Searching for Protein Classification Features

Jennifer A. Smith
Boise State University

Searching for Protein Classification Features

Scott F. Smith
Department of Electrical and Computer Engineering
Boise State University
Boise, ID 83725-2075
sfsmith@boisestate.edu

Abstract- A genetic algorithm is used to search for a set of classification features for a protein superfamily which is as unique as possible to the superfamily. These features may then be used for very fast classification of a query sequence into a protein superfamily. The features are based on windows onto modified consensus sequences of multiple aligned members of a training set for the protein superfamily. The efficacy of the method is demonstrated using receiver operating characteristic (ROC) values and the performance of resulting algorithm is compared with other database search algorithms.

1 Introduction

The need for faster sequence classification algorithms has been demonstrated by (Cameron et al 2004). They note that search times for the GenBank (Benson et al 2004) sequence database have increased by seven fold over the period 1999 to 2003 due to the exponential increase in the number of available sequences. This seven-fold increase would have been even greater if not for the increase in speed of available processors over the same period. This trend is expected to continue.

In this paper, we attempt to increase the speed of database search algorithms by finding features in the sequences of a protein superfamily which are as unique as possible. The discovered features will then be used for ungapped matching to a query sequence. These features will be defined as subsequences of a modified version of the consensus sequence of the multiple alignment of the sequences used to train for a superfamily.

In order to determine how well the superfamily classification works, a set of superfamily classifications will be used that are determined independently of the protein primary sequence. This can be done for those proteins that have known three dimensional structure. Such a classification is available in the form of the ASTRAL Compendium for Sequence Structure Analysis database (Chandonia et al 2004). ASTRAL contains sequences annotated with superfamily classification (among other things) for those proteins represented in the Protein Data Bank (Berman et al 2000). Superfamily classifications correspond to those used by the Structural Classification of Proteins (SCOP) database (Andreeva et al 2004). Version 1.67 of ASTRAL is used with sequences containing more than 40% sequence identity eliminated. Sequence redundancy elimination to 40% identity is done automatically by the ASTRAL web server.

The metric used for classification accuracy determination is the receiver operating characteristic (ROC). This value summarizes the sensitivity and specificity of the algorithm in a single value between 0 and 1, where higher values are better. The use of the ROC to measure search accuracy is discussed in (Gribskov and Robinson 1996).

Comparisons of the speed of a protein classification algorithm based on the classification features is compared with the Smith-Waterman (Smith and Waterman 1981), Fasta (Pearson and Lipman 1985), and BLAST (Altschul et al 1997) scoring methods and the proposed algorithm is found to be faster. The speed improvement comes from the small number of ungapped comparisons that are required.

The form of the features used is discussed in Section 2. The use of evolutionary computation to search the very large space of possible features for a good feature set is shown in Section 3. In Section 4, an algorithm for scoring possible superfamily membership of a query string is introduced. The effectiveness of the algorithm in terms of ROC scores for the ASTRAL database is shown in Section 5. Section 6 discusses the speed of the algorithm and Section 7 concludes.

2 Definition of Features

The usual definition of a consensus sequence is a sequence which has the symbol that occurs most frequently at each position of a multiple alignment. The definition of a consensus sequence which is used in this paper will instead be a sample sequence drawn from a population of sequences with the probability of a particular symbol at each position equal to the observed frequency of that symbol in the multiple alignment. This can also be thought of as a sample from the position-specific scoring matrix (PSSM) of the multiple alignment (Mount 2001, pp. 192-198). The mode of the consensus sequence definition in this paper is therefore the usual definition of consensus sequence.

A feature set for a particular protein family will be defined by a particular consensus sequence draw as defined above in combination with a binary window

0-7803-9363-5/05/$20.00 ©2005 IEEE.

function which selects which ranges of the consensus sequence to include and which to omit. The window function will contain one or more groups of ones which represent regions of the consensus sequence which must be matched in an ungapped manner when producing the score for a query sequence.

The number of combinations of possible consensus sequences and window functions is very large and there is no hope of trying them all. The measure of fitness of a given combination of consensus sequence and window function will be the ROC score for protein classification taken using the training set for a protein superfamily. Calculation of this ROC score is itself not trivial, making exhaustive search completely infeasible.

3 Method for Searching for Good Features

A genetic algorithm is used to find good features. An individual consists of a consensus sequence and a binary window function. A mutation in the consensus sequence takes the form of redrawing a representative residue at a randomly selected multiple alignment position from the observed frequency of residues at that position in the training set. A mutation in the window function takes the form of changing a 0 to a 1 or vice versa, where the change is constrained to take place at one of the two positions adjacent to a change location in the initial window function. This is equivalent to allowing a group of ones to lengthen or shorten by one position. It also allows groups to merge and to disappear. Insertions and deletions are also allowed to take place in the window function by changing a contiguous range of positions all to ones or all to zeros.

An initial population of size N is created by generating N consensus sequences from the observed distribution of residues at each multiple alignment position in the training set for the superfamily. The window functions for each member of the initial population are generated by randomly selecting one, two, or three groups of length uniformly distributed between three and ten positions to be set to one and all other positions remain zero.

At each generation, each individual is scored against every sequence in the database and a ROC value calculated. The scores are found using the BLOSUM62 substitution matrix (Henikoff and Henikoff 1993) and matching the consensus sequence groups determined by the window function to the query sequence in an ungapped fashion. The consensus sequence group is paired with every possible remaining substring of the query sequence with the same length and the maximum-valued pairing selected. A remaining substring is part of the database sequence starting at a higher position number than the last position number which generated the maximum value for the previous feature. The partial score determined from each feature is summed to get the final score.

Figure 1 shows the ranges used for ungapped alignment for the partial score associated with each feature. In the figure "Feature #1 Range" is shown to cover all the residue positions in the entire query sequence. If the range for feature #1 is residue positions 1 through M and feature #1 is labeled with residue positions 1 through K, then feature #1 positions will first be paired with query sequence positions 1 through K, then 2 though K+1, and so on until feature #1 positions 1 through K are finally paired with query sequence positions M-K+1 through M. Over the M-K-1 pairings, the maximum valued pairing is found and is shown in Figure 1 as "Feature #1 Max." The range for feature #2 then starts one residue toward the C-terminal end of the query sequence and goes to the end of the sequence (protein sequences are conventionally written starting at the N-terminal end of the protein and ending with the C-terminal end).

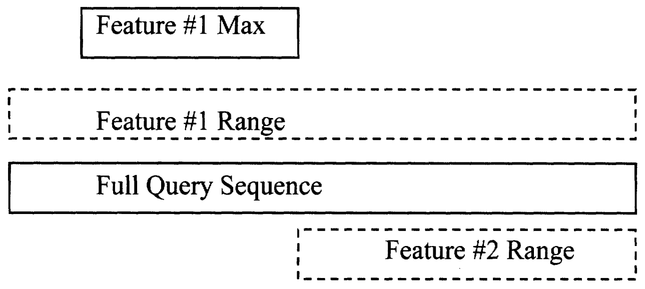

Figure 1: Scoring of a Database Sequence

The ROC value is found using the equation

$$ROC_n = \frac{1}{nL} \sum_{1 \le F \le n} u^F ,$$

where n is a false-positive cutoff (with n = 50 used in this study) and L is the number of true positives found before n false positives are found. The value u^F is the number of true positives ranked before false positive number F. A ROC_{50} value of 1.0 means that all 50 false positives rank below all of the true positives (up to the cutoff) and a score of 0.0 means that 50 false positives are ranked at the top of the list.

The population size used to generate the results of this paper is N = 101. At each new generation, the best individual was retained and five copies each of the twenty best individuals were generated and subjected to possible mutation. The consensus sequences of each individual were mutated by redrawing the consensus residue from the observed distribution with probability 0.001 per residue position. The window functions were then mutated with the following five operations in the order stated: crossover, delete group, insert group, lengthen group, shorten group. With probability 0.02 per individual, the window function of an individual was crossed with another individual with crossover point uniformly distributed over the length of the window function (which is the same for all individuals). With probability 0.02 per individual, one group of contiguous

1s was changed to 0s within the individual with the group deleted selected uniformly. If an individual only had one group, the delete operation was skipped. With probability 0.02 per individual, a range of window function positions was set to 1 where the length of the range was uniformly distributed between three and ten and the start position uniformly distributed along the window function length. This group insertion operation may or may not increase the number of groups depending on whether the inserted group ends in the middle of an existing group or not. With probability 0.2 per group end (0 to 1 or 1 to 0 transition), the 0 adjacent to the group end was changed to 1 (group lengthening operation). With probability 0.2 per group end, the 1 adjacent to a group end was changed to a 0 (group shortening operation). The GA was iterated for twenty generations and the best individual from generation twenty taken as the solution.

4 Search Algorithm Based on the Features

Searching for a superfamily similar to a query string is a two step process. In the first step, a database of sequences annotated with superfamily information is used to find a set of good features for each superfamily in the database. This step needs to be done only once for the database and can therefore be rather computationally expensive, as long as it is not so expensive as to be infeasible. The second step needs to be repeated for every query sequence. In this step, the set of features for each superfamily must be scored against the query sequence. This step needs to be fast.

The second step is done exactly as the scoring was done in the feature search of step one. The features for a given superfamily are an ordered set. The first feature is compared to every possible overlap range of the full query sequence. The highest valued comparison is taken as the partial score for feature #1. The comparisons are done as an ungapped alignment using any standard substitution matrix. The location in the query sequence is noted and feature #2 is compared using ungapped alignment in the portion of the query sequence starting one position beyond the C-terminal end of the feature #1 location. This process is continued for all features of the superfamily. If any feature does not fit into the remaining query sequence range it and all higher numbered sequences receive a partial score of zero. The overall score is the sum of the partial scores.

A large speed advantage is expected from the algorithm since the number of residues in the feature sets for each superfamily will be much less than the total number of residues in the database and the matching is an ungapped alignment. The number of residues is much smaller than the database since the sum of the lengths of the features will generally be less than the average length of the sequences in the superfamily and there is only one set of features covering all the sequences in the superfamily. The cost of this speed advantage will be a one-time preprocessing of the database to find the features.

5 Efficacy of the Search Algorithm

The effectiveness of the feature-based search algorithm for finding members of a superfamily in the ASTRAL 1.67 database limited to 40% sequence identity is shown in Table 1. All superfamilies in the database with at least 40 members are shown. This was done so that 20 superfamily members could be used to estimate the features and at least 20 superfamily members would be left as possible true positives when finding ROC values. The superfamily members used to estimate features were removed from the database so that the training set and test set would be independent. In actual application of the algorithm all known members of a superfamily would be used to estimate the features. The column labeled N is the number of superfamily members in the database.

TABLE 1
ROC_{50} FOR SUPERFAMILIES WITH AT LEAST 40 MEMBERS

Super family	**N**	**SW ROC_{50}**	**BLAST ROC_{50}**	**Query Seq.**	**Feat. ROC_{50}**	**# of Feat.**
a.4.1	43	0.640	0.850	d1hlva2	0.800	1
a.4.5	84	0.560	0.610	d1pp7u_	0.800	2
a.39.1	41	0.930	0.837	d1hqva_	0.942	1
b.1.1	122	0.573	0.518	d1cid_2	0.806	3
b.1.18	49	0.787	0.000	d1soxa1	0.000	1
b.1.2	52	0.505	0.660	d1f42a2	0.630	2
b.29.1	45	0.833	0.925	d1dypa_	0.440	1
b.40.4	65	0.895	0.210	d1pyba_	0.320	1
c.1.8	77	0.880	0.646	d1ug6a_	0.740	1
c.2.1	132	0.903	0.820	d1vj0a2	0.473	4
c.3.1	53	0.340	0.570	d1lqta1	0.685	3
c.37.1	158	0.000	0.405	d1qhxa_	0.678	3
c.47.1	64	0.834	0.500	d1eeja1	0.984	1
c.66.1	53	0.723	0.797	d1fp1d2	0.748	2
c.67.1	43	0.887	0.885	d1uu1a_	0.470	3
c.69.1	61	0.820	0.482	d1r1da_	0.540	2
Mean:		0.694	0.607		0.629	

The ability of the proposed algorithm to find superfamily members is compared with that of the Smith-Waterman and BLAST algorithms. One superfamily member was chosen at random from each superfamily to be a query sequence for the Smith-Waterman and BLAST searches. The ASTRAL database name for this sequence is given in the "Query Seq." column of Table 1 for the sake of reproducibility. The ASTRAL database was then searched using the SSEARCH and BLAST programs and ROC values calculated from the results. These ROC values are listed in the "SW ROC_{50}" and "BLAST ROC_{50}" columns of Table 1. The mean ROC value over the sixteen superfamilies is 0.694 using Smith-Waterman and 0.607 using BLAST.

The ROC values for the proposed algorithm were calculated by searching the ASTRAL database (with the 20 superfamilies used to estimate the features excluded) with the set of features for that superfamily. The resulting ROC values are shown in the Features ROC_{50} column of Table 1. The mean ROC value using the proposed algorithm is 0.629, which is not very much less than the Smith-Waterman value and slightly more than the BLAST value. The ROC value difference of 0.065 between Smith-Waterman and the proposed algorithm is equivalent to moving the true positives in a ranked list including all true positives and fifty false positives down an average of a little more than three positions.

TABLE 2
FEATURE #1 FOR EACH SUPERFAMILY

Super-family	Feature #1
a.4.1	VWFQNRR
a.4.5	LDAEEFKLLTLISIIEEGELTVKEIAEAL
a.39.1	AEVDEMFKELDTNGDGEIDFEEFLRLV
b.1.1	SLLVKEGETVTLSC
b.1.18	VGAASTIGGNVITVLGKGDVKASALLYFG
b.1.2	NSTITGYRVTYVPKN
b.29.1	IVNGPWHNVGWGREERSVVLPFDSGK
b.40.4	GKEAEVVAELLLKE
c.1.8	VVGADPDKAVTFVDNH
c.2.1	GPGGSVVVVGAAGAVGLVAVQIAKALG
c.3.1	IIVGAGLSGLAAAYRLSEAGKNVLLVE
c.37.1	RIVIEGPPGAGKST
c.47.1	PWCGPCKAAKP
c.66.1	ALFLLLPLNADARLLKDVLLEAFDDDKLSALVKK L DL
c.67.1	PNNPTGLVPPLELGEDIVDHAATKGINGHSDAAYG GFAAG
c.69.1	SYDGDYLAAGENVIVV

TABLE 3
FEATURE #2 FOR EACH SUPERFAMILY THAT HAS ONE

Super-family	Feature #2
a.4.5	AVVRAIKKLEDKGLISR
b.1.1	STLTITSAQPEDSATYYC
b.1.2	YEVSVIALNGRGES
c.2.1	VDFALDTVG
c.3.1	LVGAKLAAAGREILSVARKEDQEIQSLR
c.37.1	LLRDAEEIGKDLGFPAARYLDGDMIIADLLLELAL LK
c.66.1	LNGKGLLSILPMRRAIDATAREDDGAREIKLSKEA GF
c.67.1	AVLSAFSKAFGLRG
c.69.1	GLEDQLAALEWLKENAAAFGGDPKRITIFGESAG GLSVAALLLLPLDK

TABLE 4
FEATURE #3/#4 FOR EACH SUPERFAMILY THAT HAS THEM

Super-family	Feature #3
b.1.1	DKLIFGQG
c.2.1	VGAPGGALTAP
c.3.1	IQTADGSKGANIVVSADGTF
c.37.1	FPRRARQAEALDEAL
c.67.1	LRISVGIEDLDDLLADLEAAL
	Feature #4
c.2.1	NKSKFEEALDFLAQG

The discovered features for each superfamily are shown in Table 2 (first feature), Table3 (second feature), and Table 4 (third and fourth features). None of the sixteen cases resulted in more than four features and only one case had a fourth feature. The average number of residues in a feature is 19.1 and the average number of features is just under 2. The features tend to come from positions in the multiple alignment which are highly conserved. That is, features tend to come from portions with few gaps and where substitution matrix values are generally large and positive.

The multiple alignments of the twenty training sequences from each superfamily were found using CLUSTALW (Higgins and Sharp 1988).

6 Performance of the Search Algorithm

The inner loop of the proposed search algorithm involves taking a sum of substitution matrix values and then comparing the sum to a previous maximum sum. For an average length feature, this means looking up 19 substitution matrix values, summing the values, and comparing the sum to the maximum sum generated at all preceding N-terminal positions. The 19 table look-ups and additions dominate the computation.

The Smith-Waterman algorithm also requires a substitution matrix value look-up and an addition for each amino-acid match investigated as well as many additional computations at each attempted match. As a result, counting the number of compared amino acids in each algorithm should give an indication of the relative speed of the two algorithms which underestimates the computation time of Smith-Waterman relative to the proposed algorithm. The equations used by the Smith-Waterman algorithm are:

$$I_{i,j} = \max\{I_{i-1,j} - c,\ M_{i-1,j} - g\}$$

$$D_{i,j} = \max\{D_{i,j-1} - c,\ M_{i,j-1} - g\}$$

$$M_{i,j} = \max\{I_{i-1,j-1} + d(a_i,b_j),\ D_{i-1,j-1} + d(a_i,b_j),\ M_{i-1,j-1} + d(a_i,b_j),\ 0\}.$$

I is the score if the current sub-alignment ends with an insert, D is the core if the sub-alignment ends with a delete, and M is the score if the sub-alignment ends with a match or mutation. The penalty for initiating a gap is g and for continuing a gap is c.

The value $d(a_i,b_j)$ is the substitution matrix lookup-value which is added to three different values. Four additional subtractions, a four-way maximum, and two two-way maximums are also performed. All of these operations are done every time two amino acids are compared. The single maximum done in the proposed algorithm is done once for every sum (an average of once per 19 amino acid comparisons).

The number of amino acid comparisons for the Smith-Waterman algorithm is relatively easy to estimate. Smith-Waterman does a compare for every possible pair of residues in the query string versus the database string. For an average case we expect this algorithm to do a number of amino acid compares equal to the square of average length of a protein sequence. This is actually an underestimate since the square of an expected value is less than the expected value of the square of the same random variable that produces values greater than one.

The number of amino acid comparisons done by the proposed algorithm is harder to estimate since the number of comparisons done for feature #2 or higher depends on where the best match was found for feature #1. In order to continue overestimating the compute time of the proposed algorithm relative to Smith-Waterman, we shall assume that the best match is at the first (N-terminal) position of the query sequence. For an average length feature this means 19*(L-19) comparisons for feature #1, where L is the length of the query string. For feature #2, there would be 19*(L-2*19) comparisons when the best match for feature #1 was N-terminal. For feature #n, the number of matches is 19*(L-n*19) for average length features.

The average sequence length in the ASTRAL 1.67 database limited to no more than 40% sequence identity is 181 (L = 181). There are 6600 sequences in the database representing 1445 superfamilies. The average number of discovered features in the sixteen cases examined above was two, so the number of amino acid comparisons for a typical case using the proposed algorithm is estimated to be 19*(181-19) + 19*(181-38) = 5795. For the Smith-Waterman algorithm, the estimated number of comparisons is L^2 = 32,761. In addition to a more than five-fold decrease in the number of amino acid comparisons for each score generated, the number of times that the query sequence needs to be scored is reduced. With Smith-Waterman, the query string is scored against every database sequence (with multiple sequences existing for most superfamilies). With the proposed algorithm, the query string is scored once against one set of features for each superfamily. For the ASTRAL database figures above, this means that 6600 / 1445 = 4.57 times as many scores are generated with Smith-Waterman. Since most sequence databases with superfamily annotation include far more sequences per superfamily, this advantage of the proposed algorithm is being underestimated here. For example, the Pfam database (Bateman et al 2004) has thousands of members associated with many of its classifications. The overall result in this study is that the Smith-Waterman algorithm does 4.57 * (32,761 / 5795) = 25.8 times as many amino acid comparisons. It is likely that the additional computation time for Smith-Waterman is much more that 25.8 times as great.

Table 5 shows the relative execution times of the proposed algorithm as well as several popular pair-wise database search algorithms. The relative speed of the proposed algorithm ("Features") and the Smith-Waterman algorithm were determined above. The speed of two flavors of Fasta and of BLAST relative to Smith-Waterman are from (Brenner et al 1998).

TABLE 5
RELATIVE EXECUTION TIMES

Features	1.0*
Smith-Waterman	25.5
Fasta (ktup = 1)	3.9
Fasta (ktup = 2)	1.4
BLAST	1.0

* Very conservative estimate. Expected to be much lower in practice.

7 Conclusions

A new algorithm for classifying a query sequence into a protein superfamily has been introduced. The algorithm is faster than the Smith-Waterman algorithm at the expense of significant one-time processing of the database to find superfamily features which can be used for ungapped alignment during query sequence scoring. The algorithm has been found to be only slightly less effective at superfamily classification than the Smith-Waterman algorithm. Determination of a good set of features for a superfamily during the preprocessing phase is accomplished using a genetic algorithm.

Future work on this algorithm will examine ways to make the preprocessing step converge more rapidly on a solution either by altering the way the initial population is chosen or changing the way new generations are selected and modified. Also, the fitness function may be modified to favor shorter and fewer features in an attempt to further reduce the time required for scoring at a hopefully small penalty in reduced ROC value.

Acknowledgments

The project described was supported by NIH Grant Number P20 RR016454 from the INBRE Program of the National Center for Research Resources.

Bibliography

S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman (1997) "Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs," *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389-3402.

A. Andreeva, D. Howorth, S. Brenner, T. Hubbard, C. Chothia, and A. Murzin (2004) "SCOP Database in 2004: Refinements to Integrate Structure and Sequence Family Data," *Nucleic Acids Research*, vol. 32, no. 1, pp. D226-D229.

A. Bateman, L. Coin, R. Durbin, R. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. Sonnhammer, D. Holme, C. Yeats and S. Eddy (2004) "The Pfam Protein Families Database," *Nucleic Acids Res.*, vol. 32, pp. D138-D141.

D. Benson, I. Karsch-Mizrachi, D. Lipman, J. Ostell and D. Wheeler (2004) "GenBank: Update," *Nucleic Acids Research*, vol. 32, no. 1, pp. D23-D26.

H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Borne (2000) "The Protein Data Bank," *Nucleic Acids Research*, vol. 28, pp. 235-242.

S. Brenner, C. Chothia, and T. Hubbard (1998) "Assessing Sequence Comparison Methods with Reliable Structurally Identified Distant Evolutionary Relationships," *Proc. of the National Academy of Sciences*, vol. 95, pp. 6073-6078.

M. Cameron, H. Williams, and A. Cannane (2004) "Improved Gapped Alignment in BLAST," *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, vol. 1, no. 3, pp. 116-129.

J. Chandonia, G. Hon, N. Walker, L. Lo Conte, P. Koehl, M. Levitt, and S. Brenner (2004) "The ASTRAL Compendium in 2004," *Nucleic Acids Research*, vol. 32, no. 1, pp. D189-D192.

M. Gribskov and N. Robinson (1996) "Use of Receiver Operating Characteristic (ROC) Analysis to Evaluate Sequence Matching," *Computers and Chemistry*, vol. 20, pp. 25-33.

S. Henikoff and J. Henikoff (1993) "Performance Evaluation of Amino Acid Substitution Matrices," *Proteins Struct. Funct. Genet.*, vol. 17, pp. 49-61.

D. Higgins and P. Sharp (1988) "CLUSTAL: A Package for Performing Multiple Sequence Alignment on a Microcomputer," *Gene*, vol. 73, pp. 237-244.

D. Mount (2001) *Bioinformatics: Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press.

W. Pearson and D. Lipman (1985) "Improved Tools for Biological Sequence Comparison," *Proceedings of the National Academy of Sciences*, vol. 85, no. 8, pp. 2444-2448.

T. Smith and M. Waterman (1981) "Identification of Common Molecular Subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195-197.