMASVPLGLILVP...
PETD_CHLEU/65       PANPFATPLEILPEWYFYPVFQILRTVPNKLLGVLAMAAVPVGLLTVP...
CYB_MARPO/262       PANPMSTPAHIVPEWYFLPVYAILRSIPNKLGGVAAIGLVFVSLLALP...
CYB_HETFR/259       PANPLVTPPHIKPEWYFLFAYAILRSIPNKLGGVLALLFSILMLLLVP...
CYBB_STELO/258      PANPLSTPAHIKPEWYFLFAYAILRSIPNKLGGVLALLLSILVLIFIP...
CYBA_STELO/258      PANPLSTPPHIKPEWYFLFAYAILRSIPNKLGGVLALLLSILILIFIP...
CYB_APILI/260       IANPMNTPTHIKPEWYFLFAYSILRAIPNKLGGVIGLVMSILIL--YI...
CYB_ASCSU/249       ESDPMMSPVHIVPEWYFLFAYAILRAIPNKVLGVVSLFASILVL--VV...
CYB_TRYBB/253       IVDTLKTSDKILPEWFFLYLFGFLKAIPDKFMGLFLMVILLFSL--FL...
```

Figure 2.3: **A Multiple Alignment Segment** of the Cytochrome b C-terminal domain family seed (Pfam family PF00032, Bateman et al., 2002). These transmembrane proteins, involved in respiratory functions, appear in many living organisms across multiple lineages (Zhang et al., 1998). We show the start of the C-terminal domain, which forms a relatively conserved region around an invariant P-E-W triplet that lies in the loop that separates the fifth and sixth transmembrane segments. On the left are the Swissprot identifiers, followed by the positional location of the first amino acid in the block. Gap positions towards the end of the aligned segment are denoted by '-'.

A partial solution to the latter problem could be offered in the form of approximate matches, i.e., finding all subsequences that would exactly match a conservative regexp, if allowed to undergo one or more insertion, deletion or substitution event. While the algorithmic aspects of this direction are well studied in computer science (e.g., Gusfield, 1997), their use would increase instances of spurious hits.

Profiles

Basic sequence profiles, sometimes called weight or position specific scoring matrices (PSSM), are also derived from consecutive regions of multiple alignments of related sequences (Gribskov et al., 1987), but quite differently. A consecutive region of relatively high conservation of length l, possibly the whole alignment, is typically chosen.

Denote by Σ the 20 amino acid alphabet (Figure 1.1). A profile is a $20 \times l$ matrix $\{s_{\sigma i}\}_{i=1\ldots l}^{\sigma\in\Sigma}$ of scores for each each possible amino acid at each position of the profile. Scoring schemes vary, but they all aim to capture the likelihood of observing a given amino acid in that position of the multiple alignment.

Any novel protein segment of length l can now be scored against the profile by summing the scores each of the protein's residues obtain at each position. One then typically calibrates a scoring **threshold** t, such that a novel protein sequence $x_1 \ldots x_n$ will be accepted into the family if

$$\max\{ \sum_{i=i_0}^{i_0+l-1} s_{x_i\, i-i_0+1} \ \mid\ i_0 = 1, \ldots, n-l+1 \ \} \ \geq\ t$$

Contrary to the match or reject regular expression approach we have now defined a conceptually continuous scale of grades and can choose where to dissect it based on our tagged examples. The thresholds for different protein family profiles can be compared by observing the average score contribution per symbol, t/l. Differences attest to the different levels of inter-family conservation.

These gapless profiles can be extended, to allow insertion and deletion costs at each column. Algorithmically, using dynamic programming (similarly to Figure 2.1) in $O(nl)$ time one can examine all possible contiguous segment matches to the profile, now of only approximately length l, for the maximal scoring one.

Due to the limitations of regular expressions discussed above, **PROSITE** has been expanded to also include sequence profiles. Prosite profiles require a multiple sequence alignment as input and use a symbol comparison table to convert residue frequency distributions into weights. Profile sensitivity is then improved using iterative refinement procedures.

Unlike patterns, profiles are usually not confined to small regions with high sequence similarity. Rather, they attempt to characterize a protein family or domain over the entire length of the multiple alignment block. This scheme can lead to false hits, when a profile covering conserved as well as divergent sequence regions, obtains a significant similarity score to a sequence that is partially incorrectly aligned. Yet, in general profiles are considered to be more sensitive and more robust than patterns, partly because they assign finite weights to residues which have not been observed previously at every position, using observed amino acid compositions and observed amino acid substitutions.

Sequence Fingerprints

The term sequence fingerprint is not unique to any particular model. Rather, it denotes the conceptual fact that we no longer attempt to build a single detector model for a certain domain or family, but rather rely on a *set* of these, used in concert, for positive identification.

Fingerprinting relies on the fact that in most protein families certain parts of a sequence tend to be more conserved than others across the family. These are typically, but not always, related to key functional regions or to core structural elements of the fold. This contrasts the approach of Prosite, reviewed above, where each pattern or profile is optimized to characterize, by itself, a single important site, motif, or a domain.

A fingerprint approach offers more versatility. If a protein family multiple alignment shows, for example, two well conserved regions, we are no longer obliged to choose on which to focus, nor must we model the unstructured in-between region, which may cause false hits to rank higher. Rather, we can try to characterize both regions using mathematical models of our choice. Next we can decide what constitutes a family member. For example we can demand that an accepted novel sequence be recognized by both models, and in the same linear order as in the alignment from which they were crafted. We can also limit the allowed spacer between the two segments, etc. The two databases we next survey both use ungapped sequence profiles as the basic fingerprint unit.

PRINTS (Attwood et al., 2003) is a compendium of simple protein profile fingerprints. The starting point for fingerprint definition is, again, a reliable multiple sequence alignment, done by human experts. Typically only a few family members are included in the initial alignment, to ease the manual inspection. Once a motif, or set of motifs, has been identified, the conserved regions are manually excised in the form of short independent local alignments. Contrary to Prosite's focus on meaningful conserved motifs, here there are no rules regarding the juxtaposition of such motifs, other than that they should not substantially overlap. Each fingerprint is treated as a simple frequency matrix. Independent database scans are made with each aligned motif, summing the scores of identical residues for each position of the retrieved match, using no mutation or otherwise weighting scheme. To be considered for family membership a sequence must match all fingerprints. If novel family members are discovered they are used to update the profiles and perform a new search. This process is repeated manually until no further improvements are found. The final aligned motifs from this iterative procedure constitute the refined fingerprint that is entered into the Prints database. To address the relatively slow pace at which new families are added through this manually supervised process a mostly automatic classification scheme based on the same principles, has been recently initiated in an accompanying database called prePRINTS. This database obtains its putative family seeds from Prodom, an automated clustering of the entire sequence space, which we review later on.

Alignment blocks in the **Blocks+** database (Henikoff et al., 2000) are also multiply aligned ungapped segments corresponding to the most highly conserved regions of proteins. However the algorithmic details between the two methods differ. The generation of block fingerprints is automatic. An algorithm performs the multiple alignment of related sequences, it then searches for contiguous intervals up to sixty positions long where the aligned amino acids are highly similar in at least half of the aligned sequences. The best subset of blocks from all available ones is chosen as the representative fingerprint of the family (Henikoff and Henikoff, 1991). Contrary to Prints, Blocks profiles are weighted to avoid extensive bias due to subsets of overly similar sequences within an alignment, and sequence scoring is not governed only by simple frequency calculations, but also involves prior knowledge of the substitution rates between between different amino acids. Like Prints and other databases we review, it contains blocks derived both from biologically meaningful Prosite families, from Pfam sequence domain families and from Prodom and Domo families generated by automated clustering schemes (see below).

Fingerprints extend our ability to characterize protein families in cases where two or more relatively well conserved regions together characterize the family well. However, due to the myriad of possible combinations, the profiles used to generate fingerprints in practice are simple gapless ones. Consequently characterization of more divergent or heterogeneous families is often out of reach for such tools.

Profile Hidden Markov Models

Hidden Markov models (HMM) are rich and well studied mathematical models that have been widely applied in the field of speech recognition (Rabiner, 1986). From a bioinformatic point of view profile HMMs (Krogh et al., 1994) are a non-trivial extension of the profile model defined earlier. We begin by surveying the general HMM theory, and then focus on the specific architecture and topology of profile HMMs.

The basic HMM building blocks are termed states. An HMM has a finite set of states, $S = \{s_1, \ldots, s_N\}$. In each discrete time step t an HMM process is found at a particular state, $q_t \in S$. We denote by π the probability vector of finding the HMM at some state in time $t = 1$, such that $\pi_i = P(q_1 = s_i)$. The process then switches from one state to another (possibly back to itself) at discrete intervals, governed by a stochastic $N \times N$ transition matrix A, where $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$. Thus, a series of HMM state transitions $Q = q_1, \ldots, q_T$ is a **Markov process** of order one, since given the complete history of the process at time t, the next transition is influenced only by the current transition[2]

$$P(q_{t+1}|q_1 \ldots q_t) = P(q_{t+1}|q_t)$$

Each time an HMM arrives at state q_t it emits a symbol o_t from a finite alphabet $\Sigma = \{\sigma_1, \ldots \sigma_{|\Sigma|}\}$. Each chosen symbol is governed by a state specific stochastic emission matrix B of size $N \times |\Sigma|$, where $b_{ik} = P(o_t = \sigma_k | q_t = s_i)$. A transition series Q thus generates a second series $O = o_1 \ldots o_T$. The emission process is termed **stationary** because

$$\forall \sigma, s, t, t' : P(o_t = \sigma | q_t = s) = P(o_{t'} = \sigma | q_{t'} = s)$$

[2]Throughout we will often use expressions of the form $P(x)$ as shorthand for $P(X = x)$.

The HMM is thus completely defined by $\Lambda = (\pi, A, B)$. When we come to fit an HMM to some data we assume that the series of state transitions Q is **hidden** and cannot be directly measured. What we can directly measure is only the **observed** series O.

There are three major inferences we would like to perform using HMMs. Since the techniques used to solves these problems are not directly relevant to this thesis we suffice in outlining them.

The evaluation problem. Compute $P(O|\Lambda)$, which is the probability that a given HMM generated a given series of observations. Formally this is defined as a weighted sum over all N^T possible transition series

$$P(O|\Lambda) = \sum_{Q \in S^T} P(Q|\Lambda)P(O|Q,\Lambda)$$

A standard dynamic programming approach yields the **forward algorithm** that computes this sum in $O(N^2T)$ time.

The decoding problem. Given an observed series O and a model Λ, find a corresponding state sequence Q which best explains the observations. This problem has several different formulations. We focus on the one relevant for us subsequently, and search for the most likely path,

$$Q^* = \arg\max_Q P(Q|O,\Lambda) = \arg\max_Q P(Q|\Lambda)P(O|Q,\Lambda)$$

Using Bayes rule to obtain the right hand term we can take a very similar approach to that of the evaluation problem sum, replacing additions with maximum operators and tracking paths, to derive the **Viterbi algorithm** which has the same time complexity.

The learning problem. Given an observed series O, find the HMM that best explains it,

$$\Lambda^* = \arg\max_\Lambda P(O|\Lambda)$$

This problem is much harder than the previous two, as it requires finding the global maximum of the **likelihood function** $P(\text{data}|\text{model})$ over a continuous space of model parameters riddled with many local maxima. Indeed, in general, as in most practical applications no analytical solution can be found to this problem. The **Baum-Welch algorithm** offers an alternative where one guesses an initial model Λ^0. An iterative procedure is then performed where from each Λ^τ we derive another model $\Lambda^{\tau+1}$ for which it is guaranteed that $P(O|\Lambda^{\tau+1}) \geq P(O|\Lambda^\tau)$. This procedure, which is an instance of the **Expectation Maximization (EM)** approach thus converges to some local maximum of the likelihood function.

We turn to define **protein profile HMMs** which are a specific subset of HMMs. We begin by assuming **data preprocessing**. Namely, that we are given a multiple alignment block of length L' consisting of l aligned sequences (recall Figure 2.3). We will illustrate the building process using Figure 2.4 which shows a simple example where $L' = 3$ and $l = 5$ We will build and calibrate a specific HMM that imitates the process of matching a novel protein sequence to this MSA. We begin by performing **model selection** to determine the architecture and topology of our HMM. First, we decide which subset of the alignment columns represent sequence positions common to the whole family, and which represent positions inserted in a small, non-representative subset of the family (i.e., a column where most entries are gap symbols). We denote the size of the first subset $L \leq L'$. Let $S = \{b, e, i_0, \ldots, i_L, m_1, \ldots, m_L, d_1, \ldots, d_L\}$. We term these states in correspondence to their initial letter: begin, end, insert, match and delete states. We set $\pi_b = P(q_1 = b) = 1$, forcing all paths to start from the begin state. The transition matrix A allows only the very restricted set

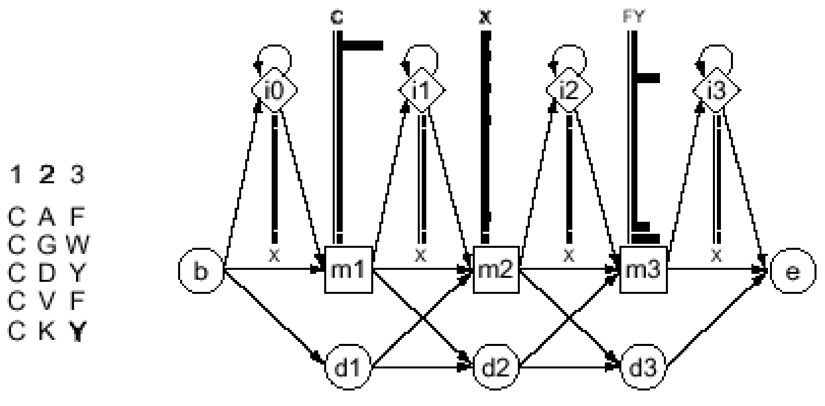


Figure 2.4: **A short profile HMM** (right) representing the multiple alignment of five sequences along three consensus columns (left). Each column is modeled by a match state (squares labeled m1, m2, m3, respectively). Above each we plot the emission probabilities of the twenty amino acids, using black bars. On top of each vector we denote the column consensus amino acid(s). Insert states (diamonds labeled i0–i3) are also associated with emission vectors. Delete states (circles labeled d1–d3) have no emission probabilities as they stand for columns where the respective amino acid was deleted in the protein sequence compared to the model. Begin and end states (b,e) are also included, and allowed transitions are shown as arrows. Note that no amino acid is precluded (by assigning zero probability) at any column. (adapted from Eddy, 1998b).

of transitions depicted by arrows in Figure 2.4. This topology defines a **left-right model** since allowed transitions either stay in place or move to the right until the rightmost state e is reached. In particular, state subscripts along a valid path can only increase by increments of one. The emission alphabet Σ is set to all allowed amino acids (Figure 1.1), and only match and insert states emit them. The begin, end and delete states are all silent states.[3]

How does this architecture relate to comparing a novel sequence $O = o_1, \ldots, o_T$ to the MSA? Just as in the pairwise sequence comparison of Figure 2.1, we start from the N- terminal of both O and the MSA. We then either delete or insert a symbol from O, or match it to the current column of the MSA. The HMM states d_j, i_j, m_j respectively perform these operations. Thus, m_j emits, or weighs amino acid σ according to its abundance in the respective MSA column. State i_j inserts one amino acid (or more, using self transitions) from O before matching column j. Finally, the silent state d_j allows a gap in sequence O skipping column j of the MSA altogether. To globally align sequence O to HMM Λ we will demand that every allowed path $Q = q_1, \ldots, q_{T'}$ ($T' \geq T + 2$ due to states b, e and insertions) ends in $q_{T'} = e$.

Under this interpretation the learning problem, or **model training** goal at hand can be formally defined as finding

$$\Lambda^* = \arg\max \left\{ P(O_1, \ldots, O_l | Q_1, \ldots, Q_l, \Lambda) \mid A, \{b_{ik}\}_{i=m_1,\ldots,m_L}^{k=1,\ldots,|\Sigma|} \right\}$$

where $O_1, \ldots O_l$ are the l MSA sequences, and $Q_1, \ldots, Q_l$ are their paths through the HMM Λ defined by the MSA itself. Parameter estimation is limited to all allowed transition probabilities and emission from match states. Insert state emissions are typically set to some background distribution of amino acids. This problem is much more restricted than the original one, and can be solved analytically. If we denote by α_{ij} the transition counts we observe along paths $Q_1, \ldots, Q_l$

[3]Formally we could have augmented Σ with the empty symbol ϵ and restrict the emission matrix such that for each silent state $P(\epsilon|s) = 1$, and elsewhere $P(\epsilon|s) = 0$.

between the different states, and by β_{ik} the number of times each σ_k appeared at state m_i along those paths, then one can show that the maximum likelihood (ML) solution yields

$$a^*_{ij} = \frac{\alpha_{ij}}{\sum_{j'} \alpha_{ij'}}, \quad b^*_{ik} = \frac{\beta_{ik}}{\sum_{k'} \beta_{ik'}}$$

While being optimal for the training set at hand, these probabilities are then corrected for small sample size effects. Note, for example, that they exclude the use of any state transition and any amino acid emission which were not observed in the training set (i.e, $\alpha_{ij} = 0 \Rightarrow a^*_{ij} = 0$).

Querying a novel sequence O against such an HMM Λ, or **model prediction** can be done by computing either $P(O|\Lambda)$ summing over all paths, or $P(O|Q^*, \Lambda)$ which the Viterbi algorithm can be altered to retrieve, using only the most probable path Q^*. Classification can be obtained by contrasting either expression with a threshold t, which, as in the profile case will be sequence length dependent. A more statistically motivated procedure performs a **likelihood ratio test** between the above probability and $P(O|R)$, where R models a random protein sequence, typically using a single column independent background distribution. The HMM is thus shown here to be a **generative model** of the data, as acceptance or rejection are seen to relate to the probability that the given HMM generated, or emitted, the novel sequence.

Another issue important in protein modeling is the need to perform **sequence weighting**. Since column emission probabilities are treated as independent events, if a large number of training sequences come for a specific subset of family sequences, all emission probabilities will be biased towards these, impairing the ability of the model to detect other family members. Finally, we note that profile HMMs architecture can be augmented to allow explicit local (partial) sequence matches to the model, as depicted in Figure 2.5. They can also be built from an unaligned set sequences. This is done using iterative applications of the Baum-Welch and Viterbi algorithms to align the training sequences against each other, which eventually results in an HMM-built MSA. These and other extensions of estimating profile HMMs from protein sequences are discussed at length in Durbin et al. (1998).

Pfam (Bateman et al., 2002) is a database of multiple alignments and profile HMMs of protein sequence domains. We recall that by sequence domain we denote a long, relatively well-conserved protein sequence region, which in many cases represents a structural domain or an otherwise evolutionary conserved structure with bearings on the protein's function.

Pfam is composed of two sets of families. Pfam-A families are based on curated multiple alignments. For each family in Pfam-A a seed multiple alignment is manually prepared from selected family members. A profile HMM is derived from the seed alignment and used to find additional family members and align them to the family model. This process can be iterated until it achieves satisfactory results. In certain cases the resulting alignment is discarded by the curator and a new attempt is made using other seed members. The resulting HMMs together with the thresholds used to train them are also stored in the database, and can be used to classify novel sequences. Pfam-B automates this process for the rest of the proteins clustered automatically into families by Prodom. Pfam-B families are candidates for protein family characterization and annotation.

The **SMART** (Letunic et al., 2002) and **TIGRFAMs** (Haft et al., 2003) databases of profile HMMs complement Pfam in specific narrower areas of interest. In principle, Pfam sequence domains attempt to cover all conserved, long-enough contiguous sequence regions which appear in at least several different protein sequences. Smart puts a special emphasis on signalling, extracellular and chromatin-associated protein domains. These domains are extensively annotated there with respect to phyletic distributions, functional class, tertiary structures and functionally important

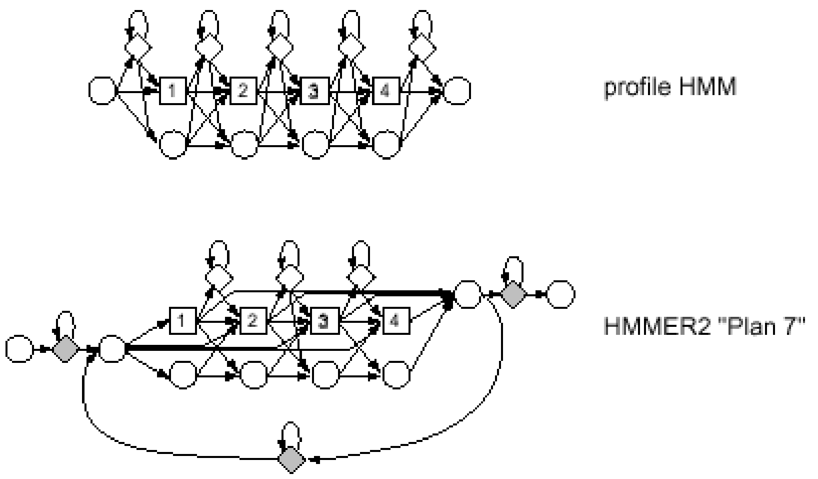


Figure 2.5: **HMM topology comparison** (legend as in Figure 2.4). (top) The profile HMM described in the text. (bottom) An augmented topology allowing for explicit local matches and repeating elements to be represented as a single path from begin to end state. Local matches are allowed by adding to the profile HMM explicit initial and terminal insert states, and allowing to jump straight from the initial insert state to any match state in the sequence, and from any match state out to the terminal insert state. Whole or fragmented repeats are accommodated by a single backwards edge which includes an intermediate insert state. (adapted from Eddy, 1998b).

residues. This focus on mostly regulatory domains stems from a realization that those domains were proving most difficult to detect and annotate using database searching methods. Regulatory domains are generally shorter and less well conserved, whereas enzymes, for example, are mostly longer and have better amino acid conservation, particularly in active site regions, allowing for better characterization using Pfam and the simpler methods discussed earlier.

The special focus in Tigrfams is on characterizing groups of proteins which are conserved with respect to function. In such groups (called equivalogs) no member has diverged functionally since their last common ancestor. Tigrfams curated multiple alignments and HMMs are thus mostly geared towards functional, rather than structural annotation. Through sequence homology, it provides the information best suited for automatic assignment of specific functions to proteins from large scale genome sequencing projects. A Pfam sequence domain family may be broken down on occasion by Tigrfams into sub-families of divergent functions. On the other hand when several structural Pfam domains characterize a single function when they appear together, they will typically be modeled by a single Tigrfams HMM of the entire region containing these domains.

Profile HMMs have been generally accepted as the preferred method for generating a discriminating mathematical model from a multiple sequence alignment. As such, it is no surprise that other Profile HMM databases also exist, which are not primarily derived from sequence data alone. One such example is the **HOMSTRAD** database (de Bakker et al., 2001) of alignments and HMMs which is primarily focused on **structural alignment**. In structural, contrary to sequence alignment, residues are not aligned to each other based on physico-chemical similarities but rather based on their actual location in the 3D structure of the proteins. A structural alignment is derived in principle only from sequences whose structures are known. These are aligned in the best possible way against each other in three dimensions, using measures of backbone proximity and other essential geometrical measures, which are very different from sequence alignment similarity functions. As noted earlier, only a fraction of all known protein sequences have solved structures, and this imbalance currently only increases. Thus, the Homstrad database takes structural alignments and augments them with available sequence data to create enriched alignments whose primary focus is spatial positioning. Overall similarity between members of an alignment can be very low, as long as structurally and functionally important residues (such as those of the active site) can be reliably aligned and thus highlighted in all members of the group.

Evolution tends to conserve structure much more than sequence. Therefore, the best quality multiple sequence alignments are generally considered to be those derived from structural superposition. Interestingly, in a recent study (Griffiths-Jones and Bateman, 2002) several structure and sequence alignment methods were compared. While structural data did improve the quality of obtained multiple sequence alignments, these did not add significantly to the ability of the derived profile HMMs to find more remote sequence homologs.

2.2.2 Unsupervised Clustering

All previous methods, with no exception require labeled sequences, typically a manually curated set of aligned sequences per targeted family. However, as mentioned, the in-flux of novel sequences together with their observed power-law abiding diversity leave many sequences outside the coverage of all curated databases. Fully automated clustering methods give partial answers with respect to global organization of all protein sequences.

Goal. Given a set of unlabeled sequences, group them into biologically meaningful clusters at the different granularity levels discussed in Section 1.2. Alternatively, allow some of the sequences to be pre-labeled, but emphasize the determination of novel clusters of biologically related sequences, which are not related to any of the given labeled ones.

From a mathematical point of view clustering is an ill-posed problem. The hierarchical classification of the protein world into folds, super-families, families and sub-families serves as an excellent example of this ambiguity. While large parts of the classification tree can be unanimously agreed upon by protein experts, the actual details of each assignment vary tremendously. They involve a myriad of sequence, structure and function related observations, which we are far from being able to quantify mathematically to obtain a unique objective function we wish to optimize.

As a result, computational clustering efforts choose very different approaches to the same task. They use different representations of the proteins to be classified, define different optimization goals, and try to achieve, or often approximate these using different algorithmic techniques. Not surprisingly the resulting partitions of the known protein space differ, and more dramatically so than in the curated databases of the previous section. Moreover, their granularity levels are not directly correlated with the biological four layer hierarchy. As a result many clusters are too heterogeneous with respect to the known sequences to allow clear labeling. The resulting clusters are often not stable, such that subsequent runs with new data can result in rather different partitions, between two updates of the same database.

Still, as far as they are from inferring the ultimate classification tree which would result from the formidable task of completely understanding each individual sequence, they offer a unique and fruitful glimpse into it. One such beneficiary is the on-going **structural genomics** effort to focus experimental structure determination on sequences which represent super-families with no similar known structure.

The clustering schemes we next review use the protein sequence as its basic representation. Typically, they then proceeds to use one or more of the common pairwise similarity measures to

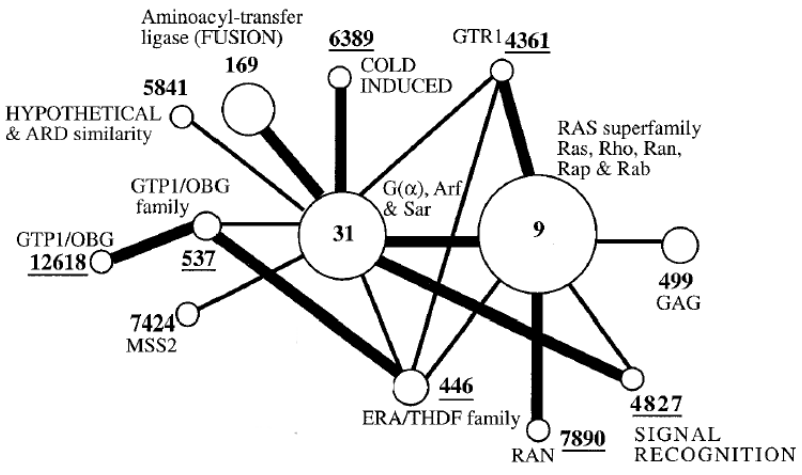


Figure 2.6: **Protomap unsupervised clustering.** A small subset of Protomap clusters is shown, each depicted by a circle whose diameter is proportional to the cluster size. Each cluster is labeled according to the annotation of the majority of the proteins within it. Edges connecting different clusters indicate similarity, and edge width is proportional to degree of similarity. In this subset the Ras superfamily is shown to be related to other small GTP-binding proteins. (adapted from Yona et al., 2000).

induce a distance measure between all pairs of sequences. Two issues worth keeping in mind when evaluating such an algorithm are the correct handling of multi-domain proteins which are affine to several single domain clusters, and the danger of false associations arising from these instances. Consider, for example, two single domain proteins, a having domain A and b coding for domain B, and a multi-domain protein c having both domains A,B. Protein c is thus similar to both a and b, yet a and b themselves have nothing in common and we would not want them clustered together. The different databases approach these issues using different methodologies.

In **ProtoMap** (Yona et al., 2000) all three common measures of pairwise similarity (Smith-Waterman, Fasta, and Blast) are combined with two different scoring matrices of amino acid substitution costs (known as Blosum 50 and Blosum 62; Henikoff and Henikoff, 1992) to create an exhaustive list of neighboring sequences per each sequence in the Swissprot and Trembl databases. From these one can devise a conceptual complete weighted graph, where each sequence is represented by a node, and each edge length is the pairwise distance between the two sequences it connects. However, for statistical soundness, the weight of an edge connecting two sequences is chosen not from the raw scores but based on expectation values of the similarities between the two sequences. Clusters of related proteins correspond to strongly connected components of this graph. Subsequent analysis is aimed at automatically detecting these sets.

The bottom-up analysis starts from a very conservative classification, based on highly significant similarities, which generates many small sets. Subsequently, classes are merged to account for less significant similarities. Merging is performed by a two phase algorithm. First, the algorithm identifies groups of possibly related clusters, based on transitivity and strong connectivity, using local considerations. Then, a global test is applied to identify nuclei of strong relationships within these groups of clusters, and clusters are merged accordingly. This process is iterated at varying thresholds of statistical significance (or confidence levels), where at each step the algorithm is applied to the classes of the previous classification, to obtain the next one, at a more permissive threshold. Consequently, a hierarchical organization of all proteins is obtained (see Figure 2.6).

Protomap, which has spawned two daughter databases, underwent several conceptual changes recently. Additional tests were incorporated in the process of deciding whether two clusters should be merged, or not, to further avoid biologically misguided actions. Conceptually, Protomap has moved from a **hard clustering** scheme, where each protein sequence belongs to a single set at each phase, to a **soft clustering** paradigm where sequences are assigned only probabilities of being a member of any given set. A set, in this terminology, consists of an appropriate weighting of all sequences which are sufficiently affine to it. This conceptual change tries mainly to address the proper handling of multi-domain proteins. In a hard clustering whole sequence approach associating a multi-domain protein to either of the single domains within it hides its relationship with all the others.

ProtoNet (Sasson et al., 2003), the first daughter database of Protomap, has kept the conceptual bottom-up agglomerative hard clustering approach. Protonet emphasizes the structure that underlies the repeated merger steps, which is generated in the following manner: At the beginning of the procedure represent each protein sequence by a node. After the first merger step, take each resulting set of proteins, and connect all corresponding nodes to a novel one. This new node represents the set of proteins. The next merger step works at the level of these new nodes, and also includes all protein nodes not grouped in the first stage. For each set of these, now merged into a new super-set, add a node, and connect the representative nodes to it. When the iteration process terminates we are left with a mathematical object known as a forest - a group of trees where the original protein sequences are the leaves, and higher level nodes represent merged sets of all the sequences at their leaves. We can now map to this structure additional standard graph theoretic terminology, such as tree roots, children nodes, etc.

A normalized distance measure is imposed, through a series of definitions, to obtain a distance measure between any pair of sequences in the same tree. Protonet currently bases its distance measure only on Blast pairwise scores, but it experiments with three types of averaging in defining the distances (and subsequent merger decisions), yielding in effect three, non-identical hierarchical views of the known protein world.

Similar approaches are used to generate the **CluSTr** (Kriventseva et al., 2003) and **SYSTERS** (Krause et al., 2000) databases. Both also start from an all against all pairwise comparison, and carry an analysis at different levels of protein similarity, yielding a hierarchical organization of clusters. However, Clustr relies on the Smith-Waterman algorithm as the basis for its scoring mechanism. It uses Monte-Carlo simulation mediated Z-scores to estimate the statistical significance of similarity between potentially related proteins. Systers starts from a set of weak Blast hits, which are symmetrized using local pairwise alignments, and assigned E-values. Both subsequently apply a standard single linkage clustering algorithms, which Systers augments with procedures for separating minimally overlapping sub-clusters.

BioSpace (Yona and Levitt, 2000), the other offspring of Protomap takes a step further and tries to merge sequence and structure classification into one coherent view. As a first step sequence based clustering is performed on the protein domains in the Scop database. While the structural information and hierarchical classification of Scop are both dropped, this initial step is still performed not on whole, possibly multi-domain proteins, but on single structural domain sequences. A profile is generated from each resulting cluster, and a database of all known protein sequences is searched using **PSI-BLAST** (Altschul et al., 1997).

Psi-blast is an iterative extension of the pairwise homology search tool Blast. Given a query sequence, the first step is to perform a Blast search against the given database. A human or an automated decision is taken as to which of the top ranking matches are retained for the next iteration. These kept sequences are processed automatically into an alignment according to their best matching path to the query sequence. A subsequent search is performed with this profile against the database, trying to discover more distant homologies. This can be iterated a sufficient number of times, or until no further sequences join in.

In Biospace this procedure is started from a phase where a profile already exists, of the family members in a set. The result of these procedures are called type-I clusters, each having at least one solved structure from the Scop seeds. The remaining sequence space is clustered using Psi-blast, repeatedly starting from each yet unclustered sequence. At the last step these structure-less type-II clusters are merged with the type-I clusters to arrive at an automated Scop-like hierarchy of folds, super-families and families.

2.3 Protein Sequence Segmentation

In a recent survey of the structural genomics goal, Liu and Rost (2002) claim that clustering efforts which treat each input sequence as a whole are doomed to failure. The authors go on to demonstrate that segmentation of protein sequences into their respective domains is a necessary step in all such attempts.

Protein segmentation is also important for other purposes. For example, nuclear magnetic resonance (NMR) methods of structure determination are generally limited to short amino acid sequences which exhibit well defined structure. Also, in recent studies of protein-protein interactions (surveyed in Salwinski and Eisenberg, 2003) one often comes up with a list of proteins all shown to interact with a given protein. It is plausible then to hypothesize that all sequences in the list share a common structural domain which confers to them this specificity.

Goal. Given a set of unaligned, unlabeled protein sequences, detect all domains within the set, and segment the sequences to delineate the correct boundaries of each domain instance.

Protein segmentation from sequence information alone is a difficult task, made hard by our lack of deep understanding of the mapping between sequence and ensuing structure. Consider the small motivating example illustrated in Figure 2.7. Eight proteins are schematically drawn there, each having one or two structural domains. Yet, they share among them but five domains, in eight different combinations. A naive all-against-all pairwise comparison yields the similarity matrix of Figure 2.7 (right). Deducing the right number of domains, let alone their correct boundaries, is not trivial.

In the previous section we have touched on methods that try to address the segmentation challenge indirectly. Such are the attempts to soft cluster whole sequences into several sets simultaneously, and those starting a clustering effort from all known structural domain seeds. In this section we review computational efforts that try to address this challenge explicitly.

2.3.1 Alignment Based Methods

Greedy Segment Elimination

ProDom (Corpet et al., 2000) defines domain borders based on the local alignment obtained by Psi-Blast recursive homology searches (Gouzy et al., 1999). First, profiles are constructed for domains with known boundaries and the matches are pulled out from the sequence database by profile searching. If the extracted domain is not terminal in a protein sequence, the remaining sequence is cut into two parts that become two independent entries in the searched database. Other families are built iteratively, also based on Psi-Blast, by repeatedly drawing the shortest remaining

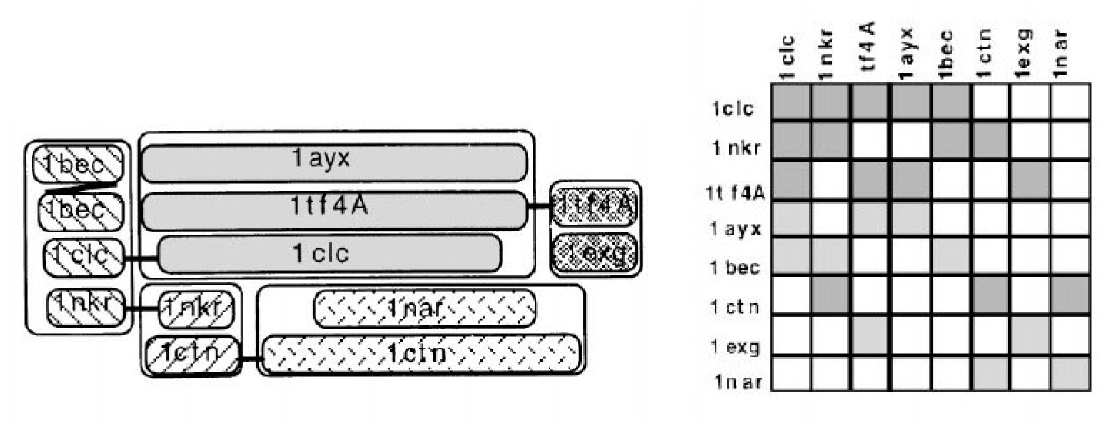


Figure 2.7: **The Diversity of Protein Domain Combinations.** (left) We illustrate eight proteins (named by their PDB codes) containing five different domain types. Domain types are drawn in a uniform pattern and parts of multi-domain proteins are joined by thick lines. The vertical arrangement of domains inside each box represents a multiple alignment. For example, there are four immunoglobulin-like domains at the far left, two of which belong to one protein. (right) Similarity matrix of the eight proteins. Each cell is shaded in proportion to the pairwise similarity between the two respective sequences. (adapted from Heger and Holm, 2000).

sequence to be used as a seed for the next Psi-Blast search. Each time a domain family is found, the corresponding fragments are extracted, as in the first step, from the depleting database. The process ends when Psi-Blast no longer finds any similarity between the remaining sequences.

The greedy elimination strategy determines to a large extent the resulting cluster identities and compositions. Also, at larger evolutionary distances, local sequence alignment methods may only detect a diminishing region of similarity, so that alignment borders no longer correspond to structural domains but represent conserved motifs, e.g., an active site vicinity, thereby inducing a problem of excessive fragmentation. Indeed, when known domains are generated by Prodom they tend to be shorter than those assigned from known protein structures (Liu and Rost, 2002). This has led to the recent integration of Pfam-A profiles and manual expert advice to correct Prodom border assignments.

Evolutionary History Tracking

Domaination (George and Heringa, 2002a) also uses a Psi-Blast driven iterative procedure. Instead of taking the minimal overlapping segment it studies the empirical distribution of observed N- and C- termini in the aligned sequences to identify potential domain boundaries. The underlying algorithm puts special emphasis on trying to infer the evolutionary history of the query sequence from related matches. Explicit searches are performed for evidence of domain deletion, domain shuffling, circular permutations of secondary structures and discontinuous domain assignments.

Special attention is also paid to tuning Psi-Blast searches to balance specificity and sensitivity, to avoid narrow pre-mature convergences on the one hand and drifting of the iterative process towards non-related sequence attractors on the other. Still, the assignment of a wrong member early on may very well pull in many other sequences unrelated to the query sequence.

Transitive Domain Delineation

DOMO (Gracy and Argos, 1998b) estimates the locations of domain borders in long sequences by transitively mapping the positions of known N- and C- termini to aligned sequences (Gracy and Argos, 1998a). First, pairwise sequence similarities are detected based on statistical significance of amino acid and di-peptide composition similarity, and local sequence alignments. Such ungapped regions are clustered into blocks called anchors. At this stage suspected fragments are eliminated, and all anchors are iteratively intersected to delineate proper sequence positions for domain termini. These fragments are then weighted and multiply-aligned, allowing for gaps for the first time in the process, through hierarchical clustering and comparison of domain profiles. Finally, the multiple alignments are extended towards sequence termini, conditioned on an acceptable level of similarity among constituents.

The fact that sequence delineation is performed prior to multiple alignment generation requires careful elimination of all fragments, which can be a difficult task given the uncertainties in eukaryotic exon prediction, and the existence of relatively short domains which appear in many domain combination contexts.

Multiple Alignment Dissection

Picasso (Heger and Holm, 2001) takes an opposite approach to Domo, aligning first and delineating only afterwards. It starts from highly overlapping sequence neighbourhoods revealed by all-against-all pairwise Blast alignment. Overlaps are then reduced by merging sequences or parts of sequences into multiple alignments. Merging proceeds agglomeratively, starting from many small clusters (multiple alignments) defined at high confidence. The decision of merging is based on the score of profile-profile comparison. At the end of this stage, each part of a sequence is covered by at least one multiple alignment. Definition of explicit domain boundaries is avoided while generating the multiple alignment covering. Instead, the domain composition is analyzed afterwards using set theoretic concepts, applied to parts of sequences. The key idea is to identify closed neighbourhoods, where one cluster contains all members that are linked by similarity, and no member has any neighbour outside of the cluster. Sets of domains that are neighbours of each other and recur, embedded in different protein contexts that are not related to each other, are searched for, based on their association with different sets of neighbours.

However, the use of extensive profile comparison, which is strongly influenced by seed alignment compositions, may lead to instability of the iterative profile searching process. Also, sequence-based domain definitions, focusing mostly on conserved blocks in a multiple alignment, has been shown to not necessarily reproduce structural domains (e.g., by Elofsson and Sonnhammer, 1999, who compared Pfam-A and Scop domain assignments).

Similarity Matrix Based Methods

GeneRAGE (Enright and Ouzounis, 2000) focuses primarily on the results of all-against-all Blast comparisons of complete sequences. First a boolean similar/dissimilar matrix of all comparisons is generated using cut-off thresholds. Pairwise asymmetries are resolved using the symmetric (but computationally costlier) Smith-Waterman scores. Multi-domain combinations are searched through non-transitivity of similarity. Namely, if A and B are similar to C, but not to each other, C is hypothesized to be a multi-domain protein (e.g., sequences 1exg, 1ayx and 1tf4A in Figure 2.7, left). Clustering is then performed on all supposedly single domain proteins and the hypothesized multi-domain ones are then used to search for instances where further cluster splitting is required.

The non-transitivity approach to exact segmentation is known to be hampered by nested domain composition similarities (such as A contained in AB contained in ABC), erroneous alignments and fragmented sequences (e.g., Gracy and Argos, 1998a). Indeed, this approach reportedly suited prokaryotic genomes, but did not scale well to eukaryotes where sequence fragments, complex domain combinations and shared domain elements abound (Enright et al., 2002).

TRIBE-MCL (Enright et al., 2002), by the same authors takes a different path from the same starting point, converting the all-against-all Blast matrix into a probabilities matrix. This is subsequently soft-clustered using an empirically-motivated iterative procedure. However, as in the revised Protomap database, multi-domain proteins appear as belonging to several clusters, and explicit domain delineation is not tackled.

2.3.2 Handling Unaligned Datasets

All the methods surveyed above rely, in one way or another, on aligning two or more sequences against each other. Pairwise alignment is indeed optimally solvable in terms of edit distance cost using the Smith-Waterman algorithm. However, especially for more distantly related proteins, the scoring function maximum does not necessarily coincide with the biologically correct alignment inferred via structure comparisons and related means. Moreover, scaling-up of the Smith-Waterman score is exponential in the number of sequences to be aligned, and thus impractical. Iterative methods, such as Psi-Blast which compare and add sequences to profiles are sub-optimal heuristic approximations to the optimal multiple alignment score. Despite these reservations alignment-less analysis of biosequences is largely restricted to the discovery of relatively short and simple patterns which are over abundant in the given set (reviewed by Brazma et al., 1998). We briefly survey several representative approaches, giving mathematical details for one method which is akin to ours.

Combinatorial Approaches

Combinatorial approaches aim to consider all patterns from a certain pre-defined set that appear within the dataset. Of these they output a subset of patterns which answer over-abundance criteria.

One such example is **Teiresias** (Rigoutsos et al., 1999) which systematically enumerates all maximal patterns in the given set that pass a predefined minimal number of occurrences.

Other tools use probabilistic modeling assumptions to define the expected abundance of different patterns, such as **Verbumculus** (Apostolico et al., 2000) which uses an assumed background probability distribution, and efficiently computes the abundance of observed patterns in units of standard deviation compared to the expected frequency of their occurrence.

Several similar approaches also exist, but are typically limited to relatively short, contiguous patterns due to their exhaustive nature.

Probabilistic Approaches

We term as probabilistic, approaches which do not attempt to consider every pattern of some predefined set. Rather such approaches typically search in a stochastic manner for over abundant patterns, from heuristically chosen starting points. We present one such tool in relative detail, as the approach it uses is akin to the one we will employ in our segmentation efforts later on.

This tool is called **MEME** (Bailey and Elkan, 1994). Meme uses an expectation maximization (EM) algorithm (Dempster et al., 1977) to search for probabilistically over-abundant gapless PSSMs of a given length range in the unaligned sequences, in the following manner.

Let $\bar{x} = x_1 \dots x_n$ represent the set of sequences, where each x_i obtains a value from an alphabet Σ of size L. We assume $\bar{x}$ has been generated by a background model we denote θ_1, except for several planted occurrences of a motif of length W, generated by another probabilistic model

θ_2. This type of model is termed a two component **finite mixture model**. We further assume the background model to be governed by a multinomial distribution $\theta_1 = \{f_{0j}\}_{j=1...L}$ where each symbol x_i is generated independently according to θ_1. Each occurrence of the motif is governed by a gapless profile model (Section 2.2.1) $\theta_2 = \{f_{ij}\}_{j=1...L}^{i=1...W}$. Each of the $i = 1 \ldots W$ symbols is generated independently from the multinomial distribution $\{f_{ij}\}_{j=1...L}$. We define a prior probability of choosing either of the two models $\lambda_j = P(\theta_j)$ for $j = 1, 2$, and assume that $\bar{x}$ was generated in the following manner: Choose either model with probabilities λ_1, λ_2; Generate W symbols according to the chosen θ_j; Repeat until you have generated all of $\bar{x}$. We will jointly denote the mixture model parameters by $\Lambda = \{\lambda_1, \lambda_2, \theta_1, \theta_2\}$.

The above procedure defines a segmentation process which we will denote by another set of random variables $z = \{z_{ij}\}_{j=1,2}^{i=1...n}$. In principle these are zero-one variables, such that $z_{ij} = 1$ iff x_i was generated by θ_j. However, as in the HMM case, this set of variables is hidden and cannot be observed directly. Instead, we use them as a set of probabilistic estimators, such that z_{ij} denotes the probability estimate that x_i was generated by θ_j.

Our objective is to find a set of model parameters Λ that maximizes the likelihood of the data given the model, $P(\bar{x}|\Lambda)$. This goal has been termed the learning problem in HMM context (Section 2.2.1), and indeed $\bar{x}$ plays the role of the observed sequence O, and the segmentation z is akin to the hidden series of state transitions Q. As in the HMM scenario we define an iterative procedure which is an instance of the EM algorithm: Start with an initial guess of model parameters Λ^0. Next, for $\tau \geq 0$ deduce a segmentation using Bayes rule

$$\forall i, j: \quad z_{ij}^{\tau} = \frac{\lambda_j^{\tau} p(x_i|\theta_j^{\tau})}{\sum_{k=1}^{2} \lambda_k^{\tau} p(x_i|\theta_k^{\tau})}$$

Then find a novel set of parameters that solves

$$\Lambda^{\tau+1} = \arg\max_{\Lambda} \sum_{i,j} z_{ij}^{\tau} \log \lambda_j p(x_i|\theta_j)$$

Here as well, the maximum can be solved analytically. It can be shown (e.g., Durbin et al., 1998) that for every such iteration $P(\bar{x}|\Lambda^{\tau+1}) \geq P(\bar{x}|\Lambda^{\tau})$. Thus, this procedure converges to some local maximum of the likelihood function.

The Meme algorithm uses additional heuristics. It transforms the input sequence into a set of overlapping W-mers, on which the segmentation is performed. During each EM step, the optimal $\Lambda^{\tau+1}$ parameters are smoothed, as in the HMM case, to address small sample size over-fitting, using user defined prior counts. The EM procedure is also strongly dependent on the initial guess Λ^0 because it is essentially a gradient ascent method. Every improvement beyond the initial set of parameters improves the likelihood function. As a result a relatively small region of the parameter space is searched prior to convergence. Meme addresses this issue by running the EM procedure multiple times employing further heuristics to select different promising starting points. The converged model that maximizes $P(\bar{x}|\Lambda)$ is returned.

Other tools use **Gibbs sampling** methods to stochastically search for similar patterns (Lawrence et al., 1993). Gibbs sampling algorithms combine gradient ascent steps with random search space jumps. The latter allow the exploration of larger parts of the parameter space per run but can also cause the procedure to spend many more increments before converging. In both approaches the resulting patterns are not guaranteed to be the optimal ones.

An interesting extension of Meme, termed **Meta-MEME** (Grundy et al., 1997) takes as input the unaligned sequence set and the MEME generated motifs. From these Meta-Meme concludes the best observed linear ordering of motifs in the set. A motif-based HMM is then built where

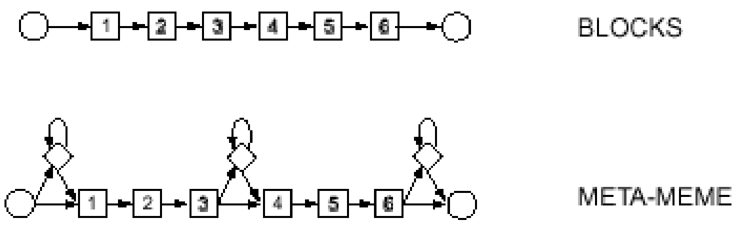


Figure 2.8: **Meta-Meme motif-based HMM.** The Meme algorithm is run on an input set of unaligned sequences and obtains several over-abundant motifs. Each of these is represented in the form of an ungapped profile, similar to a single Blocks model (top). The Meta-Meme algorithm then orders these linearly and builds a unified simplified HMM. In this model each motif profile is represented by a set of match states (e.g., states 1–3 above), and each spacer between two consecutive motifs, as well as before the first and after the last motif, is modeled by a single self-referencing insert state. The resulting model is much simpler than the profile HMMs of Figure 2.5, and resembles a Blocks fingerprint. (adapted from Eddy, 1998b).

each motif is modeled by a fixed length series of match states, and each spacer region is modeled by a single self referencing insert state (Figure 2.8). This reduction in the number of parameters compared to a profile HMM allows Meta-Meme to be more accurately trained from smaller sets of sequences. While no claim is made as to whether these heuristically found motifs are guaranteed to span the length of an unaligned domain, Meta-Meme does attempt to provide a statistical fingerprint (Section 2.2.1) of the region it models, by searching databases for homologs using the resulting model.

Homogeneity Segmentation

The approach we will subsequently develop towards sequence segmentation is also akin to another, albeit simpler computational goal. That is the segmentation of biosequences (DNA, RNA, proteins), into regions, each of which is homogeneous with respect to some property. Such efforts are applied to multiple alignments as well as unaligned sequences. Segmentation is performed according to relatively simple criteria such as base or amino acid compositional homogeneity, or regions exhibiting roughly similar positional variability (in alignments). These approaches includes works by May (2002); Ramensky et al. (2001); Xing et al. (2001) and references therein.

Our, more ambitious effort, in Chapter 5, will attempt to unify these two goals by simultaneously segmenting and clustering unaligned sequences into regions sharing higher order statistics.

2.3.3 Other Approaches

Wheelan et al. (2000) have demonstrated that simple collected statistics of domain size, and domain segment size distribution can allow one to guess with partial success the boundaries for relatively short sequences of up to 400 amino acids. This work serves as a representative example for integrating additional considerations into the complete prediction schemes above.

SnapDRAGON (George and Heringa, 2002b) employs an *ab initio* folding algorithm based on physico-chemical properties to generate a large number of 3D models for a given multiple alignment with predicted secondary structure. Assuming that hydrophobic residues tend to cluster together in space, domain boundaries are assigned automatically to each model. A final prediction is obtained by examining the consistency of these.

Several other preliminary approaches, including neural network based tools, are reviewed by Liu and Rost (2003).

2.3.4 Integrative Resources

Having realized that no single classification or segmentation approach is uniquely superior to its counter-parts, the bioinformatic community now advocates the use of practically all established tools to obtain different views on the nature of the sequences at hand. In order to facilitate this approach, meta-databases are being developed.

Goal. Given an unlabeled protein sequence, query with it several of the above databases. Then present one coherent view of the results, resolving conflicts correctly by estimating the strong points of each queried resource.

InterPro (Mulder et al., 2003) is an integrated documentation resource for protein families, domains and sites. InterPro combines the following databases, all of which have been introduced above: Prosite, Prints, Pfam, Smart, Tigrfams and Prodom. Rather than simply displaying the results, Interpro provides internal consistency checks and deeper coverage making it more efficient and reliable than using each of the pattern databases separately. The united approach improves the utility and the coverage of pattern databases, pin-points weaknesses and facilitates their further development.

Two other meta-databases are **MetaFam** (Shoop et al., 2001) and **iProClass** (Huang et al., 2003). Metafam is built from ten protein family databases, including several fully automated ones. On top of these it automatically builds supersets that facilitate comparing the different underlying approaches. Iproclass tries to provide comprehensive protein annotation at many sequence, structural and functional levels. It provides rich links to over fifty databases of protein families, functions and pathways, protein-protein interactions, post-translational modifications, structures and structural classifications, genes and genomes, ontologies, literature, and taxonomy. Comprehensive reports are generated at both the sequence level and the super-family level, summarizing and linking to the many available information sources it incorporates.

These databases face non-trivial challenges, because sequence databases generally contain poor positional pointers of functional domains, and include second-hand, erroneous annotations. These errors percolate through the databases which reference and use each other to update. As a result they often cannot be traced back to the chain of decisions which led to their formation (Gilks et al., 2002).

2.4 Discriminative Analysis

In the previous sections we have discussed the segmentation of protein sequences into their underlying domains and the classification of novel sequences into their respective families. However, many protein families contain several functional sub-types such as different substrate specificities. As discussed in Section 1.2, these functional differences define a further classification level into distinct sub-families. Sequence based classification into the different sub-families is a challenging task. While the entire sequences are subject to diverging mutational forces, functional variation often depends only on a small subset of the variable positions.

Goal. Given a set of unaligned protein sequences, tagged as belonging to distinct sub-families, generate a model which can correctly classify novel sequences into the sub-types. Furthermore, analyze the resulting model to detect residues that best discern between the different sets.

For the main objective, one supposedly could apply the classification approaches advocated in Section 2.2 in the context of families and super-families. However, Prosite regular expressions will not do, as it is seldom that sub-types differ in one short contiguous region and in a form that can be characterized by deterministic signatures. On the other hand the scores of probabilistic profiles and HMMs are accumulated over entire sequences, giving no special preference to any alignment position. From this point of view often the sequences in a family are all too similar to each other to allow a meaningful separation. The only viable approach of those is the sequence fingerprints approach taken by the Prints database. However, for the second objective, Prints fingerprints are too long and are chosen with no direct correlation to specific residues that differ between the sub-types.

Most tailored methods rely on a multiple alignment of the family sequences, as well as a phylogenetic tree inferred from it. Indeed, when the resulting tree can be constructed accurately, functional sub-types can often be identified with sub-trees within it. This observation has also been a starting point to works which try to automate the division of a protein family into its underlying sub-families by collapsing nodes of the phylogenetic tree (e.g., Wicker et al., 2001). Other methods use multiple alignments and phylogenetic trees to try and predict the functionally important sites which are conserved within the sub-families while variable between them. For example, three different representative methods are compared in del Sol Mesa et al. (2003). The first method takes as a starting point a phylogenetic representation of a protein family and, using measures of divergence between site distributions, automatically searches for a division of the family into sub-families. The second method looks for positions whose mutational behavior is similar to the mutational behavior of the full-length proteins, by directly comparing the corresponding distance matrices. The third method performs vector-based principal component analysis on distributions of sequences and amino acid positions in corresponding multidimensional spaces.

However, the division of proteins into functional sub-types cannot always be accomplished by phylogeny. In families which have evolved over a long period of time, or were subjected to rapid evolution, phylogeny often cannot give a clear division. In addition, proteins usually have multiple features that co-evolve, at different rates, making phylogenetic inference more complicated. Finally, molecular function may also evolve convergently, particularly where specificity is conferred by few residues.

In the remainder we focus on a representative work that tries to predict sub-type assignment for novel sequences, and to deduce the residues which confer sub-type specificity using HMMs. Hannenhalli and Russell (2000) start from a multiple alignment of the family representatives, where each sequence is labeled by its sub-type. For each sub-type s they train an HMM from the alignment of available sequences of that type, correcting for small sample size and sequence bias. They then normalize the HMM score profile to obtain probabilities of observing symbol x at position i in sub-type s, denoted $P_i^s(x)$, such that

$$\forall i, s : \quad \sum_x P_i^s(x) = 1$$

For a sub-type s let $\overline{s}$ denote the union of all sub-types excluding s. The score of an alignment column is defined by comparing the statistics of each sub-type to the union of all others, using the Kullback-Leibler divergence measure between two distributions (Cover and Thomas, 1991). The

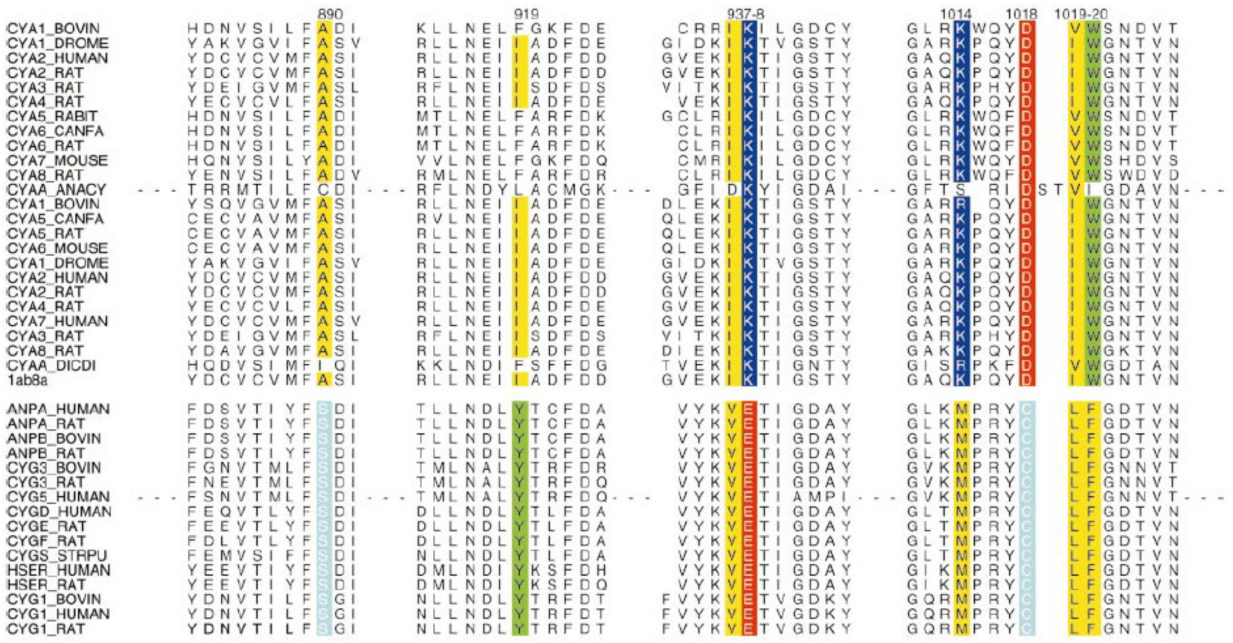


Figure 2.9: **Alignment scoring for sub-family specificity.** An alignment of representatives from the two sub-families of nucleotidyl cyclases, one acting on GTP and the other on ATP. High scoring residues according to Hannenhalli and Russell (2000) are highlighted. Only regions of high-scoring columns are shown, whereas skipped regions are indicated by dashes. Numbers above the alignment correspond to a representative family member with solved structure (PDB code 1ab8). For this set, Tucker et al. (1998) have shown that a simultaneous change in residues 938 and 1018 suffices to change protein function from that of one group to the other. (adapted from Hannenhalli and Russell, 2000).

divergences of all sub-types are summed to obtain a column score,

$$C_i = \sum_s \sum_x P_i^s(x) \log \frac{P_i^s(x)}{P_i^{\overline{s}}(x)}$$

These scores are then normalized using their empirical mean μ and standard deviation σ to obtain Z-scores,

$$Z_i = \frac{C_i - \mu}{\sigma}$$

Using empirical calibration, a threshold of $Z_i > 3$ is chosen, as it is often seen to correlate with specificity determining positions. The specificity conferring amino acids are searched for in the high scoring columns by computing the ratio

$$L_i^s(x) = \frac{P_i^s(x)/P_i^{\overline{s}}(x)}{\sum_y P_i^s(y)/P_i^{\overline{s}}(y)}$$

The threshold $L_i^s(x) \geq 0.5$ is then chosen, based on further empirical calibration, for reporting specificity conferring amino acids.

Figure 2.9 shows the results of applying this method to the nucleotidyl cyclase family. This family includes membrane attached or cytosolic domains that catalyze the reaction that forms a cyclic nucleotide monophosphate from a nucleotide triphosphate. The known cyclases act either on GTP (guanalyate cyclase) or ATP (adenylate cyclase), forming two functionally distinct sub-families. Representative members of both sub-families are shown aligned in Figure 2.9. Columns scoring $Z_i > 3$ are numbered in the figure, and individual residues are highlighted within each such column whenever $L_i^s(x) \geq 0.5$ for the respective amino acid in that particular sub-type. Tucker et al. (1998) have shown that mutations of two specific residues are sufficient to change the specificity of the enzyme from GTP to ATP, or vice versa. These residues, numbered 1018 and 938 in the figure, receive first and third Z-score ranks, respectively. However, the second ranking residue, no. 1014, does not seem to confer specificity in family members.

To classify a novel family sequence into a certain sub-type, the authors compare Blast and HMM based scoring schemes to a sub-profile score,

$$Sc(x_1 \dots x_n | P^s) = \prod_{i:\ Z_i > 0} P_i^s(x_i)$$

which eliminates contributions from non discriminating alignment positions. It is shown that in most cases the sub-profile scoring scheme indeed out-performs the other methods.

Chapter 3

Markovian Protein Family Classification

In this chapter we define the variable memory modeling framework which is at the center of this thesis. We adapt one such learning algorithm to the challenge of protein sequence modeling and demonstrate its ability to classify protein sequences into their respective families.

3.1 Markovian Sequence Modeling

We begin by recalling the task of supervised protein sequence classification, surveyed in Section 2.2.1

Goal. We are first given a family seed S_F of m proteins $\{s^1, s^2, \dots, s^m\}$, each represented by its primary sequence alone. These proteins are all tagged as members of a potentially larger protein family $F \supseteq S_F$. We are then confronted with a database D of many protein sequences, some of which belong to F, but not necessarily to S_F. Our goal is to pick from the database all proteins that belong to F, and only these.

As we saw in Chapter 1, the databases we wish to examine can be as large as hundreds of thousands of individual sequences. The discrimination task must therefore be performed automatically. Given a family seed we shall build a computational model M_F. When observing a novel sequence $s \in D$, this model will be capable of deciding whether $s \in F$, or not.

All protein family classification models reviewed in Chapter 2 can be cast this way. The most prominent of these, the HMM, relied on the fact that a common evolutionary source allows to align family members to each other meaningfully, and that different positions along a protein multiple alignment show different preferences for specific amino acids. We turn to define a different approach to the same problem, termed **variable memory modeling** (VMM), which will be the focal point of this thesis. This approach, originally defined by Rissanen (1983), will have certain noticeable advantages over HMMs, and will also lead us to further, more ambitious applications.

As with HMMs, we will try to fit the data with a generative model. We recall from Chapter 2 that for each query sequence s we will compute the probability $P(s)$ that this sequence was generated by our trained model. To obtain a binary decision we will threshold either $\frac{P(s)}{|s|}$, where $|s|$ is the length of the query sequence, or $\frac{P(s)}{P_0(s)}$, where P_0 models a random protein sequence from no particular family. In order to recognize members of the family F, we would like to improve the probability of generating $P(s)$ for all sequences we are given, S_F. However, we must do so primarily using features we believe are shared by all members of F, or we will **over-fit** the training data, and will be able to recognize only sequences from S_F itself.

To obtain a probabilistic framework, we treat each protein sequence as a series of random variables $s = s_1 s_2 \dots s_l$, where each s_j obtains a discrete value from a finite alphabet Σ, of 20 amino acids (Figure 1.1). The chain rule of probabilities states that the joint occurrence of any set of random variables can be decomposed in the following manner

$$P(s_1 \dots s_l) = P(s_1) \prod_{j=2}^{l} P(s_j | s_1 \dots s_{j-1})$$

Theoretically, We can now estimate all $P(s_j|s_1 \dots s_{j-1})$ terms from the sequences in S_F to try and maximize $P(s)$ for all $s \in F$. This approach, however, is clearly both intractable, and extremely over-fitted to S_F itself. Consider that for the j-th such term there are $|\Sigma|^{j-1}$ possible contexts in which we need to be able to predict reliably the identity of the next symbol, s_j. Tractable mathematical models condense the information available in the past, $s_1 \dots s_{j-1}$, which is relevant to predict the present symbol s_j using **state variables**.

Recall that in a profile HMM context, each amino acid in s is emitted at a specific match or insert state. These, together with the silent delete states constitute the state variables of an HMM. Once a path Q through the model is chosen, the sequence context of symbol s_j is completely replaced by the single state q_{t_j} from which s_j is emitted

$$P(s_j|s_1 \dots s_{j-1}) \approx P(s_j|q_{t_j})$$

We shall take a different modeling approach to summarize past events. First, we approximate each past term using its fixed length suffix

$$P(s_j|s_1 \dots s_{j-1}) \approx P(s_j|s_{j-L} \dots s_{j-1})$$

By doing so we effectively try to approximate the sequence generation using a Markov process of order L. Notice that now our state variables are no longer hidden. For each s_j we can deduce the corresponding state variable from its observed past. This formulation, however, is only meaningful biologically in the context of an alignment. Due to frequent insertion and deletion events in related protein sequences, combining $P(s_j|s_{j-L} \dots s_{j-1})$ statistics from these makes sense only when s_j relates to the same alignment column. HMM formulation makes this restriction explicit, by replacing the context altogether with an alignment position variable. Taking an opposite approach, we will leave only the context, and drop any alignment information, by defining

$$P(s_j = \sigma_0 | s_{j-L} \dots s_{j-1} = \sigma_{-L} \dots \sigma_{-1}) = P(c_0 = \sigma_0 | c_{-L} \dots c_{-1} = \sigma_{-L} \dots \sigma_{-1})$$

where $c_{-L}, \dots, c_0$ are position independent random variables, each obtaining a value from Σ. Thus, in our generative model wherever the immediate past equals $\sigma_{-L} \dots \sigma_{-1}$, we use the same prediction vector for the next symbol, regardless of the actual position within the sequence at which prediction is made. Such models are termed **stationary**, or time-invariant.

So far we have chosen to model our data using an L-order Markov model, which is effectively a table of $P(\sigma_0|\sigma_{-L} \dots \sigma_{-1})$ terms, of size $|\Sigma|^L \times |\Sigma|$. However, this modeling approach is problematic for our needs. On the one hand, choosing a small L yields a very tractable model, but as we will demonstrate later, it poorly differentiates members of a protein family from other sequences. On the other hand, increasing L requires an amount of sequence data we typically do not have in S_F. For example, to estimate a full model of size $L = 3$, we need some idea of the frequency of all 160,000 possible 4-mers.

The following property helps refine this model. Let the combined length of all sequences in S_F be denoted n. While there are $|\Sigma|^L$ possible L-mers in general, there are less than $n-L$ different (overlapping) L-mers in S_F itself. Therefore instead of modeling using an order L Markov model, for which we typically do not have enough statistics, we will model sequence generation using contexts of *varying* lengths, up to length L. Consider the following modeling approach:

For $k = 1 \dots L$, examine all $P(\sigma_0|\sigma_{-k} \dots \sigma_{-1})$ statistics that appear in S_F,

Memorize only $P(\,\cdot\,|\sigma_{-k} \dots \sigma_{-1})$ vectors considered *typical* of F for future predictions.

Before we define which contexts are memorized let us see how the resulting set of contexts, C_F, is used: For each $c \in C_F$, $c = \sigma_{-k} \dots \sigma_{-1}$ we have memorized the distribution vector $P(\sigma|c)$. For prediction we use the *longest* memorized suffix at each position

$$P(s_j = \sigma_0 | s_1 \dots s_{j-1} = \sigma_{-j+1} \dots \sigma_{-1}) \approx P(\sigma_0 | \max_{k \geq 0}\{\sigma_{-k} \dots \sigma_{-1} \in C_F\})$$

If we ensure that the empty context is also added to C_F then the above expression is always well defined. Also note that we examine less than nL different k-mers which can appear in S_F, of all allowed lengths.

Having defined its subsequent use, we return to resolve the issue of which prediction vectors $P(\,\cdot\,|\sigma_{-k} \dots \sigma_{-1})$ get memorized during training. Intuitively, there are two relevant aspects:

1. How frequent the suffix $\sigma_{-k} \dots \sigma_{-1}$ is in the training set S_F. Infrequent suffixes may yield under-sampled non-representative distributions which are best avoided.

2. Aiming to obtain compact models, it is also useful to ask whether adding the suffix will yield significantly different predictions.

Throughout the thesis we will experiment with different quantitative answers that try to combine these two criteria, according to our modeling needs. We turn to define and evaluate a first variant, put forth by Ron et al. (1996).

3.2 Theory

A **Prediction** or **Probabilistic Suffix Tree** (PST) over an alphabet is a non empty tree, whose nodes vary in degree between zero (for leaves) and the size of the alphabet. Each edge in the tree is labeled by a single symbol of the alphabet, such that no symbol is represented by more than one edge branching out of any single node (hence the degree of each node is bounded by the size of the alphabet). Nodes of the tree are labeled by a string, which is the one generated by walking *up* the tree from that node to the root. Each node is assigned a probability distribution vector over the alphabet. When the PST is used to predict significant patterns within a query string (i.e., segments of high probability), this probability distribution vector comes into play. It corresponds to the probabilities the tree assigns to a query symbol, given that the longest subsequence of symbols that have been observed before it in the query matches that particular node's label. An example of a PST model is given in Figure 3.1.

It should be noted that the PST differs from, albeit is related to, the classical suffix tree, which contains all the suffixes of a given string (see Gusfield, 1997). Consider, for example, the PST in Figure 3.1, where we refer to a node using the unique label associated with it. In a suffix tree the father of node(bra) would have been node(br), whereas in a PST the father of a node is a

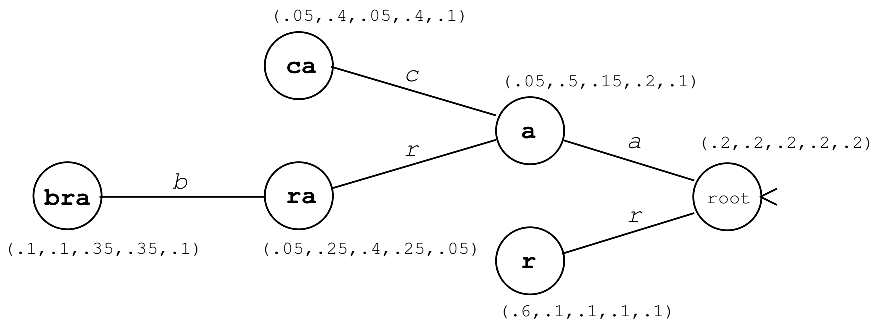


Figure 3.1: **An example of a PST model** over the alphabet $\Sigma = \{a, b, c, d, r\}$. The tree is shown in landscape mode, which makes the prediction step (Section 3.2.3) easier to follow. The root is the rightmost node. The vector that appears near each node is the probability distribution over the next symbol. For example, the probability distribution associated with the subsequence *ra* is 0.05, 0.25, 0.4, 0.25 and 0.05 for the symbols a, b, c, d and r respectively. Thus, the probability to observe *c* after a subsequence, whose largest suffix in the tree is *ra*, is 0.4.

node without the first (as opposed to last) symbol. Here the father of node(*bra*) is node(*ra*). The following observation specifies the relation between the two data structures: The skeleton (nodes, edges and labels) of a PST for a given input string is simply a subtree of the suffix tree associated with the *reverse* of that string. The differences become clear when following the tree construction procedure, described in Section 3.2.2.

3.2.1 Definitions

Let Σ be the alphabet (e.g., the alphabet of 20 amino acids for protein sequences, or 4 nucleotides for DNA sequences), and let $r^1, r^2, ..., r^m$ be the sample set of m strings over the alphabet Σ, where the length of the i-th ($i = 1..m$) string is l_i (i.e., $r^i = r^i_1 r^i_2 ... r^i_{l_i}$ $\forall r^i_j \in \Sigma$).

First, we define the empirical probability of a subsequence s over the given sample set as the number of times this subsequence was observed in the sample set divided by the maximal number of (possibly overlapping) occurrences a pattern of the same length could have had, considering the sample size. Formally speaking, given a string s of length l ($s = s_1 s_2 ... s_l$) we define a set of variables

$$\chi_s^{i,j} = \begin{cases} 1 & \text{if } s_1 s_2 ... s_l = r^i_j r^i_{j+1} ... r^i_{j+(l-1)} \\ 0 & \text{otherwise} \end{cases}$$

for each $i = 1..m$ and $j = 1..l_i - (l-1)$. Such indicator variable $\chi_s^{i,j}$ has a value of one if and only if the string s is a subsequence of r^i starting at position j.

The number of (possibly overlapping) occurrences of string s in the string set $\{r^i\}$ is given by

$$\chi_s = \sum_{i,j} \chi_s^{i,j}$$

The total number of (overlapping) subsequences of length $|s| = l$ within the set $\{r^i\}$ is

$$N_{|s|} = \sum_{i \ \ s.t. \ \ l_i \geq l} (l_i - (l-1))$$

We choose to define the empirical probability of observing string s as the ratio between these last two quantities

$$\tilde{P}(s) = \frac{\chi_s}{N_{|s|}}$$

The exact empirical probability depends on the number of possible occurrences of s in the sample set. In general, computing the maximal number of possible occurrences of a *specific* string s is more complicated and is dominated by the period of that string (the minimal interval which will allow it to overlap itself). Our definition implicitly disregards the fact that the subsequences accounted for are indeed overlapping, and therefore are not independent of each other. However, this definition suffices for our purposes. Note that it also leads to a natural definition of a probability distribution over all strings of length l since $\sum_{s \in \Sigma^l} \tilde{P}(s) = 1$.

We go on to define the conditional empirical probability of observing a symbol right after a given subsequence. This probability is defined as the number of times this symbol has shown up right after the given subsequence divided by the total number of times this subsequence has shown up at all, followed by any symbol. Specifically, let χ_{s*} be the number of non-suffix occurrences of the string s in the string set $\{r^i\}$, i.e.,

$$\chi_{s*} = \sum_{\sigma' \in \Sigma} \chi_{s\sigma'}$$

Then the conditional empirical probability of observing the symbol σ right after the string s is defined by

$$\tilde{P}(\sigma|s) = \frac{\chi_{s\sigma}}{\chi_{s*}}$$

It is easy to verify, using the above definitions, that the obtained $\tilde{P}(\sigma|s)$ distribution satisfies the marginal consistency condition, such that $\tilde{P}(\sigma|s) = \sum_{\hat{\sigma} \in \Sigma} \frac{\tilde{P}(\hat{\sigma}s)}{\tilde{P}(s)} \tilde{P}(\sigma|\hat{\sigma}s)$, yielding a stationary set of distributions (Ron et al., 1996). Finally, we define $suf(s) = s_2 s_3 ... s_l$, and $s^R = s_l ... s_2 s_1$.

3.2.2 Building a PST

First, we define L to be the memory length of the PST (i.e., the maximal length of a possible string in the tree). We work our way gradually through the space of all possible subsequences of lengths 1 through L, starting at single letter subsequences, and abstaining from further extending a subsequence whenever its empirical probability has gone below a certain threshold (P_{min}), or once it reaches the maximal L length boundary. The P_{min} cutoff avoids an exponentially large (in L) search space.

At the beginning of the search we hold a PST consisting of a single root node. Then, for each subsequence we decide to examine, we check whether there is some symbol in the alphabet for which the empirical probability of observing that symbol right after the given subsequence is non negligible, and is also significantly different from the empirical probability of observing that same symbol right after the string obtained from deleting the leftmost letter from our subsequence[1]. Whenever these two conditions hold, the subsequence, and all necessary nodes on its path, are added to our PST.

The reason for the two step pruning (first defining all nodes to be examined, then going over each and every one of them) stems from the nature of PSTs. A leaf in a PST is deemed useless if its prediction function is identical (or almost identical) to that of its parent node. However, this in itself is no reason not to examine its sons further while searching for significant patterns. Therefore, it may (and does) happen that consecutive inner PST nodes are almost identical.

Finally, the node prediction functions are added to the resulting PST skeleton, using the appropriate conditional empirical probability, and then these probabilities are smoothed using a standard

[1]This string corresponds to the label of the direct father of the node we are currently examining (note that the father node has **not** necessarily been added itself to the PST at this time).

```
Build-PST ($P_{min}$, $\alpha$, $\gamma_{min}$, $r$, $L$):
  1. Initialization: let $\bar{T}$ consist of a single root node, with an empty label $\lambda$,
     and let $\bar{S} \leftarrow \{\sigma \mid \sigma \in \Sigma \ and \ \tilde{P}(\sigma) \geq P_{min}\}$.
  2. Building the PST skeleton: While $\bar{S} \neq \phi$, pick any $s \in \bar{S}$ and do
     (A) Remove $s$ from $\bar{S}$
     (B) If there exists a symbol $\sigma \in \Sigma$ such that
             $(I) \quad \tilde{P}(\sigma|s) \geq (1+\alpha)\gamma_{min}$
         and
             $(II) \quad \frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|suf(s))} \begin{cases} \geq r \\ or \\ \leq 1/r \end{cases}$
         then add to $\bar{T}$ the node corresponding to $s$ and all the nodes on the path to $s$
         from the deepest node in $\bar{T}$ that is a suffix of $s$.
     (C) If $|s| < L$ then for every $\sigma' \in \Sigma$, if
             $\tilde{P}(\sigma' \cdot s) \geq P_{min}$,
         then add $\sigma' \cdot s$ to $\bar{S}$.
  3. Smoothing the prediction probabilities: For each $s$ labeling a node in $\bar{T}$, let
             $\bar{\gamma}_s(\sigma) \equiv (1 - |\Sigma|\gamma_{min})\tilde{P}(\sigma|s) + \gamma_{min}$        (3.1)
```

Figure 3.2: **The PST Learning Algorithm,** adapted from Ron et al. (1996).

technique so that no single symbol is absolutely impossible right after any given subsequence (even though the empirical counts may attest differently).

In Figure 3.2 we present the procedure for building a PST out of a sample set. The procedure uses 5 external parameters: L the memory length, P_{min} the minimal probability for string occurrence, r which is a simple measure of the difference between the prediction of the candidate at hand and its direct father node, γ_{min} the smoothing factor, and α, a parameter that together with the smoothing probability defines the significance threshold for a conditional appearance of a symbol (an example of an effective set of parameters is given in the legend of Table 3.1). We use $\bar{T}$ to denote the tree, $\bar{S}$ to denote the set of (unique) strings that we need to check, and $\bar{\gamma}_s$ to denote the probability distribution (over the next symbol) associated with the node s.

The final step (step 3) of the learning algorithm is the smoothing process, which assures that no symbol is predicted to have zero probability, independent of the suffix observed before it. The value of γ_{min} defines the minimum probability for a symbol, and the empirical probabilities should be adjusted to satisfy this requirement. This is done by decreasing the empirical probabilities, such that a total of $|\Sigma|\gamma_{min}$ is "collected", to be later shared by all symbols. The decrement of each empirical probability is done in proportion to its value, by solving the set of equations $\forall\sigma \in \Sigma$, $\bar{\gamma}_s(\sigma) = k * \tilde{P}(\sigma|s) + \gamma_{min}$ under the normalization constraint $\sum_{\sigma\in\Sigma} \bar{\gamma}_s(\sigma) = 1$. This yields $k = (1 - |\Sigma|\gamma_{min})$ constrained by $\gamma_{min} < \frac{1}{|\Sigma|}$ to assure non-negative values, and Equation 3.1 follows.

Returning to the PST of Figure 3.1, we can now give two exemplary observations on the data set from which the model was learned: First, in the training set there must be an overall clear preference for observing the letter b after the letter a (as $\bar{\gamma}_a(b) = 0.5$), unless the a itself was preceded by an r, in which case the preference is for c (as $\bar{\gamma}_{ra}(c) = 0.4$). Second, assuming that γ_{min} was set to 0.05, and examining the probability vector associated with node(ca), we can infer that only three different symbols, b, d and r, were observed in the training set succeeding the subsequence ca, in quantities that obey the ratio 7 : 7 : 1, respectively.

Theoretical Motivation

PSTs are a subclass of probabilistic finite automata (PFA). We use a PST model to approximate the natural source, learned from the family seed of protein sequences. Machine learning theory holds strong indications that efficient learning of general PFA sources is not possible (e.g., Kearns et al., 1994). Related works show that general HMMs cannot be trained or inferred efficiently (Abe and Warmuth, 1992; Gillman and Sipser, 1994, respectively). For PSTs, however, the opposite is true, as the following theorem shows.

Theorem. (Ron et al., 1996) Given a bound L on the depth of a source PST, and a bound n on its size, the algorithm of Figure 3.2 will generate a model arbitrarily close to the source PST, from a set of sequences generated by the source model, in time polynomial in L, n, the alphabet size $|\Sigma|$, and the desired accuracy parameters.

This result was later extended by Bühlmann and Wyner (1999) to also include certain classes of non-stationary PST sources. Empirically, these results indicate that PST training is potentially easier than HMM calibration, and may require a smaller, less optimal family seed to obtain separation.

A PST variant incorporating biological considerations

The basic PST model does not require any assumption on the input data, nor does it utilize any a-priori information we may have about the data. Incorporating such information may improve the performance of this model, since it adjusts the general purpose model to the specific problem of identifying significant patterns in macromolecules, where some characteristic features and processes are observed. Specifically, the information we would like to consider is the amino acids background probabilities, and the amino acids substitution probabilities. These distributions are integrated into the PST model and some changes are made to account for this a-priori information.

Few additional definitions are needed here: Let q_{ab} be the probability that amino acid a is replaced by amino acid b ($a, b \in \Sigma$). Also, define $\bar{\chi}_s$ as the number of *distinct* sequences which include the subsequence s. This number is required to exceed a certain threshold N_{min}, which is defined in proportion to the total number m of strings in the sample set (i.e., $N_{min} = c \cdot m$ where c is chosen to be, say, 0.2 so that the subsequence s is required to exist in at least 20% of the member proteins). Alternatively, this parameter can be set to a constant value regardless of the actual size of the training set.

Finally, the last step (step 3) of the learning algorithm (the smoothing step), is modified, and it is now based on a position-based pseudo-counts method (similarly to the method suggested by Henikoff and Henikoff, 1996). This method adds hypothetical counts to the sample set in order to avoid zero probabilities which are due to under-sampling. The number and the distribution of pseudo-counts is “position-based” (i.e., different for each node) and takes into account the *diversity* (the number of different amino acids observed after the corresponding subsequence), as well as the counts of actually observed amino acids.

For a node represented by the subsequence s, denote by R_s the diversity at s (number of different amino acids observed after s),

$$R_s = |\{\sigma \mid \chi_{s\sigma} > 0\}|$$

Denote by B_s the total number of pseudo-counts added to node s. We set $B_s = \mu R_s$, as suggested in Henikoff and Henikoff (1996), where μ has been optimized for best performance. Then, the number of pseudo-counts of amino acid a at this node is given by

$$\begin{aligned} b_a &= B_s \sum_{i=1}^{20} Prob(i|s) \cdot Prob(a|i) \\ &= B_s \sum_{i=1}^{20} \frac{\chi_{si}}{\chi_{s*}} \cdot \frac{q_{ia}}{Q_i} \end{aligned}$$

Where $Q_i = \sum_{k=1}^{20} q_{ik}$. The probability of observing a after the string s is defined as the weighted average of the empirical probability $\tilde{P}(a|s)$ and the a-priori probability as defined by the pseudo-counts, $P_{pse}(a|s) = b_a/B_s$.

The modified procedure is described in Figure 3.3. We briefly summarize the changes with respect to the external parameters: N_{min} substitutes P_{min}, while the pseudo counts matrix replaces γ_{min} in smoothing, leaving γ to take over for α and γ_{min} in threshold determination.

Incremental model refinement

A useful and very practical feature of the PST learning algorithm is the ability to refine any given PST model (over the same training set it was originally grown from) without the need to re-iterate calculations which have already been made while building the original model. Given the PST model to be further expanded, denoted T_0, one can, by relaxing the original learning parameters, in any of the ways described below, extend the model without repeating any of the calculations taken in building T_0 itself. In order to increase the number of nodes to be considered for inclusion in the resulting PST model one may lower P_{min} or increase L. In order to alleviate the criteria for acceptance into the tree one may lower r towards 1, or α towards (-1). Once the relaxed set of parameters has been chosen (any subset of them may be relaxed simultaneously) and T_0 has been passed to Build-PST, the initialization step of the algorithm should be altered to the following:

1. **Initialization:** if T_0 is empty
   (a) As before.
   (b) Else, let $T \leftarrow T_0$,
   and let $\bar{S} \leftarrow \{s \mid suf(s) \in T_0 \;\; and \;\; s \notin T_0 \;\; and \;\; \tilde{P}(s) \geq P_{min} \;\; and \;\; |s| \leq L\}$

The second variant can be improved incrementally much in the same way.

3.2.3 Prediction using a PST

Given a string s its prediction by a certain PST is done letter by letter, where the probability of each letter is calculated by scanning the tree in search of the longest suffix that appears in the tree and ends just before that letter. The conditional probability of this letter given this suffix is given by the probability distribution associated with the corresponding node in the PST. For example,

Build-Bio-PST $(N_{min},\ \gamma,\ r,\ L)$:

1. **Initialization:** let $\bar{T}$ consist of a single root node, with an empty label λ, and let $\bar{S} \leftarrow \{\sigma \mid \sigma \in \Sigma \ \ and \ \ \chi_\sigma \geq N_{min}\}$.
2. **Building the PST skeleton:** While $\bar{S} \neq \phi$, pick any $s \in \bar{S}$ and do
   (A) Remove s from $\bar{S}$
   (B) If there exists a symbol $\sigma \in \Sigma$ such that
   $$(I) \qquad \tilde{P}(\sigma|s) \geq \gamma$$
   and
   $$(II) \qquad \frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|suf(s))} \begin{cases} \geq r \\ or \\ \leq 1/r \end{cases}$$
   then add to $\bar{T}$ the node corresponding to s and all the nodes on the path to s from the deepest node in $\bar{T}$ that is a suffix of s.
   (C) If $|s| < L$ then for every $\sigma' \in \Sigma$, if
   $$\tilde{P}(\sigma' \cdot s) \geq P_{min},$$
   then add $\sigma' \cdot s$ to $\bar{S}$.
3. **Smoothing the prediction probabilities:** For each s labeling a node in $\bar{T}$, let
$$\begin{aligned} \bar{\gamma}_s(\sigma) &\equiv \frac{\chi_{s*}}{\chi_{s*} + B_s}\tilde{P}(\sigma|s) + \frac{B_s}{\chi_{s*} + B_s}P_{pse}(\sigma|s) \\ &= \frac{\chi_{s*}}{\chi_{s*} + B_s}\frac{\chi_{s\sigma}}{\chi_{s*}} + \frac{B_s}{\chi_{s*} + B_s}\frac{b_\sigma}{B_s} = \frac{\chi_{s\sigma} + b_\sigma}{\chi_{s*} + B_s} \end{aligned}$$

Figure 3.3: **A biologically motivated variant** of the PST learning algorithm.

to predict the string $s = abracadabra$ with the PST given in Figure 3.1 the following procedure is carried out:

$$\begin{array}{rcccccccc} P^T(abracadabra) = & P^T(a) & P^T(b|\underline{a}) & P^T(r|ab) & P^T(a|ab\underline{r}) & P^T(c|a\underline{bra}) & P^T(a|abrac) & \ldots & P^T(a|abracadab\underline{r}) \\ = & \bar{\gamma}_{root}(a) & \bar{\gamma}_a(b) & \bar{\gamma}_{root}(r) & \bar{\gamma}_r(a) & \bar{\gamma}_{bra}(c) & \bar{\gamma}_{root}(a) & \ldots & \bar{\gamma}_r(a) \\ = & 0.2 & 0.5 & 0.2 & 0.6 & 0.35 & 0.2 & \ldots & 0.6 \\ = 4.032 \cdot 10^{-6} & & & & & & & & \end{array}$$

The underlined subsequences represent the longest suffices that appeared in the tree (no characters are underlined when the longest suffix is the empty string). The probability of each letter is given by the prediction function that is associated with the corresponding node ($\bar{\gamma}_{root}(), \bar{\gamma}_a()$, etc.).

Note that a random uniform distribution over the alphabet would assign a probability of $0.2^{11} = 2.048 \cdot 10^{-8}$ to the string s, making it roughly 200 times more plausible under T than under the simple uniform model.

Two-way prediction

The prediction step described in the previous paragraph proceeds from left to right, starting from the leftmost letter. An obvious side effect of this property is that letters that mark the beginning of significant patterns are predicted with non significant probabilities. Only after a significant subsequence has been observed (i.e., once sufficient information is accumulated) the subsequent letters are predicted with high probability. Consequently, the (left) boundaries between significant and non significant patterns (i.e., domain/motif boundaries) are somewhat blurred. To accommodate for this property we have implemented a variant of the prediction step. Given a set of input sequences, two PST models are created, T and T^R. T is built from the sequences as they are, and T^R is built from the reversed sequences. The prediction step is repeated twice. The input sequence is predicted using T, and the reverse sequence is predicted using T^R. Then, the predictions are combined by taking the maximal prediction assigned by the two models. Thus, for $s = \tau\sigma\rho$ where $\sigma \in \Sigma$ and $\tau, \rho \in \Sigma^\star$, we set $P_{T,T^R}(\sigma|s) = \max\{P_T(\sigma|\tau), P_{T^R}(\sigma|\rho^R)\}$.

3.2.4 Computational Complexity

Denote the total length of the training set by n, the depth bound on the resulting PST by L, and the length of a generic query sequence by m. In these terms, the learning phase of the algorithm can be bounded by $O(Ln^2)$ time and $O(Ln)$ space, as there may be $O(Ln)$ different subsequences of lengths 1 through L, each of which can be searched for, in principle, in time proportional to the training set length, while each contributes but a single node beyond its father node to the resulting structure. The prediction phase is bounded by $O(Lm)$ time, since every symbol predicted requires traversing a path from the root down a tree of maximal depth L. A speed up in subsequent predictions to can always be achieved by a conversion of the tree into a not much larger probabilistic finite automaton, of identical predictions (see Ron et al., 1996). However, the time complexity of this procedure is $O(Ln^2)$. In the next chapter we develop an alternative, more powerful, algorithmic scheme to achieve the **optimal bounds** of learning in $O(n)$ time and space (regardless of L, allowing theoretically for unbounded trees) and predicting in $O(m)$ time.

The implementation described in this chapter was coded in ANSI C, compiled using `gcc`, and ran on Linux and BSDI based Pentium II and III machines. Actual run time on a Pentium II 300 MHz PC, for a protein family, varied between 30 seconds and 90 minutes, resulting in models of size 10–300 KB. In general, a tree of an average Pfam family contains some 6,500 nodes using the set of parameters given in Table 3.1. This is a small portion of all potential nodes that are checked during the learning algorithm (on average, nearly 33,000 potential nodes were inspected per model).

3.3 Results and Discussion

To test our approach, a PST was created for each of the 175 families in the Pfam database (Bateman et al., 2002) release 1.0, and tested against the Swissprot database (Boeckmann et al., 2003) release 33. We briefly recall from Section 2.2.1 that Pfam is a database of profile HMMs. These are built from expert curated multiple alignments of a small subset of the family, through a semi-automatic iterative procedure that identifies other family members in the Swissprot database (Boeckmann et al., 2003).

Each Pfam family is divided into a training set and a test set, in ratio 4 : 1, such that 4/5 of the family members in the form of complete, unaligned sequences serve as the PST training set. Then, for each sequence in the Swissprot 33 database, we calculate its probability as predicted by the PST. To avoid bias due to length differences, the probability is normalized by the length, and each sequence is reported along with its average probability per letter. Hits are sorted by decreasing probability.

The quality of the PST model is estimated by applying the **equivalence number criterion**

(Pearson, 1995). This criterion sets the family membership threshold at the point where the number of false positives (the number of non member proteins, according to Pfam, with probability above the threshold) equals the number of false negatives (the number of member proteins with probability below the threshold). In the terminology of Section 2.2.1 (page 12) it is the point where FP=FN, which sets a balance between specificity and sensitivity[2]. We will term this point the **iso-point**. A hit that belongs to the family (true positive) and scores above this threshold, is considered successfully detected. The quality of the model is measured by the number of true positives detected relative to the total number of proteins in the family. The results for the 170 protein families in the Pfam database release 1.0, with more than 10 members each, are given in Table 3.1. When averaged over all 170 families, the PST detected 90.7% of the true positives.

3.3.1 Performance Evaluation

The performance evaluation procedure that we applied assesses the quality of the PST model in terms of its ability to predict the correct assignments of proteins to *a-priori* defined groups, and our reference set here is the HMM based Pfam database. Note that this assessment does not measure the relative merit of the PST model with respect to the HMM in general, since the reference set depends on the HMM itself. In order to compare the performance of the PST model to the performance of the HMM in a more objective manner, we built an HMM for each family, out of the same training set that was used to build the PST model, and tested its ability to detect family members using the same equivalence number criterion. These HMMs were built automatically, without any manual calibration, using two public domain software packages which are available through the web, namely, the SAM package version 2.2 (Hughey and Krogh, 1998), and the HMMER package version 2.1 (Eddy, 1998a). An HMM is built there from a set of input sequences using the Expectation Maximization (EM) algorithm (Dempster et al., 1977). An initial model is constructed based on background distributions. The model is then improved iteratively by aligning the sequences to the current model and then recalculating the transition and emission probabilities based on these alignments. The process is repeated until it converges.

The compared methods

For each family, three HMMs were built. The first one was built directly from the set of unaligned sequences using the program ‘buildmodel’ of the SAM package. The model was then used to search the Swissprot database using the ‘hmmscore’ program. The other two models were built after the sequences were aligned first using the ClustalW program, version 1.75 (Higgins et al., 1996). These models were created using the program ‘hmmbuild’ which is part of the HMMER package[3]. In this package the mode of the search (local or global) is part of the model itself. Therefore, one of the models was created in a local/global mode (allows local match with respect to the sequence, and a global match with respect to the model, i.e., only a complete match with the model is reported), and the second was created in a local/local mode. Both allow multiple matches with the same sequence (corresponding to multiple copies of the same domain). These models were then used to search the Swissprot database using the ‘hmmsearch’ program of the HMMER package. The results of our assessment are summarized in Table 3.1.

For reference, we have also evaluated the performance of the HMMs which are part of the Pfam database itself (made available through the Pfam website). These HMMs are based on

[2]To see this note that lowering the threshold causes some sequence labels to turn from negative to positive. This in turn may increase sensitivity, and decrease specificity. And vice versa when the threshold is increased.

[3]In version 2.1, this program can build an HMM only from a multiple alignment of the input sequences.

Family	Size	Coverage	Sequences	% True Pos.	No. of sequences missed by					
					HMM	HMM	HMM	HMM	BLAST	BLAST
			PST missed	PST detected	Pfam	local	global	SAM	best	average
7tm_1	515	0.707	36	93.0%	13	1	7	1	12	64
7tm_2	36	0.735	2	94.4%	0	0	0	0	0	0
7tm_3	12	0.805	2	83.3%	0	0	0	0	0	0
AAA	66	0.378	8	87.9%	0	1	1	1	1	2
ABC_tran	269	0.518	44	83.6%	1	3	2	6	5	12
actin	142	0.965	4	97.2%	4	2	2	1	0	3
adh_short	180	0.661	20	88.9%	0	8	2	3	8	55
adh_zinc	129	0.970	6	95.3%	1	2	1	3	2	7
aldedh	69	0.907	9	87.0%	0	0	0	0	0	1
alpha-amylase	114	0.750	14	87.7%	0	2	1	3	2	18
aminotran	63	0.942	7	88.9%	0	1	0	1	16	28
ank	83	0.151	10	88.0%	3	9	26	3	9	39
arf	43	0.951	4	90.7%	0	0	0	0	0	0
asp	72	0.771	12	83.3%	7	1	5	1	0	3
ATP-synt_A	79	0.649	6	92.4%	3	1	1	0	1	11
ATP-synt_ab	180	0.694	6	96.7%	6	1	3	0	1	4
ATP-synt_C	62	0.855	5	91.9%	12	0	1	1	0	6
beta-lactamase	51	0.863	7	86.3%	0	0	0	0	9	17
bZIP	95	0.217	10	89.5%	1	4	6	2	22	46
C2	78	0.175	6	92.3%	3	7	16	7	23	47
cadherin	31	0.503	4	87.1%	0	1	1	1	2	5
cellulase	40	0.584	6	85.0%	0	1	1	2	8	17
cNMP_binding	42	0.466	3	92.9%	2	1	7	2	2	15
COesterase	60	0.900	5	91.7%	7	1	4	1	0	2
connexin	40	0.687	1	97.5%	0	0	0	0	0	0
copper-bind	61	0.835	3	95.1%	0	0	0	0	14	26
COX1	80	0.215	13	83.8%	1	4	3	5	2	6
COX2	109	0.897	2	98.2%	11	0	2	0	0	3
cpn10	57	0.953	4	93.0%	1	0	1	0	0	1
cpn60	84	0.948	5	94.0%	0	0	0	0	0	0
crystall	53	0.851	1	98.1%	0	0	0	0	0	2
cyclin	80	0.635	9	88.8%	2	2	2	2	4	12
Cys-protease	91	0.682	11	87.9%	11	2	9	0	0	4
cystatin	53	0.742	4	92.5%	1	1	20	3	13	27
Cys_knot	61	0.502	4	93.4%	0	1	-	6	12	25
cytochrome_b_C	130	0.313	27	79.2%	2	20	19	18	1	17
cytochrome_b_N	170	0.658	3	98.2%	22	2	3	0	0	1
cytochrome_c	175	0.891	11	93.7%	2	4	6	5	30	66
DAG_PE-bind	68	0.112	7	89.7%	1	7	5	13	9	33
DNA_methylase	48	0.846	8	83.3%	2	0	2	0	0	2
DNA_pol	46	0.650	9	80.4%	1	-	-	0	0	3
dsrm	14	0.226	2	85.7%	1	0	6	1	7	10
E1-E2_ATPase	102	0.636	7	93.1%	3	0	2	0	0	2
efhand	320	0.401	25	92.2%	27	28	52	25	42	105

Table 3.1: **PST performance for all Pfam families (part 1).** Families are ordered alphabetically. The names of the families are abbreviated as in the Pfam database. The number of proteins in the family is given in the second column. Coverage (third column) is the total portion of the sequences which is included in the multiple alignment used to define the domain or the family in the Pfam database. Each family was divided into a training set and a test set and the PST was built from the training set. To test the quality of this PST, we calculate the probability that the PST induces on each sequence in the Swissprot 33 database, and each family sequence (from the training set **and** the test set) whose probability is above the iso-point is considered successfully detected (see text for more details). The number of sequences missed (i.e., the equivalence number) and the percentage of true positives detected are given in the fourth and fifth columns respectively. The other columns give the number of sequences missed by: the Pfam HMM; an HMM trained on a multiple alignment in a local search mode (see text); an HMM trained on a multiple alignment in a global search mode; an HMM trained on the unaligned sequences; best Gapped-BLAST search, and average Gapped-BLAST search. The sign "-" denotes that results were not available (the program crashed). The set of parameters used to train the PST is $P\_min = 0.0001$ $\alpha = 0$ $\gamma\_min = 0.001$ $r = 1.05$ *and* $L = 20$. Additional improvement in the performance is expected if the parameters are tuned for each family (see text). To train a PST on a typical family with this set of parameters takes about half an hour on average, on a pentium II 300 Mhz (the range is between 30 seconds and 90 minutes). Additional 5 minutes are needed to predict all sequences in the Swissprot database. For comparison, training an HMM from unaligned sequences takes about two hours on average, and searching the Swissprot database with a typical HMM takes several hours.

Family	Size	Coverage	Sequences PST missed	% True Pos. PST detected	No. of sequences missed by HMM Pfam	HMM local	HMM global	HMM SAM	BLAST best	BLAST average
EGF	169	0.133	18	89.3%	28	120	118	53	46	98
enolase	40	0.983	0	100.0%	3	0	2	0	0	1
fer2	88	0.785	5	94.3%	2	1	2	1	0	5
fer4	152	0.559	18	88.2%	7	3	6	2	16	46
fer4_NifH	49	0.928	2	95.9%	5	0	3	0	0	1
FGF	39	0.691	1	97.4%	0	0	0	0	0	0
fibrinogen_C	18	0.469	4	77.8%	0	1	-	0	0	1
filament	139	0.607	5	96.4%	14	0	7	0	3	10
fn1	15	0.107	2	86.7%	1	1	1	4	6	9
fn2	20	0.141	2	90.0%	0	1	2	2	0	10
fn3	161	0.242	23	85.7%	1	85	95	43	61	116
GATase	69	0.605	8	88.4%	0	2	1	1	2	17
gln-synt	78	0.807	5	93.6%	3	0	1	0	1	6
globin	681	0.974	15	97.8%	5	3	3	2	48	136
gluts	144	0.849	14	90.3%	0	3	1	2	23	59
gpdh	117	0.977	3	97.4%	3	0	3	0	0	4
GTP_EFTU	184	0.802	15	91.8%	0	3	1	2	1	5
heme_1	55	0.250	4	92.7%	0	6	17	0	0	4
hemopexin	31	0.458	3	90.3%	0	1	0	0	0	2
hexapep	45	0.184	8	82.2%	1	2	10	1	5	16
histone	178	0.887	6	96.6%	0	0	0	1	52	104
HLH	133	0.194	7	94.7%	1	8	14	2	27	70
homeobox	383	0.333	27	93.0%	13	2	16	3	9	57
hormone	111	0.961	4	96.4%	0	2	0	2	2	4
hormone2	73	0.613	2	97.3%	0	0	0	0	4	18
hormone3	53	0.760	5	90.6%	0	0	0	0	2	5
hormone_rec	127	0.313	7	94.5%	0	1	1	2	3	5
HSP20	129	0.625	7	94.6%	1	0	1	0	10	34
HSP70	163	0.906	7	95.7%	24	0	9	0	3	6
HTH_1	101	0.476	16	84.2%	0	3	1	2	11	34
HTH_2	63	0.348	9	85.7%	0	1	7	1	3	18
ig	884	0.414	51	94.2%	12	-	-	35	248	553
il8	67	0.662	4	94.0%	0	0	0	0	2	18
ins	132	0.715	3	97.7%	0	0	0	0	7	20
interferon	47	0.987	2	95.7%	0	0	0	0	0	0
kazal	110	0.735	6	94.5%	1	1	1	2	3	13
ketoacyl-synt	38	0.741	7	81.6%	0	0	0	0	0	2
KH-domain	36	0.195	4	88.9%	4	6	23	13	21	28
kringle	38	0.298	2	94.7%	0	1	2	1	0	8
Kunitz_BPTI	55	0.665	5	90.9%	0	2	38	0	0	4
laminin_EGF	16	0.215	3	81.2%	1	0	-	-	2	5
laminin_G	19	0.248	2	89.5%	0	1	-	1	3	11
ldh	90	0.910	6	93.3%	16	1	14	4	11	27
ldl_recept_a	31	0.150	5	83.9%	0	2	6	2	5	16
ldl_recept_b	14	0.209	1	92.9%	0	1	1	0	1	1
lectin_c	106	0.478	14	86.8%	1	1	5	1	3	44
lectin_legA	43	0.356	3	93.0%	0	1	5	0	0	0
lectin_legB	38	0.749	7	81.6%	6	5	5	10	2	5
lig_chan	29	0.836	1	96.6%	2	0	0	0	0	0
lipase	23	0.779	3	87.0%	6	0	0	0	0	0
lipocalin	115	0.858	7	93.9%	4	1	4	0	50	74
lys	72	0.907	1	98.6%	5	0	3	0	1	3
MCPsignal	24	0.107	4	83.3%	0	0	0	0	0	0
metalthio	56	0.963	0	100.0%	5	0	0	0	2	4
MHC_I	151	0.494	3	98.0%	1	0	0	0	0	0
mito_carr	61	0.874	7	88.5%	1	0	0	0	0	1
myosin_head	44	0.439	10	77.3%	1	4	-	0	0	6
NADHdh	57	0.933	4	93.0%	5	0	2	0	0	0
neur	55	0.799	2	96.4%	3	0	2	0	0	2
neur_chan	138	0.882	4	97.1%	5	1	2	2	1	4
oxidored_fad	101	0.244	12	88.1%	0	3	19	1	15	49
oxidored_molyb	35	0.487	1	97.1%	2	0	0	0	0	1
oxidored_nitro	75	0.800	8	89.3%	20	3	18	3	5	33

Table 3.1: **PST performance for all Pfam families (part 2)**

Family	Size	Coverage	Sequences PST missed	% True Pos. PST detected	No. of sequences missed by					
					HMM Pfam	HMM local	HMM global	HMM SAM	BLAST best	BLAST average
p450	204	0.917	17	91.7%	0	2	1	2	2	7
peroxidase	55	0.745	7	87.3%	6	0	3	0	12	32
PGK	51	0.984	3	94.1%	1	0	1	0	0	0
PH	75	0.150	5	93.3%	1	4	17	15	43	67
phoslip	122	0.938	3	97.5%	8	0	5	0	0	3
photoRC	73	0.888	1	98.6%	2	0	1	0	0	1
pilin	56	0.700	6	89.3%	10	2	3	2	0	3
pkinase	725	0.523	108	85.1%	23	-	-	1	6	52
pou	47	0.234	2	95.7%	1	0	0	0	0	0
Pribosyltran	45	0.831	5	88.9%	1	0	2	3	18	23
pro_isomerase	50	0.780	3	94.0%	1	0	0	0	0	1
pyr_redox	43	0.938	7	83.7%	0	0	0	0	0	0
ras	213	0.930	8	96.2%	1	2	1	2	1	3
recA	72	0.928	3	95.8%	7	0	4	0	0	0
response_reg	128	0.424	19	85.2%	1	25	20	2	5	27
rhv	40	0.431	2	95.0%	0	2	-	1	0	5
RIP	37	0.716	2	94.6%	8	0	7	0	4	15
rnaseA	71	0.926	1	98.6%	1	0	1	0	0	1
rnaseH	87	0.186	12	86.2%	0	7	7	7	8	13
rrm	141	0.353	22	84.4%	2	-	-	11	21	79
RuBisCO_large	311	0.995	4	98.7%	20	0	19	0	0	0
RuBisCO_small	99	0.721	3	97.0%	0	0	0	0	0	0
rvp	82	0.156	12	85.4%	2	14	-	13	12	26
rvt	147	0.237	17	88.4%	0	10	12	10	24	56
S12	60	0.958	2	96.7%	3	0	3	0	0	1
S4	54	0.952	4	92.6%	0	2	1	2	1	3
serpin	98	0.908	9	90.8%	8	1	2	1	0	3
SH2	128	0.157	5	96.1%	3	23	18	29	16	50
SH3	137	0.126	16	88.3%	0	43	36	57	18	72
sigma54	56	0.663	9	83.9%	6	3	4	0	2	6
sigma70	61	0.679	5	91.8%	0	0	0	0	0	2
sodcu	66	0.912	5	92.4%	7	1	4	1	3	6
sodfe	69	0.931	5	92.8%	7	0	6	0	2	5
STphosphatase	86	0.810	5	94.2%	2	0	0	0	0	0
subtilase	82	0.563	9	89.0%	0	1	1	1	0	13
sugar_tr	107	0.880	15	86.0%	2	6	5	6	14	33
sushi	75	0.454	8	89.3%	0	1	21	1	3	31
TGF-beta	79	0.296	6	92.4%	3	1	4	1	1	4
thiolase	25	0.965	3	88.0%	0	0	0	0	0	0
thiored	76	0.723	11	85.5%	4	1	3	0	2	6
thyroglobulin_1	32	0.161	3	90.6%	0	2	27	3	0	7
TIM	40	0.973	3	92.5%	1	0	0	0	0	0
TNFR_c6	29	0.353	4	86.2%	0	0	3	2	2	7
toxin	172	0.936	4	97.7%	2	1	2	1	2	9
trefoil	20	0.361	3	85.0%	0	3	11	0	1	8
tRNA-synt_1	35	0.743	7	80.0%	0	0	0	0	1	2
tRNA-synt_2	29	0.702	5	82.8%	0	0	0	0	0	2
trypsin	246	0.730	22	91.1%	0	1	0	1	0	4
tsp_1	51	0.152	6	88.2%	0	3	4	1	2	30
tubulin	196	0.943	1	99.5%	13	0	5	0	1	2
UPAR_LY6	14	0.908	2	85.7%	0	0	0	0	2	5
vwa	29	0.277	6	79.3%	0	2	5	4	10	20
vwc	23	0.090	6	73.9%	0	8	-	2	8	15
wap	13	0.566	2	84.6%	0	0	0	0	1	3
wnt	102	0.936	6	94.1%	66	0	0	0	0	0
Y_phosphatase	92	0.577	8	91.3%	34	0	3	1	4	19
zf-C2H2	297	0.362	23	92.3%	4	13	22	11	15	56
zf-C3HC4	69	0.093	10	85.5%	2	3	28	2	18	48
zf-C4	139	0.152	6	95.7%	1	1	3	1	0	1
zf-CCHC	105	0.072	12	88.6%	1	11	7	12	7	40
zn-protease	148	0.029	21	85.8%	4	110	111	13	92	125
Zn_clus	54	0.061	10	81.5%	0	2	8	1	3	24
zona_pellucida	26	0.484	3	88.5%	0	0	0	0	0	7

Table 3.1: **PST performance for all Pfam families (part 3)**

manually calibrated alignments and are in principle verified to recognize all family sequences (missed sequences may be manually inserted into the seed subset). As a third reference test we ran Gapped-BLAST (Altschul et al., 1997) with each of the member sequences in these families as a query. The performance of this pairwise search algorithm is given both in terms of the best query, and the average query.

Evaluation of results

Overall, the manually calibrated Pfam HMMs detected 96.1% of the true positives (averaged over 170 families with more than 10 members). This is only slightly better than the average performance of HMMs that were built from a multiple alignment of the input sequences, in a local/local mode of comparison (96.0% over 166 families). When the HMMs were built from the same multiple alignments, but in a local/global search mode, the performance dropped to 91.5% (averaged over 158 families). The HMMs that were built directly from the unaligned sequences using the SAM package performed surprisingly well, with 96.7% success over 169 families. This is slightly more discriminative in comparison with the manually calibrated Pfam HMMs. (This may be explained by the fact that Pfam HMMs are based on seed alignments of a small sample set of the family, while the SAM HMMs are trained on 4/5 of the family members.) A marginal improvement (0.1%) was observed with the SAM package when the rest of the sequences were used as a test set by the program 'buildmodel' (not shown). Gapped-BLAST searches performed quite well (92.5% over 170 families), when the "best" query was chosen for each family. However, a typical BLAST search (average performance for all member proteins) performed much worse (78.5% over 170 families).

According to our assessment, already in its very simplistic form and with a preliminary set of parameters, the PST model has detected 90.7% of the true positives in the reference set. This is much better than a typical BLAST search, and almost as good as an HMM that was trained from a multiple alignment of the input sequences in a global search mode. This result is surprising in view of the fact that the model was built in a fully automated manner, without any human intervention or biological consideration, and without utilizing any prior knowledge, such as multiple alignments or scoring matrices.

When the **build-bio-pst** procedure was applied to all Pfam families, with substitution probabilities corresponding to the Blosum62 scoring matrix (Henikoff and Henikoff, 1992), the overall performance was improved and 91.7% of the true positives were detected (not shown). For many families, this procedure resulted in much smaller trees that performed same or better than the trees of the basic procedure (for example, the detection rate for the peroxidase family improved from 87.3% with 10,226 nodes using the **build-pst** procedure, to 96.4% with 5,579 nodes, using the **build-bio-pst**). This is not true for all families, probably since our pruning criteria still need to be refined. In some families we observe minor fluctuations and one or two member proteins were missed by the new model. For small families each protein missed equals to few percentages less in performance, and consequently, the overall performance is affected. However, overall the new model performed better and detected 178 more sequences than the original model.

Critique of the evaluation methodology

Comparing the PST model with HMM based on Pfam families is not a totally objective test, since the groups are defined based on an HMM. An optimal evaluation test would be based on an independent "true" classification of proteins into families. However, no such classification clearly exists. Another reservation is that false positives may be over-counted. It often happens that supposedly false positives with respect to the reference database are actually true positives (Yona

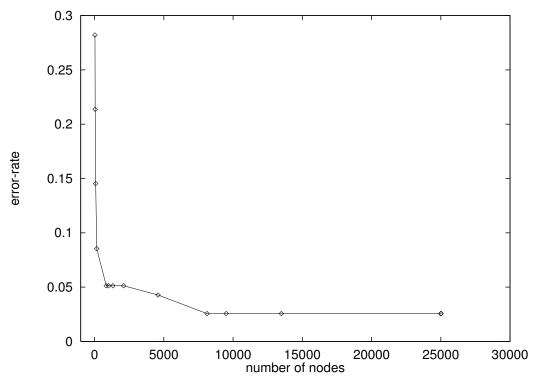


Figure 3.4: **Improving prediction by increasing the number of nodes.** A PST was built for the glyceraldehyde 3-phosphate dehydrogenases family, for different values of P_{min}, and the quality of the PST was estimated by applying the equivalence number criterion (see text). The graph plots the error rate (the number of family members which were not identified, as their score was below the threshold set by the iso-point) vs. the number of nodes in the PST (which increases as P_{min} decreases). Note that the error rate decreases as the number of nodes increases. At some value of P_{min} the quality does not improve much, while the tree keeps growing. If the results are satisfactory, then it is suggested to stop the incremental process at this point.

et al., 2000; Letunic et al., 2002). For example, among the first 250 hits reported by the PST of the pkinase family four are supposedly false positives. However, these four are short proteins (21–42 amino acids long) documented as various types of kinases and their sequence similarity to other kinases is very significant. This problem is inherent in any evaluation procedure which assesses a new classification by means of another, man-made, classification. However, as no clear baseline distribution exists and in the absence of other reliable evaluation procedures, this so called "external validation test" is commonly used (Yona et al., 2000; Gracy and Argos, 1998b; Pearson, 1995). Our specific choice of the Pfam database was motivated by the high quality of this database.

Factors affecting PST performance

It should be noted that Table 3.1 only demonstrates the potential of the PST model, but must not be taken as an upper bound on its performance. To obtain these results we simply ran the PST learning procedure with a fixed set of parameters for all families which we found to result in good performance in a reasonable run time. However, the performance of a PST can be improved by simply tuning the values of the parameters, either globally or per each family. One can either decide to examine more nodes (lower P_{min}), or lower the criteria of acceptance of a candidate (lower α or lower r) or even deepen the tree (increase L). This can be done in an efficient, incremental manner (see Section 3.2.2).

The effect of parameter tuning on the performance is demonstrated in Figure 3.4 for the glyceraldehyde 3-phosphate dehydrogenases family. In general, adding more nodes would tend to increase sensitivity **without** decreasing specificity, simply because more (longer) subsequences that are observed in the training set are "recorded" in the PST. This means that only leaves are further expanded, while the probability distributions of internal nodes are not affected. This way, the predictions over the test set are, hopefully, refined. However, since long subsequences observed in the training set are not expected to occur in unrelated sequences, the prediction of unrelated sequences is based on short subsequences corresponding to internal nodes close to the root, and therefore it is not expected to change.

The only limitation one should keep in mind is that the size of the tree (the number of nodes), as well as the run time, may increase significantly as the parameters are refined, while the improvement in the quality may be insignificant. However, if computation time is of no concern then the PST can run as long as the sensitivity improves (i.e., more family members are detected above the iso-point). In two cases no further improvement is expected: when all sequences of the family are detected, and when all strings in the training set are exhaustively recorded, and predicted with very high probabilities.

An additional improvement is expected if a larger sample set is used to train the PST. Currently, the PST is built from the training set alone. Obviously, training the PST on all strings of a family should improve its prediction as well. Such, for example, is the case for the Pfam HMMs, where all detected members are aligned against the seed HMM to obtain a much larger model.

Screening short sequences and score normalization

The performance of the PST model is even better if very short protein sequences are ignored. Since the "score" assigned to each string by the PST model is normalized by the string's length, very short sequences (i.e., shorter than 20 amino acids) may be assigned relatively high probabilities. These sequences are too short to "activate" the long memory prediction (i.e., the variable memory does not take effect), and hence the prediction is based on nodes closer to the root of the tree. Consequently, these strings may get high probabilities simply by chance and are harder to distinguish from random strings. By discarding all sequences in the Swissprot database which are shorter than 20 amino acids (754 sequences), better prediction results are obtained. Specifically, the performance of the PST model improves by 2.3% (272 sequences) to detect 93% of the true positives in the reference set (not shown). As only 5 out of 15,604 sequences in the Pfam database release 1.0 are shorter than 20 amino acids, the HMMs are hardly affected by discarding those sequences.

While discarding these sequences can be justified because very short peptides are usually biologically meaningless, a different scoring scheme can neutralize to some extent the effect of these sequences. For example, a log odds score normalizes the raw probability of a string s by the chance probability to observe that string (Altschul, 1991). Formally speaking, let $P^T(s)$ be the probability assigned by the PST and let $P_0(s)$ be the chance probability defined as $P_0(s) = \prod_{i=1..|s|} P_0(s_i)$, where $P_0(s_i)$ are defined by the background probabilities in the Swissprot database. The log odds score is defined as

$$\log \frac{P^T(s)}{P_0(s)}$$

This ratio compares the probability of an event under two alternative hypotheses. Thus, the score for each amino acid along the sequence is defined as the logarithm of the amino acid's PST probability divided by its probability of occurrence under independent selection.

The evaluation procedure was repeated with the normalized log odds scores, this time without screening the short sequences. Surprisingly, with these new scores the success rate improved only by 0.2% to 90.9%, suggesting that more sophisticated normalization schemes should be applied.

Performance for protein families vs. domain families and local predictions

As described in Section 3.2.3, the probability the PST model assigns to a protein sequence accounts for all symbols in the sequence. Specifically, it is the product of symbol probabilities along the **whole sequence**. Therefore, it may happen that family members which are similar to other family members along a relatively small fraction of their sequence will be assigned a low overall probability. Consequently, families in which the common pattern is only a small part of the sequence (i.e., domain families) are expected to perform worse than families that are conserved along all or most of the sequence. We tested the effect of the size of the domain on the performance. We define the **coverage** (ranging between 0 and 1) as the total portion of the family sequences which is included in the multiple alignment used to define the domain or the family in the Pfam database. We expect better detection with the PST model for families with high coverage. Indeed, the

Family	L_{domain}	N_{local}	P_{local}	N_{gap}	% True	Positives
					Global prediction	Local prediction
myosin_head	20	0.5	0.1	3	77.3%	86.4%
vwa	10	0.7	0.15	3	79.3%	86.2%
cytochrome_b_C	30	0.7	0.2	3	79.2%	83.1%
vwc	10	0.9	0.15	3	73.9%	82.6%

Table 3.2: **Improvement of PST performance in local prediction mode.** This variant was tested on five families on which the PST model performed worst in global-prediction mode (see text). Four out of the five families were better detected with the model that was tuned to local prediction mode.

performance of the PST model improves once only families with high coverage are considered (see Table 3.1). It detects 92% of the true positives in 108 families with more than 0.5 coverage (while the Pfam HMMs detect 94.9%). Surprisingly, the PST detects more true positives (94.2%) than the HMM (93.9%) for all 38 families with at least 0.9 coverage. This last feature of the PST model can lead to a better performance for proteins with several repeats of the same domain. Two such cases are the EF-hand and the EGF families, that were best detected with the PST model. The proteins in both these families are known for having several copies of these domains.

In view of the discussion above, the PST model is a prediction tool with a "global" flavor. However, in many cases the similarity of two sequences is limited to a specific motif or domain, the detection of which may yield valuable structural and functional insights, while outside of this motif/domain the sequences may be essentially unrelated. Moreover, many proteins contain several different domains. In such cases "global" prediction may not be the appropriate scheme. Local similarities may be masked by long unrelated regions. Consequently, the probability assigned to such sequences can be nearly as low as that assigned to totally unrelated sequences.

To accommodate for the multi-domain trait of proteins we have tested a variant of our prediction step. We calculate the probability only for those regions which are at least L_{domain} amino acids long, and in which at least N_{local} percent of the amino acids have probability above P_{local}. Each such subsequence is considered a domain or a motif. A continuous subsequence of at least N_{gap} amino acids with probability below P_{local} each, marks the end of a domain. We tested this variant on five families on which we performed worst (vwc, myosin_head, fibrinogen_C, cytochrome_b_C and vwa, with performance ranging between 73.9% and 79.3% using the original prediction procedure). Clearly, these parameters need to be optimized for each family separately. For each family we evaluated the performance with several sets of parameters, and picked the best one. For four families the performance improved using the new prediction procedure (see Table 3.2).

3.3.2 Biological Implications

To demonstrate the general performance of the PST model, we examine the scores assigned by a PST model to the entire Swissprot database, which contains the training sequences from which the PST was trained, the test set of family members the PST should detect, and many more sequences that do not belong to the family. Figure 3.5 holds the scores given by two PSTs, one trained on the Neurotransmitter-gated ion-channels and another on the MHC class I family. The cumulative log odds (as predicted by the PST) of the training set sequences, the test set sequences and all the other sequences in the database are plotted vs. the sequences lengths. Note that the unrelated sequences show a clear linear relation in log scale. The training set and the test set samples are located far below this line, hence are well distinguished from the unrelated (effectively random) sequences. Taking a more permissive threshold for P_{min}, resulted in an improved model with better predictions

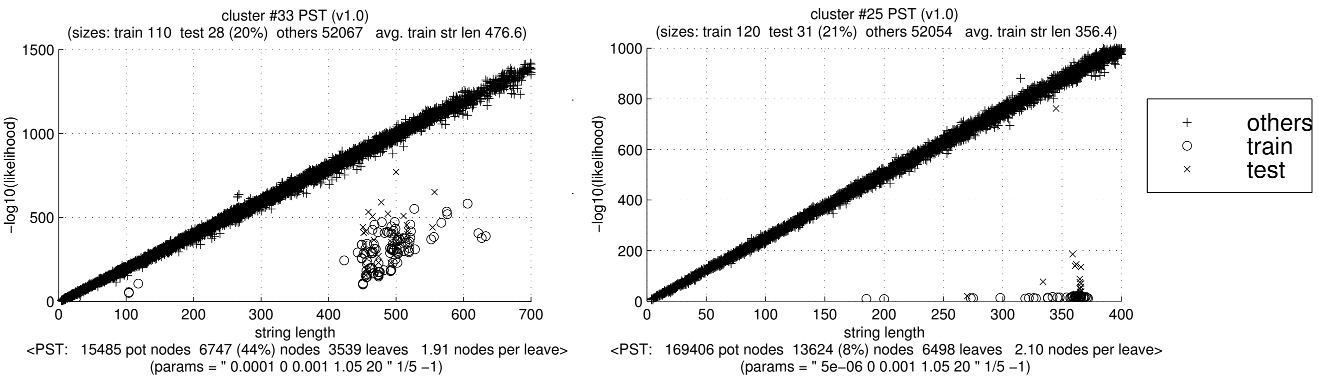


Figure 3.5: **Left: Performance of a PST of the Neurotransmitter-gated ion-channels ($P_{min} = 0.0001$). Right: Performance of a PST of the MHC class I family ($P_{min} = 0.000005$).** The graphs plot the log likelihood of each database sequence (including the training set and the test set) as a function of the sequence length. Likelihood is the product of symbol probabilities along the whole sequence. The PST of the MHC class I family, being trained with a lower value of P_{min}, is an example of an extreme fit of the PST to the training set. Note that the prediction of the test set improves as well, while the unrelated sequences are left unaffected. When the PST of the MHC class I family was trained with the same set of parameters as the PST on the left, its performance graph resembled the graph on the left.

(Figure 3.5, right), i.e., better separation between family members and unrelated sequences.

Family conservation

One possible implication of the PST model is that it can be used to quantify the degree of sequence conservation within protein families. Let us define $-\log p_i$, minus the log likelihood of a given predicted symbol, as its associated level of **surprise**. The bigger the probability of observing that symbol in that context, the smaller the related surprise, and vice versa. The average surprise per symbol for a sequence predicted by a PST model quantifies the relative departure of that sequence from the predictions of the model. This quantity can also be interpreted in terms of coding length, in bits, of the sequence by the model, but we shall differ the introduction of this interpretation to Chapter 5. Several related quantities are of interest here: the average surprise of unrelated sequences, the average surprise of related sequences vs. model size, and the average depth of the PST nodes used for prediction, in both cases.

The slope of the line in Figure 3.5 is a measure of the average surprise associated with unrelated (essentially random) sequences using a specific PST. For example, 6.64 bits per letter are recorded when using the PST of the Neurotransmitter-gated ion-channels family, while 8.3 bits are recorded when using the PST of the MHC class I family. For comparison, 4.18 bits are the average surprise associated with random sequences, when using the background distribution derived from the Swissprot 33 database. The slope is correlated with the **uniqueness** of the source distribution, and particularly, it is higher for families for which the amino acid distribution differs markedly from the overall amino acid distribution in the database.

On the contrary, the training set and test set entail far lower surprise levels per letter. This reflects the modeling performance of the PST model and can also serve as a measure of **divergence** within a protein family. Small average surprise reflects strong conservation within the protein family, while larger surprise reflects high divergence. The average surprise for MHC class I sequences, as predicted by the PST of that family, is but 0.68 bits per letter while the average surprise of the Neurotransmitter-gated ion-channels is 2.21 bits per letter (using the PST of that family with the same set of parameters), suggesting that the former is more conserved. The divergence may be related to structural diversity, suggesting that the transmembrane proteins of the Neurotransmitter-gated ion-channels may adopt a larger variety of shapes than the MHC class I family.

Prediction of significant patterns

The obvious use of the PST model is its application in prediction of family membership. A good model will discern family members from unrelated sequences. Our evaluation procedure shows that the PST model is indeed successful in this respect. The actual prediction process, letter by letter (as shown for the Snake toxins family in Figure 3.6) can further help in assessing the relevance of high scoring hits (and possibly screen out false positives).

The PST can also be used to predict which segments of a given query sequence are suspected to be functionally or structurally important and suggest domain boundaries. These segments correspond to regions of high probability. This is demonstrated in Figure 3.7 for a protein from the Zinc finger, C4 type family. The Pfam zf-C4 domain starts at position 79 and ends at position 154 (according to Swissprot annotation, this region contains two fingers in positions 81-101 and in positions 117-141). The PST assigns high probabilities to residues in this region, though it starts a bit closer to the N-terminus (at position 59). Note that another domain is predicted with high probability (between positions 190 and 413). This region corresponds to the ligand-binding domain of the nuclear hormone receptors (which according to Swissprot annotation starts at position 193 and ends at position 412). Out of the 139 proteins in the zf-C4 family, 127 also belong to the hormone receptors family. Thus, we see that both domains were recorded in the PST model during the learning phase. This explains why the PST predicted both domains with high probability. Note that along the two regions not all letters are essentially of high probability, which may point to a substitution or a gap in the query sequence, and different evolutionary pressures.

Recall that the PST does not require the input sequences to be aligned nor does it make use of such information during learning. The input sequences were not fragmented according to domain boundaries before the learning phase, and therefore this information was solely self attained. This will serve us as a motivating example when in later chapters we attempt **protein sequence segmentation**. The PST model can thus also help to guide a multiple alignment of a set of related sequences, by suggesting an initial seed, which is based on the regions of high probability.

In Figure 3.8 the PST model of the EGF family was used to predict EGF motifs within the protein sw:fbp3_strpu (570 residues long). This protein has 8 EGF repeats (the starting positions are marked by arrows). Note that all these domains, except for the first, correspond to regions of high probability as predicted by the PST model. Indeed, the Prosite database (Sigrist et al., 2002) regular expression (Section 2.2.1) defining the EGF motif `C-x-C-x(5)-G-x(2)-C` allows a lot of variability between the conserved cysteines and glycine. The probabilities in Figure 3.8 are derived from the combined two-way prediction (see Section 3.2.3), for better prediction of domain boundaries. The prediction was smoothed using a sliding window of length 20 by averaging over the probabilities of symbols within each window.

Protein prediction mechanism

We turn to examine how the PST predictions map to the protein sequences themselves. We illustrate our arguments using the paired box domain family. The complete sequences of the 139 proteins in which the paired box domain is found were randomly split, in ratio 4:1 as before, into train and test sets. A model was learned from the training set using the parameter set of Table 3.1. Figure 3.9(a)

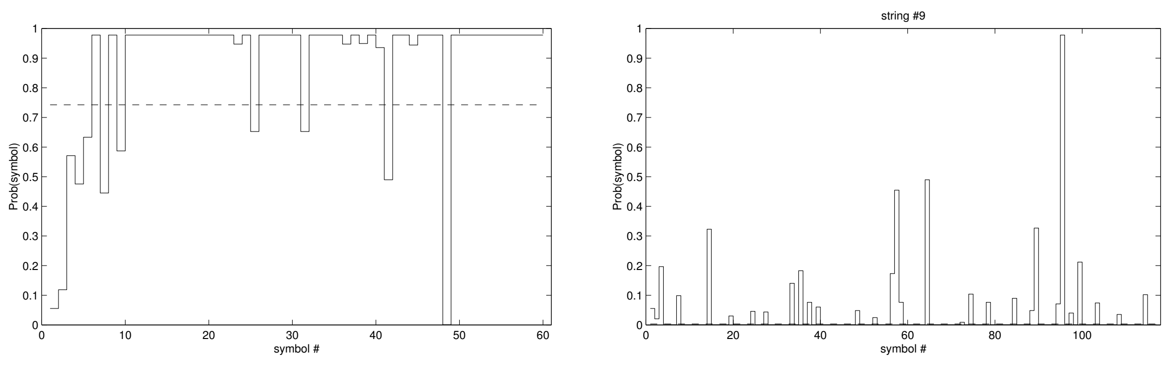


Figure 3.6: **Prediction using PSTs**. The PST of the Snake toxins family was used to calculate the probability, letter by letter, of a protein sequence (sw:P01445) from the Snake toxins family test set (left), and of a protein sequence (sw:P09837) from the WAP-type (Whey Acidic Protein) family. The average probability is 0.75 and 0.001 respectively.

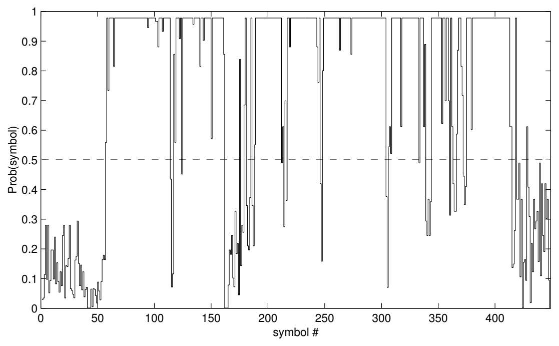


Figure 3.7: **Identifying significant patterns using PSTs**. The PST of the Zinc finger, C4 type family was used to predict significant patterns in protein sw:P10826. The protein belongs to the test set and its average probability per symbol is 0.5. More importantly, two different regions of conservation, matching the two functional domains of the protein are also detected (see text).

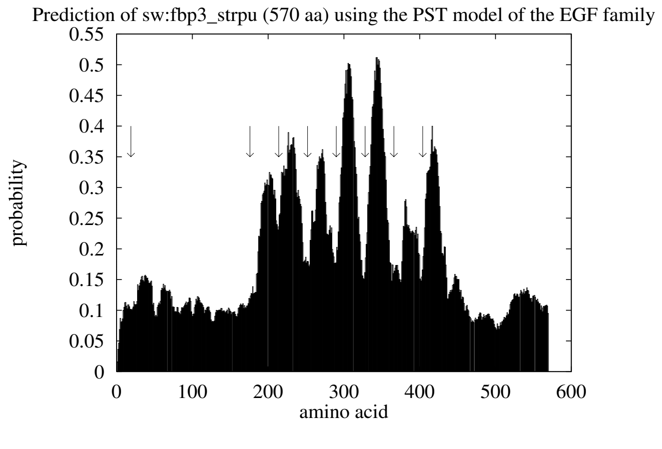


Figure 3.8: **EGF domains Prediction**. We combine the prediction of two EGF models for the sequence sw:fbp3_strpu into a single average score. One PST was trained to predict from the 5' terminus to the 3' terminus of the sequences, as before, while the other predicts from the 3' terminus to the 5' terminus. The actual starting positions of the EGF domains (according to Swissprot) are marked by arrows.

shows the prediction, symbol by symbol of a test set family member. The paired box domain is located between residues 4 – 128 in this protein, and is seen to correlate very nicely with the area of high prediction. In contrast, we draw in Figure 3.9(b) the predictions of simple Markov models, of orders 1,2 when trained from the same data. We see that the first model does not capture the domain at all, while the other already captures parts of the signal, but not its entirety. A full third order model requires much more data than the available 111 sequences used for training.

Returning to the PST model predictions, Figure 3.9(c) plots the depth of the node (or length of the suffix) from which prediction was made, per each symbol. We can see that prediction depth also correlates well with the existence of the paired box domain in all training sequences. However, the signal itself is much less contiguous than for the actual predictions. To see the source of these deep contexts we schematically draw in Figure 3.9(d) the multiple alignment of the training set sequences, against the test set sequence. Whenever an amino acid in an aligned training set sequence is identical to the corresponding amino acid in the test set sequence, it is shaded black. Otherwise, it is shaded gray. The figure zooms in on the paired box domain and flanking regions, and clearly shows the sources of the long contexts, as well as the relatively unique region preceding residue 100. Another aspect of the comparison to the MSA is shown in Figure 3.9(e), where we plot the conservation score assigned by ClustalW (Higgins et al., 1996) to each column corresponding to a residue of our query sequence.

Finally, we note that the paired box domain has been relatively well conserved during evolution. Less conserved families pose a greater challenge for the PST algorithm. For example, we plot in Figure 3.9(f) the Pfam database curated MSA of the Aldose 1-epimerase family, against an arbitrary member of the family. Comparison to Figure 3.9(d) shows that while the general structure of the alignment is similar, the exact short patterns are much rarer and shorter, reflecting on the resulting PST model quality.

Left-right causality in protein sequences?

It is interesting to see if there is a sense of directionality in protein sequences. I.e., whether protein sequences are "generated" by a source which has a clear directionality. Obviously proteins are assembled from the N-terminus to the C-terminus. They are built at the ribosome, being translated from an RNA molecule, one amino acid at a time, as the translation process propagates from the N terminus to the C terminus. While it has been speculated by some that this process affects the folding process of the amino acids chain (Fedorov and Baldwin, 1995; Kolb et al., 1995), there has been no rigorous proof whether or not left-right causality is encoded in the corresponding gene. That is, a causality that dictates the next amino acid given that we have observed a certain sequence of amino acids. Such causality may follow some physical-chemical rules and constraints that govern the processes of creating secondary structure elements in proteins, and in general, the whole folding process.

If left-right causality exists in protein sequences then one might expect differences in prediction when we propagate from left to right along the sequence compared with when we propagate along the other direction. To test this hypothesis, we trained a PST on the reversed sequences of each Pfam family. These PSTs were used to predict the sequences in the Swissprot database (after being reversed) in the same way described in Section 3.2.3 and the performance was evaluated using the equivalence number criterion. Surprisingly, perhaps, our evaluation shows that there is no difference in performance between left-right prediction and right-left prediction. Both perform almost exactly the same with some minor fluctuations. Such observation is consistent with current knowledge and perceptions of protein evolution, according to which, the genetic pressure is mainly

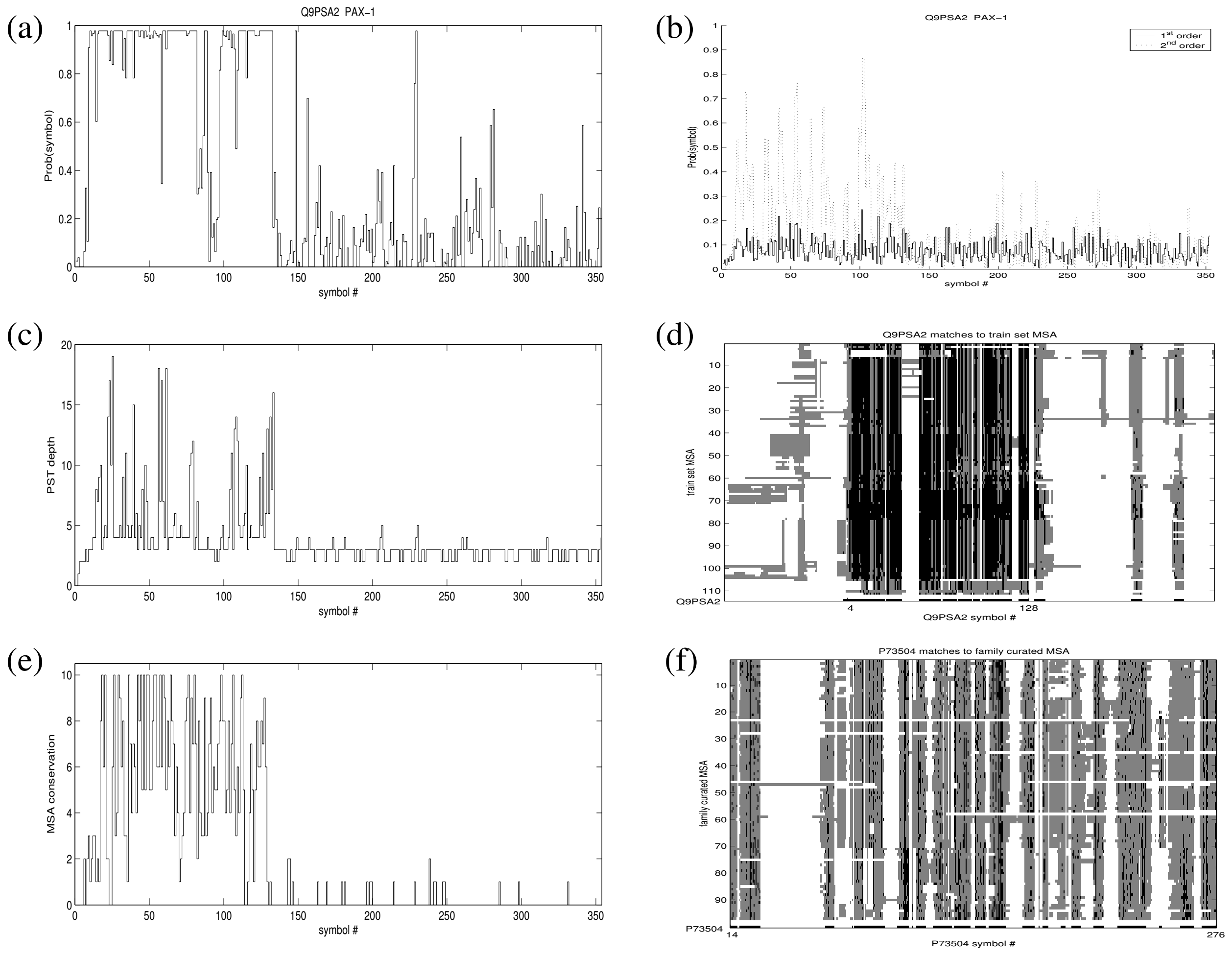


Figure 3.9: **PST prediction analysis.** (a) Prediction of a family member test set sequence (Q9PSA2) against the PST generated from the paired box domain family training set. The family domain is located between residues 4 – 128 in the query sequence. (b) A comparison to first and second order Markov model predictions given the same training set and query sequence. Third order Markov requires much more data than the available sequences. (c) Prediction depth, or suffix length, is drawn per position, as used in the PST prediction above. (d) A schematic depiction of the MSA of the 111 training sequences against the query sequence (bottom line). Training set residues are shaded black if they are identical to the corresponding residue in the query sequence, or gray otherwise. The x-axis is zoomed at a region containing the family domain. This plot shows the origin of the long contexts observed during prediction, in the figure on its left. (e) The ClustalW conservation score of each MSA column to which a query sequence residue had been matched. (f) A similar plot to the one above it, for the Aldose 1-epimerase family. One can easily see that this family has been much less conserved during evolution. Such families, offering less, shorter contexts, are harder for the PST model to capture accurately.

at the protein level, and the constraints on the sequence are determined by spatial considerations, sometimes between residues which are far apart in the sequence. These perceptions are further supported by evidences of correlated mutations in protein sequences (Gouldson et al., 1997; Pazos et al., 1997), stability under circular permutations (Feng et al., 1999; Hennecke et al., 1999; Otzen and Fersht, 1998; Tsuji et al., 1999), and experimental studies on the initiation of the protein folding process (discussed in Rabow and Scheraga, 1993).

3.4 Summary

This chapter presented a novel approach for modeling a group of related proteins without incorporating any prior information about the input sequences. The method adapts probabilistic suffix trees (Rissanen, 1983; Ron et al., 1996) to capture short term statistical dependencies in the input family by recording significant occurrences of subsequences of variable lengths. The method induces probability distributions over symbols from the empirically observed distributions. These are extended to score and classify whole sequences. The balance between possibly several recognizable motifs and unknown regions within a query sequence determines its degree of similarity to the given model.

Our modeling assumptions are in fact motivated by biological insights into the evolution and composition of protein families. A group of evolutionary related protein sequences should exhibit many identical short range statistical patterns of amino acids. This follows from the underlying evolutionary process. All protein family members descend from a single ancestral sequence. The different copies of this sequence underwent mutation events. But viable mutations are almost always pinpointed at a single or a few residues. As a result, homolog sequences will have identical short segments which have been either preserved by selection, or simply have not diverged long enough. Fixation of mutations must respect the functional role of the gene, and its encoded protein. As a result, while mutation would work separately in the different lineages, all would in general be subject to very similar constraints. Thus, the mutations themselves would be biased, again contributing to short range similarities. Moreover, by re-examining Figure 2.3 (page 14), we note that the majority of the longer conserved segments are unique to their particular location along family sequences. Thus, when we collect their counts from seemingly anywhere within the seed sequences we often approximate position specific suffixes, without alignment.

The variable memory model with its flexible, data dependent, architecture is well equipped to pick up these locally conserved short segments, of various lengths, and in various subsets. Indeed, when the model was applied to protein families in the Pfam database, it was shown to identify the other members of protein families with surprising success, compared to the state of the art in family modeling. According to our evaluation procedure, already in its basic form the PST model outperforms classic pairwise comparison methods such as Gapped-BLAST, and performs as well as an HMM that is built in a global search mode. These results were obtained when the PST was trained using a fixed set of parameters for all families. The performance of the PST model is expected to improve if the parameters are tuned for each family separately, and when all known sequences of a family are included in the training set, as is the case in established databases of protein family models (Chapter 2).

Compared to our novel approach, the HMM suffers several disadvantages: For pre-processing it requires delineation of the conserved region, its multiple alignment, and tagging of non-representative columns. Sequence weighting is often required to avoid biasing column emission probabilities towards a sub-set of the family. Theoretical hardness results, such as Abe and Warmuth (1992) and Gillman and Sipser (1994) indicate that they require relatively high quality training sets to perform adequately. They are also expensive to run, using quadratic dynamic programming procedures for both training and prediction.

On the other hand, the flexibility of the PST model does not come without a cost. Our reference set in this study contained mainly well defined protein families. Modeling more diverged sequence families using PSTs is prone to be more difficult as short contiguous patterns of conservation become more scarce. This is indicated in Figure 3.9(f), and has been shown in a recent study by Sohler and Zien (2001). In such cases, good alignments (typically requiring expert crafting for the same reasons) may confer more robustness to an HMM approach by pinpointing single conserved positions (such as active site residues) which can then be used in prediction to anchor the alignment of a query sequence to the model.

The success of the PST modeling approach in this domain suggests several interesting extensions: Algorithmically, we have shown that already in its current formulation the PST learning and prediction phases are much faster than those of HMMs. However, they are still quadratic in the size of the input set. The increasing growth of protein sequence databases motivates an attempt to further improve performance. This challenge is met in the next chapter where we optimize PST performance both in terms of run-time and space requirements. From a statistical point of view, we have demonstrated that PSTs are relatively robust to over-fitting the training set (Figure 3.4). Yet, the set of parameters we have used to train these models, motivated by the formulation of Ron et al. (1996), does not try to answer directly whether a given training set pattern is worthy of memorizing or not. One such guiding principle, termed minimum description length (Barron et al., 1998) will be put to use in Chapter 5. In that chapter we will also develop tools that are motivated by the ability of one PST model to capture several domains simultaneously (Figure 3.7) and attempt to spilt this model into several ones, each specializing in a certain domain in a heterogeneous input set. Another challenge is the attempt to enhance the PST models to better capture the data. Kermorvant and Dupont (2002) use more sophisticated smoothing techniques to compensate for small sample effects on longer suffixes, to obtain smaller yet accurate PST-like models. Eskin et al. (2000) add wildcards into the suffix tree structure, thus allowing in principle better handling of gapped patterns and longer range correlations. However, the sheer magnitude of the combinatorial space of possible patterns spun by the use of these match-all symbols limits their use in practice to a minimal number. One final worthwhile direction calls for analysis of the very features the PST chooses to model a protein family with. One can use these, as Sohler and Zien (2001) did, to select relevant features for a support vector machine driven classification (see Vapnik, 1998). This way features selected by the generative PST learning algorithm are fed to a discriminative learning method. Another discriminative approach (Eskin et al., 2000) classifies any new protein directly to one of all known families, without going through an explicit generative phase. In Chapter 6 we will take a third discriminative direction, which is more biologically motivated, when we try to learn one PST model for several families, simultaneously. Such a model can focus on discriminative features between the given groups while ignoring many features they all share.

This chapter is based on an extended abstract (Bejerano and Yona, 1999) which was a co-winner of the best paper by a young scientist award at the RECOMB 1999 conference, and was later developed to journal format (Bejerano and Yona, 2001; Bejerano, 2003a).

Chapter 4

Algorithmic Optimization

In the previous chapter we have demonstrated the applicability of variable memory models in analysis of protein sequences. The algorithms we used there were quadratic in the length of the input set. In this chapter we completely replace the underlying data structure and resulting algorithms to obtain novel learning and prediction algorithms, which are equivalent to the original ones, yet are optimally linear in terms of both runtime and memory requirements.

4.1 Reminder

In Chapter 3 we have adapted a family of variable memory models, termed Probabilistic Suffix Trees (PST), to the task of classifying protein sequences into known protein families. A PST, first defined by Ron et al. (1996), is a compact, tree-shaped sub-class of probabilistic automata which approximates the input sequences during training using an underlying Markov process of variable memory length not exceeding some maximum depth L. The probability distribution generated by these automata is equivalent to that of a Markov chain of order L, but the description of the automaton itself is much more succinct (recall Figure 3.1).

Novel protein sequences are being discovered, as a result of genomic sequencing efforts, at an ever increasing pace, and the relevant databases are quickly approaching the one million sequence mark. While the above algorithms are already faster than their equivalent HMM procedures, they are quadratic in the input set length. The process of learning the automaton from a given training set S of sequences, as defined in Section 3.2.2, requires $\Theta(Ln^2)$ worst-case time, where n is the total length of the sequences in S and L is the length of a longest substring of S to be considered for a candidate state in the automaton. Once the automaton is built, predicting the likelihood of a query sequence of m characters (Section 3.2.3) may cost time $\Theta(m^2)$ in the worst case.

In this chapter we will present automata equivalent to PSTs but having the properties that, on one hand, learning the automaton takes $O(n)$ time, regardless of L, and on the other, prediction of a string of m symbols by the automaton takes $O(m)$ time. Along the way, we address notions of empirical probability and their efficient computation, possibly a by-product of more general interest. We begin by briefly reviewing the important definitions from Chapter 3.

We deal with a (possibly singleton) collection S of strings over a finite alphabet Σ, and use λ to denote the empty string. The *length* of S is the sum of the lengths of all strings in it and is denoted by n. With reference to S and a generic string $s = s_1 s_2 ... s_l$, the *empirical probability* $\tilde{P}$ of s is defined provisionally as the number of times s occurs in S divided by the maximum "possible" number of such occurrences. The *conditional empirical probability* of observing the symbol σ immediately after the string s is given by the ratio

$$\tilde{P}(\sigma|s) = \frac{\chi_{s\sigma}}{\chi_{s*}} \quad ,$$

where χ_w is the number of occurrences of string w in S and $s*$ is every single-symbol extension of s having an occurrence in S. Finally, $suffix(s)$ denotes $s_2 s_3 ... s_l$.

We briefly recount the structure of a PST (Figure 3.1). In any such tree, each edge is labeled by a symbol, each node corresponds to a unique string – the one obtained by traveling from that node to the root – and nodes are weighted by a probability vector giving the distribution over the next symbol. In the following, $\mathcal{T}$ is the PST, $\bar{S}$ is the set of strings that we want to check, or learn, and γ_s is the probability distribution over the next symbol associated with node s. The construction starts with a tree consisting of only the root node (i.e., the tree associated with λ) and adds paths as follows. For each substring s considered, it is checked whether there is some symbol σ in the alphabet for which the empirical probability of observing it after s is both significant and significantly different from the probability of observing it after $suffix(s)$. Whenever these conditions hold, the path relative to the substring (and possibly its necessary but currently missing ancestors) are added to the tree. As detailed below, the time complexity of this construction is $O(Ln^2)$, where L is the length of a longest string considered for possible inclusion in $\mathcal{T}$.

Given a string, its weighting or prediction by a PST is done by scanning the string one character after the other while assigning a probability to every character, in succession. The probability of a character is calculated by walking down the tree in search for the longest suffix that appears in the tree and ends immediately before that character, the corresponding conditional probability is then used in calculating the product for all characters (Section 3.2.3). Since, following each input symbol, the search for the deepest node must be resumed from the root, this process cannot be carried out on-line nor in linear-time in the length of the tested sequence, the worst-case time being in fact $\Theta(m^2)$ for a sequence of m characters. In Appendix B of Ron et al. (1996) a solution is offered to this issue: a procedure is given to turn the PST into an equivalent not much larger Probabilistic Finite Automaton (*PFA*) on which every prediction step does take constant time (as it translates to a single transition on the PFA). However this procedure may, by itself, cost $\Theta(Ln^2)$ time in the worst case.

4.2 Learning Automata in Linear Time

In Section 3.2.2 two variants of the learning algorithm introduced by Ron et al. (1996) were defined. For clarity of exposition we will focus our attention on the first, presented in Figure 3.2 (page 39). We briefly recall the parameters used there: L is, again, the maximum length for a string to be considered, P_{min} is the minimum value of the empirical probability in order for the string to be considered, r (> 1) measures the multiplicative prediction difference between the candidate and its father per any given character, while α and γ_{min} limit the minimal empirical probability for a particular character to be of interest. The parameter γ_{min} is also used as the smoothing factor at the last stage of the construction.

We are interested primarily in the asymptotic complexity of the main part (tree construction) of the procedure, and in possible ways to improve it. The last steps of smoothing probabilities have no substantial bearing on the performance and no consequence on our considerations. We see that the body of the algorithm consists of checking all substrings having empirical probability at least P_{min} and length at most L. Although the number of substrings passing these tests may be smaller in practice, there are in principle $n - l + 1$ possible different strings for each l, which would lead to $\Theta(Ln^2)$ time just to compute and test empirical probabilities (nL strings in total each requiring at least $\Theta(n)$work to test). The discussion that follows shows that in fact overall $O(n)$ time suffices.

Our approach must depart considerably from the algorithm of the figure. There, word selection and tree construction go hand in hand in Steps B and C. In our case, even though in the end the two can be re-combined, we decouple these tasks. We concentrate on word selection, hence on the tests of Step B. Essentially, we want to show that all those words can be tested in overall linear time, even if those word lengths may add up to more than linear space. For simplicity of exposition we assume henceforth that S consists of only one string, which will be denoted by x.

4.2.1 Computing Conditional Probabilities and Ratios Thereof

The goal of this Subsection is to establish the following

Lemma 4.1. There is an algorithm to perform the collection of all tests under Step B for all substrings of S in overall linear time and space.

Notice that S may contain $\Theta(n^2)$ distinct strings as substrings. Thus, there are two qualifications to the lemma: one is to show that computation can be limited to $O(n)$ words, the other is that this can be achieved in overall linear time and space. We now begin with the proof of the lemma.

Given two words x and y, the *start-set* of y in x is the set of *occurrences* of y in x, i.e., $pos_x(y) = \{i : y = x_i...x_j\}$ for some i and j, $1 \leq i \leq j \leq n$. Two strings y and z are equivalent on x if $pos_x(y) = pos_x(z)$. The equivalence relation instituted in this way is denoted by $\equiv_x$ and partitions the set of all strings over Σ into equivalence classes. We use $C(w)$ to denote the equivalence class of w with respect to x. In the string $x = abaababaabaababaababa$, for instance, $\{ab, aba\}$ forms one such C-class and so does $\{abaa, abaab, abaaba\}$. Recall that the *index* of an equivalence relation is the number of equivalence classes in it. The following important "left-context" property is adapted from Blumer et al. (1985).

Fact 4.2. The index k of the equivalence relation $\equiv_x$ obeys $k \leq 2n$.

Proof: For any two substrings y and w of x, if $pos_x(w) \cap pos_x(y)$ is not empty then y is a prefix of w or vice versa (i.e., ($C(y) \subseteq C(w)$ or vice versa). If x is extended by appending to it a symbol not appearing anywhere else, then the containment relation on subsets of the form pos_x forms a tree with $|x| + 1$ leaves, each corresponding to a different position, and in which each internal node has degree at least 2. Therefore, there are at most $|x|$ internal nodes and $2|x| + 1$ nodes, or equivalence classes, in total. Taking now back the spurious leaf of position $(|x| + 1)$ yields the claim. □

Fact 4.2 suggests that we might restrict computation of empirical probabilities to the $O(n)$ equivalence classes of $\equiv_x$. One incarnation of the tree evoked by the above proof – in fact, an alternate proof of its own – is the *suffix tree* T_x associated with x. We assume familiarity of the reader with the structure and its clever $O(n \log |\Sigma|)$ time and linear space constructions such as in Weiner (1973); McCreight (1976); Ukkonen (1995). The word ending precisely at vertex α of T_x is denoted by $w(\alpha)$. The vertex α is called the *proper locus* of $w(\alpha)$. The *locus* of word w is the unique vertex α of T_x such that w is a prefix of $w(\alpha)$ and $w(\text{FATHER}(\alpha))$ is a proper prefix of w. One key element in the above constructions is in the following easy fact:

Fact 4.3. If $w = av$, $a \in \Sigma$, has a proper locus in T_x, then so does v.

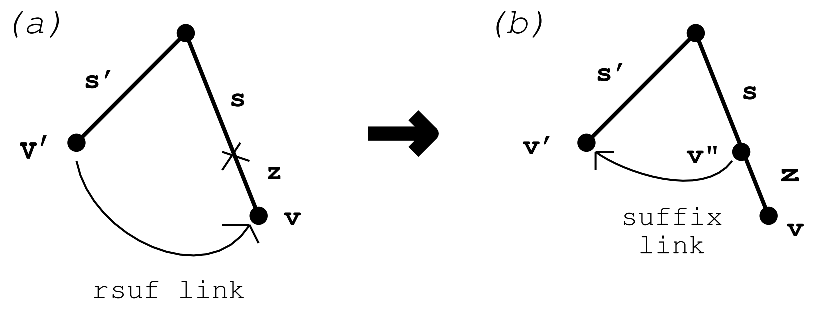


Figure 4.1: **Creating auxiliary suffix links.** See text for details.

To exploit this fact, *suffix links* are maintained in the tree that lead from the locus of each string av to the locus of its suffix v. Here we are interested in Fact 4.3 only for future reference. Having built the tree, some simple additional manipulations make it possible to count and locate the distinct (possibly overlapping) instances of any pattern w in x in $O(|w|)$ steps.

Consider now conditional empirical probabilities, which were defined as the ratio between the observed occurrences of $s\sigma$ to the occurrences of $s*$. The first observation is that the value of this ratio persists along each arc of the T_x, i.e.,

$$\tilde{P}(\sigma|s) = \chi_s/\chi_{s\sigma} = 1$$

for any word s ending in the middle of an arc of T_x and followed there by a symbol σ. Therefore, we know that every such word passes the first test under (B), while continuation of s by any other symbol would have zero probability and thus fail. These words s have then some sort of an obvious implicit vector and need not be tested or considered explicitly. On the other hand, whenever both s and $suffix(s)$ end in the middle of an arc, the ratio

$$\frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|suffix(s))} = \frac{1}{1} = 1 \ .$$

Since $r > 1$, then neither r nor $1/r$ may be equal to 1, so that no such word passes either part of the second test under B. The fate of any such word with respect to inclusion in a PST (whence also in the final version of our tree) would depend thus on that of its shortest extension with a proper locus in it. The cases where both s and $suffix(s)$ have a proper locus in T_x are easy to handle, since there is only $O(n)$ of them and the corresponding tests take linear time overall. By Fact 4.3, it is not possible that s has a proper locus while $suffix(s)$ does not. Therefore, we are left with those cases where a proper locus exists for $suffix(s)$ but not for s. There are still only $O(n)$ such cases of course, but in order to handle them we need first to perform a slight expansion on T_x.

Let ν' be the proper locus of string s'. We define $\texttt{rsuf}(\nu', \rho)$ to be the node ν which is the proper locus of the shortest extension of $\rho s'$ having a proper locus in T_x. In other words, node ν has the property that $w(\nu) = sz$ with $s = \rho s'$ and z as short as possible if $z \neq \lambda$ (see Figure 4.1a). If $\rho s'$ has no occurrence in x then $\texttt{rsuf}(\nu', \rho)$ is not defined.

Clearly, `rsuf` coincides with the reverse of the suffix link whenever the latter is defined. When no such original suffix link is defined while $\rho s'$ occurs in x, then `rsuf` takes us to the locus of the shortest word in the form $\rho s' z$. Since ν' is a branching node, then there are occurrences of s' in x that are not followed by the first character of z. In other words, not all occurrences of s' occur precisely at the second position of an occurrence of $sz = \rho s' z$, whence $pos_x(\rho s') \neq pos_x(\rho s' z)$. In

LOCUS OF $suffix(s)$	LOCUS OF s	ACTION FOR TEST I	ACTION FOR TEST II
(1) proper locus ν'	proper locus ν	read weights of locus ν	use suffix link from ν to ν'
(2) middle of an arc	proper locus ν	impossible	impossible (see Fact 4.3)
(3) proper locus ν'	middle of an arc	single possible extension	use `rsuf` from ν' to aux or surrogate locus of s
(4) middle of an arc	middle of an arc	irrelevant	always fails (see text)

Table 4.1: **Synopsis of tests on conditional probabilities.**

these cases, we know *a priori* that $\tilde{P}(\sigma|s) = 1$ only for σ equal to the first character of z, but the value of $\tilde{P}(\sigma|s')$ and hence also of the ratio $\tilde{P}(\sigma|s)/\tilde{P}(\sigma|s')$ have to be computed and tested explicitly. We can do so by treating ν as a surrogate locus of that of $s = \rho s'$, but it is more convenient for our discussion to add to T_x explicit unary nodes for this purpose. Thus, a special unary node ν'' is created as the proper locus of $\rho s'$, and endowed with a suffix link directed towards ν' (see Figure 4.1b). It should be clear that the total number of such auxiliary nodes in our tree is bounded by $n|\Sigma|$, hence linear for finite alphabets. Table 4.1 summarizes the possible cases and their respective treatments.

Expanding T_x and computing `rsuf`'s is an easy linear post-processing of the tree. We have also seen that attaching empirical conditional probabilities only to the branching nodes of T_x suffices. As there are $O(n)$ such nodes, and the alphabet is finite, the collection of all conditional probability vectors for *all* sub-words of x takes only linear space. Given T_x, the computation of such probabilities is trivially done in linear time. With reference to Table 4.1, the only tests to be taken are of type (1) and (3), and there are $O(n|\Sigma|)$ such tests of each kind, both associated with the nodes of the tree. Specifically, there are $|\Sigma|$ comparisons at the nodes ν and ν' under (1), and $|\Sigma|$ possible extensions of the words s' associated with nodes ν' under (3).

This concludes the proof of Lemma 4.1. □

4.2.2 Building the New Amnesic Automaton

At this point we can already outline an $O(n|\Sigma|)$ procedure by which words are selected for inclusion in our automaton and the automaton itself is built.

1. **Substrate preparation:** Build a compact suffix tree T_x for x. Add auxiliary unary nodes as described.

2. **Word selection:** Determine the words to be included in $\bar{S}$ and thus in the final automaton. For this, visit the nodes of T_x bottom up, compute χ-counts and conditional probabilities, and run the tests of Step B on these nodes. Mark the root and all nodes passing the test. For every node marked, follow the path of suffix links to the root and mark all nodes on this path currently unmarked.

3. **Tree Pruning:** Visit the tree in some bottom up order, and prune the tree cutting all edges immediately below every deepest unmarked nodes.

In practice, the operations above would be more suitably arranged and combined without this affecting their global complexity. Let $\mathcal{H}$ denote the pruned version of T_x resulting from this treatment. As is easily seen, the set of words having proper loci at a marked node of $\mathcal{H}$ contains that of the words associated with the nodes of the PST resulting from the algorithm of Figure 3.2. In particular, the marking of all nodes on the suffix path of a marked node corresponds to admitting into the tree all suffixes of every admitted word.

Fact 4.4. If word w is spelled out on some path from a node to the root of the PST $\mathcal{T}$, then w has a marked proper locus in $\mathcal{H}$.

This fact shows just how the PST $\mathcal{T}$ is embedded in $\mathcal{H}$: to extract $\mathcal{T}$ from $\mathcal{H}$, take the marked nodes of $\mathcal{H}$ and the `rsufs` edges connecting these nodes, and then possibly prune some fringes at the bottom of the tree thus obtained. Any marked node v in $\mathcal{H}$ that does not appear in $\mathcal{T}$ corresponds to a node that the PST algorithm would have inserted had it gotten to it (or to a marked descendant of it). However, due to the top-down nature of the PST algorithm combined with a possibly *non-monotone* notion of $\tilde{P}$, if any node along the `rsuf` path of v fails test C (even though v itself passes it), node v would never be examined by the PST algorithm[1]. One might argue that these nodes should have also been included in $\mathcal{T}$ and hence must stay, or modify the pruning of T_x so that these nodes are excluded from $\mathcal{H}$ as well (this requires one walk on `rsufs`). Yet another alternative is to defer the tests of Step C altogether to the weighting phase, in which they may be performed on the fly, where desired, without this affecting the time complexity of that phase. This issue shall be further discussed in Section 4.5.

Essentially, $\mathcal{H}$ is a compact Trie resembling the basic structure of a Multiple Pattern Matching Machine (MPMM) (see Aho and Corasick, 1975). The import of this is that, on such a machine, substrings undergoing tests are scanned in the forward, rather than reverse, direction while traveling on paths that go from the root to the leaves of the automaton. The full-fledged structure of MPMMs, with failure-function links etc., ensures that while the input string is scanned symbol after symbol, we are always at the node of the MPMM that corresponds to the longest suffix of the input having a node in the MPMM. Also by the structure of MPMMs, running a string through it takes always overall linear time in the length of the string. However, our MPMM is non-standard in that explicit nodes (and associated failure pointers) might be missing along the arcs of T_x. The total number of such nodes might amount to $\Theta(n^2)$ in the worst case. One might consider to add such nodes on the fly during prediction at a cost of constant time per character, and charge the predicted sequence(s) with the corresponding $O(m)$ work. In the next section, we study means of surrogating the missing nodes and links within the $O(n)$ time allocated to learning. We conclude this section by recording the following

Theorem 4.5. The probabilistic automaton $\mathcal{H}$ contains $\mathcal{T}$ and all the information stored in $\mathcal{T}$, and can be learned in linear time and space.

4.3 Implementing Linear Time Predictors

In this section, we assume we are given a pruned tree $\mathcal{H}$ with its nodes suitably weighted and marked, and we tackle the problem of how to use this tree for prediction. We consider two different

[1]Note, for example, that the notion of $\tilde{P}$ we use is non-monotone, i.e. there may be $w \in \Sigma^\star$ and $\sigma \in \Sigma$ s.t. $\tilde{P}(w) < \tilde{P}(\sigma w)$. Consider the case where $\Sigma = \{a, b\}$ and $x = baabaa$. A simple calculation shows that $0.4 = \tilde{P}(aa) < \tilde{P}(baa) = 0.5$. This means that if the threshold in test C is set to 0.45 the node corresponding to baa will not be examined for inclusion in $\mathcal{T}$, because its father node, corresponding to aa, will fail to pass test C. However, As $\mathcal{H}$ prunes bottom-up, it will encounter the node corresponding to baa. This node may very well pass test B, and as a result both it *and* its father will be included in $\mathcal{H}$.

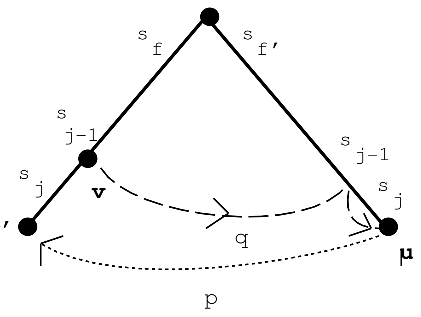


Figure 4.2: **Adding direct auxiliary links.** (a) If ν' does not exist and yet $f > f'$ then the suffix path p leads us to a contradiction. (b) Thus, if ν' does not exist, it must be that $f < f'$. We wish then to hold a link from ν to μ, the end point of path q. (c) Finally, If ν' exists, we wish to hold a link from ν' to μ, the end point of the reverse suffix path p^{-1}. Refer to the text for details.

scenarios for prediction, depending on whether the string s is assumed to be fed to $\mathcal{H}$ one character at a time from left to right or backward, beginning with the last character. We first sketch our treatment of the first case, and then discuss the second one in full detail.

In a left-to-right scanning, we want to maintain that at the generic step where we have read the prefix $s_1s_2...s_{j-1}$ we find ourselves at the marked node ν of $\mathcal{H}$ that is the proper locus for the longest suffix of $s_1s_2...s_{j-1}$ among those suffixes that have a marked proper locus in $\mathcal{H}$.

Let us say that a node μ has a *direct* σ-child in $\mathcal{H}$ if μ has a child node μ' reachable through an edge labeled only by the character $\sigma \in \Sigma$. Node μ is then the *direct* father of μ'. Back to the discussion, our approach distinguishes two cases, depending on whether or not the node ν has a direct s_j-child. We consider first the case where ν does not have a direct s_j-child. This is the easier case, as the following lemma gives the handle for it.

Lemma 4.6. Let $w(\nu) = s_fs_{f+1}...s_{j-1}$ and $w(\mu) = s_{f'}s_{f'+1}...s_j$ be the longest suffixes of $s_1s_2...s_{j-1}$ and $s_1s_2...s_j$, respectively, having a marked proper locus in $\mathcal{H}$. If ν has no marked direct s_j-child, then $f' > f$.

Proof: The condition $f = f'$ is impossible, since the only way for this to happen would be if ν had a marked direct s_j-child. Since this is denied by hypothesis, then we are left with one of the following three possibilities: there is an edge to a child ν' of ν labeled by a string that begins with s_j but consists of more than one character; there is no edge from ν whose label begins by s_j altogether; ν had a direct s_j-child ν' but ν' is not marked. For each of these cases, we have to look elsewhere in $\mathcal{H}$ than among the children of ν to find the deepest possible marked proper locus μ of a suffix of $s_1s_2...s_j$. Assume now for a contradiction that we found μ such that $f > f'$. Since μ must be a marked node in $\mathcal{H}$ then so must be by construction all nodes that are proper loci of the suffixes of $w(\mu) = s_{f'}s_{f'+1}...s_j$. Among these nodes, we find, in particular, the marked proper locus of $s_fs_{f+1}...s_j$. But then ν has a direct s_j-child, which contradicts the hypothesis (seeFigure 4.2a). □

Consider now our second case, in which ν has a marked direct s_j-child ν' in $\mathcal{H}$. This case is trivial to handle whenever ν' cannot be reached from a marked node through a path of suffix links labeled by some suffix of $s_1...s_{f-1}$. Indeed, if no such path exists then traversing the edge to ν' propagates our invariant condition to $s_1s_2...s_j$, in constant time. If, on the other hand, such a path does exist, then the node on the longest possible such path is the node μ that we are seeking. Note that node μ depends on the structure of s and does not necessarily coincide with the deepest marked node encountered on an `rsuf` path from ν'.

We now outline the computations involved in prediction when s is fed to $\mathcal{H}$ one symbol at a time from left to right. The cases contemplated in Lemma 4.6 are handled in constant time per symbol if we add to $\mathcal{H}$ links from every marked node and alphabet symbol to the closest node reachable by a number of direct transitions on suffix links followed by exactly one transition on a direct downward tree edge (see Figure 4.2b). These links are easy to set in time linear in the size of $\mathcal{H}$.

To handle the case of a marked direct child node ν' of ν, we need to access, on the fly, the deepest marked node μ that is found on a reverse suffix path from node ν' above, and such that $w(\mu)$ corresponds to a suffix of $s_1 s_2 ... s_j$ (see Figure 4.2c). This is made possible by a preprocessing on s which consists of running a multiple pattern matching for the (longest) substrings ending at marked nodes of $\mathcal{H}$. For this, $\mathcal{H}$ itself is suitably adapted (in linear time) in order to be treated as a standard MPMM. The information collected in this way is used during the weighting stage. Intuitively, we use the tracks left behind by s on its trail in the MPMM, and this enables us now to locate, in constant time, the deepest `rsuf` descendant of ν' which is compatible with a suffix of $s_1 s_2 ... s_j$. The net worth is that now there is one transition to the appropriate marked node for every symbol of s, whence s is weighted in linear time.

Note that $\mathcal{H}$ is in compact form, so that specifying failure transitions on it while keeping the $O(n)$ time and space is not obvious. The details are deferred to a forthcoming paper. In what follows, we concentrate on the alternative assumption that s is available *off-line* so that it can be fed *backwards* to $\mathcal{H}$. Since we are interested only in the product of all sub-terms, the order in which they are calculated may be altered at will. We show a simple and elegant linear time prediction phase that works for this case.

Theorem 4.7. Given the automaton $\mathcal{H}$, there is an algorithm to weight any string s in overall $O(|s|)$ time.

Proof: We retain the preprocessing that assigns, to every node ν, a pointer to the closest marked node μ that can be reached following suffix links from ν (Figure 4.2b). The bulk of the algorithm consists of walking on the `rsuf` links of $\mathcal{H}$ in response to consecutive symbols of $s^R = s_m s_{m-1} ... s_1$, making occasional steps "sideways" along an edge of $\mathcal{H}$. The work is partitioned in *batches* of operations where each batch advances our knowledge of the deepest marked nodes for a certain number of suffixes of s^R. Each batch is associated with a substring of s^R and the work it performs is linear in that substring. Consecutive batches parse s^R into consecutive non-overlapping substrings of s^R, whence the linear overall bound. Batches are issued at a subset of the set of positions of s^R, and each batch faces a primary task and a maintenance task. If a batch is invoked in connection with $s_j s_{j-1} ... s_1$, the primary task of the batch is to find the two nodes $\mu = reach(j)$ and $\nu = mark(j)$ which correspond, respectively, to the deepest and deepest-marked node on the path of `rsuf`s from the root that is labeled by a prefix of $s_j s_{j-1} ... s_1$. A by-product of the primary task is to weight symbol s_j. The maintenance task is explained in what follows.

The batch for j starts having being handed a node $\theta = start(j)$ on the `rsuf` path for $s_j s_{j-1} ... s_1$ (consult Figure 4.3a). Let $s_j s_{j-1} ... s_h$ be the word labeling the `rsuf` path from the root to node θ, and consider the path P of original T_x edges that connect the root of $\mathcal{H}$ to θ. Prior to inception of this batch, the following conditions hold.

1. For all suffixes $s_k s_{k-1} ... s_1$ with $k > j$, $mark(k)$ is already known.

2. Consider the collection of all `rsuf` paths that are defined by walking from the root of $\mathcal{H}$ until the path ends or a node of P is met (Each such path is the path or a prefix of the path to $mark(k)$ for $j \geq k \geq h$). For $k = j, j-1, ...h$, $mark(k)$ is currently set to the deepest marked node on its corresponding path.

The work begins at θ by following `rsuf`s while parsing the symbols that follow s_h in s^R until node μ is found. The algorithm climbs to θ' =FATHER(μ), the father node of μ, which will be passed

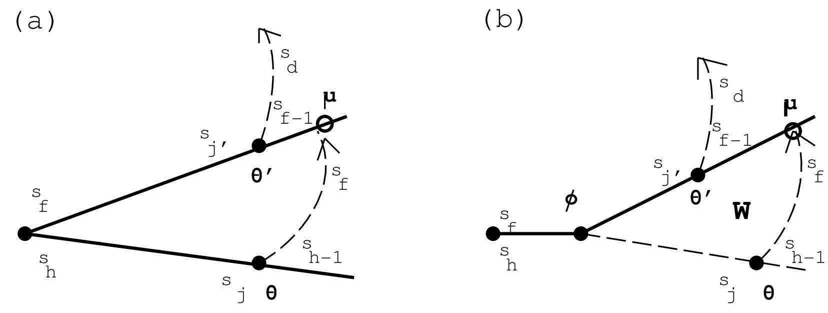


Figure 4.3: **Parsing a batch in backwards mode.** The text holds the full details.

on to the next batch where it will take the place of θ. The string $\bar{s} = s_{h-1}s_{h-2}...s_f$, connecting θ to μ, is the substring of s^R contributed by this batch to the parse of s^R mentioned above. At this point, we know the final value of $mark(j)$, since this must be either the deepest node known when the batch begun, or the deepest node encountered while scanning $\bar{s}$.

Since $w(\text{FATHER}(\mu))$ is a prefix of $w(\mu)$, then the **rsuf** path from the root to $\text{FATHER}(\mu)$ corresponds to some suffix $s_{j'}s_{j'-1}...s_f$ of $s_js_{j-1}...s_f$. The scan beginning at $start(j') = \text{FATHER}(\mu)$ will map into a substring $s_{f-1}s_{f-2}...s_d$ of s^R that immediately follows $\bar{s}$ and thus has no overlap with this string . Before the new batch at j' can begin, however, Invariants 1 and 2 must be restored. We thus address the maintenance task of the batch.

Let ϕ be the lowest common ancestor of μ and θ in $\mathcal{H}$ (consult Figure 4.3b). The only nodes where the Invariants might have been infringed are those found in the subtree W of $\mathcal{H}$ which is rooted at ϕ and has leaves at the nodes encountered on the path of $\bar{s}$ from θ to μ. The invariants are restored by visiting this subtree and checking, for every node in it, whether the pointer to the closest marked node gives an improvement over the current corresponding value of $mark$. Application of an argument already used in Lemma 4.6 to nodes μ and $\text{FATHER}(\mu)$ shows that, in particular, this treatment propagates Invariant 1 to all values of k in the interval $[j, j')$, i.e., Invariant 1 now holds for all $k > j'$. The number of nodes encountered in the visit is bounded by $|\bar{s}|$, the number of leaves, whence the work involved is linear in $|\bar{s}|$. □

4.4 Computing Empirical Probabilities

We consider here, in greater detail, the notion of empirical probability for a string and its related computations. This notion is not straightforward. Fortunately in the algorithm – in so far as the construction of the automaton goes – we are interested primarily in *conditional* probabilities which turn out to be less controversial.

One ingredient in the computation of empirical probabilities is the count of occurrences of a string in another string or set of strings. As seen, although there can be $\Theta(n^2)$ distinct substrings in a string of n symbols, Fact 4.2 and the very structure of T_x show that linear time and space suffice to build an implicit table of χ_w counts of all strings w in x.

One way to define the empirical probability $\tilde{P}(w)$ of w in x is to take the ratio of the count χ_w to $|x|-|w|+1$, where the latter is interpreted as the maximum number of possible starting positions for w in x. This corresponds to viewing $\tilde{P}(w)$ as χ_w divided by $\chi_{|w|}$ (i.e., how many of the overall $n-l+1$ substrings of length l were actually w)[2]. From the computational standpoint, for w and v

[2]This definition has a convenient probabilistic quality in that $\forall l = 1, 2, ..., L \quad \sum_{|w|=l} \tilde{P}(w) = 1$.

much shorter than x we have that the difference between $|x|-|w|+1$ and $|x|-|wv|+1$ is negligible (this is not automatically true for any set S of k strings, where we would have $|x|-k|wv|+1$), which means that the probabilities computed in this way and relative to words that end in the middle of an arc of T_x do not change, i.e., computing probabilities for strings with a proper locus is enough to know the probabilities of all substrings.

This notion of empirical probability, however, assumes that every position of x compatible with w length-wise is an equally likely candidate. This is not the case in general, since the maximum number of possible occurrences of one string within another string crucially depends on the compatibility of self-overlaps. For example, the pattern *aba* could occur at most once every two positions in *any* text, *abaab* once every four, etc. Compatible self-overlaps for a string z depend on the structure of the *periods* of z. A string z has a *period* w if z is a prefix of w^k for some integer k. Alternatively, a string w is a period of a string z if $z = w^l v$ and v is a possibly empty prefix of w. When this causes no confusion, we will use the word "period" also to refer to the length or *size* $|w|$ of a period w of z. A string may have several periods. The shortest period (or period length) of a string z is called *the period* of z. A string is trivially always a period of itself. It is not difficult to see that two consecutive occurrences of a word may overlap only if their distance equals one of the periods of w. Along this line of reasoning, we have

Fact 4.8. The maximum possible number of occurrences of a string w into another string x is equal to $(|x|-|w|+1)/|u|$, where u is the smallest period of w.

If we wanted to compute the empirical probabilities of, say, all prefixes of a string along the definition of Fact 4.8, we would need first to know the periods of all those prefixes. In fact, by a classical result of string matching, the period computations relative to the set of prefixes of a same string can be carried out in *overall* linear time, thus in amortized constant time per prefix. We refer for details and proofs to e.g., Aho and Corasick (1975); Apostolico and Galil (1997). Such a construction may be applied, in particular, to each suffix suf_i of a string x while that suffix is being inserted as part of the direct tree construction. This would result in an annotated version of T_x in overall quadratic time and space in the worst case.

Perhaps more interestingly, we have that for empirical probabilities defined per Fact 4.8 the following holds.

Theorem 4.9. The set of values $\tilde{P}(w) = \chi_w \cdot |u|/(|x|-|w|+1)$ can be computed for all words of x that have a proper locus in T_x in overall linear time and space.

Proof: Simply compute periods while walking on suffix links "backward", i.e., traversing them in their reverse direction, beginning at the root of T_x and then going deeper and deeper into the tree. This walk intercepts all nodes of T_x. Correctness rests on the fact that for any word w the periods of w and w^r coincide. □

Note, however, that since the period may vary in the middle of an arc, so could the empirical probabilities computed in this way. This weakens the assumption that the probability of a short word ending in the middle of an arc is surrogated by that of the shortest extension of that word with a proper locus. Fortunately, the discussion that led to Fact 4.4 shows that $\mathcal{T}$, being nothing but a subgraph of $\mathcal{H}$ connected by `rsufs`, only needs the $O(n)$ probabilities at the $O(n)$ nodes of $\mathcal{H}$, irrespective of how such probabilities are defined.

4.5 Final Remarks

Using the known duality between direct and reverse suffix links, it is natural to revolve our previous construction around and learn trees for the reverse of the strings in set S. Indeed, the PST tree structure itself is but a subtree of the expanded suffix tree of S^R. In such a dual construction, the learning phase is concerned with building a suitable, reverse-annotated tree of x^R while the weighting phase will traverse this tree.

Another algorithmic challenge set forth by this work concerns setting up procedures of unsupervised learning that can follow some initial training phase. Once some version of the automaton is constructed from an initial set of positive examples, one wishes to easily learn a new example, i.e. update it in linear time, in the very same manner all previous examples in S were assimilated one by one. The same goes for removing a sequence from our pool. Along these lines, we develop a simple incremental learning scheme – start off with some initial seed S from which the concept is first built. Then, while predicting over query sequences, one may, when coming across a sequence that, with high likelihood, belongs to the family – efficiently assimilate it into the learned structure before proceeding. Similarly, one may from time to time go over the list of sequences composing S and check whether due to the evolution of the concept some members no longer fit the concept - these may then be efficiently rejected. This is a useful feature to have when learning from noised or error-prone data, as is our case.

Other, closely related, benefits stem from deliberately abstaining from pruning our trees or pre-smoothing the head count vectors implicit in them. These facts will allow us in Chapter 5 to couple the incremental nature presented above with a form of an *annealing* schedule (see Rose, 1998). Namely, we may start off with a rather permissive notion of a significant pattern and an appropriate smoothing technique, and gradually during learning, while we evolve our notion of a family, and hopefully put it on firmer grounds, we may "cool down" our system by increasing the threshold for significance, while lowering the impact of smoothing. We may also alter L, now taken as the maximal prediction (but not learning) depth, similarly.

This chapter is based on an extended abstract (Apostolico and Bejerano, 2000a) which was presented at RECOMB 2000, and was later developed to a journal version (Apostolico and Bejerano, 2000b).

Chapter 5

Markovian Protein Sequence Segmentation

In the last two chapters we have shown that variable memory models can capture the notion of a protein family from a given seed of unaligned sequences, using very efficient training and subsequent prediction algorithms. Motivated by these results we turn to the much harder task of segmenting a given set of unlabeled, unaligned sequences into the domains that compose them. In this chapter we present and calibrate a novel algorithmic approach to this task, and apply it to protein datasets.

5.1 Introduction

In Chapters 3–4 we have focused on a specific variant of variable memory modeling (VMM), termed Prediction Suffix Trees (PST). These models were adapted there to the task of modeling protein sequence families from labeled seeds, and subsequent clustering of novel sequences to their respective families. In that task we have extensively compared PST performance to that of profile hidden Markov models (HMM), considered the state of the art in this field.

A much harder task is that of searching protein sequences for novel domains, of which no previously labeled examples are available.

Goal. Given a set of unaligned, unannotated, possibly multi-domain protein sequences, segment these into their underlying, recurring, domains.

One such example is produced by protein-protein interaction measurements. Through a set of such experiments one often comes up with a subset of proteins, all of whom interact with a certain other protein. A plausible hypothesis in this case would be that the ability to interact with the latter is conferred by a domain found in all (or many) of the interacting proteins.

We recall from Section 2.2.1 that HMMs are predefined parametric models and their success crucially depends on the correct choice of the state model and observation distributions attached to each of the states of the Markov chain. In our context the architecture and topology of the profile HMM are predetermined by the family seed multiple sequence alignment (MSA), typically hand crafted in advance by an expert. Subsequent HMM training is thus limited to calibration of the resulting model parameters, such as emission probabilities, etc. The memory property of these models is limited to first order dependency on the assumed MSA column position. As a result it is rather difficult to directly generalize these models to hierarchical structures with unknown a-priory state topology. This, however, is the situation in our case, when presented with a set of sequences on which very little, possibly nothing, is known in advance.

Meta-Meme (Grundy et al., 1997), described in Section 2.3.2, proposes one indirect approach to this problem, in the context of protein sequences. We briefly recall that Meta-Meme first searches the group of unaligned sequences for over-abundant sequence segments, which can be represented by simple ungapped profiles. The algorithm then goes on to make a very restrictive assumption, that all given input sequences share a single N- to C-terminal structure, and tries to infer the best underlying linear ordering of the found motifs. This results in a very simple HMM architecture and topology (Figure 2.8), composed of a set of profiles, each given by a series of match states, with single insert states between consecutive profiles. More flexible approaches to HMM architecture and topology inference do exist (e.g., Fine et al., 1998; El-Yaniv et al., 1998), albeit not in our context.

As we have already seen, PST models are very different in this respect. In particular, model architecture determination is central to the PST learning process (Section 3.2.2), and requires no external pre-processing. While they can be weaker than HMMs as generative models, we have seen that they are able to capture longer correlations and higher order statistics of the underlying sequence. In fact, we have shown in our experiments that when presented with a dataset containing repetitions of several domains, a single PST model may be able to capture all of these (Figure 3.7–3.8). This property stems from two observations: that different protein domains have different conserved k-mer vocabularies, and that our data driven learning algorithm captures all of these together in a single model. This last property has lead us to attempt to take such a PST model and try to split it in an automated manner into several separate PSTs, each modeling a single domain in the given set. Our resulting model can in fact be viewed as an HMM with a VMM attached to each of its states. However, our learning algorithm allows for a completely adaptive structure and topology both for each state and for the whole model.

The approach we take for this task is information theoretic in nature. For this purpose we will first revise our learning algorithm for a single PST model to fit an unsupervised setting. We will then define a novel algorithm for sequence segmentation and clustering of statistically similar segments into different PST models. The problem of learning a stochastic mixture of generative models is known to be computationally hard in general, similarly to data clustering, with only very simple generation schemes that can be segmented correctly in an efficient manner (Freund and Ron, 1995). And yet, as we demonstrate later in this chapter, they are very much applicable to protein sequence segmentation.

5.2 Non-Parametric PST Training

Both the learning algorithm of Figure 3.2 (page 39) and its optimized successor from Section 4.2.2 require a set of five user-controllable parameters. Each of these is involved in different aspects of the algorithm: determining the maximal length of memorized suffixes, how abundant must they be and how much must their prediction vector differ from that of shorter suffixes to justify inclusion in the resulting tree. These parameters give the researcher the needed freedom when trying to model in a **supervised** manner protein families which differ in size, in length, and in conservation levels and patterns. Given a subset of family members one can, through techniques of cross validation, search for the best combination of parameters that model this family. For our new **unsupervised** segmentation goal this definition is unwieldy. We turn to replace it with a non-parametric variant which strives for the same modeling goal, yet requires no external user supplied parameters.

To do so, we turn to the **minimum description length** (MDL) principle (Barron et al., 1998), which weights the suggested model complexity against the complexity of the observed data. Given a model of the input sequences, a PST in our case, we use it to code as compactly as possible the data itself into a uniquely decodable binary string. This coding is also termed **lossless compression**. We define the description length (DL) of the PST T and data $\bar{x}$ pair as the combined cost of describing the PST model itself and coding the sequences using it,

$$DL(T,\bar{x}) = DL(T) + DL(\bar{x}|T)$$

Given a dataset $\bar{x}$, we would like to model it using the PST T that minimizes the above combined expression. This approach contrasts the two opposed modeling extremes. A trivial PST minimizes $DL(T)$ but results in a long inefficient coding of the data. On the other hand, the PST that best describes the given sequences and minimizes $DL(\bar{x}|T)$ runs a high risk of over-fitting the data by modeling statistically insignificant patterns. However, this excessive modeling results in an increase of $DL(T)$ disfavouring the choice of this model. Thus we see that this simple principle results in a model whose complexity is governed by the length of the input sequences as well as by their statistical richness, which is compressed using the PST data structure. From a theoretical point of view the MDL principle can also be shown to relate to other model selection criteria such as Bayesian inference (see Barron et al., 1998).

We turn to present the algorithm, extending the notations of Section 3.2.1. The inputs to the algorithm are a string $\bar{x} = x_1..x_n$ and a vector of weights $\bar{w} = w_1..w_n$, where each w_i is a weight associated with x_i ($0 \leq w_i \leq 1$). A single string is used for ease of exposition, and can be easily generalized (as was done in Section 3.2.1). For clarity we will also denote $w(x_i) \equiv w_i$. Later on, in a multiple PST setting, $w(x_i)$ will serve as a measure of confidence that a given PST generated the observation x_i. For now we may simply assume $\forall w_i = 1$.

For a string s we denote $sx_i \in \bar{x}$ if it is a substring of $\bar{x}$ ending at index i. We define

$$w_s(\sigma) \equiv \sum_{x_i=\sigma \ \ and \ \ sx_i \in \bar{x}} w(x_i)$$

$$w(s) \equiv \sum_{\sigma \in \Sigma} w_s(\sigma)$$

Clearly $\frac{w_s(\sigma)}{w(s)}$ is an empirical estimate for $P_s(\sigma) \equiv \tilde{P}(\sigma|s)$. We denote by P_s the resulting probability distribution vector for all $\sigma \in \Sigma$. We smooth these probabilities uniformly using Krichevsky-Trofimov estimators resulting in

$$\overline{\gamma}_s(\sigma) = \frac{w_s(\sigma) + \frac{1}{2}}{w(s) + \frac{1}{2}|\Sigma|}$$

as these ensure good bounds on the distance between the two distributions for small sample sizes (Krichevsky and Trofimov, 1981).

We turn to compute the description length, $DL(T)$, of a given PST. Coding for the tree skeleton requires $|\Sigma|$ bits per node. Each such bit denotes the non/existence of the respective son node. Note that ordering these bits according to Σ saves us the coding of the node label explicitly. In addition we should code all probability distribution vectors $\{P_s\}_{s \in T}$. Note, that the distribution vector P_s is used to code only those x_i's, for which $suf_T(x_1..x_i) = s$. Thus the total amount of data that is coded using P_s is at most $w(s)$, and exactly $w(s)$ for leaf nodes. In order to achieve a minimal description length of the vector P_s together with the fraction of the data that is coded using P_s, the counts $w_s(\sigma)$ should be coded to within accuracy of $\sqrt{w(s)}$ (see Barron et al., 1998). Each node s holds $|\Sigma|$ such counts, thus the total description length of a node s is

$$Size(s) = |\Sigma| + \frac{|\Sigma|}{2} \cdot \log_2(w(s))$$

Denoting by T_s the subtree of T rooted at node s,

$$Size(T_s) = Size(s) + \sum_{\sigma s \in T} Size(T_{\sigma s})$$

and $DL(T) = Size(T_\lambda)$, where λ denotes the empty context at the root node of $T \equiv T_\lambda$.

Now, we compute the description length of the data given the PST, $DL(\bar{x}|T)$. Information theory teaches us that the minimal attainable average code length per symbol, for all symbols coded using node s, is the **entropy** of the distribution P_s (Cover and Thomas, 1991),

$$H(P_s) \equiv -\sum_{\sigma \in \Sigma} P_s(\sigma) \log_2 P_s(\sigma)$$

Note that this is also the average per symbol surprise, discussed in Section 3.3.2. Consider all input sequence data coded by nodes at the subtree T_s, i.e., all x_i for which s is a suffix of $suf_T(x_1..x_{i-1})$. They can either be coded by s itself or by a deeper node within T_s:

- If $x_1..x_{i-1}$ ends with $\hat{\sigma}s$ for $\hat{\sigma}s \notin T$, then x_i is coded using P_s and the average code length of x_i is $H(P_s)$. This will happen $w(\hat{\sigma}s)$ times out of the $w(s)$ times we visit s.
- If $\hat{\sigma}s$ is present in T, then x_i is coded according to the distribution of some node in $T_{\hat{\sigma}s}$, and the average code length will be the entropy of the distribution in that node. This will also happen $\frac{w(\hat{\sigma}s)}{w(s)}$ out of the times we visit s.

Thus the entropy of a tree node T_s satisfies the recursive definition:

$$H(T_s) = \sum_{\hat{\sigma}s \in T} \frac{w(\hat{\sigma}s)}{w(s)} \cdot H(T_{\hat{\sigma}s}) + \sum_{\hat{\sigma}s \notin T} \frac{w(\hat{\sigma}s)}{w(s)} \cdot H(P_s)$$

and the code length of all data coded by T_s is thus $w(s) \cdot H(T_s)$. We conclude that

$$TotalSize(T_s) = Size(T_s) + w(s) \cdot H(T_s)$$

Our goal is thus to minimize

$$DL(T, \bar{x}) = TotalSize(T_\lambda) = Size(T) + H(T) \cdot \sum_{i=1}^{n} w_i$$

In a simple single source scenario each $w_i = 1$ and thus the second term $DL(\bar{x}|T) = n \cdot H(T)$, where n is the length of $\bar{x}$ and $H(T)$ is the average length of a code-word in T.

Our algorithm proceeds in two steps. In step I we extend all the nodes that are potentially beneficial, i.e. by using them we *may* decrease the total size. Of interest are thus only those nodes whose description length is smaller than the code length of data passing through them when that data are coded using the parent node distribution. Then, in step II the tree is recursively pruned so that only truly beneficial nodes remain. If a child subtree $T_{\sigma s}$ of some node s gives better compression (respecting its own description length) than that of its parent node, that subtree is kept, otherwise it is pruned. The resulting algorithm is given in Figure 5.1.

5.3 Markovian Segmentation Algorithm

Having defined a self-regulated non-parametric variant of the PST learning algorithm, we turn to define the segmentation task, through the use of mixture models.

```
Learn_PST(String $\bar{x}$, Weights $\bar{w}$)
        1. $T$ = Build_PST($\bar{x}$, $\bar{w}$)
        2. Prune($T$, $\lambda$)
Where the two steps are:
I. Build_PST (String $\bar{x}$, Weights $\bar{w}$)
        1. Start with $T$ having a single node $\lambda$.
        2. Recursively for each $s \in T$ and $\sigma \in \Sigma$
              If $Size(\sigma s) < H(P_s) \cdot w(\sigma s)$
                  Then add node $\sigma s$ to $T$.
II. Prune (Tree $T$, node $s$)
        1. For each $\sigma \in \Sigma$ such that $\sigma s \in T$:
           (a) Prune($\sigma s$)
           (b) If $TotalSize(T_{\sigma s}) > H(P_s) \cdot w(\sigma s)$
                  Then delete subtree $T_{\sigma s}$
```

Figure 5.1: **A non-parametric MDL driven PST learning algorithm.**

5.3.1 Mathematical Goal

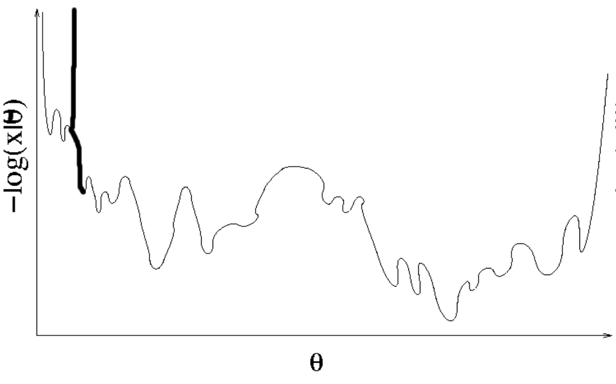
Given an input string $\bar{x} = x_1..x_n$ (which for ease of notation represents our set of sequences, as before), we will try to approximate its generation using a mixture of PST models. We will assume the existence of k distinct PST models $\mathcal{T} = \{T_j\}_{j=1}^{k}$. To generate $\bar{x}$, we start by probabilistically choosing one of the models T_{j_1}. We then generate a series of symbols using this model, as in Section 3.2.3. At some random time i we switch to using a second model T_{j_2}, which is used to generate the next segment, and so forth, until some T_{j_l}, for $l > k$, has been used to complete the emission of all of $\bar{x}$.

Ideally we would like to fit, say T_1, to some background distribution of amino acids and all other models, each to a single sequence domain, and have $\bar{x}$ generated according to the exact structural boundaries, with T_1 emitting all unstructured regions and each T_j emitting the occurrences of the domain it models. The generation of the sequence set $\bar{x}$ is thus fully described by:

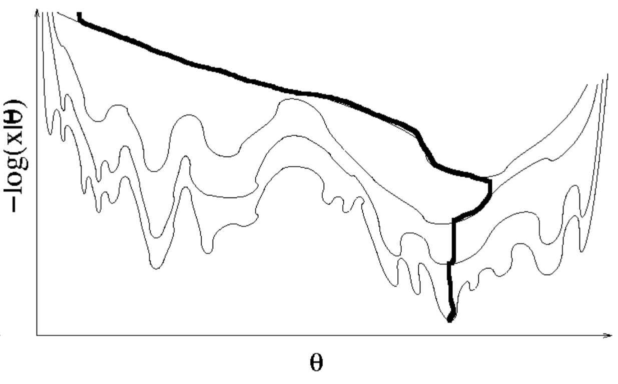
- An appropriate number of mixture components k,
- A set of k PST models $\mathcal{T}$,
- The exact switching times and order in which this mixture has emitted $\bar{x}$.

Denoting the above triplet as θ, our goal is to find a combination that maximizes $p(\bar{x}|\theta)$. By trying to fit this modeling scheme to our data we rely on two strong assumptions: That each structural domain appears a sufficient number of times in the input set, such that knowledge of the exact domain boundaries would allow us to build a reliable PST for it, using the technique of the previous section. And that each domain occurrence in the input set is long enough to allow significantly better modeling using the correct model, compared to the other models generating the set. Clearly, the more instances of a given domain in the set, the better, more distinctly we can model it.

Finally, note that this formulation performs **clustering** of sequence segments of the given input sequences into the different PST models.



(a) Gradient descent



(b) Annealing

Figure 5.2: **Schematic comparison of optimization approaches.** (a) A schematic depiction of the parameter space, whose likelihood we wish to maximize. In bold we show how gradient descent from an initial guess greedily converges to the nearest local optimum, having explored very little of the parameter space. (b) By solving a succession of less and less smoothed versions of the parameter space, an annealing approach tends to explore more of it, before converging to an optimum which can be very far from our initial guess, possibly the global one.

5.3.2 Conceptual Framework

If we assume that k, the number of generating models is known, we face a k components finite mixture model, similar to the two component mixture model defined by MEME (Bailey and Elkan, 1994). In Section 2.3.2 we reviewed the expectation-maximization (EM) approach used by MEME. The same approach could be extended to handle k PSTs. Starting from an initial guess of models $\mathcal{T}^0$, one can estimate a segmentation induced by the set using Bayes rule, and then hold the segmentation fixed and retrain a new set of models based on the segmentation to obtain $\mathcal{T}^1$. After each iteration of these two steps the likelihood is guaranteed to increase, i.e. $p(\bar{x}|\theta^{t+1}) \geq p(\bar{x}|\theta^t)$, ensuring that the process converges to a local maximum. However, as noted in Chapter 2, this greedy hill climbing approach typically converges to the nearest local maximum, after exploring very little of the riddled parameter space.

Rather than running many restart points, using different amounts of models we will opt for a **deterministic annealing** (DA) approach (Rose, 1998), which tries both to aim for the global maximum and directly infer the number of mixture components. Instead of directly trying to solve the original optimization problem, the DA approach solves a series of successive problems. The first problem solved is an extremely smoothed version of our optimization problem, where typically only a single extremum exists. The (global) solution of this convex problem serves as the starting point to the next surface, which is less smooth than its predecessor, and thus resembles more the surface of interest. Thus, we can employ EM to solve a series of increasingly harder optimization problems, in the hope that the gradual refinement of the parameter surface will allow us to explore more than the immediate neighbourhood of our initial guess and converge to a better local optimum, possibly the global one. The difference between the two approaches is drawn schematically in Figure 5.2.

To guide us in the successive refinement of the parameter space we employ further information theoretic concepts. We consider the clustering of sequence segments into the different PSTs as a form of **lossy coding**. Namely, we consider each sequence symbol as replaced by the model that emitted it. Since the emitting PST is stochastic, we define the **distortion** due to this replacement based on the probability the PST assigns to that symbol. Now, instead of directly trying to maximize the likelihood, we turn to minimize the distortion between models and sequences, through a succession of intermediate distortion levels. This framework can also be shown to prefer solutions using a smaller amount of models (Rose, 1998). Thus, we will gradually increase the number of models available to the algorithm at every phase. Some of these will model no data, others will become identical to existing models. The former will be discarded, and the latter will be merged. The end result is a self regularizing algorithm which also tries to infer the number of mixture components while minimizing clustering distortion.

5.3.3 Soft Segment Clustering

Let $\mathcal{T} = \{T_j\}_{j=1}^k$ be the set of k PSTs we are currently working with. We define $w_j(x_i) \equiv P(T_j|x_i)$ to be the probability that a symbol x_i is assigned to model T_j, given the context of x_i in $\bar{x}$, which is omitted for clarity of notation. The vector $\bar{w}_j$ of weights will later be used to retrain T_j.

As described in the previous section we will cast the clustering problem as a lossy coding one, and assume that each sequence symbol is replaced by the PST which has generated it. To quantify the imprecision introduced by this substitution, a natural distortion measure would be $d(x_i, T_j) = -\log P_{T_j}(x_i|x_1 \ldots x_{i-1})$. This measure is non-negative, it is equal to zero only when the PST would emit the sequence symbol x_i with probability one, and it is additive in a natural way. We enhance this measure by introducing a smoothing window of size $2M+1$ around symbol x_i, setting

$$d(x_i, T_j) = -\sum_{\alpha=i-M}^{i+M} \log P_{T_j}(x_\alpha|x_1 \ldots x_{\alpha-1}) \qquad (5.1)$$

The purpose of the symmetric window around x_i is to force the models to alternate in a controllable manner, allowing for smoother segmentation, and reliable likelihood estimation. This local measure can then be summed to define the (global) distortion between a segmentation into a set of PSTs, and the input sequences,

$$\langle d \rangle = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} d(x_i, T_j) \cdot P(T_j|x_i)$$

Here, and elsewhere, we approximate the unknown average over $p(x_i)$ with the normalized sum over the sample $\bar{x}$. Minimizing the distortion is thus seen to be closely related to maximizing the likelihood of the models and segmentation.

We can now define the inner loop of our algorithm, of soft clustering the data into models at a prescribed distortion level. We use an EM-like approach that alternates between finding the optimal sequence segmentation for a fixed set of models, and model retraining given a fixed segmentation. We begin by finding the optimal assignment, or segmentation probabilities $P(T_j|x_i)$ for a fixed set of PSTs, $\mathcal{T}$, constrained by the allowed distortion level D. Rate distortion theory (Cover and Thomas, 1991, Ch. 13) teaches us that the optimal assignment is obtained by solving

$$R(D) = \min_{\{P(T_j|x_i)\ :\ \langle d \rangle \le D, \sum_{j=1}^{k} P(T_j|x_i)=1\}} I(\bar{x}, \mathcal{T}) \qquad (5.2)$$

where I is the mutual information between $\bar{x}$ and $\mathcal{T}$

$$I(\bar{x}, \mathcal{T}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} P(T_j|x_i) \cdot \log \frac{P(T_j|x_i)}{P(T_j)}$$

```
**Blahut-Arimoto($P(\mathcal{T})$, $\beta$)**
  Repeat until convergence:
      1. $\forall i, j :\ P(T_j|x_i) = \frac{P(T_j)e^{-\beta d(x_i, T_j)}}{\sum_{\alpha=1}^{k} P(T_\alpha)e^{-\beta d(x_i, T_\alpha)}}$
      2. $\forall j :\ \ P(T_j) = \frac{1}{n}\sum_{i=1}^{n} P(T_j|x_i)$
```

```
**Soft_Clustering($\mathcal{T}$, $P(\mathcal{T})$, $\beta$)**
  Repeat until convergence:
      1. Blahut-Arimoto($P(\mathcal{T})$, $\beta$)
      2. $\forall j :\ T_j$ = Learn_PST($\bar{x}$, $\bar{w}_j$)
```

Figure 5.3: **Soft clustering at a given distortion level.**

and $P(T_j)$ is the proportion of data assigned to model j

$$P(T_j) = \frac{1}{n}\sum_{i=1}^{n} P(T_j|x_i)$$

By minimizing the joint information between models and sequences we aim to find the segmentation which makes the least assumptions (or constraints) on the relationship between models and sequences, while still maintaining the desired distortion level. As such it is also the most probable assignment.

While we cannot write an analytical solution to Equation 5.2, we can employ a computation procedure, known as the Blahut-Arimoto (BA) algorithm, that provably converges to the optimal solution (Csiszar, 1974). Given some initial (prior) distribution $P(T_j)$, the algorithm iterates between two self consistent equations for estimating $P(T_j|x_i)$ and $P(T_j)$. The constraint on the maximally allowed distortion D is replaced by its Lagrange multiplier β in the Lagrangian of Equation 5.2. The two relate to each other through $\beta = -\frac{dR}{dD}$ and since $R(D)$ can be shown to be convex, a one to one relation exists between values of the two, such that small values of β correspond to high distortion levels, and vice versa (Cover and Thomas, 1991). The BA algorithm is given in Figure 5.3(left), where we introduce an abbreviated notation $P(\mathcal{T}) \equiv \{P(T_1), \ldots, P(T_k)\}$.

We can now take advantage of the non-parametric PST training algorithm, defined in Section 5.2 to obtain an EM-like iterative loop. We fix the obtained segmentation and train a new set of PST models, using the $P(T_j|x_i)$ values as weight vectors. The resulting procedure, shown in Figure 5.3(right), is performing soft clustering of sequence symbols into k PST models, at a given distortion level.

5.3.4 Annealing Main Loop

Once the soft clustering is defined for a prescribed distortion level, embedding it in a deterministic annealing loop is straightforward. We begin by defining the phase through which new models will be added to the clustering procedure, as in Figure 5.4. This we do by substituting each PST T in $\mathcal{T}$ with two new PSTs. We create two exact copies of T and do random anti-symmetric perturbations of the counts vectors in each node of the two copies (this corresponds to a perturbation of the probability measures each of the copies induces on Σ^+). We then distribute $P(T_j)$ equally among the two, and have them replace the original model.

At every distortion level, (corresponding to the given value of β) a limited number of PST models K suffice to achieve D. When more than the required models are supplied, i.e. $k > K$, some models become near-identical, while others are not used by the clustering solution, indicated by $P(T_j) = 0$. The first phenomena is typical of DA procedures (see Rose, 1998). The latter results from our combination of window based likelihood smoothing (Equation 5.1), which enforces contiguous segmentation, together with the MDL governed PSTs (Figure 5.1). Models which do not do significantly better than other on some data segments, inevitably lose complexity, until they degenerate and are out-performed by the others. We take advantage of this self regularization in the main DA loop of the algorithm.

We are finally ready to outline the complete algorithm. We start with $\mathcal{T}$ including a single PST T_0 that is trained on the full sequence $\bar{x}$ with $w^0(x_i) = 1$ for all i. We pick an initial low value of β (corresponding to a high distortion value), split $\mathcal{T}$ and partition $\bar{x}$ among the resulting models, T_1 and T_2. Once clustering has converged we split $\mathcal{T}$ again and repeat. If a model is found to have lost its data, it is eliminated. When the effective number of models stops increasing, we increase β (decreasing the desired distortion level) and repeat the process. The pseudo-code and schematic description of the resulting algorithm are given in Figure 5.5. As we shall next demonstrate, sets of segments that are assigned with high probability to the same model over several values of β are stable clusters that contain important information about the statistical structure of our sample.

Before proceeding to the applications part, we introduce a variant of the segmentation algorithm which executes a single iteration of the Blahut-Arimoto algorithm between successive model re-training. This results in the replacement of the procedures of Figure 5.3 with that of Figure 5.6(left). The latter figure also holds a schematic drawing of how this affects the general behaviour of the algorithm. The motivation for this variant is empirical. Our runs indicate that the altered algorithm often converges to a superior solution. In the coming sections we will therefore resort to using this variant. From a computational point of view it appears that it is beneficial to spend more time on re-training models, while seeking for the optimal segmentation, rather than optimizing every re-training instance, thus moving through a more constrained sub-space of possible PST configurations.

5.4 Textual Calibration Tests

Before approaching the protein segmentation task, we analyze several synthetic data sets to calibrate and examine our approach. Our synthetic data sets will be composed of alternating fragments from five running texts in five different languages: English, German, Italian, French and Russian. We pre-process the five original texts, by converting them into lower case accent-free Latin letters with white space substituting all separators. For Russian, a commonly used phonetic transcript was applied.

The first test set is constructed by taking 100 consecutive letters from each text in turn, re-

```
Split_PSTs($\mathcal{T}$, $P(\mathcal{T})$)
  Replace each $T_j$ in $\mathcal{T}$ by two new models:
      1. Start with two exact copies of $T_j$: $T_{j_1}$ and $T_{j_2}$
      2. For each node $s$ in $T_j$ and for each $\sigma \in \Sigma$:
          (a) Select $\{\zeta = 1, \xi = 2\}$ or $\{\zeta = 2, \xi = 1\}$
              with probability $\frac{1}{2}$ / $\frac{1}{2}$.
          (b) Perturb and normalize the counts vectors:
                For $T_{j_\zeta}$: $w_s(\sigma) = (1+\gamma) \cdot w_s(\sigma)$   $(|\gamma| \ll 1)$
                For $T_{j_\xi}$: $w_s(\sigma) = (1-\gamma) \cdot w_s(\sigma)$
      3. $P(T_{j_1}) = \frac{1}{2}P(T_j)$, $P(T_{j_2}) = \frac{1}{2}P(T_j)$
```

Figure 5.4: **The PST splitting procedure.**

```
Algorithm main part:
Initialization:
        For all $i$, $w^0(x_i) = 1$
        $T_0 = \text{Learn\_PST}(\bar{x}, \bar{w}_0)$
        $\mathcal{T} = \{T_0\}$, $P(T_0) = 1$
        $\beta = \beta_0$, $k_{prev} = 0$
Annealing loop:
        While $|\mathcal{T}| < \frac{n}{2M+1}$
            1. While $|\mathcal{T}| > k_{prev}$
                (a) $k_{prev} = |\mathcal{T}|$
                (b) Split_PSTs($\mathcal{T}$, $P(\mathcal{T})$)
                (c) Soft_Clustering($\mathcal{T}$, $P(\mathcal{T})$,$\beta$)
                (d) Remove all $T_j$ s.t. $P(T_j) = 0$ from $\mathcal{T}$.
            2. Increase $\beta$
```

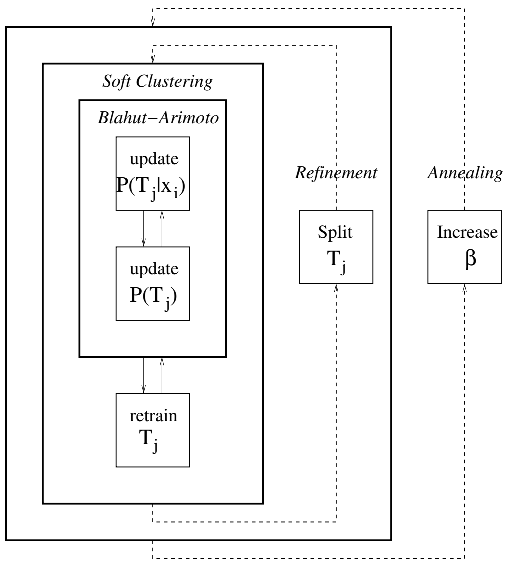


Figure 5.5: **The Markovian sequence segmentation algorithm.**

peating this concatenation 300 times. Thus, the first 100 letters open the English text, the next four segments, 100 letters each, open the other four texts, respectively. We then concatenate the next 100 letters from the English text, etc. In linguistic terms this is a relatively high switching rate, as 100 letters typically allow for two sentences per segment. The resulting interleaved text is 150,000 letters long. We made several independent runs of our algorithm, using a window size of 21 ($M = 10$ in Equation 5.1), which seemed to produce the best results for the language experiments. In every run, after $2000-3000$ accumulated innermost iterations of the non-BA soft clustering loop (Figure 5.6) we obtained a clear-cut, correct segmentation of the text into segments corresponding to the different languages. As Figure 5.7 shows, the segmentation boundaries are accurate up to a few letters. Moreover, in all runs, subsequent steps that further split the five language models resulted in starvation and subsequent removal of the five redundant models, resulting in the same, stable segmentation.

In Figure 5.8 we show a temporal picture of a segmentation run, by plotting all $P(T_j)$ values after each iteration of the soft clustering procedure. We recall that $P(T_j)$ is the relative amount of data assigned to an individual model. Thus, at each call to Split_PSTs, we get a discontinuity point as a given $P(T_j)$ ceases to exist, and is replaced by two models, each weighted at $\frac{1}{2}P(T_j)$. When a model loses all data, the respective $P(T_j)$ drops to zero. Such models are then removed. It is interesting to note that in most runs linguistically similar languages (English and German, French and Italian) tended to separate at later stages of the segmentation process. This observation suggests that in addition to the segmentation one may also attempt to obtain a hierarchical structure over the discovered data sources, as is obtained, for example, in El-Yaniv et al. (1998). Finally, by comparing the BA and non-BA segmentation runs in Figure 5.8 we note the different typical bifurcation shapes after model splitting between the two, and the fact that the non-BA solution obtains model separation at lower levels of β.

To better understand the limitations of the method, we performed several more experiments in this context.

First, we examined the location and size of the smoothing window in the definition of local

```
Soft_Clustering-NoBA($\mathcal{T}$, $P(\mathcal{T})$, $\beta$)

  Repeat until convergence:

      1. $\forall i, j$ :  $P(T_j|x_i) = \frac{P(T_j)e^{-\beta d(x_i,T_j)}}{\sum_{\alpha=1}^{k} P(T_\alpha)e^{-\beta d(x_i,T_\alpha)}}$
      2. $\forall j$ :  $P(T_j) = \frac{1}{n}\sum_{i=1}^{n} P(T_j|x_i)$
      3. $\forall j$ :  $T_j$ = Learn_PST($\bar{x}$, $\bar{w}^j$)
```

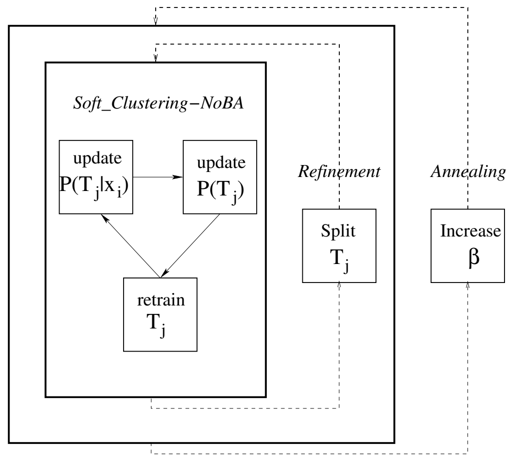


Figure 5.6: **Segmentation algorithm variant without the BA procedure.**

distortion between sequence symbol and its modeling PST (Equation 5.1). By comparison, the symmetric window was found to be preferable to non-symmetric ones. For example, averaging over $x_{i-2M} \ldots x_i$, caused both a decrease in the ability to detect quickly alternating sources, as well as a skew in the point of transition between most likely models. As expected, the predicted transition point moves in the opposite direction to the center of the smoothing window. The magnitude of M was seen to also influence the performance of the algorithm. A large M with respect to underlying segment size would diminish our ability to distinguish between quickly alternating sources, although it does provide more confident local segmentation when $P(T_j|x_i)$ is compared between the generating model and the others, during sequence weight redistribution. On the other hand, very small values would result in very noisy segmentation, as well as occasional convergence to inferior local minima. Averaging window size was thus deemed to be data driven, and was targeted for a later automation effort.

In another experiment we searched for the model alternation rate beyond which the segmentation algorithm no longer segments correctly. When keeping the total amount of data as above, segmentation could be reliably achieved when sources alternated every 30 letters or so, as in Figure 5.9(a). On average this would mean less than one whole sentence in each segment. Segmentation at this high alternation level, was however less stable, as two language models would on occasion merge, and later split back. Completely stable segmentation was obtained at a segment length of 40. We also tried to minimize the total length of the data, while alternating every 100 letters as before. In this case, accurate segmentation was obtained from a total length of 6,000 letters per language, as in Figure 5.9(b), while stable segmentation appeared only when 8,000 letters or more are supplied per language. This appears to be mainly the result of our MDL single model training which would not allow complex models to evolve for smaller sample sizes. Finally, we examined the benefits of using a variable memory approach to this problem. When all PST training is restricted to suffixes of length one at most, segmentation can still be obtained in the original setting, as Figure 5.9(c) attests. However, segment borders, as well as model confidence are inferior to those using unrestricted PSTs (compare to Figure 5.7). When PST depth is further restricted to depth zero (single root node), segmentation capability deteriorates dramatically. Figure 5.9(d) holds a minimal segmentation, of only two very distinct languages, English and Russian, at an increased segment size of 200. Thus, we conclude that the variable memory approach does play an important role in segmentation, and may play a more significant role in segmenting related protein sequences,

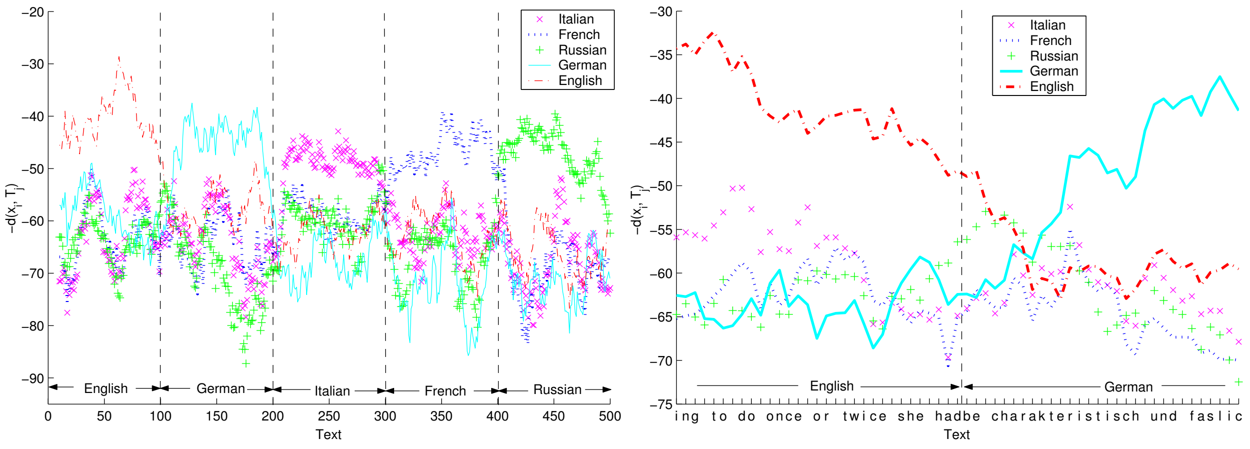


Figure 5.7: **Multilingual text segmentation**. (left) We plot $-d(x_i, T_j)$ for the first 500 letters of text, for the five models in $\mathcal{T} = \{T_{English}, T_{German}, T_{Italian}, T_{French}, T_{Russian}\}$, which are produced by a typical run of the segmentation algorithm (here $M = 10$ and $\beta = 0.6$). The true segmentation of the data appears above the x-axis. (right) A zoom in of the first transition point, where the x-axis corresponds to text letters $70-130$. Note the correspondence between the transition of prediction dominance from $T_{English}$ to T_{German} and the English-like beginning "*be chara*kteristisch..." of the German segment.

as discussed in Section 3.3.2.

5.5 Protein Domain Signatures

The input to the segmentation algorithm is a group of unaligned sequences in which we search for regions of one or more types of conserved statistics. The different training sets were constructed using the Pfam (release 5.4, Bateman et al., 2002) and Swissprot (release 38, Boeckmann et al.,

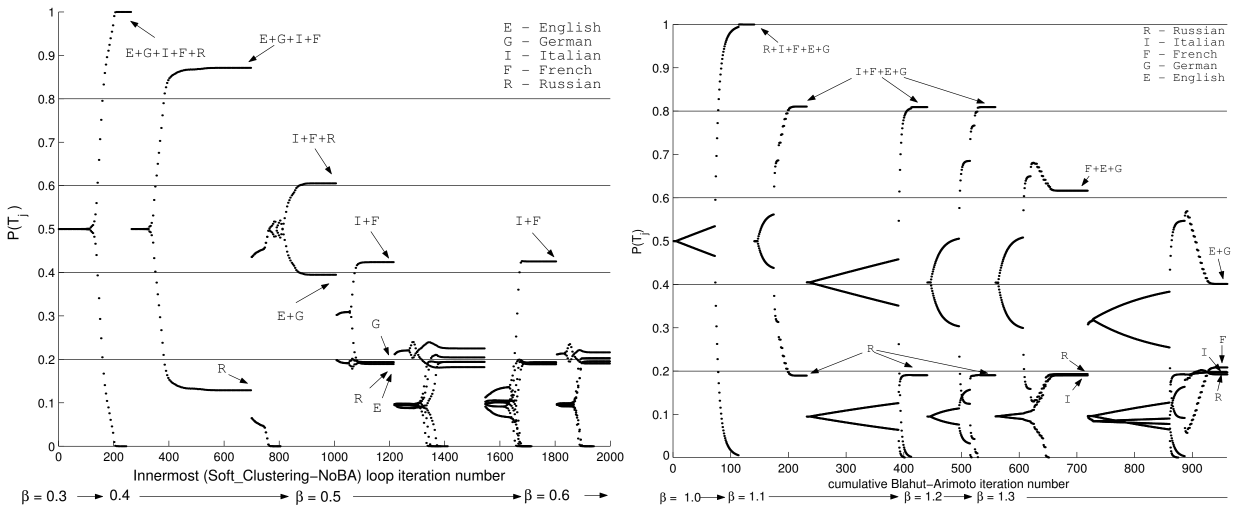


Figure 5.8: **Source separation**. The proportion of text assigned to each model is shown for all models, as a function of the iteration number of the innermost loop of the no-BA procedure (left) and the BA employing procedure (right). The bifurcations in both graphs follow Split_PSTs steps, while curves dropping off to zero show emptied models. We increment β after the number of models converges at a given distortion level.

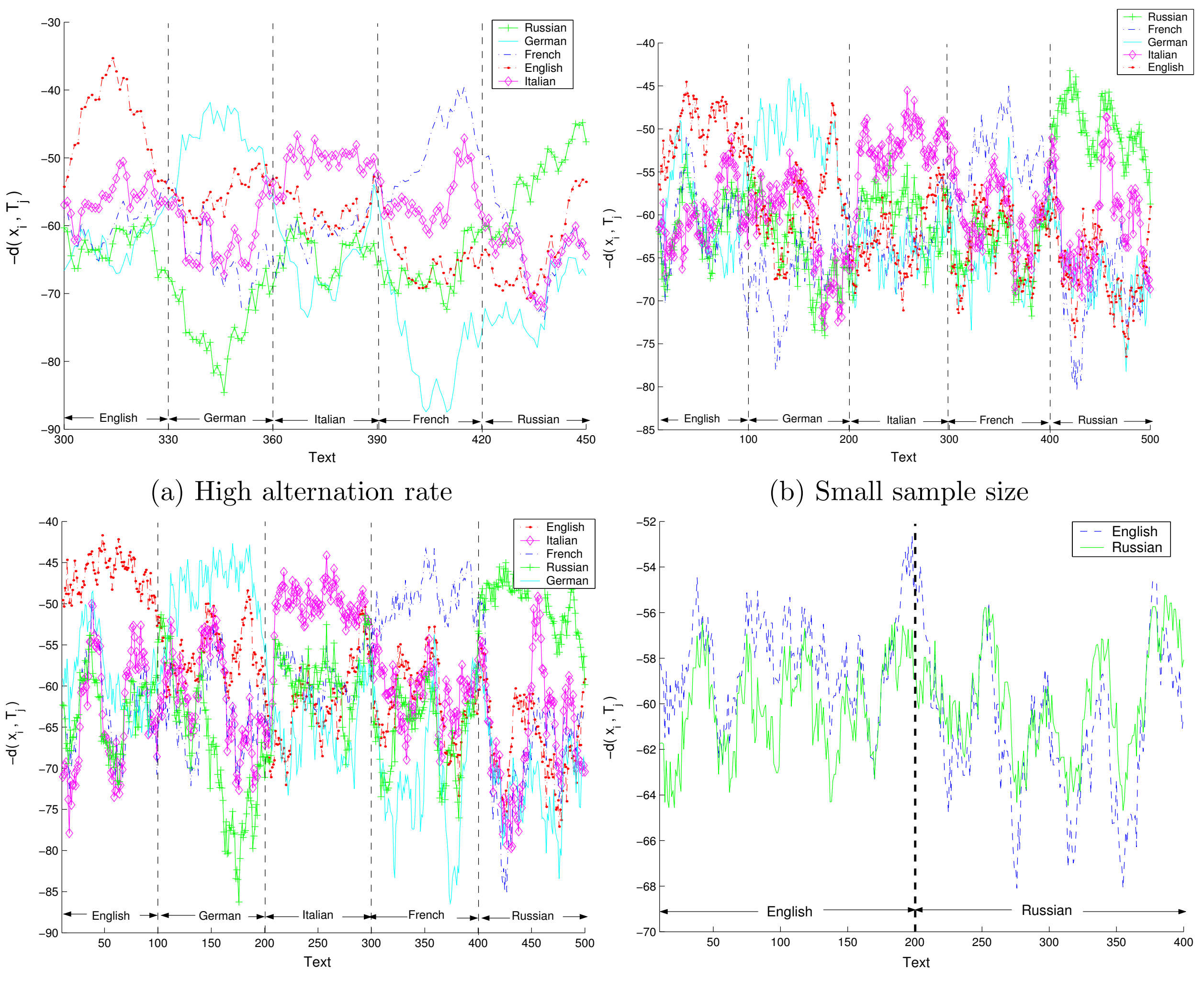


Figure 5.9: **Factors influencing segmentation success.** (a) An example of a successful segmentation at minimal per segment size of 30. (b) Successful segmentation at alternation rate of 100 letters, and minimal sample size of 60 segment per language. (c) Inferior segmentation obtained using only PSTs limited to depth 1 suffixes. (d) When PST depth is limited to root node only, even two distinct languages, alternating every 200 symbols can only be segmented at very low quality.

2003) databases. We briefly recall from Chapter 2 that Pfam contains various conserved sequence domain families. In each Pfam family all members share a domain. An HMM detector is built for that domain segment based on an MSA of a seed subset of family member domains. The HMM is then verified to detect that domain in the remaining known family members. Multi domain proteins therefore belong to as many Pfam families as there are different characterized domains within them. In order to build realistic, more heterogeneous sets, we collected from Swissprot the *complete sequences* of all chosen Pfam families. All members of each set thus contain a certain domain. Some of them may also contain various other domains in different combinations. We also experimented with sets which are the union of two or more Pfam families to examine the ability of our algorithm to detect and resolve the different repeating sources.

As we saw above, given such a set of unaligned sequences our algorithm returns as output several PST models. The number of models returned is determined by the algorithm itself. In practice two types of PSTs emerge for protein sequence data: Models that significantly outperform others on relatively short regions along the different sequences (and generally do poorly on most other regions). These we call detectors. The other type are models that perform averagely over all sequence regions. These are noise (or baseline) models and we can discard them automatically. As we shall next show, detector models often correspond to conserved domains, or parts of them, while the noise models capture the existing statistical correlations in the unstructured regions of different family members.

Three of the Pfam-based sets we ran experiments on will next be presented to demonstrate the ability of the resulting models to pin-point the prevalent domain/s, simultaneously detect others which appear less frequently in the data set, and even surpass supervised HMM detection capabilities in certain cases. Several independent runs of the (stochastic) segmentation algorithm, were carried out per family. For each family the different runs converged to the same stable segmentation. Each PST detector we present is run over the complete segmented data set in order to determine its nature. In the following we will present the segmentation of a single representative protein sequence out of each explored set. The algorithm itself is implemented in C++. On a Pentium III 600 MHz Linux machine clear segmentation was usually apparent within an hour or two of run time.

5.5.1 The Pax Family

Pax proteins (reviewed in Stuart et al., 1994) are eukaryotic transcriptional regulators that play critical roles in mammalian development and in oncogenesis. All of them contain a conserved domain of 128 amino acids called the paired or paired box domain (named after the *drosophila* paired gene which is a member of the family). Some contain an additional homeobox domain that succeeds the paired domain. Pfam nomenclature names the paired domain "PAX", after the Pax protein which is a family member.

The Pax proteins show a relatively high degree of sequence conservation. We attempted to segment a set of 116 unaligned, unannotated complete sequences of family members, as described above. In Figure 5.10 we superimpose the prediction of all resulting PST detectors over a representative family member. This Pax6 SS protein contains both the paired and homeobox domains. Both are shown to have matching signatures. Furthermore, in this set the obtained signatures exactly overlap the entire span of the two domains. Note that only half of the proteins contain the homeobox domain and yet its signature is very clear.

The stable break-up of the paired domain signature into three separate specialized models in Figure 5.10 merits further discussion. Through a literature search we discovered that crystallographic and biochemical studies have indicated that the paired domain is actually composed of two sub-domains separated by a short linker (e.g., Stuart et al., 1994; Xu et al., 1999). This segmentation is evident from the crystal structures of two of the family members, drosophila paired protein and the human Pax6 protein, whose 3D structures have been solved when bound to their DNA target sequences. While the crystal structure of the paired protein-DNA complex indicated that the N-terminal sub-domain is responsible for interactions with the DNA, in the Pax6-DNA complex both the N-terminal and the C-terminal sub-domains bound the DNA through the major groove by a helix-turn-helix motif and the linker contacted the DNA through the minor groove. Further scrutiny of our algorithm revealed, however, that in its current formulation, the break up of the paired box signature is a function of the window size parameter M of Equation 5.1. It appears that as we drive our algorithm to increasingly harden the clustering of the segments, less conserved

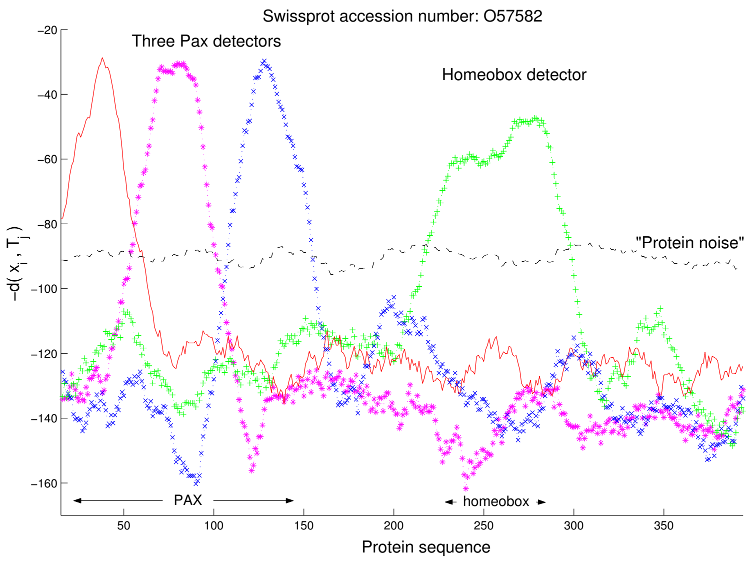


Figure 5.10: **Unaligned domain detection and segmentation.** We present the results of segmenting an unaligned set of Paired/PAX domain sequences, half of which also contain the homeobox domain. We superimpose the predictions of all four detector PSTs generated by the segmentation algorithm, and a baseline model (dashed), against the sequence of the PAX6 SS protein. Above the x-axis we denote in Pfam nomenclature the location of the experimentally verified paired box and homeobox domains. These are seen to be in near perfect match with the high scoring sequence segments.

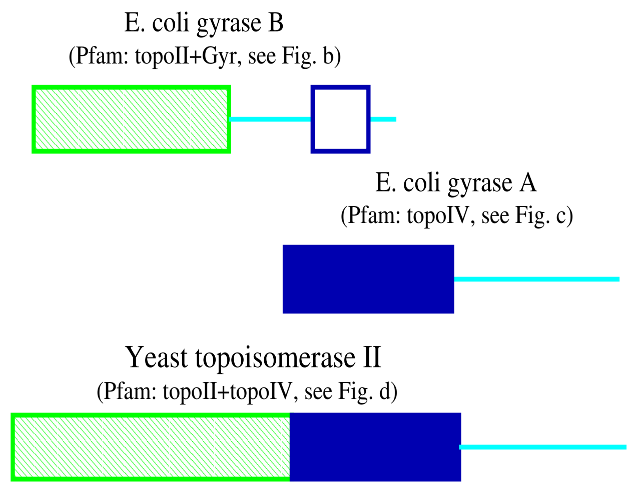
domains tend to break into succeeding segment models, of roughly equal length, proportional to the magnitude of M.

5.5.2 DNA Topoisomerase II

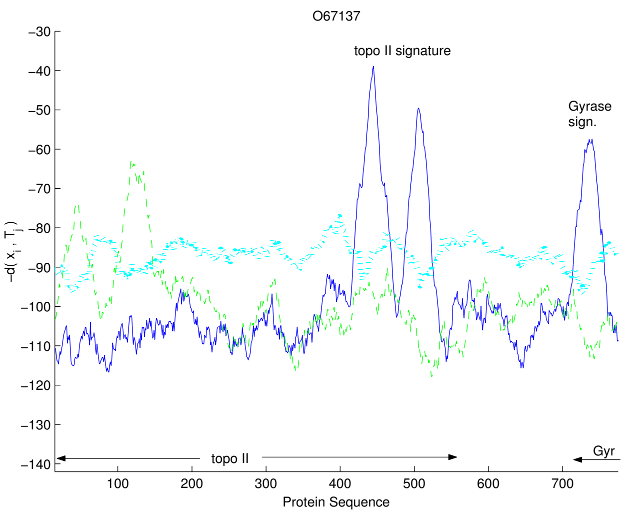
Type II DNA topoisomerases (reviewed in Roca, 1995) are essential and highly conserved in all living organisms. They catalyze the interconversion of topological isomers of DNA and are involved in a number of mechanisms, such as supercoiling and relaxation, knotting and unknotting, and catenation and decatenation. In prokaryotes the enzyme is represented by the *Escherichia coli* gyrase, which is encoded by two genes, gyrase A and gyrase B. The enzyme is a tetramer composed of two GyrA and two GyrB polypeptide chains. In eukaryotes the enzyme acts as a dimer, where in each monomer two distinct domains are observed. The N-terminal domain is similar in sequence to Gyrase B and the C-terminal domain is similar in sequence to GyraseA. We illustrated this relationship in Figure 5.11(a). In Pfam 5.4 terminology GyrB and the N-terminal domain belong to the "DNA_topoisoII" family (which has been sub-divided in later Pfam releases), while GyrA and the C-terminal domain belong to the "DNA_topoisoIV" family[1]. In the remainder we term the pairs GyrB/topoII and GyrA/topoIV, respectively.

For the analysis we used a group of 164 unaligned, unannotated sequences that included both eukaryotic topoisomerase II sequences and bacterial Gyrase A and B sequences. These were gathered from the union of the DNA_topoisoII and DNA_topoisoIV Pfam 5.4 families. As we next show, our algorithm successfully differentiates them into sub-classes. Figure 5.11(d) plots the segmentation of a representative of the eukaryotic topoisomerase II sequences. We see that the PSTs model the

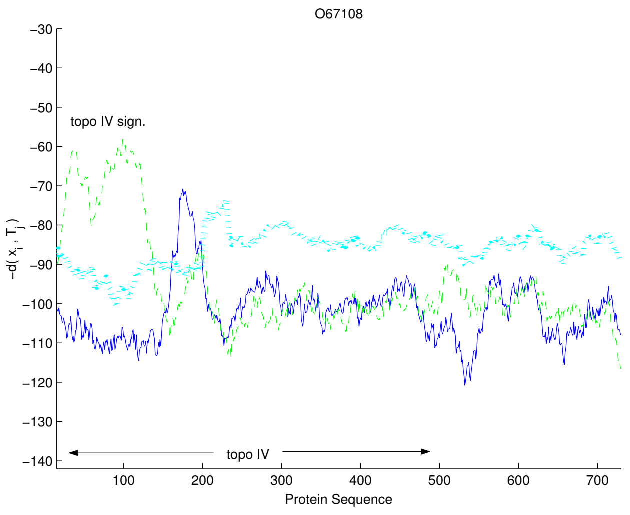
[1]The name should not be confused with the special type of topoisomerase II found in bacteria, that is also termed topoisomerase IV, and plays a role in chromosome segregation.


E. coli gyrase B
(Pfam: topoII+Gyr, see Fig. b)
E. coli gyrase A
(Pfam: topoIV, see Fig. c)
Yeast topoisomerase II
(Pfam: topoII+topoIV, see Fig. d)

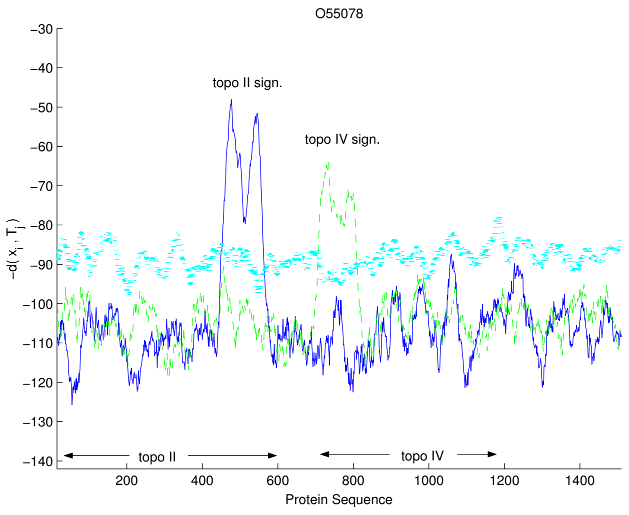
(a) Type II DNA topoisomerases



(b) Bacterial GyrB/topoII signature



(c) Bacterial GyrA/topoIV signature



(d) Eukaryotic Topoisomerase II signature

Figure 5.11: **Detection of a fusion event in unaligned sequences.** (a) The relationship between prokaryotic and eukaryotic Type II DNA topoisomerases is depicted schematically. (adapted from Marcotte et al., 1999) We see that a single gene in yeast codes for two domains which are homologous to those coded by two separate genes in *E. coli*. (b),(c) Typical PST segmentation signatures for the Gyrase B and A domains, respectively. (d) By comparison to the previous two figures, the signature of the eukaryotic Type II DNA topoisomerase is shown to be composed of the two prokaryotic signatures, in the right order and placing. Also note that the C-terminal signature of the Gyrase B protein is missing from its eukaryotic counter-part, as is the conserved region it models.

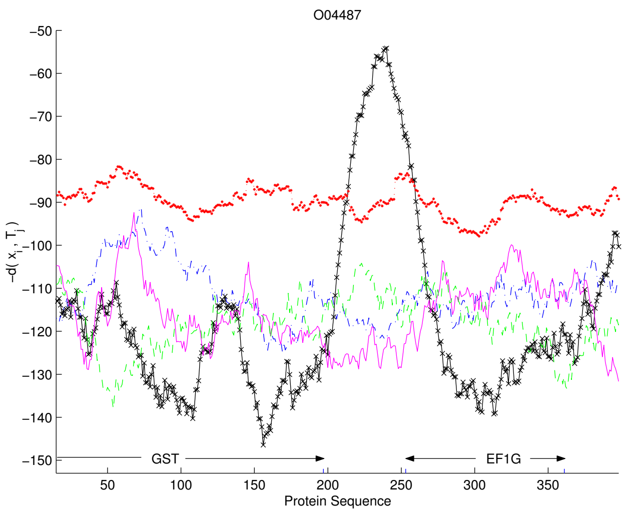
C-terminal part of the GyrB/topoII domain, and the N-terminal part of the GyrA/topoIV domain. Figure 5.11(b) and (c) demonstrate the results for representatives of the bacterial Gyrase B and Gyrase A proteins, respectively. As we denote above the graphs, the *same* two signatures are found in all three sequences, at the appropriate locations. Interestingly, in Figure 5.11(b) in addition to the signature of the GyrB/topoII domain another signature appears at the C-terminal region of the sequence. This signature is compatible with a known conserved region at the C-terminus of Gyrase B, that is involved in the interaction with the Gyrase A molecule. It also corresponds to the Pfam "DNA_gyraseB_C" family.

The relationship between the *E. coli* proteins GyrA and GyrB and the yeast topoisomerase II of Figure 5.11 provides a prototypical example of a fusion event of two proteins that form a complex in one organism into one protein that carries a similar function in another organism. Such examples have lead to the idea that identification of such similarities may suggest the relationship between the first two proteins, either by physical interaction or by their involvement in a common pathway (Marcotte et al., 1999; Enright et al., 1999). The computational scheme we present can be useful in searching for these relationships.

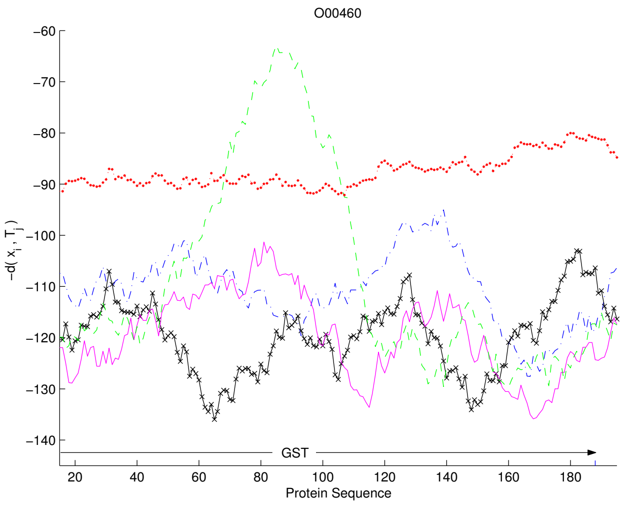
5.5.3 The Glutathione S-Transferases

The glutathione S-transferases (GST) represent a major group of detoxification enzymes (reviewed in Hayes and Pulford, 1995). There is evidence that the level of expression of GST is a crucial factor in determining the sensitivity of cells to a broad spectrum of toxic chemicals. All eukaryotic species possess multiple cytosolic GST isoenzymes, each of which displays distinct binding properties. A large number of cytosolic GST isoenzymes have been purified from rat and human organs and, on the basis of their sequences they have been clustered into five separate classes designated class alpha, mu, pi, sigma, and theta GST. The hypothesis that these classes represent separate families of GST is supported by the distinct structure of their genes and their chromosomal location. The class terminology is deliberately global, attempting to include as many GSTs as possible. However, it is possible that there are sub-classes that are specific to a given organism or a group of organisms. In those sub-classes the proteins may share more than 90% sequence identity, but these relationships are masked by their inclusion in the more global class. Also, the classification of a GST protein with weak similarity to one of these classes is sometimes a difficult task. In particular the definition of the sigma and theta classes is imprecise. Indeed the PRINTS database (Attwood et al., 2003) of discriminating signatures, reviewed in Chapter 2, only the first three classes, alpha, pi, and mu have been defined by distinct sequence signatures. For lack of sufficient similarity, Pfam models all these sequences using a single GST domain HMM.

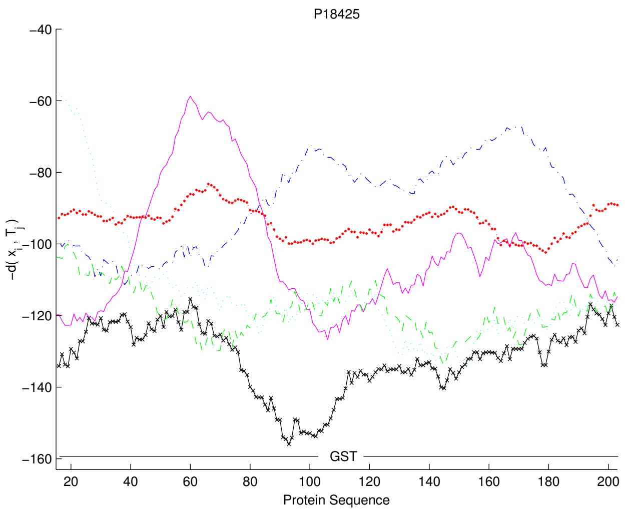
Our algorithm segmented 396 unaligned, unannotated Pfam family members into distinct signature groups: (1) A typical weak signature common to many GST proteins that contain no sub-class annotation. (2) A sharp peak after the end of the GST domain appearing exactly in all 12 out of 396 (3%) proteins where the elongation factor 1 gamma (EF1G) domain succeeds the GST domain. Interestingly, as shown in Figure 5.12(a) the captured region corresponds mostly to the linker region between the GST and EF1G domain, which upon inspection of the multiple alignment between members is also seen to be relatively conserved. (3) A clear signature common to almost all PRINTS annotated alpha and most pi GSTs, shown in Figure 5.12(b). (4) The theta and sigma classes are abundant in non-vertebrates. As more and more of these proteins are identified it is expected that additional classes will be defined. The first evidence for a separate sigma class was obtained by sequence alignments of S-crystallins from mollusc lens. Although these refractory proteins in the lens probably do not have a catalytic activity they show a degree of sequence similarity



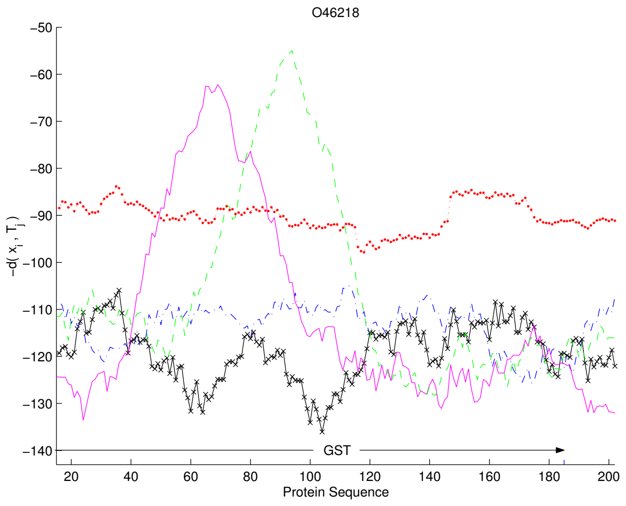
(a) GST + EF1G signature



(b) Joint α and π GST signature



(c) S-crystallin signature



(d) Putative θ GST signature

Figure 5.12: **Differentiating functional sub-types without alignment.** We show four of the five distinct signature patterns that arise when segmenting the GST domain sequences. Interestingly, the clear EF1G signature of (a) appears in only about 3% of the set, and the putative theta signature of (d) is seen to be composed of two detector models, each used by a different sub-class (b and c). These observations can help us analyze the phylogenetic relations behind the divergent sub-types we see nowadays.

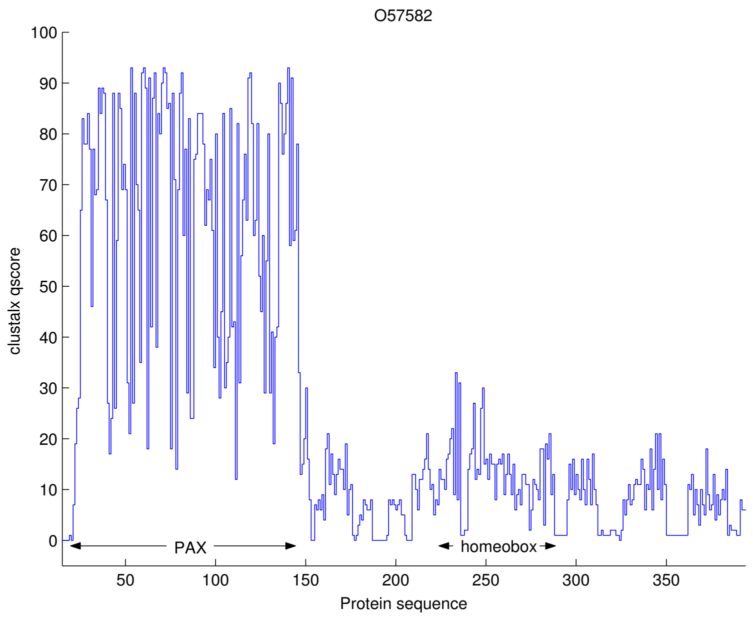


Figure 5.13: **Domain detection using automatically aligned sequences.** We plot the clustalX conservation score of the PAX6 SS protein against an MSA of all Pax proteins. While the predominant paired/PAX domain is discerned, the homeobox domain (appearing in about half the sequences) is lost in the background noise. Compare with Figure 5.10 where the same training set and plotted sequence are used.

to the GSTs that justifies their inclusion in this family and their classification as a separate class of sigma (Buetler and Eaton, 1992). This class, defined in PRINTS as S-crystallin, was almost entirely identified by the fourth distinct signature, shown in Figure 5.12(c) to be composed of prediction peaks of two adjacent detector models. (5) Interestingly, the last distinct signature, is composed of two detector models, one from each of the previous two signatures (alpha + pi and S-crystallin). Most of these two dozens proteins come from insects, and of these most are annotated to belong to the theta class. Note that many of the GSTs in insects are known to be only very distantly related to the five mammalian classes. Our segmentation pinpoints, without alignment, conserved regions shared by the differer classes. It does however fail to detect the mu class signature.

5.5.4 Comparative results

In order to evaluate our findings we have performed three unsupervised alignment driven experiments using the same sets described above: An MSA was computed for each set using clustalX (Linux version 1.81, Jeanmougin et al., 1998). We then let clustalX compare the level of conservation between individual sequences and the computed MSA profile in each set. Qualitatively these graphs resemble ours, apart from the fact that they do not offer separation into distinct models, which sometimes provides extra discerning power.

As expected this straightforward approach can be less powerful. The Pax alignment did not clearly elucidate the homeobox domain existing in about half the sequences. As a result, when we plot the graph comparing the same PAX6 SS protein we used in Figure 5.10 against the new MSA in Figure 5.13, the homeobox signal is lost in the noise. For type II topoisomerases the picture is slightly better. The Gyrase B C-terminus unit from Figure 5.11(b) can be discerned from the main unit, but with a much lower peak. However, the clear sum of two signatures we obtained for the eukaryotic sequences, as in Figure 5.11(d), is lost here. In the last and hardest case the MSA approach tells us nothing. All GST domain graphs look nearly identical precluding any possible subdivision. And the 12 (out of 396) instances of the EF1G domain are completely lost at the alignment phase.

5.5.5 Model Enhancements

In the three sets presented above, and others, our algorithm has achieved meaningful segmentation, pin-pointing whole or parts of conserved domains. These sets, however, are characterized by relatively high conservation of family members. However, many domains that are less conserved could not be detected using our method. To address this challenge we attempted to enhance our modeling scheme. We briefly describe enhancements which have improved the behaviour of the algorithm:

A fixed background model. As part of the initialization, the algorithm trains an additional order-1 model of the entire data set. Then, during the main loop, this model participates in data re-partitioning (steps 1,2 in Figure 5.6) but it is not re-trained (step 3 there). This fixed background model tends to repress the appearance of other noise models, and typically causes the algorithm to converge earlier on.

Wildcard PST nodes. Consider an inner node of a PST, labeled s. Its main use is in predicting contexts which extend beyond its depth, but are not modeled in the tree. In the notation of Section 5.2, $s \in T$, but its one letter extension $*s \notin T$. Instead of predicting the next letter σ using $P_s(\sigma) \equiv \frac{w_s(\sigma)}{w(s)}$, we model it using $P_{*s}(\sigma) \equiv \frac{w_{*s}(\sigma)}{w(*s)}$, where $w_{*s} = w_s(\sigma) - \Sigma_{\sigma' s \in T} w_{\sigma' s}(\sigma)$, and $w(*s)$ is defined analogously. Thus, we predict the next symbol based not on the entire statistics of node s, but only on those contexts which are not refined by son nodes, and thus predicted elsewhere. By revising the MDL expressions of Section 5.2 accordingly, we obtain more precise stochastic modeling of the sequence segments.

Bi-directional modeling. Increasing space and time demands by a factor of two, we replace every PST with a pair. One models the sequences in the N- to C-terminal direction, as before, and the other models them in the reverse direction. During data partitioning, the distortion between such a pair and a sequence symbol (Equation 5.1) is now a cumulative sum of the two respective distortions. During model re-training they use the same resulting weight vector. As PST modeling focuses on prediction given the sequence past, bi-directional modeling allows us to capture more sequence correlations at a low computational cost.

While improving the stability and performance of our algorithm, room for improvement remain, especially for highly divergent domain families. We discuss further directions for enhancement in the next section.

5.6 Discussion

The sequence segmentation algorithm we describe and evaluate in this chapter is a combination of several different information theoretic ideas and principles, naturally combined into one new coherent procedure. The core algorithm, the construction of Prediction Suffix Trees, is essentially a source coding loss-less compression method. It approximates a complex stochastic sequence by a probabilistic automaton, or a Markov model with variable memory length. The power of this procedure, as demonstrated on both natural texts and on protein sequences, is in its ability to capture short strings (suffixes) that are significant predictors, and thus good features, for the statistical source. We combined the PST construction with another information theoretic idea, the MDL principle, to obtain a more efficient estimation of the PST, compared with its original learning algorithm.

Our second key idea is to embed the PST construction in a lossy compression framework by adopting the rate-distortion theory into a learning procedure. Here we treat the PST as a model of a single statistical source and use the rate distortion framework (i.e., the Blahut-Arimoto algorithm) to partition the sequences between several such models in an optimal way. By doing so we specifically obtain a more expressive statistical model, as mixtures of short memory, ergodic Markov models lay outside of this class, and can be captured only by much deeper Markov models. This is a clear advantage of our current approach over mixtures of HMMs (as in Fine et al., 1998) since mixtures of HMMs are but HMMs with constrained state topology.

The analogy with rate-distortion theory enables us to take advantage of the trade-off between compression (rate) and distortion, and use the Lagrange multiplier β, required to implement this trade-off, as a resolution parameter. The deterministic annealing framework follows naturally in this formulation and provides us with a simple way to obtain segmentation of very complex sequences, such as protein sequences.

From a biological perspective the method has several advantages: It is fully automated; it does not require or attempt an MSA of the input sequences, which may very well not be amenable to a particular linear ordering; it handles heterogeneous groups well and locates domains appearing only a few times in the data; by nature it is not confused by different module orderings within the input sequences; it appears to seldom generate false positives; and it may even surpass HMM clustering in instances of very similar separation to sub-types. However, as we saw, it is hard to separate and segment highly similar statistical sources, domains with few occurrences in the set or those which are very short, or very divergent. Rather than compare our algorithm to alignment-based models, it would be interesting to use it in conjuncture with such methods. The segmentation can first be applied to separate heterogeneous groups of proteins into groups sharing similarities. Those groups can then be profiled by HMMs or similar tools, using our signatures as guides to the alignment and domain boundaries.

Algorithmically, several natural extensions of our ideas are possible. We may attempt to replace the window parameter M, which is currently calibrated manually, with an explicit HMM of PSTs framework, where transitions between the different PST models are estimated from the data. One may replace the PST with even more efficient loss-less coding schemes, such as the Context-Tree-Weighting algorithm (Willems et al., 1995). This powerful method trades the MDL principle for a Bayesian formulation, essentially averaging efficiently over all possible PSTs for a given sequence. The disadvantages are the much larger data structures required, and the absence of the clear features that emerge from the VMM model. Instead of enhancing our generative modeling ability we may benefit by replacing these with discriminative ones, aimed not so much at capturing the statistical richness in each source but rather at differentiating between them. This approach will be explored in the next chapter.

Algorithmic optimization, similar to that performed in Chapter 4 for a single model, may allow us to consider running the tool on much larger sets, such as whole proteomes, or even the entire set of known proteins. Currently, scaling of the convergence run time with data set size prohibits these experiments. From a theoretical point of view, a proof of convergence to a stable segmentation, is still missing. In fact, the soft clustering procedure sometimes does enter oscillations around the point of convergence, when small amounts of data pass back and forth between two or more models. In itself, this situation is easy to automatically detect and resolve. Perhaps it can be resolved algorithmically in a manner that facilitates a proof of convergence.

This chapter is based on an extended abstract (Seldin et al., 2001) which was presented at ICML 2001, and was later developed to a journal version (Bejerano et al., 2001) and presented at the In Silico Biology 2001 conference. This work was a co-winner of the best poster award at AIBS 2002.

Chapter 6

Discriminative Markovian Protein Analysis

Throughout the previous chapters we have used the probabilistic suffix tree as a generative model, aiming to capture as many statistical features of the underlying source as possible. This chapter introduces discriminative variable memory modeling, which focuses only on source discriminating features. This novel approach is shown to obtain more compact and accurate models which pinpoint protein residues of potential functional significance.

6.1 Introduction

Feature selection is one of the fundamental problems in pattern recognition and machine learning. In this approach, one wishes to sort all possible features using some predefined criteria and select only those that are most appropriate for the task at hand. It thus may be possible to significantly reduce model dimensionality without impeding the performance of the learning algorithm. In some cases one may even gain in generalization power by filtering irrelevant features (cf. Almuallim and Dietterich, 1991). The need for a good feature selection technique also stems from the practical concern that estimating the joint distribution between the different classes and the feature vectors when either the dimensionality of the feature space or the number of classes is very large, requires unrealistically large training sets. Indeed, increasing the number of features while keeping the number of samples fixed can actually lead to a decrease in the accuracy of the classifier (Hughes, 1968; Baum and Haussler, 1989).

In this thesis we explore the power of variable memory modeling (VMM) in the context of protein sequence analysis. When modeling a single protein family using a probabilistic suffix tree (PST), in Chapter 3, our main concerns in terms of feature selection were the frequency of the feature under consideration in the training set, and its predictive novelty with respect to less complex features which may take its place (shorter suffixes). In Chapter 5, these two criteria were accounted for simultaneously using the minimum description length principle. However, in both cases our use of the PST model was for generative purposes. Model training was aimed at producing probabilistic models which generate training set-like sequences with high probability. In particular, we aimed at collecting all significant features of a class of short term context dependencies from the training set. The ultimate goal of classification, segmentation or discrimination was achieved based on generative PST predictions. It was never explicitly taken into account during model growing step.

As reviewed in Section 2.4, generative modeling of sequences is often not sensitive enough to differentiate between protein sub-families. This is due to the high sequence similarities between all members of a given family, and the fact that among all residue variations between members only specific residue changes confer the sought functionality variation.

Goal. Given a set of unaligned sequences, labeled as belonging to different families or sub-families of proteins, devise a sensitive classification scheme of novel sequences into the given sub-families, which will also highlight highly discriminating residues, of potential functional importance.

In this chapter we address this challenge by defining a novel discriminative framework for PST modeling. First, we extend the generative PST modeling technique to handle several sources simultaneously. Next, we prune from the resulting model all features which have low discriminating power between the different classes. Rather than obtaining several separate PSTs focusing on generative modeling of highly similar sources, we obtain a much smaller combined model which focuses on detection of discriminating features between the different classes. We term this approach **discriminative variable memory modeling** (DVMM).

Our feature selection, or pruning scheme is based on maximizing **conditional mutual information**. More precisely, for any subsequence s we estimate the information between the next symbol in the sequence and each statistical source $c \in C$, given that subsequence, or suffix s. We use this estimate as a new measure for pruning less discriminative features out of the model. This yields a criterion which is very different from the one used by the generative VMM model of Chapter 3-5. In particular, many features may be important for good modeling of each source independently although they provide minor discrimination power. These features are pruned in the DVMM framework, resulting in a much more compact model while attaining high classification accuracy. We further suggest a natural sorting of the features retained in the DVMM model. This allows an examination of the most discriminating features, which may help us gain insight into the nature of the grouping.

6.2 Feature Selection

The use of mutual information (MI) for feature selection is well established in the machine learning literature, although it is usually suggested there in the context of deterministic rather than stochastic modeling. The original idea may be traced back to Lewis (1962). The approach is motivated by the fact that when the a-priory class uncertainty $P(C)$ is given, maximization of the mutual information $I(C;X)$ is equivalent to the minimization of the conditional entropy $H(C|X)$. This in turn links mutual information maximization and the decrease in classification error P_{err}, through the following two inequalities (e.g., Goodman and Smyth, 1988)

$$H\left(P_{err}\right) + P_{err}\log\left(C-1\right) \geq H\left(C|X\right) \geq 2P_{err} \tag{6.1}$$

where $H\left(\cdot\right) = -\sum P\left(\cdot\right)\log P\left(\cdot\right)$ and $H\left(\cdot|\cdot\right) = -\sum P\left(\cdot,\cdot\right)\log P\left(\cdot|\cdot\right)$ are the entropy and the conditional entropy, respectively.

Since then, a number of methods have been posed, differing essentially in their method of approximating the joint and marginal distributions, and their direct usage of the mutual information measure (cf. Battiti, 1994; Barrows and Sciortino, 1996; Yang and Moody, 1999). One of the difficulties in applying MI based feature selection methods, is the fact that evaluating the MI measure involves integrating over a dense set, which leads to a computational overload. To circumvent that, Torkkola and Campbell (2000) have recently suggested to perform feature transformation (rather than feature selection) to a lower dimension space in which the training and analysis of the data is more feasible. Their method is designed to find a linear transformation in the feature space that will maximize the mutual information between the transformed data and their class labels, and the reduction in computational load is achieved by the use of Renyi's entropy based definition of mutual information (Cover and Thomas, 1991) which is much more easy to evaluate.

Out of numerous feature selection techniques found in the literature, we also point out the work of Della Pietra et al. (1997) who devised a feature selection (or rather, induction) mechanism to build n-grams of varying lengths, and the "U-Tree" of McCallum (1997) which builds PSTs based on the ability to predict the future discounted reward in the context of reinforcement learning.

Another popular approach in language modeling is the use of pruning as a mean for parameter selection from a higher-order n-gram backoff model. The backoff model recursive rule (Chen and Goodman, 1998) represents n-gram conditional probabilities $P(w_n|w_1 \ldots w_{n-1})$ using $(n-1)$-gram conditional probabilities multiplied by a backoff weight, $\alpha(w_1 \ldots w_{n-1})$, associated with the full history. Thus, $P(w_n|w_1 \ldots w_{n-1}) = \alpha(w_1 \ldots w_{n-1})P(w_n|w_2 \ldots w_{n-1})$, where α is selected such that $\sum P(w_n|w_1 \ldots w_{n-1}) = 1$. One successful pruning criterion, suggested by Stolcke (1998), minimizes the divergence (measured using relative entropy) between the distributions embodied by the original and the pruned models. By relating relative entropy to the relative change in training set perplexity, or average branching factor of the language model, a simple pruning criterion is devised, which removes from the model all n-grams that change perplexity by less than a threshold. Stolcke (1998) shows that in practice this criterion yields a significant reduction in model size without increasing classification error. More recently, Kermorvant and Dupont (2002) have used a different backoff pruning criterion to obtain models which are smaller than their equivalent generative PSTs and as accurate.

A selection criterion, similar to the one we propose here, was suggested by Goodman and Smyth (1988) for decision tree design. Their approach chooses the highest scoring feature at any node in the tree, conditioned on the features previously chosen, and the outcome of evaluating those features. Thus, they suggested a top-down algorithm based on greedy selection of the most informative features. Their algorithm is equivalent to the Shannon-Fano prefix coding (Cover and Thomas, 1991), and can also be related to communication problems in noisy channels with side information. For feature selection, Goodman and Smyth (1988) noted that with the assumption that all features are known a-priori, the decision tree design algorithm will choose the most relevant features for the classification task, and ignore irrelevant ones. Thus, the tree itself yields valuable information on the relative importance of the various features.

A related use of MI for stochastic modeling is the maximal mutual information (MMI) approach for multi-class model training. This is a discriminative training approach attributed to Bahl et al. (1986), designed to directly approximate the posterior probability distribution, in contrast to the indirect approach, via Bayes' formula, and maximum likelihood (ML) training. The MMI method was applied successfully to HMM training in speech applications (e.g., Normandin et al., 1994; Woodland and Povey, 2000). However, MMI training is significantly more expensive than ML training. Unlike ML training, in this approach all models affect the training of every single model through the denominator. In fact this is but one reason why the MMI method is considered to be more complex. Another reason is that there are no known easy re-estimation formulas (as in ML). Thus one needs to resort to general purpose optimization techniques.

Our approach stems from a similar motivation but it simplifies matters: we begin with a simultaneous ML training for all classes and then select features that maximize the discriminative objective function. While we do not directly maximize the mutual information, we provide a practical approximation which is far less computationally demanding.

6.3 Theory

Consider the classification problem to a set of categories $C = \{c_1, c_2, \ldots, c_{|C|}\}$. The training data consists of a set of labeled examples for each class. Each sample is a sequence of symbols over some alphabet Σ. A Bayesian learning framework trains generative models to produce good estimates of class conditioned probabilities. Upon receiving a new test sample d, the generative models are employed to yield a maximum aposteriori decision rule:

$$\max_{c \in C} P(c|d) \propto \max_{c \in C} P(d|c)P(c), \quad d \in \Sigma^*$$

Thus, good estimates of $P(D|C)$ are essential for accurate classification.

6.3.1 The Generative Approach

Let $d = \sigma_1\sigma_2, \ldots \sigma_{|d|}$, $\sigma_i \in \Sigma$, and let $s_i \in \Sigma^{i-1}$ denote the subsequence of symbols preceding σ_i, then

$$P(d|c) = \Pi_{i=1}^{|d|} P(\sigma_i|\sigma_1\sigma_2 \ldots \sigma_{i-1}, c) = \Pi_{i=1}^{|d|} P(\sigma_i|s_i, c) \tag{6.2}$$

Denoting by suff(s_i) the longest suffix of s_i, we recall from Section 3.1 that if $P(\sigma|s_i) = P(\sigma|\text{suff}(s_i))$, for every $\sigma \in \Sigma$, then predicting the next symbol using s_i is equivalent to a prediction using the shorter context given by suff(s_i). Thus, in this case it is clear that keeping only suff(s_i) in the model should suffice for the prediction.

The PST training algorithm of Figure 3.2 (page 39) aims at building a model which will hold only a minimal set of relevant suffixes. To this end, a PST $\hat{T}$ is built in two steps: First, only suffixes $s \in \Sigma^*$ for which the empirical probability in the training data, $\hat{P}(s)$, is non-negligible, are kept in the model. Thus, rare suffixes are ignored. Next, all suffixes that are not informative for predicting the next symbol are pruned out of the model. Specifically, this is done by thresholding $r \equiv \frac{P(\sigma|s)}{P(\sigma|\text{suff}(s))}$. If $r \approx 1$ for all $\sigma \in \Sigma$, then predicting the next symbol using suff(s) is nearly identical to using s. In such cases s will be pruned out of the model.

The PST algorithm is designed to statistically approximate a single source. A straightforward extension to handle multiclass categorization tasks would build a separate PST for each class, based solely on its own data, and would classify a new example to the model with the highest likelihood score. This so called one-against-all approach, was used in Chapter 3. Motivated by a generative goal, this approach disregards the possible similarities and dissimilarities between the different categories. Each model aims at best approximating its assigned source. However, in a discriminative framework these interactions may be exploited to our benefit.

6.3.2 Multiclass Discriminative PST

Assume, for example, that for some suffix s and every symbol $\sigma \in \Sigma$, $\hat{P}(\sigma|s,c) = \hat{P}(\sigma|s) \;\; \forall c \in C$, i.e., the symbols and the categories are independent given s. Since we are now only interested in the relative magnitude of the posteriors $\hat{P}(c|s)$, these terms may as well be neglected. In other words, preserving s in the model will yield no contribution to the classification task, since this suffix has no discrimination power with respect to the given categories.

We turn to generalize and quantify this intuition. In general, two random variables are independent iff the mutual information between them is zero (cf. Cover and Thomas, 1991). For every $s \in \Sigma^*$ we consider the following (local) conditional mutual information,

$$I_s \equiv I(\Sigma; C|s) = \sum_{c \in C} \hat{P}(c|s) \sum_{\sigma \in \Sigma} \hat{P}(\sigma|c,s) \log \frac{\hat{P}(\sigma|c,s)}{\hat{P}(\sigma|s)}$$

```
First step - tree growing:
    1. Initialize $\hat{T}$ to include the empty suffix $e$, $\hat{P}(e) = 1$.
    2. For $l = 1 \dots L$
        (a) For every $s_l \in \Sigma^l$, where $s_l = \sigma_1\sigma_2 \dots \sigma_l$,
            estimate $\hat{P}(s_l|c) = \Pi_{i=1}^{l} \hat{P}(\sigma_i|\sigma_1 \dots \sigma_{i-1}, c)$.
            i. If $\hat{P}(s_l|c) \geq \varepsilon_1$, for some $c \in C$, add $s_l$ into $\hat{T}$.
Second step - tree pruning:
    1. Estimate for all $s \in \hat{T}$,
       $I_s = \sum_{c \in C} \hat{P}(c|s) \sum_{\sigma \in \Sigma} \hat{P}(\sigma|s,c) \log(\frac{\hat{P}(\sigma|s,c)}{\hat{P}(\sigma|s)})$.
    2. For $l = L \dots 1$
        (a) Define $\hat{T}_l \equiv \Sigma^l \cap \hat{T}$
        (b) For every $s_l \in \hat{T}_l$,
            i. Let $\hat{T}_{s_l}$ be the subtree spanned by $s_l$
            ii. Define $\bar{I}_{s_l} = max_{s' \in \hat{T}_{s_l}} I_{s'}$
            iii. If $\bar{I}_{s_l} - I_{suff(s_l)} \leq \varepsilon_2$, prune $s_l$.
```

Figure 6.1: **The discriminative PST training algorithm.**

where $\hat{P}(c|s)$ is estimated using Bayes formula, $\hat{P}(c|s) = \hat{P}(s|c)\hat{P}(c)/\hat{P}(s)$; the prior $\hat{P}(c)$ can be estimated by the relative number of training examples labeled with category c, or from domain knowledge; and $\hat{P}(s) = \sum_{c \in C} \hat{P}(c)\hat{P}(s|c)$. If $I_s = 0$, s can surely be pruned, as above. However, we may define a stronger pruning criterion, which considers also the suffix of s. Specifically, if $I_s - I_{\text{suff}(s)} \leq \varepsilon_2$, where ε_2 is some threshold, one may prune s and settle for the shorter memory suff(s). In other words, this criterion implies that suff(s) effectively induces more dependency between Σ and C than its extension s. Thus, preserving suff(s) in the model should suffice for the classification task. While adding conditioning in general reduces entropy, and therefore increases MI, the individual terms of the MI summation may still exhibit an opposite relation (cf. Cover and Thomas, 1991).

Finally, note that as in the case of the original PST discussed in Section 4.2.2, the pruning criterion defined above is non-monotone. Thus, it is possible to get $I_{s_1} > I_{s_2} < I_{s_3}$ for $s_3 = \text{suff}(s_2) = \text{suff}(\text{suff}(s_1))$. In this case we may be tempted to prune the suffix s_2 along with its child, s_1, despite the fact that $I_{s_1} > I_{s_3}$. To avoid that, we define the pruning criterion more carefully. We denote by $\hat{T}_s$ the sub-tree spanned by s, i.e., all nodes in $\hat{T}_s$ that correspond to sub-sequences with the same suffix, s. We can now calculate $\bar{I}_s = max_{s' \in \hat{T}_s} I_{s'}$, and define the pruning criterion using $\bar{I}_s - I_{\text{suff}(s)} \leq \varepsilon_2$. Therefore, we prune s (along with all its descendants), only if there is no descendant of s (including s itself) that induces more information (up to ε_2) between Σ and C, compared to suff(s), the parent of s. The pseudo-code of the resulting algorithm is given in Figure 6.1.

Given a novel sequence d, prediction scores are obtained for each class c from Equation 6.2, using the longest suffixes remaining after the pruning phase.

6.3.3 Sorting the Discriminative Features

As we shall show below, the above procedure yields a rather compact discriminative model between several statistical sources. Naturally not all its features have the same discriminative power. We denote the information content of a feature by

$$I_{\sigma|s} \equiv \sum_{c \in C} \hat{P}(c|s)\hat{P}(\sigma|s,c) \log(\frac{\hat{P}(\sigma|s,c)}{\hat{P}(\sigma|s)})$$

Note that $I_s = \sum_{\sigma\in\Sigma} I_{\sigma|s}$, thus $I_{\sigma|s}$ is simply the contribution of σ to I_s. If $\hat{P}(\sigma|s,C) \approx \hat{P}(\sigma|s)$, meaning σ and C are almost independent given s, then $I_{\sigma|s}$ will be relatively small, and vice versa.

This criterion can be applied to sort all the DVMM features. Still, it might be that $I_{\sigma_1|s_1} = I_{\sigma_2|s_2}$, while $\hat{P}(s_1) \gg \hat{P}(s_2)$. Clearly in this case one should prefer the first feature, $\{s_1 \cdot \sigma_1\}$, since the probability to encounter it is higher. Therefore, we should balance between $I_{\sigma|s}$ and $\hat{P}(s)$ when sorting. Specifically, we score each feature by multiplying the two factors $\hat{P}(s)I_{\sigma|s}$, and sorting the resulting scores in decreasing order.

The pruning and sorting schemes above are based on *local* conditional mutual information values. Let us review the process from a global standpoint. The *global* conditional mutual information is given by (e.g. Cover and Thomas, 1991)

$$I(\Sigma;C|S) = \sum_{s\in\Sigma^*} \hat{P}(s)I(\Sigma;C|s) = \sum_{s\in\Sigma^*} \hat{P}(s)I_s = \sum_{s\in\Sigma^*}\sum_{\sigma\in\Sigma} \hat{P}(s)I_{\sigma|s}$$

In the DVMM training algorithm (Figure 6.1) we first neglect all suffixes with a relatively small prior $\hat{P}(s)$. Then we prune all suffixes s for which $\bar{I}_s$ is small with respect to $I_{\text{suff}(s)}$. Finally, we sort all remaining features by their contribution to the global conditional mutual information, given by $\hat{P}(s)I_{\sigma|s}$. Thus, we aim for a compact model that still strives to maximize $I(\Sigma;C|S)$.

Expressing the conditional mutual information as the difference between two conditional entropies, $I(\Sigma;C|S) = H(C|S) - H(C|S,\Sigma)$, we see that maximizing $I(\Sigma;C|S)$ is equivalent to minimizing $H(C|\Sigma,S)$. In other words, our procedure effectively tries to minimize the entropy, i.e., the uncertainty, over the category identity C given the new symbol Σ and the suffix S, which in turn decreases the classification error (Equation 6.1).

6.4 Results

To allow an evaluation of our method we present a comparative analysis over several representative data sets.

6.4.1 Experimental Design

For every dataset we will compare the DVMM framework with two different, albeit related, algorithms. A natural comparison is of course with the original generative VMM (GVMM) modeling approach (Section 6.3.1). We build $|C|$ different generative models, one per class. A new example is then classified into the most probable class using these models.

We further compared our results to A. Stolcke's perplexity pruning SRILM language modeling toolkit (Stolcke, 2002, discussed in Section 6.2). Here, again, $|C|$ generative models are trained and classification is to the most probable class. Since the SRILM toolkit is limited to hexagrams, we bounded the maximal depth of the PSTs (for both DVMM and GVMM) to the equivalent suffix length of five. For all three models, in the first step of ignoring small $\hat{P}(s)$, we neglected all suffixes appearing infrequently in the training sequences. In principle, these two parameters can be fine tuned for a specific data set using standard methods, such as cross validation.

For pruning purposes we vary the analogous local decision threshold parameter in all three methods to obtain different model sizes. These are ε_2, r, and the perplexity threshold for DVMM, GVMM and SRILM respectively. In order to compute model sizes we sum the number of class specific features ($s \cdot \sigma$ combinations) in each model. For example, for the DVMM this will be the number of retained nodes multiplied by $|\Sigma||C|$.

Finally, there is the issue of smoothing zero probabilities. Quite a few smoothing techniques exist, some widely used by language modeling researchers (see Chen and Goodman, 1998 for a survey). Most of these incorporate two basic ideas: Modifying the true counts of the n-grams to pseudo counts (which estimate expected rather than observed counts), and interpolating higher-order with lower-order n-gram models to compensate for under-sampling. For SRILM we use absolute-discounting (see Chen and Goodman, 1998). The GVMM uses proportional smoothing (Figure 3.2). For the DVMM, we follow Krichevsky and Trofimov (1981) as in Chapter 5, and add a pseudo count of 0.5 to each possible suffix-symbol combination. We note that in the protein realm (as well as in the textual example we give below) the alphabet size is fairly small, below thirty symbols. Arguably, this implies that sophisticated smoothing is less needed here, compared to large vocabularies of up to 10^5 symbols (words).

6.4.2 Textual Calibration Test

We begin by examining the behaviour of the DVMM algorithm in a standard text classification task. In this experiment we set Σ to be the set of characters present in the documents. Our pre-processing included lowering upper case characters and ignoring all non alpha-numeric characters. Obviously, this representation ignores the special role of the blank character as a separator between the different words. On the other hand, it seems more realistic in light of our ultimate protein analysis goal, where no clear parsing is apparent.

For data set we chose the Reuters-21578 collection, which is a widely used categorization benchmark in the text mining community.[1] This data set contains short items which appeared on the Reuters newswire in 1987. Item length range is comparable to that of individual protein sequences. Each of these items is indexed for topic, location and additional indices. For comparison purposes we chose to focus on the modified Apte (ModApte) sub-set of the database. From it we chose all items relating to the ten most frequent topic categories, resulting in a training set of 7194 documents and a test set of 2788 documents. We note that about 9% of these documents are multi-labeled while our implementation induces uni-labeled classification (where each document is classified only to its most probable class).

We randomly chose half of the sequences as the training set and used the other half as test set. This process was repeated ten times, using different splits. In each iteration we used the training set to build three types of models for the ten topics, and then used these to classify the test set items into the most probable topic for each model type. Reported results are an average over these ten runs. In all runs, for all three models, we neglected all suffixes appearing less than 50 times in the training sequences.

To compare our text classification results to those in the text mining literature we introduce several measures of classification success, based on the terminology introduced in Section 2.2.1 (page 12). Precision is defined as the number of items correctly assigned to a class divided by

[1]The Reuters-21578 collection is freely available through the UCI KDD repository at http://kdd.ics.uci.edu/.

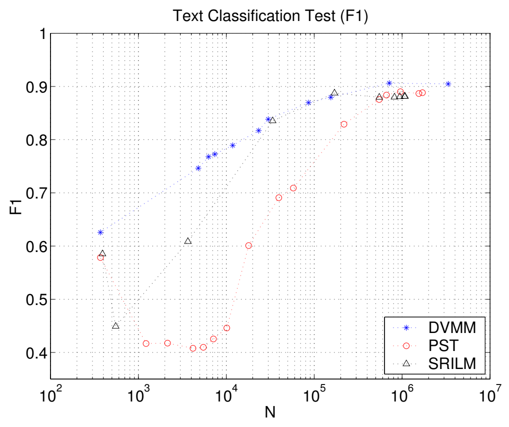


Figure 6.2: **Classification accuracy comparison on a textual data set.** The micro-averaged F1 statistic, combining recall and precision scores, is plotted against model size, measured in number of class specific features. All three methods are shown to ultimately achieve similar performance, with the DVMM performing slightly better. However, the DVMM performs much better when model size is limited. Interestingly, both generative models show an intermediate regime where model size increase results in classification decrease.

the total number of items assigned to it, or $\frac{TP}{TP+FP}$ (see Figure 2.2). Recall is a synonym to the sensitivity measure we have used in previous chapters, and is defined as the number of items correctly assigned to a class divided by the total number of items in that class, or $\frac{TP}{TP+FN}$. When measuring multi-classification success each of these two measures can be macro or micro-averaged. A macro average computes the statistic for each category separately and then takes an (unweighted) average over categorical results. A micro average computes the value of the statistic over all documents. Obviously we wish to maximize both the recall and precision of our algorithm. These goals can be combined by defining the $F1$ statistic as the harmonic average between recall r and precision p, or $\frac{2pr}{p+r}$. This statistic always ranges between zero and one, the latter being the optimal score.

In Figure 6.2 we present the micro-averaged $F1$ results for different model sizes for the three algorithms. The DVMM results are consistently comparable or superior to the other algorithms. The two generative modeling approaches are shown to suffer a decrease in performance at intermediate model sizes, probably due to modeling features which are common to all classes, reflecting English language statistics rather than class specific ones.

At its extreme, minimally pruned model, the micro-averaged precision and recall of the DVMM are 95% and 87%, respectively. This difference is to be expected as we are unilabeling a multi-labeled set, lowering our recall potential. Another way of combining these two results interpolates a break even point at which the two measures would be equal. In our case the break-even performance is at least 87% (probably higher). We compare these results with the break-even performance reported by Dumais et al. (1998) for the same task. In that work the authors compared five different classification algorithms: FindSim, naive Bayes, Bayes nets, decision trees and support vector machines (SVM). The (weighted) averaged performance of the first four were 74.3%, 84.8% 86.2% and 88.6%, respectively. The DVMM is thus superior or comparable to each of these. The only algorithm which outperformed the DVMM was the SVM with averaged performance of 92%. However, these results are even more encouraging, as all of the above algorithms were used with the words representation (i.e., every word is a symbol in the now much expanded alphabet), while the DVMM was using the low level unparsed character representation.

class	protein family name	#seq.
c_1	*Fungal lignin peroxidase*	29
c_2	*Animal haem peroxidase*	33
c_3	*Plant ascorbate peroxidase*	26
c_4	*Bacterial haem catalase/peroxidase*	30
c_5	*Secretory plant peroxidase*	102

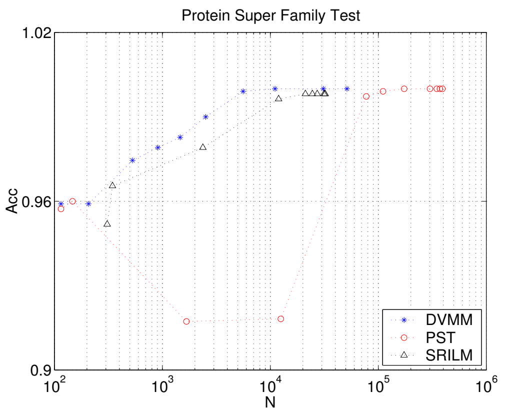


Figure 6.3: **Classification accuracy comparison for the Peroxidase Superfamily.** (left) The list of five peroxidase families used in this test, and number of sequences in each. (right) Classification accuracy is plotted against model size. All three algorithms are shown to obtain near optimal accuracy, with the DVMM performing consistently best per number of features.

6.4.3 Protein Classification

Out of the discriminative methods discussed in Section 2.4, we have chosen to compare our results to those of the Prints database (Attwood et al., 2003). We briefly recall that this established database is a collection of protein family fingerprints. Each family is matched with a fingerprint of one or more short profiles which have been iteratively refined using database scanning to maximize their discrimination power in a semi-automatic procedure involving human supervision. This method does not rely on phylogenetic information, and while it does utilize multiple alignment, it does so primarily to ease the manual curation work by suggesting potential regions of conservation. The resulting short profiles used for prediction are akin to the richer, yet positionless Markovian suffixes we use.

In terms of experimental design we will continue to report the average ten-fold cross validation runs. In screening infrequent suffixes we shall now ignore all those appearing twice or less. For performance measurement, it is easy to verify that for a uni-labeled dataset and a uni-labeled classification scheme, the micro-averaged precision and recall are equivalent, and hence equal to the $F1$ measure. Therefore, we will compute only micro-averaged precision, and simply term it accuracy.

The Peroxidase Superfamily

Peroxidases are Haem-containing enzymes that use hydrogen peroxide as the electron acceptor to catalyze a number of oxidative reactions. Five related protein families, all members of the Haem peroxidase super-family, were taken from the Prints database (version 29.0). Figure 6.3 holds both the protein family names and sizes and the classification accuracy of the three models. In this test, all algorithms are seen to achieve perfect, or near perfect classification using minimally pruned models. However, for more intensive pruning (and hence, smaller model size), the DVMM consistently outperforms the two generative algorithms. The PST model is again seen to display non-monotonic behaviour, and requires more features than the other two methods to obtain optimal accuracy.

We turn to examine the best discriminating model features. To do that we run the DVMM training algorithm once with all available data, and sort the obtained features as advocated in

class	feature	$\hat{P}(\sigma\|s,c)$	$\hat{P}(s\|c)$	seq-corr.	fing-corr.
c_1	`ARDS|R`	0.65	0.0019	62%	62%
c_3	`GLLQ|L`	0.64	0.0029	73%	73%
c_5	`ARDS|V`	0.66	0.0009	26%	0%
c_5	`GLLQ|S`	0.38	0.0006	11%	11%
c_3	`IVAL|S`	0.68	0.0035	88%	88%
c_5	`GLLQ|T`	0.29	0.0006	8%	8%
c_5	`IVAL|A`	0.28	0.0002	4%	4%
c_4	`PWWP|A`	0.59	0.0008	64%	64%
c_4	`ASAS|T`	0.40	0.0005	20%	20%
c_2	`FSNL|S`	0.49	0.0004	30%	0%

Table 6.1: **Peroxidase superfamily best discriminating features.** The ten best DVMM features of length four are shown, top down from best. For each we show the class where the feature probability is maximal (see Figure 6.3 for class names), as well as the feature probability itself in that class, $\hat{P}(\sigma|s,c)$. For example, `R` follows `ARDS` in 62% of the fungal lignin peroxidases. In the other categories, $\hat{P}(\sigma|s,c')$ was usually close to zero, and never exceeded 0.1. We also show the probability of observing suffix s in the maximizing class. The next column gives the percentage of sequences in the maximizing class that contain this feature. The last column compares the percentage of sequences in this class for which this feature appears in the Prints chosen family fingerprint consensus sequence. For example the feature `ARDS|R` is contained in a motif of the first family. It appears in this motif for 62% of the proteins assigned to it. In this table all features either came from a Prints motif or from elsewhere in the protein sequences, explaining the all or nothing correspondence between the last two columns.

Section 6.3.3. In Table 6.1 we present the top ten features with respect to all suffixes of length four. Eight of them coincide with the fingerprint chosen semi-manually by the Prints database to represent the respective class. The other two short motifs which have no match in the Prints database are however good features as they appear in no other class but their respective one. Such features can suggest improvements over obtained Prints fingerprints, to try and improve model recall or precision. The features we highlight, can in principle also draw attention to conserved motifs, of possible biological importance, which a multiple alignment program or a human curator may have failed to notice. Notice also that the first seven entries in Table 6.1 share but three different suffixes between them, where for each such suffix the next symbol separates between two different classes (e.g., `R`, `V` separate `ARDS` into classes 1 and 5 respectively. Neither appears in any of the other classes). This allows to highlight polymorphisms which are family specific and thus of special interest when considering the molecular reasoning behind a biological sub-classification. In the current set, however, a literature search could not link any of the high scoring features with functionally important residues.

The Glutathione S-Transferase Domain

We turn to test our discriminative approach in the more challenging task of discriminating functional sub-families. The next data set, prepared based on the Prints database, contains different members of the glutathione S-transferase (GST) domain family, which we have already used in a somewhat different context in the previous chapter. In most of these proteins, the GST domain participates in the detoxification of reactive electrophilic compounds by catalyzing their conjugation to glutathione. However, a sub-family termed the S-crystallins, while sharing an evolutionary history with the rest, has lost this function and appears to have taken a different, more structural

class	family name	#seq.
c_1	*GST* - no class label	298
c_2	*S crystallin*	29
c_3	*Alpha class GST*	40
c_4	*Mue class GST*	32
c_5	*Pi class GST*	22

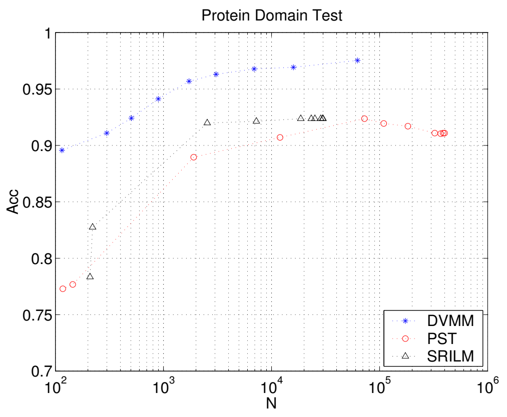


Figure 6.4: **Classification accuracy comparison for the GST domain.** (left) The five GST sets, of which c_1 is much larger than the rest, containing all GST domain sequences with no known sub-type classification. (right) We see that despite the skew in class sizes, and high sequence similarities between members of the different sets, the DVMM outperforms the generative algorithms, at all model size, including the minimally pruned one which obtains near 98% classification accuracy.

role. Within the former majority of GST domain proteins, sub-families with more subtle functional differences are currently being defined and analyzed. We have thus chosen to define five families: the S-crystallin, three other GST classes retaining the detoxification function, and a fifth class made of all class-unlabeled GST domain proteins. Figure 6.4 holds resulting set names and sizes.

Before turning to analyze the results, we note that all GST sequences share significant sequence similarity, to the point that the Pfam database of HMMs (Bateman et al., 2002), currently considered the state of the art in generative modeling of protein families (Section 2.2.1), has chosen to model all these sequences using a single HMM which does not distinguish between the different sub-types. Additionally, in the table in Figure 6.4 we see that relative class size, reflected in the empirical prior probability $\hat{P}(C)$ was especially skewed in this test, since we used all GST proteins with no known sub-classification as one of the groups. This skewing towards the label-less class is a known difficulty for classification schemes in general.

Despite of the above reservations, the performance graph in Figure 6.4 shows the DVMM algorithm to perform particularly well in this test, significantly surpassing the generative modeling approaches, at all model sizes. For example, the minimally pruned DVMM scored almost 98% in accuracy, while the generative methods peaked at 93%. Moreover, the DVMM accuracy using about 500 features was comparable to the accuracy of the generative PST approach, using some 400,000 features. The distinctive advantages of the DVMM approach in this test may be explained when considering the high similarities between members of all classes. In terms of accuracy per number of features, both the generative approaches take up modeling the rich features common to all sequences who share this doamin, in order to best approximate sequence generation statistics. The discriminative approach on the other hand, gives precedence to the most discriminating features, ignoring the many statistical correlations common to all classes. In addition, as it directly tries to minimize the discrimination error, the disregard for common features allows it to perform best, even with minimal pruning.

Again, in Table 6.2 we discuss the top ten sorted features with respect to all suffixes of length four. It is further demonstrated there how this automated method can help suggest improvements to the Prints semi-manual fingerprints generation procedure.

class	feature	$\hat{P}(\sigma\|s,c)$	$\hat{P}(s\|c)$	seq-corr.	fing-corr.
c_3	`AAGV|E`	0.74	0.0037	77%	52%
c_2	`AAGV|Q`	0.38	0.0020	31%	0%
c_2	`YIAD|C`	0.49	0.0016	34%	31%
c_5	`LDLL|L`	0.43	0.0029	45%	0%
c_1	`YIAD|K`	0.46	0.0003	4%	–
c_3	`YFPV|F`	0.42	0.0011	20%	0%
c_2	`GRAE|I`	0.70	0.0043	93%	0%
c_5	`DGDL|T`	0.49	0.0031	54%	50%
c_5	`YFPV|R`	0.45	0.0026	45%	0%
c_5	`KEEV|V`	0.51	0.0029	54%	0%

Table 6.2: **Best discriminating features for the GST domain.** The ten best DVMM features of length four are shown, top down from best, for the GST domain sub-families classification test of Figure 6.4. Table legend is as in Table 6.1. Note that class c_1 was constructed from all GST domain proteins without class labeling, and thus has no Prints fingerprint. Testimony of the relative difficulty of this task can be found in the fact that now only three of the top ten features are unique to their class. Moreover, six of these appear solely outside the Prints fingerprints, suggesting a refinement of the Prints GST domain sub-division.

6.5 Discussion

This chapter describes a well defined framework for learning variable memory Markov models in the context of discriminative analysis. It extends the approach taken in Chapter 3 to an explicit multiclass setting. The switch from generative one-against-all approach to a discriminating training framework is shown to be worth while. In tests where the different classes are easier to separate, such as a protein superfamily classification test, it is shown to improve performance at any restricted model size. Discriminative analysis is shown to be particularly beneficial when modeling protein sub-families where the similarity between members of the different classes is very high. In such a test the superiority of the discriminative approach is maintained even in a minimally pruned setting, as it disregards many features common to the different classes, and as such of no discriminating power. In comparison, HMM approaches, such as that of Hannenhalli and Russell (2000), reviewed in Section 2.4, is much more demanding computationally (see Chapter 3). It also relies on the quality of the multiple alignment from which its models are learnt, which the PST approach does not require.

As we have seen, the sorting of the resulting discriminating features allows the highlighting of features of variable lengths which differ between the classes. These can be used to suggest improvements to the semi-manual crafting of discriminating fingerprints done by the Prints database. While of proven value for discriminating sub-families, the PST top ranking features are not easily tied to function altering positions between the different classes. To see why this is so, reconsider the nature of the features we use. In order to capture the functional importance of a specific residue, we need to model some consistent context that precedes it. When one returns to examine the multiple alignment of Figure 2.9, where functionally important residues are numbered, it is seen that while the position itself is conserved within sub-types, and differs between them, the context that precedes it is far less conserved. In this particular aspect, a correct multiple alignment is very valuable, as Hannenhalli and Russell (2000) have demonstrated.

An interesting extension of the current work can try and extend the features chosen by the PST, or indeed to merge these, in a post-processing fashion, to allow modeling more complex discriminative contexts. One way to do so would be to extend the work of Eskin et al. (2000) to a discriminative framework. It would also be interesting to enrich the mulitple alignment dependent methods, such as that of Hannenhalli and Russell (2000), to allow for context dependencies which extend through more than one column, as these dependencies clearly exist in the proteins themselves.

From a theoretical point of view, a formal analysis of the algorithm is required. Indeed, it may even be possible to extend the theoretical results of Ron et al. (1996), to the context of discriminative VMM models.

This chapter is based on an extended abstract (Slonim et al., 2002) which was presented at ICML 2002, and was later developed to a journal version (Slonim et al., 2003).

Chapter 7

Concluding Remarks

This thesis has focused on the introduction of variable memory Markov models into bioinformatic research, and the field of protein sequence analysis in particular. We briefly review our contribution, and survey promising directions for future work.

7.1 Overview

The sequence of any novel protein remains by far its most accessible characteristic. For all proteins, the sequence is also the main determinator of structure. The particular shape of a protein, together with residue identities confer to it its functional role, or roles. A well established fact in protein sequence analysis determines that protein sequence similarity correlates well with both structural similarity and functional similarity (for a recent quantitative study see Wilson et al., 2000).

The conceptual building blocks of all proteins are the domain or domains from which they are composed. A protein domain is generally defined as a sequence region which folds into a particular compact structure, typically one with a hydrophobic core. Domains present one of the most useful levels for understanding protein function, as in most cases a single domain, or a combination thereof is directly implicated in conferring specific functionality to a protein that contains it. The ever-increasing numbers of sequences in databases provide new sources for domain detection. Indeed, as Figure 7.1 shows, for the past decade or more, the discovery rate of novel domains, using sequence homology or structure determination methods has maintained roughly linear growth. With the growing number of known protein families came the realization that domains themselves may have evolved from smaller structural units such as repeats or the assembly of smaller folding motifs and other ancient motifs into the larger structures seen today (Peng and Wu, 2000; Lupas et al., 2001). It also became clear that while many domains appear by themselves, or in very similar combinations with other domains, some domain types are found in combination with very different domains in different proteins. These modules or mobile domains, allow the transfer of functional information, such as being involved in a particular kind of interaction, between distinct protein classes (Copley et al., 2002). Our lack of deep understanding of sequence-structure relationship, and the diversity of protein domains, and domain combinations pose a challenge to our understanding of the protein world.

The definition of reliable and efficient computational models for the analysis of protein sequences at these different descriptive levels is an active field of research. In this thesis we have introduced variable memory Markov (VMM) modeling into this field. As evident from our survey in Chapter 2,

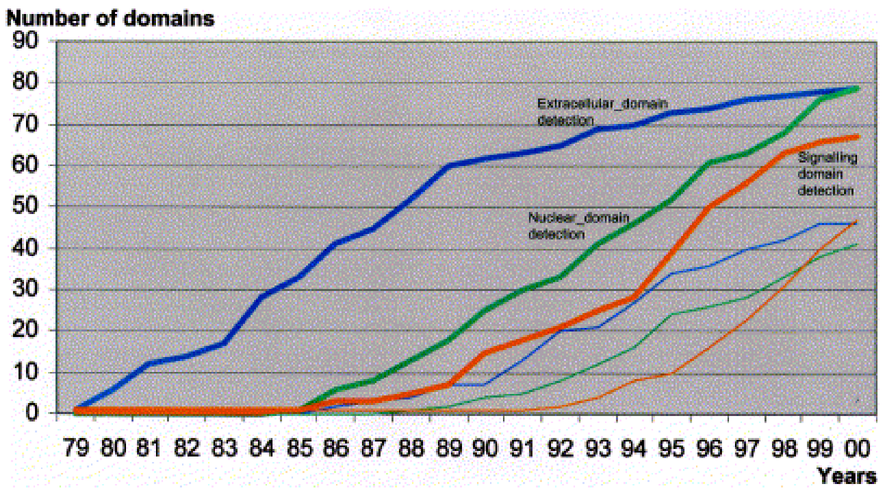


Figure 7.1: **The accumulation of detected sequence domains and solved structures along the years.** The cumulative plot shows the increase in novel sequence domains and solved structures over the last two decades. Characterized domains (in bold) and solved structures are broken by function into extracellular, nuclear and signalling (in blue, green and orange, respectively). (adapted from Copley et al., 2002)

this approach differs markedly from the modeling techniques currently in use in protein sequence analysis. Multiple sequence alignments (MSA) based profiles and hidden Markov models (HMM) aim for position based analysis of the underlying sequences. In defining the positional relationship between the different sequences, these models then resort to assume, for computational tractability, statistical independence between the different positions. From a biological point of view this simplification is certainly not true, as the different residues observed at a given position reflect the constraints imposed on the position by its spatial neighbours. The VMM approach to modeling protein sequences foregoes explicit positional alignment. As such it also does not rely on the simplifying independence assumption. Rather, it focuses on significant appearances of short contexts within unaligned related sequences, often capturing meaningful sequence motifs (Bork and Koonin, 1996). As we have shown in Chapter 3, related protein domain instances use very restricted sets of such contexts. These distinct contexts allow the recognition of novel family members through alignment-free analysis. As a result we were able to build compact computational models which capture the notion of a protein family from a seed set of unaligned family members. By bypassing the alignment phase altogether we make away with what is currently the least automated task in portion based methods. Following the algorithmic optimization of the VMM training and prediction algorithms in Chapter 4, these protein family models also take linear time and space to train, and classify with, compared to the quadratic complexity of the equivalent HMM algorithms.

The stochastic tiling of related protein sequences with common contexts carries additional advantages. First, one may typically model several domains using a single VMM. The reason being that the different domains use different context vocabularies, which are naturally modeled without interference in a single tree model. This property makes our algorithm especially valuable in analyzing sets of proteins where the same domains appear in different combinations (Bashton and Chothia, 2002). It can also serve to facilitate the detection of shorter swaps that have taken place during evolution between different domain families (Fliess et al., 2002).

Having realized this ability, we set forth in Chapter 5 to segment a set of unannotated sequences into the underlying recurring domains. By first modeling the entire set using a single VMM model, we then split this model in successive steps into several VMMs, each of which typically modeling a distinct subset of the sequences. By correlating model complexity with the amount of data assigned to it on the one hand, and dividing the data between the models in proportion to their relative modeling accuracy on the other, we obtained a self regularizing soft clustering algorithm. This algorithm was shown to capture, without aligning, instances of different domains within the given sequences. Finally, in Chapter 6, we applied VMM modeling to the closely related challenge of differentiating domain families into sub-types of sufficiently distinct functions. As all sequences in such sets are typically highly similar, it is often impossible to differentiate between the sub-types using generative modeling methods. With these methods, which include the VMM and HMM discussed above, sequence scoring, and subsequent classification is done based on observed correlations in the entire sequence. Discriminative modeling, on the other hand, aims at modeling only the features which best discriminate between the sub-types. As such it foregoes many features which are reliable descriptors of each group, but are also common to several or all groups due to their high sequence similarity. This approach allowed us to design much smaller, and yet more precise models to determine sub-class traits. It also has the potential of highlighting function conferring residues that differentiate between the different classes, as parts of its used features.

7.2 Future Directions

This work can be further extended in several exciting directions.

As we have demonstrated, the probabilistic suffix tree (PST) variable memory model performs well when modeling highly conserved protein families. However, as family conservation decreases, the variety of short consecutive motifs grows considerably, while their statistical significance levels drop (recall Figure 3.9). Based on the work of Kermorvant and Dupont (2002) it appears that more sophisticated smoothing techniques may improve PST performance, by compensating for small sample size effects in longer features, and less conserved families. Chen and Goodman (1998) survey several established methods in natural language processing to interpolate the observed counts of more complex features, using those of simpler ones. These may perhaps be combined with the work of Sjölander et al. (1996), which take a more Bayesian approach to probability smoothing in protein context, by using Dirichlet mixture models of typical amino acid distributions. A similar effect can be obtained if one tries to reduce the size of the alphabet by grouping together amino acids that share certain properties. One could do the grouping based on physico-chemical properties (see Figure 1.1), or based on empirical measures of substitution propensity, as was done by Cannata et al. (2002).

One especially appealing way to handle less conserved families would be to explicitly incorporate into the learning procedure the assumption that the observed sequences are a corrupted version of the original ones. Angluin and Csűrös (1997) have made such an attempt, where they extend the learning algorithm of Figure 3.2 to assume that the input sequences were passed through a memoryless noisy channel before observation. The general framework suggested there can be augmented by estimating explicit patterns of protein sequence evolution. Eskin et al. (2000) have tried to address this issue by introducing wild-cards into the learned features. While highly attractive from a biological point of view, application of wildcards causes the space and time complexity of the algorithm to grow rapidly, allowing for only very limited use of this feature in practice. It would be interesting to restrict the use of wildcard instances, guided by biological insights, such as those obtained by Lupas et al. (2001), who focus on the detection and characterization of conserved short motifs in broad classes of proteins. Another interesting direction would be to compare the modeling power of a PST (or similar) based approach, with that offered by a Bayesian framework. Willems et al. (1995) have shown that instead of generating a single hypothesis (model) per learned set of strings (protein family), one can efficiently sum over a weighted set of all possible PST-like models. The original approach of Willems et al. (1995) would have been prohibitively expensive in terms of memory requirements for an alphabet size of twenty. Algorithmic improvements, such as the work by Kennel and Mees (2002), have recently made the approach more viable. Note however, that while most of the previous methods can all be easily extended to the context of segmentation or discriminative analysis, the Bayesian approach which does not commit to any single model is probably harder to accommodate within those schemes.

Apart from the above enhancements, the segmentation algorithm of Figure 5.6, may benefit from a more accurate model selection criterion than the asymptotic minimum description length (MDL) variant. In our experience, while the MDL driven PST learning algorithm of Figure 5.1 crucially restrains model sizes during the cluster splitting process, it does so rather aggressively. While we have tried to re-weight the two terms of the MDL equation (model and data description lengths), other model selection methods may perform consistently better. One such approach is the extension of the MDL principle to on-line estimation of effective code lengths, discussed by Rissanen (1989), although this approach may require additional algorithmic work to improve its runtime performance. Other, more general criteria to PST model selection, explored by Bühlmann (2000), may prove beneficial in our context. Another important direction in which one may enhance the general segmentation framework, is to explicitly define the model we attempt to estimate as a hidden Markov model (HMM) of PSTs. Currently, this switching is implicit in interpreting the segmentation results, such as those presented in Figure 5.10. Namely, one may define a transition matrix which governs the switching between several PST models, each ultimately generating the instances of a single domain. While theoretically a relatively straight-forward extension, direct estimation of transition probabilities adds to the runtime complexity of the algorithm, and may prove hard to infer in the deterministic annealing framework. An attempt at extending the optimization results of Chapter 3 beyond single model estimation will be very useful in this context.

Our optimization approach may be simpler to extend to the discriminative analysis framework of Figure 6.1. While already very fast, unified training and optimized pruning of the discriminative VMM will turn this tool even more powerful, and may facilitate the incorporation of more complex features, such as those discussed above. Approaches akin to that of Eskin et al. (2000) can greatly assist in highlighting features that involve class differentiating residues.

Finally, the general rich feature based approach advocated in this thesis can be extended to both broader modeling schemes and related bioinformatic challenges. An example of the former is the extension of VMM modeling to encode the correlation of one sequence with respect to another, in a similar approach to that taken in input-output hidden Markov modeling (IO-HMM) or by transducers in general (see Bengio, 1999). This can allow one to incorporate secondary structure predictions into the basic PST classification scheme, and may also carry over to the more complex segmentation and discrimination algorithms.

From the biological point of view, it would be interesting to combine the strengths of the alignment-based methods with those of the VMM approach. For example, by using PST training as a pre- or post-processing stage to guide the generation of accurate multiple sequence alignments, from heterogeneous sets of protein sequences. It would also be interesting to attempt to generate a fast hierarchical classification scheme, built from a cascade of discriminative analysis VMM models, following one of the common structural classification schemes. A much more challenging effort may attempt to reconstruct and augment the hierarchical view, based on statistical distances between the VMM models of the different families and super-families.

Variable memory modeling can also assist in conceptually different tasks such as learning prior probabilities for the occurrences of different domains in combinations of varying lengths and repetitions. Genomic data also abounds in contextual dependent events. One such area is the field of detection and segmentation of coding genes within unannotated genomic sequences, where indeed similar approaches have already shown promise (Salzberg et al., 1999). The related fields of molecular sequence evolution and comparative genomics also hold great potential for incorporating this approach.

Bibliography

Abe, N. and Warmuth, M. (1992). On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9:205–260.

Aho, A. V. and Corasick, M. E. (1975). Efficient string matching: an aid to bibliographic search. *Comm. ACM*, 18:333–340.

Alberts, B., Bray, D., Johnson, A., Lewis, J., Raff, M., Robert, K., Walter, P., and Roberts, K. (1997). *Essential Cell Biology: An introducton to the Molecular Biology of the Cell.* Garland.

Almuallim, H. and Dietterich, T. G. (1991). Learning with many irrelevant features. In *Proc. Nat'l Conf. Artificial Intelligence*, volume 9, pages 547–552. AAAI Press.

Altschul, S. F. (1991). Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.*, 219(3):555–565.

Altschul, S. F., Madden, T. L., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25(17):3389–3402.

Angluin, D. and Csűrös, M. (1997). Learning Markov chains with variable length memory from noisy output. In *Proc. Conf. on Comput. Learning Theory*, volume 10, pages 298–308. ACM Press.

Apostolico, A. and Bejerano, G. (2000a). Optimal amnesic probabilistic automata or how to learn and classify proteins in linear time and space. In *Proc. Int'l Conf. Computational Molecular Biology*, volume 4, pages 25–32. ACM.

Apostolico, A. and Bejerano, G. (2000b). Optimal amnesic probabilistic automata or how to learn and classify proteins in linear time and space. *J. Comput. Biol.*, 7(3–4):381–393.

Apostolico, A., Bock, M. E., Lonardi, S., and Xu, X. (2000). Efficient detection of unusual words. *Journal of Computational Biology*, 7(1/2):71–94.

Apostolico, A. and Galil, Z., editors (1997). *Pattern Matching Algorithms.* Oxford University Press.

Argaman, L., Hershberg, R., Vogel, J., Bejerano, G., Wagner, E. G., Margalit, H., and Altuvia, S. (2001). Novel small RNA-encoding genes in the intergenic regions of Escherichia coli. *Curr. Biol.*, 11(12):941–950.

Attwood, T. K., Bradley, P., Flower, D. R., Gaulton, A., Maudling, N., Mitchell, A. L., Moulton, G., Nordle, A., Paine, K., Taylor, P., Uddin, A., and Zygouri, C. (2003). PRINTS and its automatic supplement, prePRINTS. *Nucleic Acids Res.*, 31(1):400–402.

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing*, pages 49–52.

Bailey, T. L. and Elkan, C. (1994). Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proc. Int'l Conf. Intell. Syst. Mol. Biol.*, volume 2, pages 28–36.

Barash, Y., Bejerano, G., and Friedman, N. (2001). A simple hyper-geometric approach for discovering putative transcription factor binding sites. *Lecture Notes in Computer Science*, 2149:278–293.

Barron, A., Rissanen, J., and Yu, B. (1998). The minimum description length principle in coding and modeling. *IEEE Trans. Inform. Theory*, 44:2743–2760.

Barrows, G. and Sciortino, J. (1996). A mutual information measure for feature selection with application to pulse classification. In *IEEE Int'l Symposium on Time-Frequency and Time-Scale Analysis*, pages 249–253.

Bashton, M. and Chothia, C. (2002). The geometry of domain combination in proteins. *J. Mol. Biol.*, 315(4):927–939.

Bateman, A., Birney, E., Cerruti, L., Durbin, R., Etwiller, L., Eddy, S. R., Griffiths-Jones, S., Howe, K. L., Marshall, M., and Sonnhammer, E. L. L. (2002). The Pfam protein families database. *Nucleic Acids Res.*, 30(1):276–280.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550.

Baum, E. and Haussler, D. (1989). What size net gives valid generalization? *Neural Computation*, 1(1):151–160.

Bejerano, G. (2003a). Algorithms for variable length Markov chain modeling. *Bioinformatics.* To appear.

Bejerano, G. (2003b). Efficient exact p-value computation and applications to biosequence analysis. In *Proc. Int'l Conf. Computational Molecular Biology*, volume 7, pages 38–47. ACM.

Bejerano, G., Friedman, N., and Tishby, N. (2003). Efficient exact p-value computation for small sample, sparse and surprising categorical data. *Submitted.*

Bejerano, G., Margalit, H., and Tishby, N. (2000). An information theoretic quantitative approach to natural selection and evolutionary processes at the genomic level. Unpublished manuscript.

Bejerano, G., Seldin, Y., Margalit, H., and Tishby, N. (2001). Markovian domain fingerprinting: statistical segmentation of protein sequences. *Bioinformatics*, 17(10):927–934.

Bejerano, G. and Yona, G. (1999). Modeling protein families using probabilistic suffix trees. In *Proc. Int'l Conf. Computational Molecular Biology*, volume 3, pages 15–24. ACM.

Bejerano, G. and Yona, G. (2001). Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, 17(1):23–43.

Bengio, Y. (1999). Markovian models for sequential data. *Neural Computing Surveys*, 2:129–162.

Blumer, A., Blumer, J., Ehrenfeucht, A., Haussler, D., Chen, M. T., and Seiferas, J. (1985). The smallest automaton recognizing the sub-words of a text. *Theoretical Computer Science*, 40:31–55.

Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.-C., Estreicher, A., Gasteiger, E., Martin, M. J., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S., and Schneider, M. (2003). The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res.*, 31(1):365–370.

Bork, P. and Koonin, E. V. (1996). Protein sequence motifs. *Curr. Opin. Struct. Biol.*, 6(3):366–376.

Branden, C.-I. and Tooze, J. (1999). *Introduction to Protein Structure.* Garland, 2^{nd} edition.

Brazma, A., Jonassen, I., Eidhammer, I., and Gilbert, D. (1998). Approaches to the automatic discovery of patterns in biosequences. *J. Comput. Biol.*, 5(2):279–305.

Buetler, T. M. and Eaton, D. L. (1992). Glutathione S-transferases: amino acid sequence comparison, classification and phylogenetic relationship. *Environ. Carcinogen. Ecotoxicol. Rev.*, C10:181–203.

Bühlmann, P. (2000). Model selection for variable length markov chains and tuning the context algorithm. *Annals of Inst. Stat. Math.*, 52:287–315.

Bühlmann, P. and Wyner, A. J. (1999). Variable length Markov chains. *Annals of Stat.*, 27:480–513.

Cannata, N., Toppo, S., Romualdi, C., and Valle, G. (2002). Simplifying amino acid alphabets by means of a branch and bound algorithm and substitution matrices. *Bioinformatics*, 18(8):1102–1108.

Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University.

Copley, R. R., Doerks, T., Letunic, I., and Bork, P. (2002). Protein domain analysis in the era of complete genomes. *FEBS Lett.*, 513(1):129–134.

Corpet, F., Servant, F., Gouzy, J., and Kahn, D. (2000). ProDom and ProDom-CG: tools for protein domain analysis and whole genome comparisons. *Nucleic Acids Res.*, 28(1):267–269.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory.* John Wiley.

Csiszar, I. (1974). On the computation of rate distortion functions. *IEEE Trans. Inform. Theory*, 20:122–124.

de Bakker, P. I., Bateman, A., Burke, D. F., Miguel, R. N., Mizuguchi, K., Shi, J., Shirai, H., and Blundell, T. L. (2001). HOMSTRAD: adding sequence information to structure-based alignments of homologous protein families. *Bioinformatics*, 17(8):748–749.

del Sol Mesa, A., Pazos, F., and Valencia, A. (2003). Automatic methods for predicting functionally important residues. *J. Mol. Biol.*, 326:1289–1302.

Della Pietra, S., Della Pietra, V., and Lafferty, J. (1997). Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4).

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. B*, 39:1–38.

Dubnov, S., Assayag, G., Lartillot, O., and Bejerano, G. (2003). Using machine-learning methods for musical style modeling. *IEEE Computer*, 36(10):73–80.

Dumais, S. T., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proc. ACM Int'l Conf. Information and Knowledge Management*, volume 7, pages 148–155, Bethesda, US. ACM Press.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge.

Eddy, S. R. (1998a). HMMER user's guide: biological sequence analysis using profile hidden markov models. http://hmmer.wustl.edu/.

Eddy, S. R. (1998b). Profile hidden Markov models. *Bioinformatics*, 14(9):755–763.

El-Yaniv, R., Fine, S., and Tishby, N. (1998). Agnostic classification of markovian sequences. In *Adv. Neural Inform. Proc. Sys.*, volume 10, pages 465–471. MIT press.

Elofsson, E. and Sonnhammer, E. L. L. (1999). A comparison of sequence and structure protein domain families as a basis for structural genomics. *Bioinformatics*, 15(6):480–500.

Enright, A. J., Iliopoulos, I., Kyrpides, N. C., and Ouzounis, C. A. (1999). Protein interaction maps for complete genomes based on gene fusion events. *Nature*, 402(6757):86–90.

Enright, A. J. and Ouzounis, C. A. (2000). GeneRAGE: a robust algorithm for sequence clustering and domain detection. *Bioinformatics*, 16(5):451–457.

Enright, A. J., Van Dongen., S., and Ouzounis, C. A. (2002). An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, 30(7):1575–1584.

Eskin, E., Grundy, W. N., and Singer, Y. (2000). Protein family classification using sparse Markov transducers. In *Proc. Int'l Conf. Intell. Syst. Mol. Biol*, volume 8, pages 134–145.

Fedorov, A. N. and Baldwin, T. O. (1995). Contribution of cotranslational folding to the rate of formation of native protein structure. *Proc. Nat'l Acad. Sci. USA*, 92(4):1227–31.

Feng, Y., Minnerly, J. C., Zurfluh, L. L., Joy, W. D., Hood, W. F., Abegg, A. L., Grabbe, E. S., Shieh, J. J., Thurman, T. L., McKearn, J. P., and McWherter, C. A. (1999). Circular permutation of granulocyte colony-stimulating factor. *Biochemistry*, 38(14):4553–63.

Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32:41–62.

Fliess, A., Motro, B., and Unger, R. (2002). Swaps in protein sequences. *Proteins*, 48(2):377–387.

Freund, Y. and Ron, D. (1995). Learning to model sequences generated by switching distributions. In *Comput. Learning Theory*, volume 8, pages 41–50. ACM Press.

Friedl, J. E. F. (2002). *Mastering Regular Expressions.* O'Reilly, 2^{nd} edition.

George, R. A. and Heringa, J. (2002a). Protein domain identification and improved sequence similarity searching using PSI-BLAST. *Proteins*, 48(4):672–681.

George, R. A. and Heringa, J. (2002b). SnapDRAGON: a method to delineate protein structural domains from sequence data. *J. Mol. Biol.*, 316(3):839–851.

Gilks, W. R., Audit, B., De Angelis, D., Tsoka, S., and Ouzounis, C. A. (2002). Modeling the percolation of annotation errors in a database of protein sequences. *Bioinformatics*, 18(12):1641–1649.

Gillman, D. and Sipser, M. (1994). Inference and minimization of hidden Markov chains. In *Proc. Annual ACM Conf. Computational Learning Theory*, volume 7, pages 147–158. ACM Press.

Goodman, R. M. and Smyth, P. (1988). Decision tree design from communication theory stand point. *IEEE Trans. Inform. Theory*, 34(5):979–994.

Gouldson, P. R., Bywater, R. P., and Reynolds, C. A. (1997). Correlated mutations amongst the external residues of G-protein coupled receptors. *Biochem. Soc. Trans.*, 25(3):529S.

Gouzy, J., Corpet, F., and Kahn, D. (1999). Whole genome protein domain analysis using a new method for domain clustering. *Comput. Chem.*, 23(3-4):333–340.

Gracy, J. and Argos, P. (1998a). Automated protein sequence database classification. I. integration of compositional similarity search, local similarity search, and multiple sequence alignment. II. delineation of domain boundries from sequence similarity. *Bioinformatics*, 14(2):164–187.

Gracy, J. and Argos, P. (1998b). DOMO: a new database of aligned protein domains. *Trends Biochem. Sci.*, 23(12):495–497.

Gribskov, M., McLachlan, A. D., and Eisenberg, D. (1987). Profile analysis: detection of distantly related proteins. *Proc. Nat'l Acad. Sci. USA*, 84(13):4355–8.

Griffiths-Jones, S. and Bateman, A. (2002). The use of structure information to increase alignment accuracy does not aid homologue detection with profile HMMs. *Bioinformatics*, 18(9):1243–1249.

Grundy, W. N., Bailey, T. L., Elkan, C. P., and Baker, M. E. (1997). Meta-MEME: motif-based hidden Markov models of protein families. *Comput. Appl. Biosci.*, 13(4):397–406.

Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press.

Hadley, C. and Jones, D. T. (1999). A systematic comparison of protein structure classifications: SCOP, CATH and FSSP. *Structure Fold Des.*, 7(9):1099–1112.

Haft, D. H., Selengut, J. D., and White, O. (2003). The TIGRFAMs database of protein families. *Nucleic Acids Res.*, 31(1):371–373.

Hannenhalli, S. S. and Russell, R. B. (2000). Analysis and prediction of functional sub-types from protein sequence alignments. *J. Mol. Biol.*, 303:61–76.

Hayes, J. D. and Pulford, D. J. (1995). The glutathione S-transferase supergene family: regulation of GST and the contribution of the isoenzymes to cancer chemoprotection and drug resistance. *Cri. Rev. Biochem. Mol. Biol.*, 30(6):445–600.

Heger, A. and Holm, L. (2000). Towards a covering set of protein family profiles. *Progress in Biophysics and Molecular Biology*, 73(5):321–337.

Heger, A. and Holm, L. (2001). Picasso: generating a covering set of protein family profiles. *Bioinformatics*, 17(3):272–279.

Henikoff, J. G., Greene, E. A., Pietrokovski, S., and Henikoff, S. (2000). increased coverage of protein families with the Blocks database servers. *Neucleic Acids Res.*, 28(1):228–230.

Henikoff, J. G. and Henikoff, S. (1996). Using substitution probabilities to improve position-specific scoring matrices. *Comput. Appl. Biosci.*, 12(2):135–43.

Henikoff, S. and Henikoff, J. G. (1991). Automated assembly of protein blocks for database searching. *Nucleic Acids Res.*, 19(23):6565–72.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Nat'l Acad. Sci. USA*, 89(22):10915–9.

Hennecke, J., Sebbel, P., and Glockshuber, R. (1999). Random circular permutation of DsbA reveals segments that are essential for protein folding and stability. *J. Mol. Biol.*, 286(4):1197–215.

Hershberg, R., Bejerano, G., Santos-Zavaleta, A., and Margalit, H. (2001). Promec: An updated database of escherichia coli mrna promoters with experimentally identified transcriptional start sites. *Nucleic Acids Res.*, 29(1):277.

Higgins, D. G., Thompson, J. D., and Gibson, T. J. (1996). Using CLUSTAL for multiple sequence alignments. *Methods Enzymol.*, 266:383–402.

Holm, L. and Sander, C. (1998). Touring protein fold space with Dali/FSSP. *Nucleic Acids Res.*, 26(1):316–319.

Huang, H., Barker, W. C., Chen, Y., and Wu, C. H. (2003). iProClass: an integrated database of protein family, function and structure information. *Nucleic Acids Res.*, 31(1):390–392.

Hughes, G. F. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Trans. Inform. Theory*, 14:55–63.

Hughey, R. and Krogh, A. (1998). SAM: Sequence alignment and modeling software system. Technical Report UCSC-CRL-96-22, University of California, Santa Cruz, Jack Baskin School of Engineering.

Jeanmougin, F., Thompson, J., Gouy, M., Higgins, D., and Gibson, T. (1998). Multiple sequence alignment with clustal X. *Trends Biochem. Sci.*, 23:403–405.

Kearns, M., Mansour, Y., Ron, D., Rubinfeld, R., Schapire, R. E., and Sellie, L. (1994). On the learnability of discrete distributions. In *Proc. Annual Sympos. Theory of Computing*, volume 26, pages 273–282. ACM Press.

Kennel, M. B. and Mees, A. I. (2002). Context-tree modeling of observed symbolic dynamics. *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, 66(5/2):056209.

Kermorvant, C. and Dupont, P. (2002). Improved smoothing for probabilistic suffix trees seen as variable order Markov chains. In *Lecture notes in Artif. Intell.*, volume 2430, pages 185–194. Springer-Verlag.

Kolb, V. A., Makeyev, E. V., Kommer, A., and Spirin, A. S. (1995). Cotranslational folding of proteins. *Biochem. Cell Biol.*, 73(11–12):1217–20.

Krause, A., Stoye, J., and Vingron, M. (2000). The SYSTERS protein sequence cluster set. *Nucleic Acids Res.*, 28(1):270–272.

Krichevsky, R. E. and Trofimov, V. K. (1981). The performance of universal coding. *IEEE Trans. Inform. Theory*, IT-27:199–207.

Kriventseva, E. V., Servant, F., and Apweiler, R. (2003). Improvements to CluSTr: the database of SWISS-PROT+TrEMBL protein clusters. *Nucleic Acids Res.*, 31(1):388–389.

Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994). Hidden markov models in computational biology. applications to protein modeling. *J. Mol. Biol.*, 235(5):1501–31.

Lartillot, O., Dubnov, S., Assayag, G., and Bejerano, G. (2001). Automatic modeling of music style. In *Proc. Int'l Computer Music Conf.*, pages 447–453.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993). Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–14.

Letunic, I., Goodstadt, L., Dickens, N. J., Doerks, T., Schultz, J., Mott, R., Ciccarelli, F., Copley, R. R., Ponting, C. P., and Bork, P. (2002). Recent improvements to the SMART domain-based sequence annotation resource. *Nucleic Acids Res.*, 30(1):242–244.

Lewin, B. (2000). *Genes VII.* Oxford.

Lewis, P. M. (1962). The characteristic selection problem in recognition systems. *IEEE Trans. Inform. Theory*, 8:171–178.

Liu, J. and Rost, B. (2002). Target space for structural genomics revisited. *Bioinformatics*, 18(7):922–933.

Liu, J. and Rost, B. (2003). Domains, motifs and clusters in the protein universe. *Curr. Opin. Chem. Biol.*, 7(1):5–11.

Lo Conte, L., Brenner, S. E., Hubbard, T. J. P., Chothia, C., and Murzin, A. G. (2002). SCOP database in 2002: refinements accommodate structural genomics. *Nucleic Acids Res.*, 30(1):264–267.

Lupas, A. N., Ponting, C. P., and Russell, R. B. (2001). On the evolution of protein folds: are similar motifs in different protein folds the result of convergence, insertion, or relics of an ancient peptide world. *J. Struct. Biol.*, 134(2-3):191–203.

Marcotte, E. M., Pellegrini, M., Ng, H. L., Rice, D. W., Yeates, T. O., and Eisenberg, D. (1999). Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–753.

May, A. C. W. (2002). Definition of the tempo of sequence diversity across an alignment and automatic identification of sequence motifs: Application to protein homologous families and superfamilies. *Protein Sci.*, 11(12):2825–2835.

McCallum, A. K. (1997). Reinforcement learning with selective perception and hidden state. Technical Report TR611, University of Rochester, Computer Science Department.

McCreight, E. M. (1976). A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2):262–272.

Mulder, N. J., Apweiler, R., Attwood, T. K., Bairoch, A., Barrell, D., Bateman, A., Binns, D., Biswas, M., Bradley, P., Bork, P., Bucher, P., Copley, R. R., Courcelle, E., Das, U., Durbin, R., Falquet, L., Fleischmann, W., Griffiths-Jones, S., Haft, D., Harte, N., Hulo, N., Kahn, D., Kanapin, A., Krestyaninova, M., Lopez, R., Letunic, I., Lonsdale, D., Silventoinen, V., Orchard, S. E., Pagni, M., Peyruc, D., Ponting, C. P., Selengut, J. D., Servant, F., Sigrist, C. J. A., Vaughan, R., and Zdobnov, E. M. (2003). The InterPro Database, 2003 brings increased coverage and new features. *Nucleic Acids Res.*, 31(1):315–318.

Normandin, Y., Cardin, R., and de Mori, R. (1994). High-performance connected digit recognition using maximum mutual information estimation. *IEEE Trans. on Speech and Audio Processing*, 1(2):299–311.

Otzen, D. E. and Fersht, A. R. (1998). Folding of circular and permuted chymotrypsin inhibitor 2: retention of the folding nucleus. *Biochemistry*, 37(22):8139–46.

Pazos, F., Helmer-Citterich, M., Ausiello, G., and Valencia, A. (1997). Correlated mutations contain information about protein-protein interaction. *J. Mol. Biol.*, 271(4):511–23.

Pearl, F. M. G., Bennett, C. F., Bray, J. E., Harrison, A. P., Martin, N., Shepherd, A., Sillitoe, I., Thornton, J., and Orengo, C. A. (2003). The CATH database: an extended protein family resource for structural and functional genomics. *Nucleic Acids Res.*, 31(1):452–455.

Pearson, W. R. (1995). Comparison of methods for searching protein sequence databases. *Protein Sci.*, 4(6):1145–60.

Pearson, W. R. (2000). Flexible sequence similarity searching with the FASTA3 program package. *Methods Mol. Biol.*, 132:185–219.

Peng, Z. Y. and Wu, L. C. (2000). Autonomous protein folding units. *Adv. Protein Chem.*, 53:1–47.

Rabiner, L. R. (1986). Tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:257–286.

Rabow, A. A. and Scheraga, H. A. (1993). Lattice neural network minimization. application of neural network optimization for locating the global-minimum conformations of proteins. *J. Mol. Biol.*, 232(4):1157–68.

Ramensky, V. E., Makeev, V. J., Roytberg, M. A., and Tumanyan, V. G. (2001). Segmentation of long genomic sequences into domains with homogeneous composition with BASIO software. *Bioinformatics*, 17(11):1065–1066.

Rigoutsos, I., Floratos, A., Ouzounis, C., Gao, Y., and Parida, L. (1999). Dictionary building via unsupervised hierarchical motif discovery in the sequence space of natural proteins. *Proteins*, 37(2):264–277.

Rissanen, J. (1983). A universal data compression system. *IEEE Trans. Inform. Theory*, 29(5):656–664.

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

Roca, J. (1995). The mechanisms of DNA topoisomerases. *Trends in Biol. Chem.*, 20:156–160.

Ron, D., Singer, Y., and Tishby, N. (1996). The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25:117–149.

Rose, K. (1998). Deterministic annealing for clustering, compression, classification, regression and related optimization problems. *Proc. IEEE*, 80:2210–2239.

Salwinski, L. and Eisenberg, D. (2003). Computational methods of analysis of protein-protein interactions. *Curr. Opin. Struct. Biol.*, 13(3):377–382.

Salzberg, S. L., Pertea, M., Delcher, A. L., Gardner, M. J., and Tettelin, H. (1999). Interpolated Markov models for eukaryotic gene finding. *Genomics*, 59(1):24–31.

Sasson, O., Vaaknin, A., Fleischer, H., Portugaly, E., Bilu, Y., Linial, N., and Linial, M. (2003). ProtoNet: hierarchical classification of the protein space. *Nucleic Acids Res.*, 31(1):348–352.

Seldin, Y., Bejerano, G., and Tishby, N. (2001). Unsupervised sequence segmentation by a mixture of switching variable memory Markov sources. In *Proc. Int'l Conf. Machine Learning*, volume 18, pages 513–520. Morgan Kaufmann.

Shoop, E., Silverstein, K. A., Johnson, J. E., and Retzel, E. F. (2001). MetaFam: a unified classification of protein families. II. Schema and query capabilities. *Bioinformatics*, 17(3):262–271.

Sigrist, C. J. A., Cerutti, L., Hulo, N., Gattiker, A., Falquet, L., Pagni, M., Bairoch, A., and Bucher, P. (2002). PROSITE: a documented database using patterns and profiles as motif descriptors. *Brief. Bioinform.*, 3(3):265–274.

Sjölander, K., Karplus, K., Brown, M., Hughey, R., Krogh, A., Mian, I. S., and Haussler, D. (1996). Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Comput. Appl. Biosci.*, 12:327–345.

Slonim, N., Bejerano, G., Fine, S., and Tishby, N. (2002). Discriminative feature selection via multiclass variable memory markov model. In *Int'l Conf. Machine Learning*, volume 19, pages 578–585. Morgan Kaufmann.

Slonim, N., Bejerano, G., Fine, S., and Tishby, N. (2003). Discriminative feature selection via multiclass variable memory markov model. *Eurasip J. Applied Signal Processing*, 2:93–102.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, 147(1):195–197.

Sohler, F. and Zien, A. (2001). Remote homology detection using significant sequence patterns. In *Proc. Int'l Conf. Intell. Syst. Mol. Biol.*, volume 9. Poster.

Stolcke, A. (1998). Entropy-based pruning of backoff language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Stolcke, A. (2002). SRILM: The SRI language modeling toolkit. http://www.speech.sri.com/-projects/srilm.

Stuart, E. T., Kioussi, C., and Gruss, P. (1994). Mammalian Pax genes. *Annu. Rev. Genet.*, 28:219–236.

Taylor, W. R. (1968). The classification of amino acid conservation. *J. Theor. Biol.*, 119:205–218.

Torkkola, K. and Campbell, W. (2000). Mutual information in learning feature transformations. In *Proc. Int'l Conf. Machine Learning*, volume 17, pages 1015–1022. Morgan Kaufmann.

Tsuji, T., Yoshida, K., Satoh, A., Kohno, T., Kobayashi, K., and Yanagawa, H. (1999). Foldability of barnase mutants obtained by permutation of modules or secondary structure units. *J. Mol. Biol.*, 286(5):1581–96.

Tucker, C. L., Hurley, J. H., Miller, T. R., and Hurley, J. B. (1998). Two amino acid substitutions convert a guanylyl cyclase, RetGC-1, into an adenylyl cyclase. *Proc. Nat'l Acad. Sci. USA*, 11:5994–5997.

Ukkonen, E. (1995). On-line construction of suffix trees. *Algorithmica*, 14(3):249–260.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Weiner, P. (1973). Linear pattern matching algorithm. In *Proc. Annual IEEE Sympos. Switching and Automata Theory*, volume 14, pages 1–11. IEEE Press.

Westbrook, J., Feng, Z., Chen, L., Yang, H., and Berman, H. M. (2003). The Protein Data Bank and structural genomics. *Nucleic Acids Res.*, 31(1):489–491.

Wheelan, S. J., Marchler-Bauer, A., and Bryant, S. H. (2000). Domain size distributions can predict domain boundaries. *Bioinformatics*, 16(7):613–618.

Wicker, N., Perrin, G. R., Thierry, J. C., and Poch, O. (2001). Secator: a program for inferring protein subfamilies from phylogenetic trees. *Mol. Biol. Evol.*, 18(8):1435–1441.

Willems, F. M. J., Shtarkov, Y. M., and Tjalkens, T. J. (1995). The context-tree weighting method: basic properties. *IEEE Trans. Inform. Theory*, 41(3):653–664.

Wilson, C. A., Kreychman, J., and 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(1):233–249.

Woodland, P. C. and Povey, D. (2000). Large scale discriminative training for speech recognition. In *Proc. Int'l Workshop on Automatic Speech Recognition*.

Xing, E. P., Wolf, D. M., Dubchak, I., Spengler, S., Zorn, M., Muchnik, I., and Kulikowski, C. (2001). Automatic discovery of sub-molecular sequence domains in multi-aligned sequences: a dynamic programming algorithm for multiple alignment segmentation. *J. Theor. Biol.*, 212(2):129–139.

Xu, H. E., Rould, M. A., Xu, W., Epstein, J. A., Maas, R. L., and Pabo, C. O. (1999). Crystal structure of the human Pax6 paired domain-DNA complex reveals specific roles for the linker region and carboxy-terminal subdomain in DNA binding. *Genes Dev.*, 13(10):1263–1275.

Yang, H. and Moody, J. (1999). Feature selection based on joint mutual information. In *Proc. ICSC Sympos. on Advances in Intelligent Data Analysis*.

Yona, G. and Levitt, M. (2000). Towards a complete map of the protein space based on a unified sequence and structure analysis of all known proteins. In *Proc. Int'l Conf. Intell. Syst. Mol. Biol.*, volume 8, pages 395–406.

Yona, G., Linial, N., and Linial, M. (2000). ProtoMap: automatic classification of protein sequences and hierarchy of protein families. *Nucleic Acids Res.*, 28(1):49–55.

Zhang, Z., Huang, L., Shulmeister, V. M., Chi, Y. I., Kim, K. K., Hung, L. W., Crofts, A. R., Berry, E. A., and Kim, S. H. (1998). Electron transfer by domain movement in cytochrome bc1. *Nature*, 392:677–684.

Appendix A

Additional Works

Addition publications in which the author has taken part during his graduate studies, of which two major ones are reproduced below:

- The development of techniques for efficient exact p-value computation in discrete settings, such as those often found in the analysis of biosequences. Enclosed below is the extended abstract (Bejerano, 2003b) which won the best paper by a young scientist award at RECOMB 2003, and was later expanded to a journal version (Bejerano et al., 2003).
- A conceptual framework that tries to tie the evolution of protein coding sequences with information theoretic notions of coding efficiency. Enclosed is the manuscript (Bejerano et al., 2000) which was presented in poster format at the Human Genetics in the Post-Genomic Age meeting, 2000.
- An application of the variable memory models analysed in the thesis, to computer generated composition and musical improvisation, based on a given set of musical pieces. An extended abstract (Lartillot et al., 2001) was presented at ICMC 2001, and was later developed to a journal version (Dubnov et al., 2003).
- An attempt to analyse the promoter sequences in Escherichia coli, which led to the organization of a database of experimentally verified transcription start sites in E. coli (Hershberg et al., 2001), and helped in the discovery of novel small RNAs in the E. coli genome (Argaman et al., 2001).
- Detection of transcription binding sites in unaligned sets of sequences (Barash et al., 2001).

A.1 Efficient Exact p-Value Computation

replace with paper!

A.2 Protein Evolution and Coding Efficieny

replace with paper!