Mega-Classification: Discovering Motifs in Massive Datastreams

Nomi L. Harris
Lawrence Hunter
David J. States

National Library of Medicine
Building 38A
National Institutes of Health
Bethesda, MD 20894
harris@ncbi.nlm.nih.gov

Abstract

We report on the development and application of an efficient unsupervised learning procedure for the classification of an unsegmented datastream, given a set of probabilistic binary similarity judgments between regions in the stream. Our method is effective on very large databases, and tolerates the presence of noise in the similarity judgements and in the extents of similar regions. We applied this method to the problem of finding the sequence-level building blocks of proteins. After verifying the effectiveness of the clusterer by testing it on synthetic protein data with known evolutionary history, we applied the method to a large protein sequence database (a datastream of more than 10^7 elements) and found about 10,000 protein sequence classes. The motifs defined by these classes are of biological interest, and have the potential to supplement or replace the existing manual annotation of protein sequence databases.

Introduction

There are many challenges in applying machine learning methods to large, real-world problems. In this paper, we report on our exploration of a classification problem that is several orders of magnitude larger than any other application of unsupervised learning technology of which we are aware. Existing machine learning methods are impractical for this problem, primarily because they fail to scale up adequately (Schank, 1991) or because they are ineffective at handling large numbers of irrelevant features (Almuallim & Dietterich, 1991). We developed a simple and efficient heuristic classification method, and demonstrated its effectiveness on synthetic data. We also report on the application of the method to a natural dataset containing more than 10^7 elements.

Our motivating problem involves finding motifs, or repeating patterns, in protein sequences. Proteins are composed of amino acids, bonded together linearly into a *protein sequence*. Proteins fold into three dimensional structures, fully coded for in the protein sequence, which determine their biochemical functions. There are 20 different naturally occurring amino acids, and the longest known protein sequences contain more than 6000 amino acids. The space of possible proteins is therefore very large: 20^{6000}, or more than 10^{7800}. Evolution has explored only a tiny fraction of the space of possible proteins. A commonly held explanation is that evolution has made use of a set of protein building blocks: collections of amino acids that form a functional unit and are combined or inherited as a group. These building blocks are sometimes called *domains*. A variety of domains that relate to protein function have been identified by biologists. For example, the "zinc finger" domain is a constituent of many DNA-binding proteins (Berg, 1990). Our method makes use of the protein sequences themselves to identify the domains that constitute them.

Several features of protein building blocks must be kept in mind when designing machine learning methods for identifying them. First, the building blocks are not of uniform size. Of the hundreds of putative domains identified by biologists, see e.g. (Bairoch, 1991), the shortest is three amino acids long, and the longest contains nearly 400 amino acids. Second, domains evolve over time; insertions, deletions and substitutions in the amino acid sequence of each domain occur through evolution, leading to significant variation in the composition of instances of any particular domain. Despite the variation, each domain has an underlying characteristic pattern, or motif. Third, the databases we must search to find these motifs are quite large. Efforts related to the Human Genome Project and other biological research have produced large numbers of protein sequences. The Protein Information Resource or PIR, (Barker, George, Hunt, & Garavelli, 1991), the largest of the individual protein

sequence databases, includes more than 33,000 protein sequences containing a total of more than 10 million amino acids. These databases are growing exponentially, with a doubling time of just over a year. Our goal is to use as much as possible of this information to identify domains that have been used repeatedly throughout evolution.

Given the explosive growth of protein sequence data, automation of as much of the analysis of this data as possible has become an imperative. The amount of existing protein data is already too large to be analyzed by hand: only 10% of the sequences in PIR have their domains annotated, and manual annotation is falling further and further behind. Our approach is a step toward the automatic annotation of massive sequence databases.

In a nutshell, the machine learning problem we address is how to efficiently segment and classify a datastream, given the ability to identify variable length regions of similar content. We have developed a method that uses a binary similarity metric to extract objects from large, unsegmented and noisy datastreams and performs automatic unsupervised classification on those objects. This method, although tailored to our specific application, has potential applications in other domains. For example, a language learner faces a similar problem in interpreting a noisy stream of auditory data to identify and classify the different types of sounds.

Existing Classification Approaches for Finding Motifs

Traditional ML classification methods such as conceptual clustering, e.g. (Fisher, 1987); Bayesian classification, e.g. (Cheeseman, et al., 1988); and self-organizing feature maps, e.g. (Kohonen, 1988) are unsuitable for application to this problem for several reasons. First, these methods require the data to be presented in the form of discrete objects with an explicit list of attributes, and demand a real-valued distance function that can measure the degree of "similarity" between any pair of objects. Protein domains are of varying length, and can occur at varying offsets within different proteins. The representational requirements of existing methods cannot be met naturally and in a reasonable amount of space with this kind of data. For example, mappings of sequences to bitstrings or fixed length feature vectors often use a sliding window down the sequences to make fixed length vectors. This method increases the number of objects to nearly the number of amino acids in the data set, which is too large (over 10,000,000 in PIR alone) to be practical for our application.

Second, none of these methods are efficient enough to handle a genuinely large dataset. At best, these methods make on the order of one comparison between each pair of objects. These classification methods therefore require at least $O(n^2)$ time to classify n objects. Even using an efficient partial matcher to identify similar pairs in the PIR database generates nearly 2 million objects (hits). If each hit had to be compared against every other hit, and each comparison took only 10 microseconds, the n^2 comparisons would take at least 4×10^7 seconds, or 463 days. Incremental approaches do not ameliorate the difficulty with this megaclassification problem.

The HHS algorithm

Our HHS (Hunter, Harris, States) clustering algorithm works in two stages. Starting with fragmented regions provided by a binary comparison tool, it assembles fragments into objects and then groups together similar objects by their extent of overlap.

The tool we used for obtaining similarity judgements for protein sequences was BLAST (Altschul, et al, 1990), a program for rapidly finding statistically significant partial matches between pairs of protein sequences. BLAST uses amino acid mutation scores for approximate string-matching. It directly approximates the results that would be obtained by a dynamic programming algorithm, but is more than an order of magnitude faster. Given a pair of protein sequences, BLAST identifies variable length stretches that have greater than chance similarity. These pairs of similar subsequences are called *hits*.. Each hit consists of a string of amino acids from the "query" protein and the corresponding same-length string from the "subject" protein (Figure 1). The length of a BLAST hit depends on the extent of similarity between the two sequences being compared and on a sensitivity/specificity tradeoff that can be set at runtime.

The clustering task involves trying to build the transitive closure of the similarity judgments that BLAST makes. There are several complications that make this task difficult. BLAST searching is probabilistic and therefore noisy. It can miss regions of similarity, and it can fragment a single region of similarity into multiple hits. Also, BLAST handles approximate matches in the content of the sequences, but it requires exact registration for matching, and its matches have fixed extent. We need to be able to build groups that have approximately matching extents, and where the registration between regions of similarity is not perfect. The HHS clustering algorithm does this by first assembling the BLAST hits and then clustering the assembled hits.

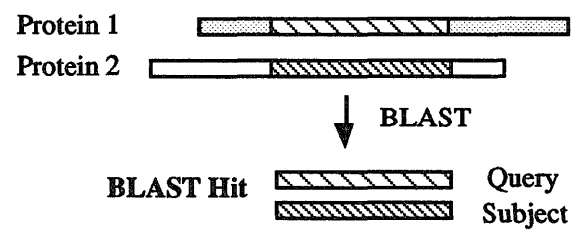


Figure 1: BLAST is applied to two protein sequences that have homologous regions, resulting in a hit.

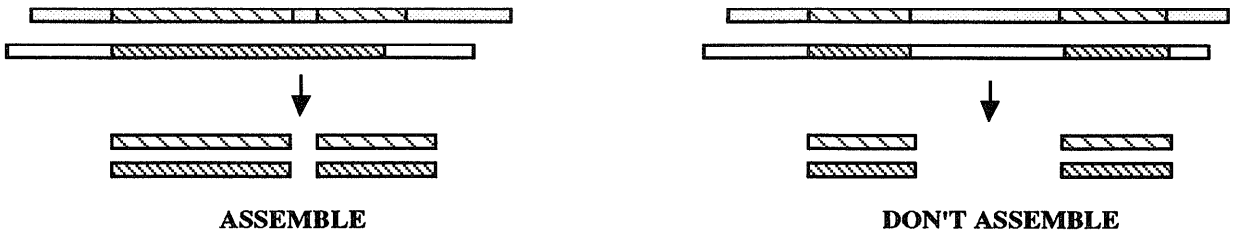


Figure 2: Protein 1 has a gap in its region of homology with Protein 2. Since BLAST can't handle gaps, it finds two separate hits. BLAST hits separated by a small gap probably come from the same domain and should be assembled. BLAST hits with a large gap between them probably come from different domains, and should not be assembled.

Hit Assembly

There are well known biological mechanisms that create differences in registration and extent of similar regions. As proteins evolve, their sequences are gradually transformed by biological events. The most common events include point mutations, where one amino acid is substituted with another; insertions, where a new amino acid is inserted into the sequence; and deletions, where an amino acid is deleted from a sequence. (Insertions and deletions are referred to as "indels.") Substitutions and indels can cause two proteins that derive from a common ancestor to gradually diverge from each other.

The best alignment between two homologous (i.e. evolutionarily related, and therefore similar) sequences may contain gaps in one or both sequences. BLAST is unable to insert gaps into its alignments. If it encounters non-matching portions in the sequences being compared, it generally breaks the hit. A new hit is likely to start after the indel, with a different offset between the query and subject portions. This means that hits do not necessarily represent complete domains; they may include only pieces of a domain. Even strongly homologous regions might show up as many separate BLAST hits.

The first stage of the HHS algorithm is to assemble the potentially fragmented raw BLAST hits into continuous (possibly gapped) regions of local similarity, which we call "assembled hits." Not all hits that involve a single pair of proteins should be assembled. A large gap between two hits indicates a lapse in homology (Figure 2). The probability that BLAST would fail to pick up such a long stretch of homologous sequence is low, so a long gap between two hits provides evidence that the two hits belong to separate domains and should not be assembled.

The hit assembly procedure works by finding all pairs of BLAST hits that share both query and subject protein and have a sufficiently small gap between both the two queries and the two subjects. This procedure is fast, since the maximum number of hits between any pair of proteins is relatively small. The program that parses the BLAST output sorts the hits by query and subject, so the hits that share both query and subject proteins are adjacent to each other in the list of hits.

The longest permissible gap between two BLAST hits that qualify for assembly can be calculated by considering BLAST's sensitivity. BLAST is very good at detecting long stretches of homology; the shorter a homologous region, the less certain it is to be detected by BLAST. We can calculate the probability that BLAST will pick up a region of similarity of any specified length. To attain a 90% probability of detection, a homologous region must be at least 57 amino acids long. We choose this gap length as our cutoff when assembling hits.

Clustering Assembled Hits

Assembling the BLAST hits has addressed one of the sources of noise in the similarity judgments: the fragmentation of essentially unitary regions of similarity. We now want to group these assembled hits into equivalence classes, forming the transitive closure of the pairwise similarity judgements. In the process, we must address the problem of variation in the extents of regions of similarity.

When assigning assembled hits to the same group, the extents of the hits are not required to match exactly. There are two reasons for this. First, the endpoints of a region of similarity identified by BLAST are much less certain than the detection of the region as a whole. Second, evolution may change the ends of a particular domain more rapidly than the central portion, both because ends may play a less important functional role, and because they have to adapt to the portions of the protein that they abut. Hits that should be grouped together may therefore have "ragged ends," and be of somewhat different lengths.

BLAST hits establish equality relations across proteins; the query and subject portions of a hit are nonrandomly similar. However, the ragged ends issue makes it problematic to determine whether two regions from the same protein are in fact the same, and, therefore, whether hits that include those two regions should be placed in the same group. Building equivalence classes is thus a matter of determining when two hits contain references to the same region.

For two hits to share a reference to a particular region, that region must be within a particular protein, and the overlap between hits must be adequate. We only need to

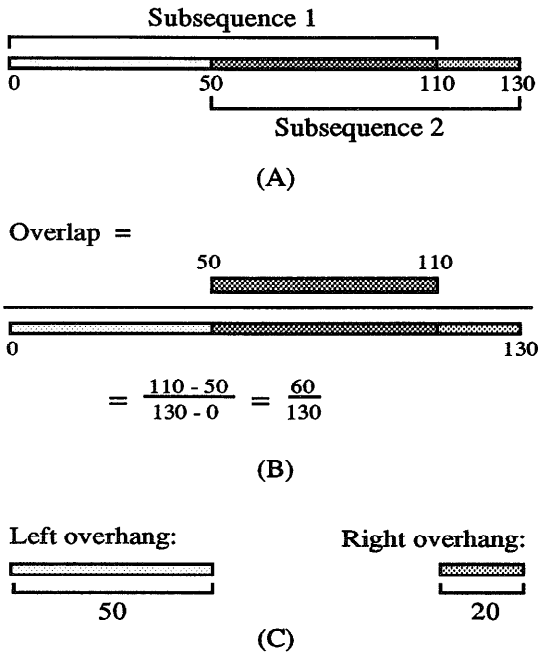


Figure 3: The proportion of overlap between two subsequences from the same protein (A) is the length of the overlap divided by the extended length of the two subsequences (B). The nonoverlapping portions are the overhangs (C).

compare hits that share a protein; call such hits "neighbors." The neighbor list for each pair of proteins is generally quite short, because a given pair of proteins is unlikely to hit each other in many separate places, and even if it does, many of those BLAST hits will have already been assembled. Comparing a hit only to other hits in its neighbor list rather than to *all* the other hits saves a lot of time.

When deciding whether two hits refer to the same region, we must consider both overlap and overhang (see Figure 3). The overlap component is the length of the overlapping portion of two sequences divided by the extended length of the sequences. To be eligible for grouping, two sequences must have a sufficient proportion of overlap. The outcome of the clustering is not highly sensitive to the exact value of this parameter; we set it to 40%. We also require that the unmatched segment at either end (the overhang) must not exceed the maximum allowable length. This overhang is like a gap that falls at the end of a hit, so we can use same analysis of BLAST's sensitivity we used in determining the maximum gap length to set the maximum overhang to 57 amino acids.

Sorting helps us avoid some unnecessary comparisons when checking overlap between pairs of hits, which in the worst case takes O(neighbors^2) time per protein. Each neighbor list is ordered by query start position and query end position. If we get to a neighbor whose start position is greater than the end position of the current hit, we know that no neighbor later in the list will have any overlap with the current hit, so we can cut short our comparisons and move on to the next hit on the list (Figure 4).

The clusterer initially assigns each assembled hit to a separate group. Whenever two hits that have a protein in common are found to have sufficient overlap, the groups that they belong to are checked to see if they should be merged. We have investigated several criteria for deciding when to merge groups. The simplest approach is to merge two groups whenever they are found to share a single sufficiently overlapping region. There is a pitfall in this approach: a single false hit might cause the merging of two groups that don't belong together. Fortunately, this type of error turns out to be extremely rare; we set the stringency for BLAST high enough to make false positives very unlikely.

We also tried requiring groups to share k overlapping hits in order to qualify for merging, where k is a constant that can be set at run time. However, tests on synthetic data showed that the best setting for k is one; higher values decrease the accuracy of the clustering.

In conceptual classification, each object is assigned to exactly one group. In Bayesian classification, objects can be assigned to more than one group as a result of uncertainty, but the underlying finite mixture model assumes that each object intrinsically belongs to a single group. Our method for classifying datastreams allows for some segments not to be classified at all, and for others to be assigned to more than one group. Regions that aren't classified are those that were unique in the database and thus didn't generate any BLAST hits. Regions that get assigned to multiple classes are those that appear in families of multiple domain proteins. For example, if two related proteins each have an *A* domain and a *B* domain, BLAST will find hits between the *A* portions, the *B* portions, and the whole *AB* region. Because the hits have very different extents, the subsequence representing the *A* domain will appear in both the *A* group and the *AB* group. In this problem, a correctly assigned region may genuinely belong to more than one class.

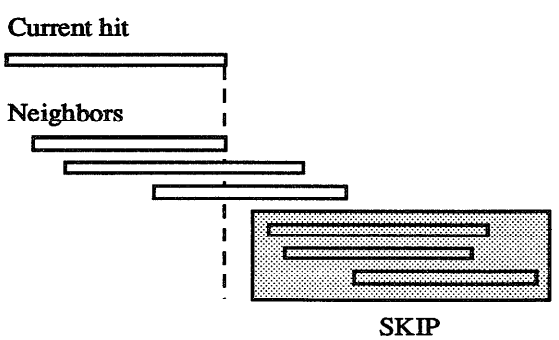


Figure 4: When checking for overlap between the current hit and its neighbors, we need not check any neighbor that starts after the end of the current hit (dotted line).

Testing the Clusterer

In order to validate our clustering approach, we developed a program to simulate protein evolution and provide us with sets of artificial proteins of known evolutionary history.. The evolver starts with a set of user-selected domains and mutates them for the desired number of generations. These synthetic proteins can then be used to test how effective our clustering method is at recognizing domains and grouping them appropriately. We found that after 50 generations of evolution, the clusterer could still recognize domains with 96% accuracy. For more details on the protein evolutionary simulator, see (Hunter, Harris, and States, 1992).

Describing Groups

In contrast with many classification tasks, the classes or groups formed by our program do not have obvious definitions: each group is a set of particular protein subsequences that have been found to resemble each other. Such a list can, by itself, be useful to a biologist. However, in order to use the classification for automated database annotation, we must create comprehensible descriptions of each class. Ideally, we would like to find a natural language description of each group. We are attempting to construct such descriptions by summarizing the names of the proteins in the group, and including additional information about the size of the group and whether the hits in the group contain whole proteins or portions of proteins. Large groups containing hits that are portions of proteins are likely to be protein building blocks. Large groups that contain whole proteins are more akin to PIR superfamilies. Our preliminary group-namer selects words or phrases from the protein names in a group that appear significantly more frequently than in the database as a whole.

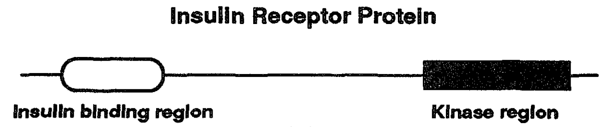
It would also be desirable to be able to describe a group by a *consensus sequence* or a *frequency matrix*. showing the frequency of each amino acid at each position along the set of sequences in a group. A consensus sequence can be derived from a frequency matrix by taking the most common amino acid at each position.

Unfortunately, calculating the frequency matrix for a group is equivalent to the problem of finding the best multiple alignment among a set of sequences. For n sequences of average length k, finding the best multiple alignment takes time $O(k^n)$ (Carillo & Lipman, 1988). Tools exist for finding multiple alignments; these could potentially be used to align the sequences in a group so that the frequency matrix could be determined. Alternatively, it might be possible to use programs that describe groups of related protein sequences, such as (Smith & Smith, 1990) and (Henikoff & Henikoff, 1991), to find patterns describing the groups obtained by our clusterer.

Representing classes with consensus sequences or frequency matrices would make it easier to incrementally classify new proteins. A new protein could be divided into chunks that fit into the already determined classes. This would enable a biologist to look for domains in a newly sequenced protein.

Application to Real World Data

Once the clusterer reached the desired level of performance on the synthetic data, we tried it on several real protein databases, the largest of which is the Non-Redundant Database (NRDB). NRDB is a collection of all of the protein sequences from several large databases (PIR, SwissProt, translated GenBank, and NCBI Backbone) with the duplicates removed. It includes 61,810 proteins comprising over 5 million amino acids.



(A)

Domains found by clusterer:

```
1  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
2  ------@@@------------------------------------------
3  ------------------------------------@@@@@@@@@@@@---
```

Proteins with Domain 1:
A05274 Insulin receptor precursor (version 1) - Human
A05275 Insulin receptor precursor (version 2) - Human

Proteins with Domain 2:
A05274 Insulin receptor precursor (version 1) - Human
A05275 Insulin receptor precursor (version 2) - Human
GQFFE Epidermal growth factor receptor - Fruit fly
GQHUE Epidermal growth factor receptor precursor
TVRTNU Kinase-related transforming protein (neu)

Proteins with Domain 3:
A05274 Insulin receptor precursor (version 1) - Human
A05275 Insulin receptor precursor (version 2) - Human
A25228 Cell division control protein 7 - Yeast
A28163 Protein kinase C delta - Rat EC-number 2.7.1.
A29597 Kinase-related transforming protein - Blowfly
A29872 Phosphorylase kinase gamma (catalytic) chain
and 140 others....

(B)

Figure 5: Two views of the insulin receptor. (A) is the traditional biological view: This protein binds insulin and signals the cell that insulin has been bound by turning on its kinase. (B) shows the view from the induced classification: One domain (1) spans the entire extent of the protein and contains complete insulin receptors. A second domain (2) defines a region which is associated with ligand binding. The third domain (3) defines the kinase domain. Domains (2) and (3) also appear in proteins other than insulin receptors.

The most time-consuming part of the whole clustering procedure is the BLAST run (which took 800 CPU hours on Sun Sparc 2 and Silicon Graphics 2D series workstations). On an IBM 3090 supercomputer, our clustering algorithm took 100 minutes to process and sort the 6.6 million BLAST hits generated by NRDB, 10 minutes to assemble the hits, and 46 minutes to cluster the assembled hits. The resulting classification has 12,548 groups.

Many of our induced domains correlate well with domains or protein families as traditionally defined by the biological community. For example, all 447 globin proteins in the PIR database are classified into a single induced domain, and no extraneous proteins are placed in this class. The whole insulin receptor appears as a single domain in our classification with two subset domains which define the ligand binding and the kinase domain of the receptor (Figure 5).

Manual definitions of patterns or signatures have also been used to define motifs, for example in the Prosite knowledge base (Bairoch, 1991). Our analysis demonstrates several limitations of that approach. Some of the domains that our program found, such as globins and immunoglobulins, cannot be represented by Prosite-like patterns. Second, the manually defined patterns have limited sensitivity and specificity. For example, only 77% of kinases actually match the Prosite kinase signature. Finally, building and maintaining patterns manually is extremely time-consuming. As a result, many recently described domains of considerable biological interest. are not yet available as patterns. Our automatic method solves each of these problems.

Our clustering method is applicable, with minimal modification, to nucleic acid databases. Nucleic acid motifs might include gene families (coding regions), signals such as promoters or splice sites, or repeated elements. We have successfully run our program on nucleic acid databases, but the results have not yet been analyzed.

The HHS approach could also be applied to other problems that involve classifying objects from noisy unsegmented datastreams, such as the problem of understanding continuous speech. A digitized stream representing time-varying sounds would be equivalent to a sequence of amino acids. The goal of the language learner is to find repeating patterns or regularities. This involves a phase analogous to hit assembly, in which noise is filtered out and phonemes are pieced together, and a clustering phase in which patterns are recognized despite differences in detail and duration.

Conclusions

We have demonstrated an efficient method (HHS) for unsupervised classification from a very large, unsegmented datastreams using probabilistic binary similarity judgements. We applied HHS to the problem of discovering protein building blocks from sequence databases. Tests on synthetic protein data showed that our method correctly clusters even relatively distantly related sequences. We then ran our clustering program on the 6.6 million BLAST hits generated by the 5 million amino acid NRDB protein sequence database, which took under three hours of supercomputer time. The induced classes correspond well to those found by biologists. Our program may provide a way of addressing the daunting problem of annotation of rapidly growing protein sequence databases. Moreover, our method is general enough to be applicable to other massive classification tasks.

References

Almuallim, H., & Dietterich, T. 1991. Learning with many irrelevant features. In *Ninth National Conference on Artificial Intelligence*, 547-552. Anaheim, CA: AAAI Press.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. 1990. A Basic Local Alignment Search Tool. *Journal of Molecular Biology* 215:403-410.

Bairoch, A. 1991. *Prosite database 7.10*. Geneva, Switzerland.

Barker, W. C., George, D. G., Hunt, L. T., & Garavelli, J. S. 1991. The PIR Protein Sequence Database. *Nucleic Acids Research* 19 (Suppl):2231-36.

Berg, J. M. 1990. Zinc finger domains: Hypotheses and current knowledge. *Annual Review of Biophysics and Biophysical Chemistry* 19:405-421.

Carillo, H., & Lipman, D. 1988. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math* 48:1073.

Cheeseman, P., Self, M., Kelly, J., Taylor, W., Freeman, D., & Stutz, J. 1988. Bayesian classification. In *Proceedings of AAAI 88*, Saint Paul, Minnesota:

Fisher, D. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2:139-172.

Henikoff, S., & Henikoff, J.G. 1991. Automated assembly of protein blocks for database searching. *Nucleic Acids Research* 19(23):6565-6572.

Hunter, L., Harris, N., & States, D. 1992. Efficient classification of massive, unsegmented datastreams. To appear at *Ninth International Machine Learning Conference*, July 1992, Aberdeen, Scotland.

Kohonen, T. 1988. *Self Organization and Associative Memory*. Berlin: Springer-Verlag.

Schank, R. C. 1991. Where's the AI? *AI Magazine* 12(4):38-49.

Smith, R.F., & Smith, T.F. 1990. Automatic generation of primary sequence patterns from sets of related protein sequences. *Proc. Natl. Acad. Sci* 87:118-122.