Evolution of a Computer Program for Classifying Protein Segments as Transmembrane Domains Using Genetic Programming[*]

John R. Koza

Computer Science Department
Stanford University
Stanford, CA 94305-2140 USA
Koza@CS.Stanford.Edu
PHONE 415-941-0336

Abstract
The recently-developed genetic programming paradigm is used to evolve a computer program to classify a given protein segment as being a transmembrane domain or non-transmembrane area of the protein. Genetic programming starts with a primordial ooze of randomly generated computer programs composed of available programmatic ingredients and then genetically breeds the population of programs using the Darwinian principle of survival of the fittest and an analog of the naturally occurring genetic operation of crossover (sexual recombination). Automatic function definition enables genetic programming to dynamically create subroutines dynamically during

classification task. In their first experiment, they equaled the error rate of the best of three human-written algorithms for this classification task.

Genetic programming is a domain-independent method for evolving computer programs that solve, or approximately solve, problems. To accomplish this, genetic programming starts with a primordial ooze of randomly generated computer programs composed of the available programmatic ingredients, and breeds the population or programs using the Darwinian principle of survival of the fittest and an analog of the naturally occurring genetic operation of crossover (sexual recombination). Automatic function definition enables genetic programming to dynamically create subroutines dynamically during the run.

that part of the protein is located on one side of the membrane, part is within the membrane, and part is on the opposite side of the membrane. *Transmembrane proteins* often cross back and forth through the membrane several times and have short loops immersed in the different milieu on each side of the membrane. The length of each transmembrane domain and each loop or other non-transmembrane area are usually different. Transmembrane proteins perform functions such as sensing the presence of certain chemicals or certain stimuli on one side of the membrane and transporting chemicals or transmitting signals to the other side of the membrane. Understanding the behavior of transmembrane proteins requires identification of their transmembrane domains.

Biological membranes are of hydrophobic (water-hating) composition. The amino acids in the transmembrane domain of a protein that are exposed to the membrane therefore have a pronounced tendency to be hydrophobic. This tendency toward hydrophobicity is an overall distributional characteristic of the entire protein segment (not of any particular one or two amino acids of the segment). Many transmembrane domains are α-helices, so all the residues of the helix are exposed to the membrane (and are therefore predominantly hydrophobic). Although some transmembrane domains are β-strands (so that only some residues that are actually exposed to the membrane), very few such transmembrane domains are annotated in the computerized databases. Thus, as a practical matter, our discussion here is limited to α-helical transmembrane domains.

Consider, for example, the 161-residue *mouse peripheral myelin protein 22* (identified by the locus name "PM22_MOUSE" in release 27 of the SWISS-PROT computerized database of proteins (Bairoch and Boeckmann 1991). The four transmembrane domains of this protein are located at residues 2–31, 65–91, 96–119, and 134–156.

of the Kyte-Doolittle hydrophobicity scale, these seven residues would be categorized into a hydrophobic category. Seven of the 24 residues of segment (1) (i.e., two Gs, two Ts, two Ys, and one S) are in the category consisting of G, T, S, W, Y, P (which the knowledgeable human would cluster into a neutral category). Only one residue of segment (1) (i.e., the Q at position 103) is in the category consisting of H, Q, N, E, D, K, R (which the knowledgeable human would cluster into a hydrophilic category). Even through there are some residues from all three categories in segments(1), segment (1) is predominantly hydrophobic and is, in fact, a transmembrane domain of PM22_MOUSE.

In contrast, 13 of the 27 (about half) of the residues of segment (2) are neutral, eight (about a quarter) are hydrophobic, and six (about a quarter) are hydrophilic. This distribution is very different from that of segment (1). Segment (2) is, in fact, a non-transmembrane area of PM22_MOUSE.

Background on Genetic Programming

John Holland's pioneering 1975 *Adaptation in Natural and Artificial Systems* described how the evolutionary process in nature can be applied to artificial systems using the genetic algorithm operating on fixed length character strings (Holland 1975, 1992). Additional information on current work in genetic algorithms can be found in Goldberg (1989), Forrest (1993), Davis (1987, 1993), and Michalewicz (1992).

Genetic programming is an extension of the genetic algorithm in which the genetic population consists of computer programs (that is, compositions of primitive functions and terminals). As described in *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (Koza 1992), genetic programming is a domain independent method that genetically recombining randomly chosen parts of two existing programs. The genetic crossover (sexual recombination) operation (described below) operates on two parental computer programs and produces two offspring programs using parts of each parent.

(3) The single best computer program in the population produced during the run is designated as the result of the run of genetic programming. This result may be a solution (or approximate solution) to the problem.

Recent advances in genetic programming are described in Kinnear (1994). A videotape visualization of numerous applications of genetic programming can be found in Koza and Rice (1992) and Koza (1994).

The genetic crossover operation operates on two parental computer programs selected with a probability based on fitness and produces two new offspring programs consisting of parts of each parent.

Automatic function definition is used to enable genetic programming to evolve subroutines during a run. Automatic function definition can be implemented within the context of genetic programming by establishing a constrained syntactic structure for the individual programs in the population as described in *Genetic Programming II: Scalable Automatic Programming by Means of Automatically Defined Functions* (Koza 1994). Each program in the population contains one (or more) function-defining branches, one main result-producing branch, and possibly other types of branches (such as iteration-performing branches). The function-defining branch(es) define the automatically defined functions `ADF0`, `ADF1`, etc. The result-producing branch may invoke the automatically defined functions. The value returned by the overall program consists of the value returned by the result-producing branch.

The initial random generation of the population (generation 0) is created so that every individual program and conditional operations on the categories, and finally performing some arithmetic calculations and conditional operations to reach a conclusion. This suggests an overall architecture for the classifying program of several automatically defined functions (say ADF0, ADF1, ADF2) to serve as detectors for categorization, an iteration-performing branch, IPB0, for performing arithmetic operations and conditional operations for examining the residues of the protein segment using the as-yet-undiscovered detectors, and a result-producing branch, RPB0, for performing arithmetic operations and conditional operations for reaching a conclusion using the

computer time or will, at worst, have unsatisfiable termination predicates and go into infinite loops. This problem can sometimes be partially alleviated by imposing arbitrary time-out limits (e.g., on each iterative loop individually and all iterative loops cumulatively).

In problems where we can envisage one iterative calculation being usefully performed over a particular known, finite set, there is an attractive alternative to permitting imposing arbitrary time-out limits. For such problems, the iteration can be restricted to exactly one iteration over the finite set. The termination predicate of the iteration is thereby fixed and is not subject to in the function set. We have used the four arithmetic functions (+, −, *, and %) and the conditional comparative operator IFLTE (If Less Than or Equal) on many previous problems, so we include them in the function set for the iteration-performing branch. The protected division function % takes two arguments and returns one

2.4. The transmembrane domains range in length from 15 and 101 residues and average 23.0 in length.

123 of the 248 proteins were arbitrarily selected to create the in-sample set of fitness cases to measure fitness during the evolutionary process. One of the transmembrane domains of each of these 123 proteins was selected at

computer program. Since raw fitness ranges between −1.0 and +1.0 (higher values being better), standardized fitness ("zero is best") can then be defined as $\frac{1-C}{2}$. Standardized fitness ranges between 0.0 and +1.0, lower values being better and a value of 0 being the best. Thus, a standardized fitness of 0 indicates perfect agreement between the predicting program and the observed reality; a standardized fitness of +1.0 indicates perfect disagreement; a standardized fitness of 0.50 indicates that

In generation 2 of run 1, the best-of-generation program achieves an incrementally better value for correlation (0.496 in-sample and 0.472 out-of-sample) by virtue of an incremental change consisting of just one residue in the definition of ADF0.

There is a major qualitative change in generation 5. The best of generation 5 is the first best-of-generation program that makes its prediction based on the entire protein segment. This program contains 62 points (i.e., 62 functions and terminals in the bodies of the branches), has occurrences of the three particular hydrophobic residues (I, A, and L) equals or exceeds the number of occurrences of the three particular hydrophilic residues (D, E, and R), the segment is classified as a transmembrane domain. This relatively simple calculation is a highly imperfect predictor of transmembrane domains, but it is often correct. Because it examines the entire given protein segment, it is considerably better than any of its ancestors.

In generation 6 of run 1, the best-of-generation program has marginally better values for correlation (0.766 in-sample and 0.834 out-of-sample). This improvement is a consequence of a small, but beneficial, evolutionary change in the definition of ADF1. This small incremental improvement (produced by the crossover operation) is typical of the intergenerational improvements produced by genetic programming.

The 62-point best of generation 8 of run 1 exhibits a substantial jump in performance over all its predecessors from previous generations. In-sample correlation rises to 0.92; out-of-sample correlation rises to 0.89.

```
(progn (defun ADF0 ()
          (values (ORN (ORN (ORN (I?) (M?))
            (ORN (V?) (C?))) (ORN (ORN (L?)
            (G?)) (N?)))))
       (defun ADF1 ()
          (values (ORN (ORN (ORN (ORN (G?)
            (D?)) (ORN (E?) (V?))) (ORN (ORN
            (R?) (E?)) (ORN (T?) (P?)))) (ORN
            (N?) (S?)))))
       (defun ADF2 ()
          (values (ORN (ORN (ORN (L?) (R?))
            (ORN (V?) (P?))) (ORN (G?) (L?)))))
       (progn (looping-over-residues (SETM1 (- (+ M1
          (ADF0)) (ADF1))))
       (values (* (+ M1 M3) (+ 6.738 (% (- M3 L) (+ M3
          M2)))))))
```

In this program, ADF0 tests for four (I, M, C, and L) of the seven hydrophobic residues, instead of three. Moreover, isoleucine (I), the most hydrophobic residue among the seven hydrophobic residues on the Kyte-Doolittle scale, has become one of the residues incorporated into ADF0. More important, ADF1 tests for three neutral residues (T, P, and S) as well as three hydrophilic residues (D, E, and R). The result-producing branch calculates $7.738M_1$. As before, a protein segment is classified as a transmembrane domain whenever the running sum M1 is positive.

The three neutral residues (T, P, and S) in ADF1, play an important role in ADF1 since a positive value of M1 can be achieved only if there are enough sampled hydrophobic residues in the segment to counterbalance the sum of the number of sampled hydrophilic and neutral residues.

In generation 11 of run 1, the 78-point best-of-generation program shown below has an in-sample correlation of 0.94 and a standardized fitness of 0.03. It scored 117 true positives, 122 true negatives, 1 false positive, and 6 false negatives over the 246 in-sample fitness cases. It has an true negatives, 2 false positives, and 3 false negatives over the 250 out-of-sample fitness cases. Its out-of-sample error rate is only 2.0%.

```
(progn (defun ADF0 ()
          (values (ORN (ORN (ORN (I?) (M?))
            (ORN (V?) (C?))) (ORN (ORN (L?)
            (G?)) (N?)))))
       (defun ADF1 ()
          (values (ORN (ORN (ORN (ORN (G?)
            (D?)) (ORN (E?) (V?))) (ORN (ORN
            (R?) (E?)) (ORN (ORN (ORN (ORN (G?)
            (D?)) (ORN (E?) (V?))) (ORN (ORN
            (R?) (K?)) (ORN (T?) (P?)))) (ORN
            (N?) (S?))))) (ORN (N?) (S?)))))
       (defun ADF2 ()
          (values (ORN (ORN (ORN (L?) (Y?))
            (ORN (V?) (P?))) (ORN (G?) (L?)))))
       (progn (looping-over-residues (SETM1 (- (+ M1
          (ADF0)) (ADF1))))
       (values (* (+ M1 M3) (+ 6.738 (% (- M3 L) (+ M3
          M2)))))))
```

The iteration-performing branch of this program uses the settable variable M1 to create a running sum of the difference between two quantities. Specifically, as the iteration-performing branch is iteratively executed over the protein segment, M1 is set to the current value of M1 plus the difference between ADF0 and ADF1.

The result-producing branch calculates $7.738M_1$. Thus, a protein segment will be classified as being a transmembrane domain whenever the running sum M1 is positive.

In this program, ADF0 tests for four (I, M, C, and L) of the seven hydrophobic residues, including isoleucine (I), the most hydrophobic residue among the seven hydrophobic residues on the Kyte-Doolittle scale.

ADF1 tests for four of the seven hydrophilic residues (D, E, R, and K) and three neutral residues (T, P, and S). D, E, R, and K are the most hydrophilic residues from among the seven hydrophilic residues according to the Kyte-Doolittle scale. The three neutral residues (T, P, and S) in ADF1 play an important role in ADF1 since a positive value of M1 can be achieved only if there are a sufficiently large number of sampled hydrophobic residues in the segment to counterbalance the sum of the number of sampled hydrophilic residues and sampled neutral residues.

The three residues V, G, and N play no role in the calculation of the running sum M1 since they appear in both ADF0 and ADF1.

ADF2 is ignored by the iteration-performing branch and therefore plays no role at all in this program.

The operation of this program from generation 11 of run 1 can be summarized as follows: If the number of occurrences of I, M, C, and L in a given protein segment exceeds the number of occurrences of D, E, R, K, T, P, and S, then classify the segment as a transmembrane domain; otherwise, classify it as non-transmembrane.

After generation 11 of run 1, the in-sample performance of the best-of-generation program continues to improve. For example, the in-sample correlation improves from 0.94 to 0.98 between generations 11 and 18 and the number of in-sample errors (i.e., false positives plus false negatives) drops from 7 to 3. However, this apparent improvement after generation 11 is illusory and is due to overfitting. Genetic programming is driven to achieve better and better values of fitness by the relentless effects of Darwinian natural selection. Fitness for this problem is based on the value of the correlation for the predictions made by the genetically-evolved program on the *in-sample* set of fitness cases. However, the true measure of performance for a classifying algorithm is how well it generalizes to other, previously unseen sets of data (i.e., the *out-of-sample* data). In this run, the out-of-sample correlation drops from 0.96 to 0.94 between generations 11 and 18 and the number of out-of-sample errors increases from 5 to 7. The maximum value of out-of-sample correlation is attained at generation 11. After generation 11, the evolved classifying programs are being fit more and more to the idiosyncrasies of the particular in-sample fitness cases employed in the computation of fitness. The classifying programs after generation 11 are not getting better at classifying whether proteins segments are transmembrane domains. Instead, they are merely getting better at memorizing the in-sample data. In fact, a continuation of this run out to generation 50 produces no result better than that attained at generation 11.

We now consider run 2. This best-of-all run produced the

false negatives over the 250 out-of-sample fitness cases. Its out-of-sample error rate is only 1.6%.

Ignoring the three residues common to the definition of both `ADF1` and `ADF2`, `ADF1` returns 1 if the current residue is I or L and `ADF2` returns 1 if the current residue is D, E, K, R, Q, N, or P. I and L are two of the seven hydrophobic residues on the Kyte-Doolittle scale. D, E, K, R, Q, and N are six of the seven hydrophilic residues, and P is one of the neutral residues.

In the iteration-performing branch of this program from generation 20 of run 2, `M0` is the running sum of the differences of the values returned by `ADF1` and `ADF2`. `M0` will be positive only if the hydrophobic residues in the protein segment are so numerous that the occurrences of I and L outnumber the occurrences of the six hydrophilic residues and one neutral residue of `ADF2`. `M3` is the same as the accumulated value of `M0` except that `M3` lags `M0` by one residue. Because the contribution to `M3` in the iteration-performing branch of the last residue is either 0 or 1, `M3` is either equal to `M0` or is one less than `M0`.

The result-producing branch is equivalent to

$$\frac{M_3^3}{M_0(M_0+M_3)(Len+0.53)}$$

The subexpression `(- LEN -0.53)` is always positive and therefore can be ignored in determining whether the result-producing branch is positive or nonpositive. Because of the close relationship between `M0` and `M3`, analysis shows that the result-producing branch identifies

Conclusions

Table 1 shows the out-of-sample error rate for the four algorithms for classifying transmembrane domains reviewed in Weiss, Cohen, and Indurkhya (1993) as well as the out-of-sample error rate of our best-of-all genetically-evolved program from generation 20 of run 2 above. We wrote a computer program to test the solution discovered by the SWAP-1 induction technique used in the first experiment of Weiss, Cohen, and Indurkhya (1993). Our implementation of their solution produced an error rate on our test data identical to the error rate

Bairoch, A. and Boeckmann, B. 1991. The SWISS PROT protein sequence data bank. *Nucleic Acids Research* 19: 2247–2249.

Davis, L. (editor). 1987. Genetic Algorithms and Simulated Annealing. Pittman.

Davis, L. 1991. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold.

Engelman, D., Steitz, T., and Goldman, A. 1986. Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins. *Annual Review of Biophysics and Biophysiological Chemistry*. Volume 15.

Forrest, S. (editor). 1993. *Proceedings of the Fifth International Conference on Genetic Algorithms*