

Protein Structure Analysis and Prediction

Steve Fairchild, Ruth Pachter, and Ronald Perrin, Wright Laboratory

Predicting the three-dimensional structure of a protein from its amino acid sequence is an important and difficult problem. We present an integrated approach that uses an artificial neural network to predict the spatial proximity of the amino acids in the sequence. *Mathematica* routines are developed to view the protein structure and to visualize and evaluate the results of the neural network.

Proteins are essential to biological processes. They are responsible for catalyzing and regulating biochemical reactions, transporting molecules, the chemistry of vision and of the photosynthetic conversion of light to growth, and they form the basis of structures such as skin, hair, and tendon. Protein function can be understood in terms of its structure. Indeed, the three-dimensional structure of a protein is closely related to its biological function. Proteins that perform similar functions tend to show a significant degree of structural homology [Chan and Dill 1993; Voet and Voet 1990, p. 109].

In general, a protein consists of a linear chain of a particular sequence of the 20 naturally occurring amino acids. Sequences vary in length, containing anywhere from tens to thousands of amino acids. The amino acid sequence of a protein is known as its primary structure, while local conformations in this sequence, namely alpha-helices, beta-sheets, and random coils are known as secondary structures. The angles between adjacent amino acids, called the torsion angles, determine the twists and turns in the sequence which result in these secondary structures. The three-dimensional configuration of the primary structure is defined as the tertiary structure, describing the fold of the protein.

The tertiary structure of the protein Crambin is illustrated in Figure 1. The alpha helices are easily identifiable. A beta sheet is a relatively straight and flat region in the sequence. Random coils are segments of nonrepetitive structure. This figure was created with the Quanta software package [Molecular Simulation 1994].

Steve Fairchild is a member of the computational material science group in the Hardened Materials Branch at Wright Laboratory. He holds B.S. and M.S. degrees from Murray State University and the University of Dayton. His background is in physics, optics, pattern recognition, and computer science.

Dr. Ruth Pachter works in computational materials science on the design of novel nonlinear optical materials and leads the computational material science group in the Hardened Materials Branch at Wright Laboratory. Her background is in theoretical chemistry, computational materials science, and biophysics.

Ronald Perrin is an electrical engineer with the Electronic and Optical Materials Branch of the Electromagnetic and Survivability Division at Wright Laboratory. He holds B.S.E.E. and M.S. degrees from Wright State University. He works on laboratory instrumentation and system automation, specializing in control systems and neural networks.

Each amino acid consists of a rigid plane formed by single nitrogen, carbon, alpha-carbon (C_α), oxygen, and hydrogen atoms, and a distinguishing side chain. The backbone of a protein is the linked sequence of these rigid planes. Figure 2 shows several amino acids linked together. The individual amino acids are distinguished from each other by a number of physical chemical properties that give rise to the three-dimensional structure [Wilcox, Poliac, and Liebman 1990]. Therefore it is reasonable to expect that the primary structure of a protein determines, in part, its tertiary structure.

Determining the actual three-dimensional structure of a protein is a time-consuming and complex process, currently being done using X-ray crystallography or nuclear magnetic resonance (NMR) techniques. On the other hand, determining the sequence (primary structure) of a protein is much easier, and there are many proteins whose primary structure is known but whose tertiary structure is still unknown. The ability to predict quickly the tertiary structure once the pri-

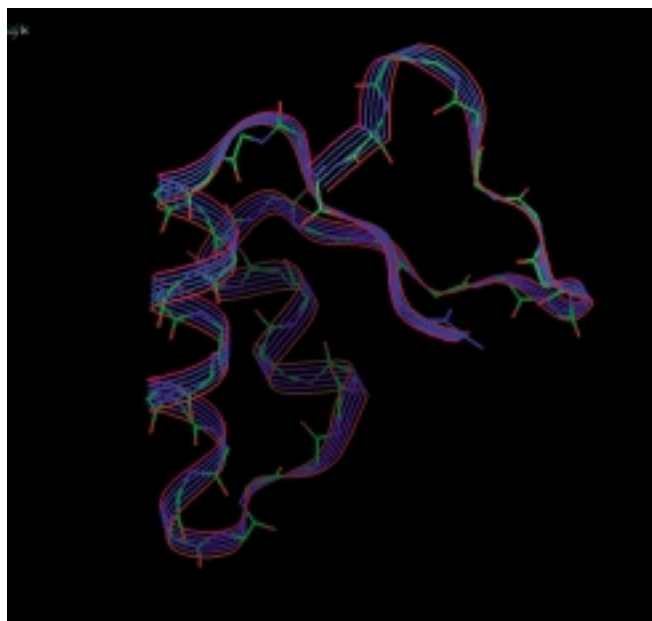


FIGURE 1. An outline of the Crambin crystal structure backbone obtained by the Quanta software package.

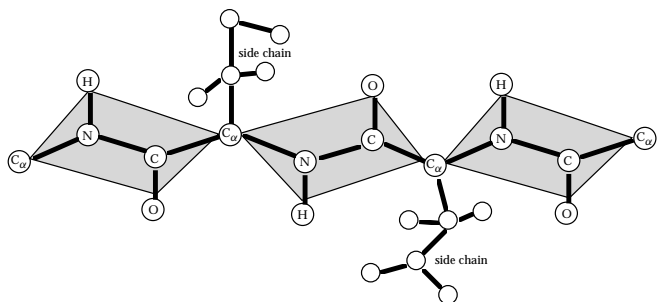


FIGURE 2. The atomic structure of the protein backbone. (Adapted from [Voet and Voet 1990, p. 145].)

primary structure has been discovered would be a tremendous asset. It would help in understanding the structures and functions of the thousands of sequences that are being discovered every day in biotechnology labs [Chan and Dill 1993]. However, predicting tertiary structure from primary structure has proved to be a very difficult problem.

This paper describes an integrated approach to predicting the tertiary structure of a protein at a low resolution. This method starts with a data set of proteins whose tertiary structures are known. A neural network is used to discover patterns that occur in this data set. It attempts to learn how primary structure affects the spatial proximity of the amino acids in the sequence. Once trained, the neural network is given a primary structure and then makes predictions about the closeness of the amino acids in three-dimensional space. A filtering technique, the so-called double-iterated Kalman filter (DIKF), is subsequently employed to elucidate the structure using a data set that includes these pairwise atomic distances predicted by the neural network, and the geometrical parameters that define the protein structure [Altman et al. 1990]. *Mathematica* routines were developed to view the distance relationships between the amino acids in the protein and to interpret the prediction results of the neural network. These routines are in the package `ProteinStructure.m`. Parts of the package are shown in Listings 1 and 2.

Viewing the Protein Backbone

Known protein structures that have been solved by X-ray crystallography and NMR are recorded in the Brookhaven Protein Database (PDB) [Bernstein et al. 1977]. Each PDB file contains the Cartesian coordinates for all the atoms of the particular protein. These coordinates can be used to calculate the torsion angles, from which the secondary structures can then be identified. Each amino acid in the protein can then be labelled as belonging to an alpha-helix, beta sheet, or random coil.

A portion of the data file created for the protein Crambin is shown below. Crambin contains 46 amino acids. Each amino acid in the sequence is represented by one line in the file. The first column lists its secondary structure (S denotes beta sheet, c is random coil, H is alpha helix). The second column gives its index, or position in the sequence, and the last three columns give the x , y , and z coordinates of the amino acid's C_α atom.

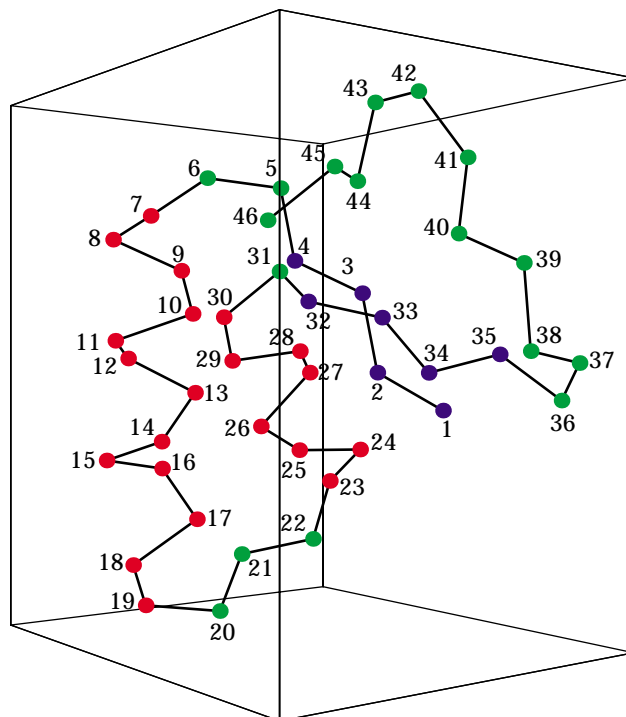
```
In[1]:= !! crambin.dat
```

S	1	16.967	12.784	4.338
S	2	13.856	11.469	6.066
S	3	13.660	10.707	9.787
S	4	10.646	8.991	11.408
c	5	9.448	9.034	15.012
c	6	8.673	5.314	15.279
H	7	8.912	2.083	13.258
H	8	5.145	2.209	12.453

An approximation to the protein backbone can be viewed by plotting the coordinates of the C_α atom in each amino acid in the sequence. Such a plot helps to visualize the distance relationships that the neural network attempts to predict. The function `ShowProteinBackbone` reads the data file and generates the plot. The points representing the amino acids are colored according to the secondary structure (red for alpha helix, blue for beta sheet, green for random coil), and are connected and labeled in their sequential order.

```
In[2]:= << ProteinStructure.m
```

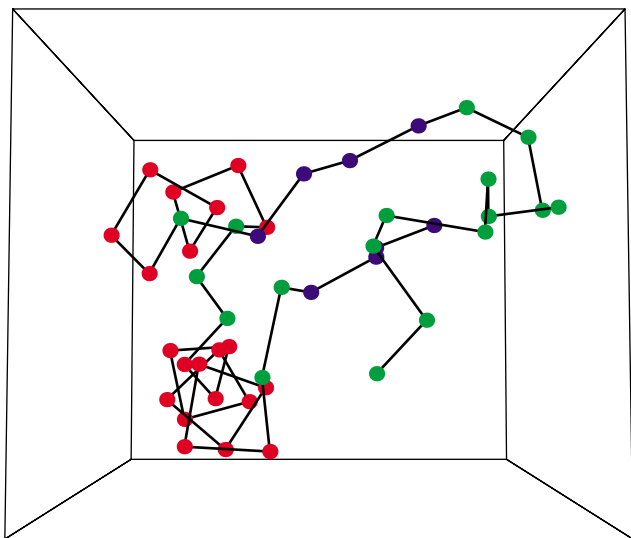
```
In[3]:= ShowProteinBackbone["crambin.dat",
ViewPoint -> {2.0, -1.5, 0.0}]
```



As can be seen from this picture, the distance relationships between the C_α atoms in the sequence determine the three-dimensional shape of the protein. Two amino acids that are far apart in the sequence may be close together in space, such as amino acids 10 and 30.

The viewpoint can be changed easily to accommodate the particular protein being observed.

```
In[4]:= ShowProteinBackbone["crambin.dat",
      SequenceNumbers -> False, ViewPoint -> {0.0, -0.1, 2.0}]
```



The Distance Matrix

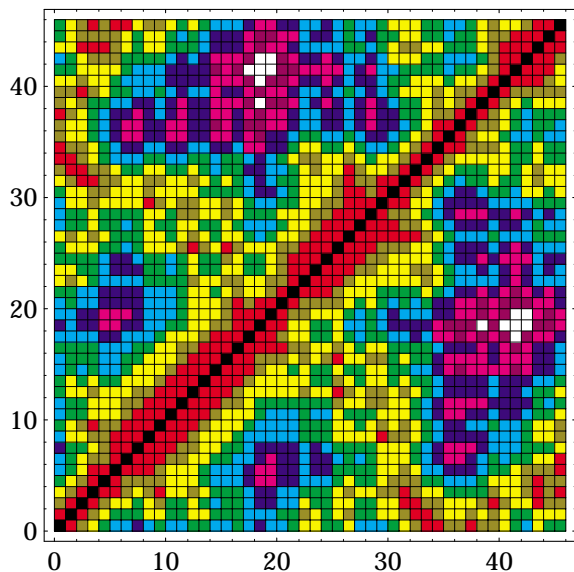
The distance matrix of a protein structure describes the proximity of the C_α atoms. Each entry a_{ij} gives the distance in angstroms between the C_α atoms at positions i and j in the sequence. The matrix is symmetric and the diagonal entries are zero since they represent a C_α 's distance to itself.

The function `ShowDistanceMatrix` displays a color-coded grid of the distance matrix. It reads a data file of distances given in the order $a_{12}, a_{13}, \dots, a_{1n}, 0, a_{23}, \dots, a_{2n}, a_{21}, 0, \dots$. The distances are calculated from the coordinates in the Protein Database.

```
In[5]:= ShowDistanceMatrix["crambin.dis", 46]
```

3 Angstrom Increments
Max Distance = 29.022 Max Scale = 30

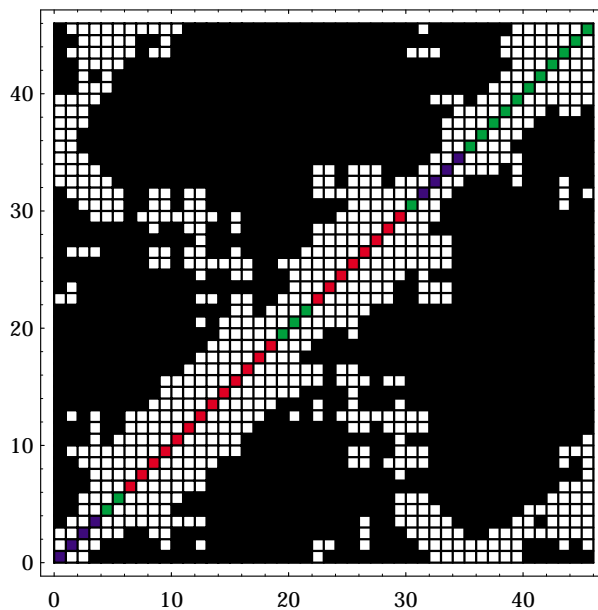
0-3 27-30



The color of each square represents the distance between the C_α atoms of the corresponding amino acids. For example, the color of square (15, 30) is yellow, indicating that the distance between amino acids 15 and 30 is 9–12 angstroms. The scaling of the distance intervals is determined by the maximum distance in the data file.

The binary distance matrix provides another representation of the distance relationships between amino acids in a sequence. The entries of this matrix indicate whether or not the amino acids are within a chosen reference distance, such as 10 angstroms. The function `ShowBinaryMatrix` displays the binary distance matrix. The binary matrix for Crambin is shown below.

```
In[6]:= ShowBinaryMatrix["crambin.dis",
      "crambin.dat", 10.0, 46]
```



White squares represent distances that are within 10 angstroms, black squares represent distances greater than 10 angstroms. Since the diagonal of the matrix contains no information, it is used to display the corresponding secondary structures for each amino acid in the sequence. Displaying the secondary structure along the diagonal shows the distance relationships between secondary structures in the protein. For Crambin, the plot shows that the two alpha helices are close around the area of amino acids 10 and 30. The backbone representations shown above confirm this observation. The neural network will attempt to predict these binary distance relationships between the C_α atoms.

The Neural Network

Neural networks can learn to identify complex patterns that occur in large sets of data. The goal of this neural network is to predict the binary distance relationships between the C_α atoms in a protein backbone, given the amino acid sequence. These distance constraint predictions are then included in the data set that is used with the DIKF to generate the protein fold.

The neural network does not attempt a distance prediction for each possible C_{α} pair. Instead, a window capturing 31 amino acids at a time is incrementally slid down the protein sequence. A distance prediction is made between the first amino acid in the window and each of the 30 that follow. It has been shown that only a partial distance matrix is needed to obtain a good reproduction of the protein backbone using minimization techniques [Bohr et al. 1990].

The neural network must first be trained on known protein structure information. The training set consists of data extracted from 47 PDB files [Kabsch and Sander 1983; Holley and Karplus 1989]. Each input-output vector pair in the training set is obtained from a section of a protein that is 31 amino acids long. The 31 amino acids in the input vector are encoded by their hydrophobicity [Wilcox, Poliac, and Liebman 1990] and associated secondary structure. Hydrophobicity is a measure of how strongly the amino acid interacts with water. The output vector contains the 30 binary distance relationships between the first amino acid and each of the following 30 in the sequence. After the network is trained, it is tested on a protein whose structure is known, but is not included in the training set. This test tells us if the neural network was able to learn sequence-distance relationship patterns in the training set well enough to make predictions about new proteins.

The neural network architecture is shown in Figure 3. The input layer contains 121 nodes. Four nodes represent each amino acid, one for hydrophobicity and three for secondary structure. The output layer consists of 30 nodes for the 30 distances in the output vector. During the training phase, the input vectors are presented to the network one at a time. A dot product is computed between the input vector and each of the weight vectors that represent each hidden node. The sum at each hidden node is input to a hyperbolic tangent transfer function, and the resulting output value is used as the input to the next layer. This same process is repeated at the output layer, and the resulting “predicted” output vector

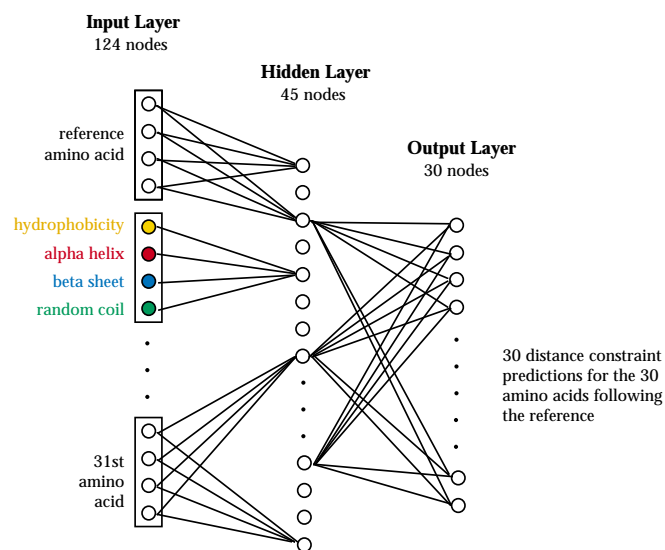


FIGURE 3. The distance-constraint predicting neural network. This is a feed-forward, fully connected network. Only a fraction of the connection weights between the nodes are shown. Each group of four nodes in the input layer represents an amino acid, by hydrophobicity and secondary structure.

```

ReadProteinData[filename_String] :=
  ReadList[filename, {{Word, Word}, {Number, Number, Number}}]

ReadDistanceData[filename_String, n_] :=
  Module[{distances},
    distances = ReadList[filename, Table[Number, {n}]];
    If[distances[[1, n]] == 0,
      MapIndexed[RotateRight[#1, #2]&, distances],
      Message[ReadDistanceData::badn, n] ] ];

distanceColor = Which[
  # <= .1, RGBColor[0,0,0], # <= 0.2, RGBColor[1,0,0],
  # <= .3, RGBColor[.5,.5,0], # <= 0.4, RGBColor[1,1,0],
  # <= .5, RGBColor[0,1,0], # <= 0.6, RGBColor[0,1,1],
  # <= .7, RGBColor[0,0,1], # <= 0.8, RGBColor[1,0,1],
  # <= .9, RGBColor[.5,0,.5], # <= 1.0, RGBColor[1,1,1]&;

secondaryColor[a_String] :=
  Switch[a, "S", RGBColor[0,0,1],
    "H", RGBColor[1,0,0], "c", RGBColor[0,1,0]]

ShowDistanceMatrix[matrix_?MatrixQ] :=
  Module[{maxDist, maxScale},
    maxDist = Max[matrix];
    maxScale = 10 Ceiling[maxDist / 10];
    distPlot = ListDensityPlot[matrix, Mesh -> True,
      ColorFunction -> distanceColor,
      DisplayFunction -> Identity];
    ShowLegend[distPlot,
      {distanceColor, 10, "0-3", "27-30",
        LegendPosition->{-0.65, 1.1}, LegendSize->{1.5, 0.3},
        LegendOrientation->Horizontal, LegendShadow -> None,
        LegendLabel -> StringJoin[
          "      3 Angstrom Increments\nMax Distance = ",
          ToString[maxDist], " Max Scale = ",
          ToString[maxScale] ] ] ] ]

ShowBinaryMatrix[distMatrix_?MatrixQ, diagonals_List,
  tolerance_Real] :=
  Module[{binMatrix, diag, n = Length[diagonals]},
    binMatrix =
      Map[If[# >= tolerance, RGBColor[0,0,0], RGBColor[1,1,1]]&,
        distMatrix, {2}];
    diag = Map[secondaryColor, diagonals];
    Do[binMatrix[[i,i]] = diag[[i]], {i, n}];
    Show[Graphics[{RasterArray[binMatrix],
      Table[Line[{{0, y}, {n, y}], {y, 0, n}],
      Table[Line[{{x, 0}, {x, n}], {x, 0, n}],
      AspectRatio -> 1, Frame -> True ] ] ]

ShowBinaryMatrix[distFile_String, dataFile_String,
  tolerance_Real, n_Integer] :=
  ShowBinaryMatrix[ReadDistanceData[distFile, n],
    Map[#[[1,1]]&, ReadProteinData[dataFile]], tolerance]

ShowProteinBackbone[data_List, opts__Rule] :=
  Module[{seqn}, seqn = SequenceNumbers /. {opts} /.
    Options[ShowProteinBackbone];
    Show[Graphics3D[{PointSize[0.02],
      Map[{secondaryColor[ #[[1,1] ]], Point[ #[[2] ] ]&, data],
      Line[Map[ #[[2] ]&, data]],
      If[seqn,
        Map[ {Text[ #[[1,2] ], #[[2] ],{2,-2}]}&, data], {}]],
      FilterOptions[Graphics3D, opts], ViewPoint -> {2, -2, 0}]]]
  
```

LISTING 1. Functions for displaying protein structure.

is compared with the “desired” output vector from the training set. The back-propagation learning algorithm uses the amount of error in the prediction to modify the initially random connection weights. During repeated cycles through the training set, the adjustment of the weights reduces the total error between the desired and predicted output vectors [Rumelhart, Hinton, and Williams 1986].

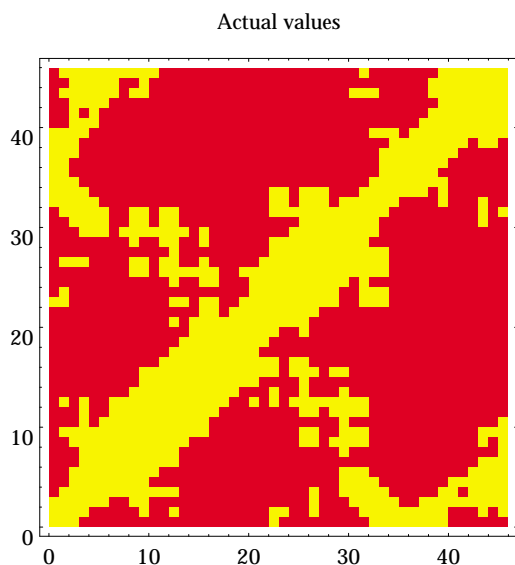
Since the linear region of the hyperbolic tangent transfer function is approximately between -0.8 and 0.8 , the output vector values in the training set are 0.8 for distances within the 10 angstrom tolerance and -0.8 otherwise. The predicted values are therefore between these two limits. These values are converted to binary values, which specify whether or not the neural net predicts that the distances are within 10 angstroms. To obtain binary values, a threshold value is chosen and output values above the threshold are converted to 1, while values below are converted to 0. Several proteins were tested to determine an optimal threshold value of -0.5 .

The 47 proteins in the training set contain 8269 amino acids, so there are 8269 input-output vector pairs. The neural network was set up on a Sun SPARCstation 10 using the NeuralWorks Professional II/Plus software package [NeuralWare 1994]. The neural network cycled through the training set approximately 250 times.

The protein Crambin, which was not included in the training set, is used to evaluate the trained network. Crambin’s amino acid sequence is presented to the network, which computes 30 distance predictions for each of the 46 amino acids in the sequence. The NeuralWorks program produces a results file which contains each input vector’s desired and predicted output vectors so that they can be compared. The functions `CompareBinaryMatrices` and `ShowCorrelation` are used to compare the predicted output vectors with the desired output vectors.

The function `CompareBinaryMatrices` reads the NeuralWorks results file and displays the desired and predicted binary distance matrices. Distances for amino acids that are more than 30 apart in the sequence are labeled as outside the 10 angstrom limit.

```
In[7]:= CompareBinaryMatrices["neuralnet.dat", -0.5]
```



The function `ShowCorrelation` quantifies the results of the test. It counts the number of correct predictions, and calculates the overall success rate and the Mathews correlation coefficient [Qian and Sejnowski 1988]. The success rate is simply the number of correct predictions divided by the total number of predictions. The correlation coefficient takes into account the number of over- and under-predicted cases, and is a better overall estimate of prediction success.

```
In[8]:= ShowCorrelation["neuralnet.dat", -0.5]
```

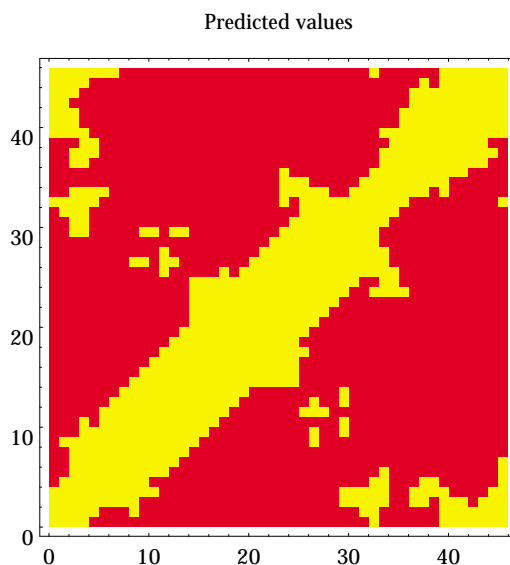
```
Out[8]//TableForm=
```

Number of amino acids	46
Number of predictions for each	30
Correct predictions / Total	1174 / 1380
Correct positive predictions	246 / 378
Correct negative predictions	928 / 1002
Success rate	0.85
Correlation coefficient	0.61

Results

Distance constraints predicted by the neural network are used in conjunction with the DIKF to determine the folded tertiary structure of the protein. Further refinement is achieved by minimizing the potential energy of the molecular system using local minimization techniques.

To test this approach, a random, unfolded geometry for the amino acid sequence of Crambin was presented to the DIKF. The final predicted structure was then compared to Crambin’s experimental X-ray structure. The comparison showed a root mean square (RMS) deviation of 6.3 angstrom for the backbone. This encouraging result indicates that such an integrated approach may be useful for a low-resolution protein structure prediction.



```

toBinary[x_, x0_] := If[x >= x0, 1, 0]
toBinary[x_List, x0_] := Map[toBinary[#, x0]&, x]
or[a_, b_] := If[ a == 0 && b == 0, 0, 1]
formatMatrix[matrix_?MatrixQ] :=
Module[{r, c, s},
  {r, c} = Dimensions[matrix];
  s = If[ r == c, matrix,
    Map[ Join[#, Table[0, {r-c}]]&, matrix] ];
  s = MapIndexed[RotateRight[#, #2]&, s];
  Do[ s[[i,i]] = 1, {i, r}];
  MapThread[ or, {s, Transpose[s]}, 2 ] ]
CompareBinaryMatrices[{act_?MatrixQ, pred_?MatrixQ}] :=
Show[GraphicsArray[
  Map[
    ListDensityPlot[#[[1]], PlotLabel -> #[[2]],
      ColorFunction ->
        (If[# == 1, RGBColor[1,1,0], RGBColor[1,0,0]]&),
      Mesh -> False, DisplayFunction -> Identity ]&,
    {{act, "Actual values"}, {pred, "Predicted values"}} ] ],
  DisplayFunction -> $DisplayFunction]
CompareBinaryMatrices[filename_String, cutoff_] :=
CompareBinaryMatrices[
  Map[formatMatrix[toBinary[#, cutoff]]&,
    ReadNNDData[filename] ] ]
CorrelationData[{act_?MatrixQ, pred_?MatrixQ}] :=
Module[{r, c, p, n, o, u, qq, cc},
  {r, c} = Dimensions[act];
  {p, n} = Map[Count[Flatten[act + pred], #]&, {2, 0}];
  {o, u} = Map[Count[Flatten[act - pred], #]&, {-1, 1}];
  qq = N[(p + n)/(r c), 2];
  cc = N[(p n - u o)/Sqrt[(n+u)(n+o)(p+u)(p+o)], 2];
  {r, c, p, n, o, u, qq, cc} ]
ff[a_, b_] := StringForm["` / `", a, b]
ShowCorrelation[{act_?MatrixQ, pred_?MatrixQ}] :=
Module[{r, c, p, n, o, u, qq, cc},
  {r, c, p, n, o, u, qq, cc} = CorrelationData[{act, pred}];
  TableForm[
    {r, c, ff[p+n, r c], ff[p, p+u], ff[n, n+o], qq, cc},
    TableHeadings ->
      {"Number of amino acids",
        "Number of predictions for each",
        "Correct predictions / Total",
        "Correct positive predictions",
        "Correct negative predictions",
        "Success rate",
        "Correlation coefficient"},
    None] ] ]
ReadNNDData[filename_String] :=
Module[{data, n},
  data = ReadList[filename, Number, RecordLists -> True];
  n = Length[First[data]] / 2;
  Transpose[Map[ Partition[#, n]&, data]] ]
ShowCorrelation[filename_String, cutoff_] :=
ShowCorrelation[
  Map[toBinary[#, cutoff]]&, ReadNNDData[filename] ] ]

```

LISTING 2. Functions for evaluating the results of the neural network.


Conclusions

Mathematica has proven to be an invaluable tool for this research project. Its numeric and graphics capabilities are ideal for analyzing the data files generated by the neural network and for viewing the protein backbones in a variety of ways. While protein structures can be viewed using a variety of specialized commercial software packages, such packages cannot usually be modified to suit particular applications. *Mathematica*'s flexibility has made it easy to integrate the data analysis and visualization described in this paper.

References

- Altman, R.B., R. Pachter, E.A. Carrara, and O. Jardetzky. 1990. PROTEAN Part II: Molecular structure determination from uncertain data. *QCPE* 10(2): 596.
- Bernstein, F.C., T.F. Koetzle, G.J.B. Williams, E.F. Meyer, M.D. Brice, J.R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. 1977. The Protein Data Bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol.* 112:535-542.
- Bohr, H., J. Bohr, S. Brunak, R.M.J. Cotterill, H. Fredholm, B. Lautrup, and S.B. Peterson. 1990. A novel approach to prediction of the three-dimensional structures of protein backbones by neural networks. *FEBS Lett.* 261(1): 43-46.
- Chan, H.S, and K. Dill. 1993. The protein folding problem. *Physics Today* (Feb.) 24-32.
- Holley, L., and M. Karplus. 1989. Protein secondary structure prediction with a neural network. *Proc. Natl. Acad. Sci. USA* 86:152-156.
- Kabsch, W., and C. Sander. 1983. How good are predictions of protein secondary structure? *FEBS Lett.* 155:179-182.
- Molecular Simulation, Inc. 1994. Quanta, Release 3.2.
- NeuralWare, Inc. 1994. NeuralWorks Professional II/Plus Version 4.
- Qian, N., and T. Sejnowski. 1988. Predicting secondary structure of globular proteins using neural network models. *J. Mol. Biol.* 202:865-884.
- Rumelhart, D.E., G.E. Hinton, and R.J. Williams. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing*, vol. 1, 318-362. Cambridge, MA: MIT Press.
- Voet, D., and J. Voet. 1990. *Biochemistry*. John Wiley and Sons.
- Wilcox, G.L., M. Poliac, and M.N. Liebman. 1990. Neural network analysis of protein tertiary structure. *Tetrahedron Computer Methodology* 3 (3-4): 191-211.

Steve Fairchild, Ruth Pachter, Ronald Perrin
 Wright Laboratory
 Wright-Patterson Air Force Base, OH 45433-7702
 fairchs@hobbes.mil.wpafb.af.mil

 The electronic supplement contains the package ProteinStructure.m, and the data files crambin.dat, crambin.dis, and neuralnet.dat.