Evolving Perl code for protein secondary structure prediction.

Robert M. MacCallum
Stockholm Bioinformatics Center, Stockholm University, 106 91 Sweden

1 Introduction & Summary

Progress in the area of secondary structure prediction has been frustratingly slow[6]. The most accurate predictors at the moment are trained to predict one of three secondary structural states (helix, strand or coil) for each residue at position i using sequence information from a "window" of residues $i \pm 7$. Information from more distant sequence positions should improve predictions further, since it is assumed that non-local interactions, like those that occur in sheet formation, can modulate the innate secondary structure preferences of a residue and its near neighbours. However, simply using a larger window does not help. First, the information content decreases rapidly as one moves away from i; because the likelihood that these residues are close in 3D space is also diminishing. Second, there is the problem of a using a fixed window to capture information from variable length secondary structures.

Recently, attempts have been made to incorporate non-local information in secondary structure predictions. Baldi and coworkers[5] have used recurrent multi-pass neural networks and have shown that information from residues $i \pm 15$ influences their predictions. Bystroff and coworkers[2] have taken another approach, which is to combine local predictors for secondary and supersecondary structures into a single large hidden Markov model which simultaneously takes into account context effects throughout the sequence. The more successful *ab initio* 3D predictors, such as Rosetta[1], may also have the side effect of producing more accurate secondary structure predictions. Unfortunately, these three approaches have not yet been shown to be superior to the established predictors in terms of percent predicted correctly into helix, strand or coil (Q_3). The best Q_3 currently stands at around 76%[4], and any future improvement on this will be a strong indicator of the successful incorporation of long-range and/or folding information into the predictors.

This paper describes another attempt to increase secondary structure prediction accuracy using long-range information. The main assumption made in this work is that some form of computer program exists, at least in theory, which can do this. Such a program might mimic the folding dynamics in some way, perhaps in one, two or three dimensions using a reduced complexity model. For example, a predictor could have a simple rule: "PREDICT weak-strand-region AS strand IF number-of-already-assigned-strands $\geq$ 2". This rule could, for example, be applied after assigning "strong-strand-regions", but before assigning regions with helical sequence patterns. The rationale here is that strands might be more likely to form in the context of an already forming sheet.

Why has such a computer program not already been written? Attempts to simplify folding simulations have not been very successful in real-world applications. Most of this research has taken a top-down approach: starting with a full atom model, then reducing its complexity until it runs in "reasonable time". However, this may still be too slow to allow proper optimisation of parameters. In this work I take a bottom-up approach: that is, to start with very simple abstract models and improve on them step by step, using a technique called genetic programming (GP).

GP is an evolutionary computation technique for generating small programs or subroutines by processes analogous to selection, reproduction and mutation in populations of living organisms. Here, as in many other implementations, the "genome" of a program is stored and manipulated as a tree, which directly corresponds to the parse-tree of the program.

In brief, the GP system uses a tournament-based selection system with ageing (death of individuals which have taken part in too many tournaments). Crossover points are chosen at random but are biased towards pairs of subtrees with similar sizes and content (to mimic homologous recombination). Mutations are of the following types: point mutation (terminal or internal), insertion and deletion of subtrees at (also internal or terminal), swapping, copying and replacing subtrees. Multiple crossovers and mutations are possible and occur with a per-node probability. All tree operations respect the strict typing

of nodes. Fitness functions are based on Q_3 and include soft-maximum penalty terms for tree size and execution time.

Both the evolved programs and the system which evolves them are written in Perl. Perl is a good choice for bioinformatics research because it allows rapid prototyping and has a powerful regular expression engine and easy string handling. Regular expressions are a special language for pattern matching in strings. While they always give a yes or no answer they can be written to allow flexibility in the number and type of matches, which should be useful for matching variable length secondary structures in proteins. The main use of regular expressions in protein sequence analysis to date has been the manual discovery of family/function specific PROSITE patterns[3], but few of these have flexible spacing.

The evolved programs are built from a few high-level Perl constructs according to a grammar. The grammar basically allows any number of `scan()` function calls and `if-then` statements. The `scan()` function takes two arguments: a regular expression (for matching the amino acid sequence) and the type of secondary structure to assign at all the positions where the regular expression matches. The `if-then` flow control statement is conditional upon matching of regular expressions against either the full amino acid sequence or the sequence of partially predicted secondary structure of the protein being predicted. It is the feedback from predicted structure which should allow dynamic behaviour to emerge.

A population of 2000 predictors takes a few days to evolve on a single CPU, and the prediction accuracy is disappointingly low. Individual predictors attain a Q_3 of around 55% and the majority vote of many of these predictors (from different populations) achieves a Q_3 of 58%. A state-of-the art predictor would be expected to get around 65% Q_3 using single sequences as input. In the next section we describe some results in more detail and discuss the apparent limitations of the GP-evolved regular expression approach.

2 Results & Discussion

2.1 Reducing the search space, tweaking the grammar

Three versions of the GP system were run, each with 30 independent populations of 2000 individuals for 60 hours. The standard system is defined using "grammar A", shown in Figure 1. If the three-part regular expression passed to the scan() function matches, only the sub-sequence corresponding to the second part is assigned a new secondary structure. The output secondary structure string is initialised with an unpredicted state, "U", and is altered as a side-effect of the scan() function.

Even a simple regular expression that matches two neighbouring amino acids can have $20 \times 20 = 400$ different combinations. Therefore steps are taken to reduce the size of the search space. "Grammar B" has a hierarchical form of $\langle AA \rangle$ which includes expert knowledge about the chemical properties of different amino acids (such as hydrophobic, aromatic or polar nature). "Grammar C" uses fixed amino acid groupings gathered from a sampling of best-of-tournament programs from a previous grammar B run.

As expected, the addition of prior knowledge significantly speeds up the training process, presumably as a result of reducing the number of "meaningless" regular expression combinations. After the 60h run, grammar C reaches the highest Q_3 of 54.8% (mean of 30 populations), compared to 52.3% and 53.9% for grammars A and B, respectively, on an unseen test set. It is expected, however, that the other grammars would reach this level, given enough time and large enough populations.

2.2 Evolved programs and their behaviour

While the regular expression based predictors are not competitive with the state-of-the-art neural network based predictors, they do give a different view of the protein sequence to structure relationship. The programs have an explicit ordering of the scan() functions and this might suggest something about the dynamics of secondary structure formation.

Another 30 runs similar to Grammar A were performed on training and test sets with equal amounts of helix and strand. In these runs, the output state was uniformly initialised with either "H", "E" or "C" specified by the evolved code. At each sequence position in the test set data, the "history" of secondary structural states was recorded for each of the 30 best-of-tournament programs (after 60h). The most common transitions are as follows: C 20.15%, C→H 15.38%, C→E 11.18%, H 8.50%, C→H→E 7.44%, H→E 6.05%, E→C 4.41%, C→E→H 3.97%, H→C 3.94%, E→H 3.82%, E 3.41%.

There appears to be a preference in this sample of 30 programs to initialise with coil

Grammar A

$\langle STAT \rangle \rightarrow$ scan($\langle SEQRE \rangle, \langle SS \rangle$); | $\langle STAT \rangle \langle STAT \rangle$ | if $\langle COND \rangle$ then $\langle STAT \rangle$

$\langle SEQRE \rangle \rightarrow (\langle R \rangle)(\langle R \rangle)(\langle R \rangle)$

$\langle SS \rangle \rightarrow$ H|E|C [H=helix E=strand C=coil]

$\langle COND \rangle \rightarrow$ match(sequence, $\langle R \rangle$) | match(secondary_structure, $\langle S \rangle$)

$\langle R \rangle \rightarrow \langle R \rangle \langle R \rangle$ | [$\langle AA \rangle$] | [^$\langle AA \rangle$] | ($\langle R \rangle$){$\langle N \rangle, \langle M \rangle$}

$\langle AA \rangle \rightarrow \langle AA \rangle \langle AA \rangle$|A|C|D|E|F|G|H|I|K|L|M|N|P|Q|R|S|T|V|W|Y

$\langle S \rangle \rightarrow$ [similar to $\langle R \rangle$ but with a 3-letter alphabet: H,E,C]

Grammar B (modifications)

$\langle AA \rangle \rightarrow \langle AA \rangle \langle AA \rangle | \langle HPHOB \rangle | \langle AROM \rangle | \langle POLAR \rangle |$...|A|C|D|E|F|G|H|I ...

$\langle HPHOB \rangle \rightarrow \langle HPHOB \rangle \langle HPHOB \rangle$|I|L|V|F|M

Grammar C (modifications)

$\langle AA \rangle \rightarrow$ GP|P|G|DGP|DGNPS|IV|FIV|CFIVWY|FILVY|CFILMVWY|ADEGHKNPQRS

Figure 1: Simplified grammars used for GP secondary structure prediction

(C), followed by helix (H). The most interesting observation is that (C→)H→E transitions are more common than (C→)E→H transitions, suggesting that a more successful strategy is to initially over-predict helix and later revert some helix to strand. The evolved regular expressions tell us even more; many helix patterns actually specify "not coil", such as `(.{1,2})(([^PG]){8,})([^P])`. Positive helix patterns also evolve, such as `([^F])((([^G]){1,2}[AELQ]){3,})([^Q])`, but these are quite rare. Although the sequence signals for strand are ultimately more long-range in nature, at a local level helix patterns are slightly more complex, involving a repeated motif of 3 or 4 residues compared to the simple alternating of strands (for example, `()(([V][^GP]){1,8})()`). Clearly GP would find it easier to learn the shorter patterns.

To state that there is a direct correspondence between the behaviour of real folding proteins and these simple evolved pattern matching predictors would be going to far. However, both systems are self-organised and initially rely on just local information, so further investigation could be worthwhile.

The real disappointment was that no programs evolved which used feedback from already predicted secondary structure, even though this was defined in the grammar. Perhaps the fitness landscape of programs with such complex internal dependencies is too rough, or could the predicted secondary structure information be of too low quality? Also note that the effective amount of training data is drastically reduced at the "per sequence" level compared to the "per short stretch of residues" level. The same reasoning may explain why few interesting long-range regular expressions evolved.

2.3 Future and other work

It seems necessary to force the GP programs to evolve interesting dynamic behaviour and to use long range information, and this is the current direction of my research. Cellular automata-like systems show interesting dynamics and the possibility to compute global information using local rules. I am also evolving programs which successfully improve neural network predictions.

References

[1] R. Bonneau, J. Tsai, I. Ruczinski, D. Chivian, C. Rohl, C. E. Strauss, and D. Baker, *Rosetta in CASP4: Progress in ab initio protein structure prediction*, Proteins: Struct., Funct., Genet. **45(S1)** (2001), 119–126.

[2] C. Bystroff, V. Thorsson, and D. Baker, *HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins*, J. Mol. Biol. **301(1)** (2000), 173–190.

[3] L. Falquet, M. Pagni, P. Bucher, N. Hulo, C. J. Sigrist, K. Hofmann, and A. Bairoch, *The PROSITE database, its status in 2002*, Nuc. Ac. Res. **30(1)** (2002), 235–238.

[4] D. T. Jones, *Protein secondary structure prediction based on position- specific scoring matrices*, J. Mol. Biol. **292** (1999), 195–202.

[5] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi, *Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles*, Proteins: Struct., Funct., Genet. **47(2)** (2002), 228–235.

[6] B. Rost, *Review: protein secondary structure prediction continues to rise*, J. Struct. Biol. **134(2-3)** (2001), 204–218.