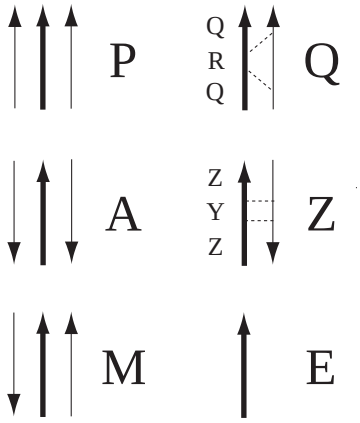

Fig. 1. The STR alphabet subdivides the letter E from DSSP into P,A,M for strands in the middle of a (parallel, anti-parallel, or mixed) sheet, and Q,Z for the strands on the edge of a sheet. The STR2 alphabet further specifies as Y the residues in anti-parallel strands which participate in the sheet hydrogen bonds. The STR3 alphabet additionally specifies as R the residues in parallel edge strands which participate in the sheet hydrogen bonds. The STR alphabets are all defined strictly from DSSP output, and so use DSSP definitions for hydrogen bonds and β-sheet partners. The E letter is used for residues that DSSP identifies as E, but which do not follow one of the defined patterns.

1 SUPPLEMENTARY MATERIALS

1.1 Local structure alphabets

Backbone geometry alphabets include the classical secondary structure alphabets and derivatives: DSSP, a seven letter version[1] of the alphabet defined by Kabsch and Sander (Kabsch and Sander, 1983); STRIDE, a six letter version[2] of the alphabet defined by Frishman and Argos (Frishman and Argos, 1995); DSSP-EHL2, a reduction[3] of DSSP to three states; STRIDE-EHL2, a reduction[4] of STRIDE to three states; STR, STR2, STR3, our enhanced versions of DSSP that subdivide the letter E (β-strand) into six, seven, or eight letters (see Figure 1), according to the particular hydrogen bonding pattern in which the strand (and residue) participates (parallel-type vs. anti-parallel-type on one, alternate, or both sides).

Other backbone geometries of interest are ALPHA, our 11-letter manual discretization of the pseudo torsion angle between C_α atoms of adjacent residues (see Figure 2 and Figure 3), and BYS, an 11-letter alphabet used in the HMMSTR (Bystroff *et al.*, 2000) program, consisting of one letter for *cis* peptide bonds and a 10-letter discretization of the Ramachandran (Ramachandran *et al.*, 1963) plot (see Figure 4 and Table 1) for *trans* peptide bonds.

We previously explored a large number of fairly similar *burial* alphabets, all of which attempt to represent whether a residue is on the surface or in the interior of a protein. These alphabets count the number of atoms or residues within a given radius of each residue. They differ in the center point of the sphere (typically the C_α or C_β atom of the residue) and the representative points of the other residues to be counted within the radius. We then discretized the distribution of counts into a fixed number of letters (such as 7 or 11), using bins of equal frequency on a sample data set that we previously described (Karchin *et al.*, 2004). We find that we have better success with these counts of neighborhood density than the surface area measures that others have used. One of our alphabets

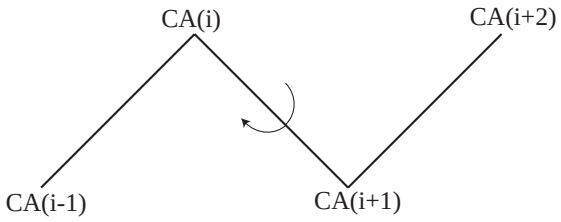

Fig. 2. The backbone angle ALPHA is defined for a residue *i* as the virtual dihedral angle between C_α atoms of residues *i-1*, *i*, *i+1*, and *i+2*.

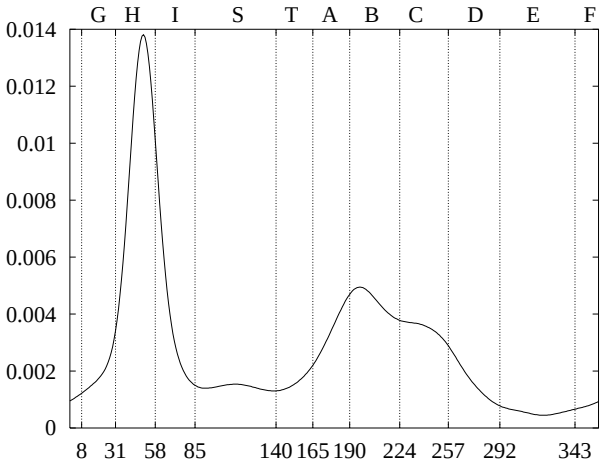

Fig. 3. Smoothed histogram of ALPHA angle distribution. The 11-letter ALPHA alphabet was chosen based on break points in this curve and in the curves for $P(\text{ALPHA}|\text{amino acid})/P(\text{ALPHA})$ for each of the amino acids.

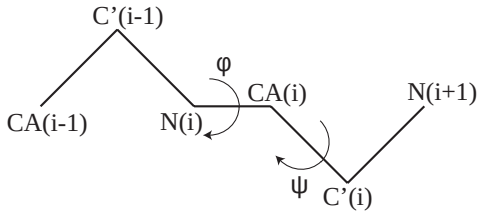

Fig. 4. The pair of dihedral backbone angles ϕ and ψ are defined by the atoms C'-N-C_α-C' (ϕ) and N-C_α-C'-N (ψ).

[1] The DSSP alphabet EHTSGBICX includes E(β-strand), H(α-helix), T(turn), S(bend), G(3-10 helix), B(short β-bridge), I(π-helix), and C(random coil) as well as the wildcard X which is not needed for prediction. We use the letters EBGHSTL for DSSP, using L(loop) for C, and mapping the rare letter I to H.

[2] We use letters EBGHTL for STRIDE, using L(loop) for C or S, and mapping the rare letter I to H

[3] DSSP-EHL2 maps G and I to H; B to E; C, T, and S to L

[4] STRIDE-EHL2 maps G and I to H; B to E; C, T, and S to L

BYS	Bystroff	ϕ	ψ
C	c	cis peptide	
H	H	-61.91	-45.20
G	G	-109.78	20.88
P	B	-70.58	147.22
E	E	-132.89	142.43
D	d	-135.03	77.26
N	b	-85.03	72.26
Y	e	-165.00	175.00
L	L	55.88	38.62
T	l	85.82	-0.03
S	x	80.00	-170.00

Table 1. Centers of the ten classes in the BYS alphabet. The first column is the letter we assign to the class, the second column is the letter that Bystroff assigned to the class. We could not use Bystroff's nomenclature, because uppercase/lowercase distinctions already have a different meaning in our software. The last two column give the center of the class in Ramachandran space. To classify a residue after a trans peptide, select the class with the smallest sum of squared differences in angles.

that we have found to perform very well in the past is BURIAL-CB-14-7, a 7-letter discretization of the number of C_β atoms within a 14Å radius of the C_β atom of the residue in question. More recently we have had improved results with NEAR-BACKBONE-11 which is an 11-letter alphabet.

NEAR-BACKBONE-11 is a burial count alphabet that counts all residues within a sphere near the residue. The center of the 9.65Å radius sphere is at a fixed location relative to the backbone. The sphere is centered at (-2.66, -5.15, 3.48) where C_α is at the origin, N is on the positive *x* axis, and C is on the *xy* plane, with a positive *y* value. The second spot defines the location of the residues to count. The spot is near the backbone at (1.24, 0.64, 0.23). The spot location and sphere radius were optimized to maximize mutual information between the identity of the residue and the measured burial.

1.2 Iterative search to generate multiple alignments

To generate a multiple alignment as input to PREDICT-2ND we use the iterative search method implemented in SAM (Karplus *et al.*, 1998). The first step consists of performing a BLAST search of the nonredundant protein database NR (NR, ????) with the single input sequence as query, to generate a set of putative homologs, which are then used to build an initial HMM. The HMM is then used to search for more remote homologs, which are iteratively used to update the HMM.

In the current work we used the SAM-T2K search method that was introduced in 2000. We also tested other iterative search methods (SAM-T04) that was developed for our CASP6 efforts and (SAM-T06) that was developed for our CASP7 efforts. These differ in several minor ways from the SAM-T2K search. The most notable differences are in the prefiltering and in the regularizers used for transition probabilities. In SAM-T04, prefiltering of the database is done using one iteration of PSI-BLAST (Altschul *et al.*, 1997; Schäffer *et al.*, 2001) at each iteration of the search. This change allows the search to be much more sensitive, without requiring extremely loose thresholds on the prefilter. Also, in SAM-T04, a regularizer is used that keeps the costs of gaps fairly low even in the later iterations of the iterative search. The resulting multiple alignments look worse to the human eye, but seem to work better for predicting local structure and contacts. SAM-T04 and SAM-T06 are similar in that they use the same prefiltering and regularizers, with SAM-T06 having slightly different thresholds to attempt to increase the sensitivity of the multiple alignments. We also noticed that some of the SAM-T04 alignments were contaminated with unrelated protein sequences, and SAM-T06 attempted to correct this contamination at each iterative step. However, the fixes we tried did not seem to work and some of our alignments still contain the contamination.

1.3 Converting multiple alignments to profiles

This section describes in more detail the weighting scheme used in converting the multiple alignments into profiles used as inputs for the neural networks.

Each alignment column of the multiple alignment corresponds to one position in the target sequence. For each alignment column, we need to create a probability vector giving the probability of seeing each amino acid at that position in the sequence for homologs of the target. Computing these probabilities is a two-step process: first converting the alignment into weighted counts of amino acids, then converting the count vector into a probability vector.

To get the weighted counts, we assign a sequence weight to each sequence. The relative weights are determined by the Henikoff method (Henikoff and Henikoff, 1994, 1996). That is, each column is initially assigned a weight of 1, which is divided up among the residues of column so that if k distinct residue types appear in the column, the total weight for each of the k is $1/k$, with all residues of the same type in a given column getting the same weight. Each sequence is then assigned the sum of the weights of all the residues in alignments columns for that sequence. These sequence weights define a relative weighting of the sequences, and can be scaled to get any arbitrary total weight.

The weight counts for a particular column are obtained by summing the sequence weights for each residue type appearing in the column. To convert these counts to probabilities, we use a mixture of 20 Dirichlet distributions (recode3.20comp) that we have been using successfully for regularizing the emission probabilities of match states in hidden Markov models. Sjölander et al. have a good explanation of Dirichlet mixtures (Sjölander *et al.*, 1996).

We usually set the total weight of the sequences so that the average relative entropy of the probability vectors after regularization is 1.3 bits/column. The higher the sequence weights, the closer the probability distribution is to the maximum-likelihood estimate, and the higher the relative entropy. In our initial tests, we thought that using a guide sequence would allow us to generalize the profile more (perhaps even as far as the 0.5 bits/column we use for fold-recognition searches). We found that very little generalization (we used 1.3 bits/column) worked better.

1.4 Training the nets

This section describes the training protocol in more detail.

We initially used the approach of train/cross-train/test sets in order to maximize the bits-saved measure. We would train three networks using our split data set, using different sets for training and cross-training for each of the networks. Looking at the results for these three networks, we would sometimes see a big difference in

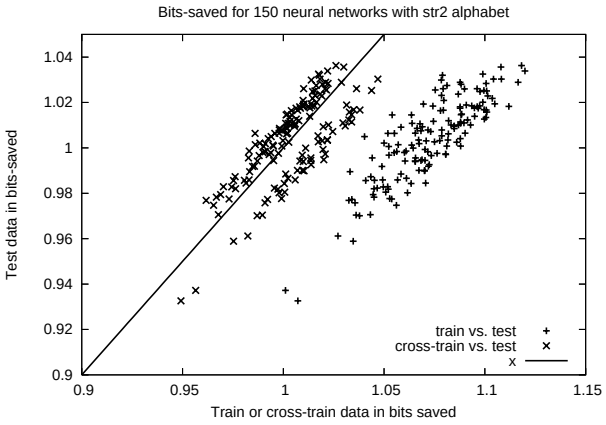

Fig. 5. The training and cross-training data against the test data. This plot shows two main groupings. The x's represent the cross-training vs. test bits-saved after neural network training. The crosses represent the training vs. test bits-saved. Both groups show a linear relationship in bits-saved for the training data and the test data. Both groups also show a wide range of bits-saved for the 150 networks (around 1-1.12 for training data, 0.95-1.04 for cross-training data, and 0.93-1.04 for test data).

the resulting bits-saved measures from one of the three networks, and the difference could not be attributed to one of the three training sets. To check this difference, we ran the protocol fifty times and looked at the differences between the resulting 150 neural networks. Figure 5 plots the results for the test data against the training data.

The figure shows two important results. The first is the wide range of results we acquired from the 150 neural networks, with around a 0.1 bit spread for each of the sets. This spread shows that different random starts can hinder the training process. The second result is that both the training and the cross-training sets show a linear relationship with the test set. The training set is a bit more dispersed than the cross-training data, but the linear relationship shows us that we can eliminate the cross-training set and instead train on two-thirds of our dataset to improve our results.

Since we can attribute the poor training to random starts, we developed a new method for training the networks. For each set of training data, we start 100 neural networks for each training set and train them for fifty epochs. After all the networks have been trained, we select the top ten networks based on the bits-saved result for the training set. We continue to train these networks for another 100 epochs (for 150 total epochs), and we select the top three networks for each training set based on the bits-saved result. We train these networks for a last 100 epochs (for a total of 250 epochs), and we select the best network for each training set based on the bits-saved result. Once we select the final network for each training set, we test the network using the one-third of the data that we held back for testing. This is our final result for the neural network. This training protocol seems to give us consistent results across the three training sets.

REFERENCES

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

Bystroff, C., Thorsson, V., and Baker, D. (2000). HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*, **301**(1), 173–190.

Frishman, D. and Argos, P. (1995). Knowledge-based protein secondary structure assignment. *Proteins: Structure, Function, and Genetics*, **23**, 566–579.

Henikoff, J. G. and Henikoff, S. (1996). Using substitution probabilities to improve position-specific scoring matrices. *Computer Applications in the Biosciences*, **12**(2), 135–143.

Henikoff, S. and Henikoff, J. G. (1994). Position-based sequence weights. *Journal of Molecular Biology*, **243**(4), 574–578.

Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, **22**(12), 2577–2637.

Karchin, R., Cline, M., and Karplus, K. (2004). Evaluation of local structure alphabets based on residue burial. *Proteins: Structure, Function, and Genetics*, **55**(3), 508–518. Online: http://www3.interscience.wiley.com/cgi-bin/abstract/107632554/ABSTRACT.

Karplus, K., Barrett, C., and Hughey, R. (1998). Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, **14**(10), 846–856.

NR (????). NR (All non-redundant GenBank CDS translations+PDB+SwissProt+PIR+PRF Database) Distributed via anonymous FTP from `ftp://ftp.ncbi.nih.gov/blast/db`. Information on NR is available at `http://www.ncbi.nlm.nih.gov/BLAST/blast_databases.html`.

Ramachandran, G., Ramakrishnan, C., and Sasisekharan, V. (1963). Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, **7**, 95–99.

Schäffer, A. A., Aravind, L., Madden, T. L., Shavirin, S., Spouge, J. L., Wolf, Y. I., Koonin, E., and Altschul, S. F. (2001). Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. *Nucleic Acids Research*, **29**(14), 2994–3005.

Sjölander, K., Karplus, K., Brown, M. P., Hughey, R., Krogh, A., Mian, I. S., and Haussler, D. (1996). Dirichlet mixtures: A method for improving detection of weak but significant protein sequence homology. *Computer Applications in the Biosciences*, **12**(4), 327–345.