Application of a neural network in high-throughput protein crystallography

A. Berntson, V. Stojanoff* and H. Takai

Brookhaven National Laboratory, Upton, NY 11973, USA.
E-mail: stojanof@bnl.gov

High-throughput protein crystallography requires the automation of multiple steps used in the protein structure determination. One crucial step is to find and monitor the crystal quality on the basis of its diffraction pattern. It is often time-consuming to scan protein crystals when selecting a good candidate for exposure. The use of neural networks for this purpose is explored. A dynamic neural network algorithm to achieve a fast convergence and high-speed image recognition has been developed. On the test set a 96% success rate in identifying properly the quality of the crystal has been achieved.

Keywords: X-ray diffraction; neural networks; protein crystallography.

1. Introduction

High-throughput protein crystallography requires the automation of multiple steps from cloning and expression to three-dimensional model construction *via* crystallization. Several of these steps have been, or are being, automated (Terwilliger, 2002). The so-called crystallization robots allow for screening of large numbers of conditions to produce X-ray diffraction quality crystals. The selection of good diffracting crystals, however, is still a tedious process that can require many hours. Even with the advent of automated sample changers the selection of a good diffracting protein crystal relies on the experimenter, who also checks for deterioration owing to radiation damage. Therefore, for high-throughput structure determination an automated diffraction pattern diagnostic tool is highly desirable. There have been different approaches to solving this problem. A traditional way is to analyze each diffraction pattern using diffraction-analysis packages and relying on the value of, for example, mosaicity and resolution limit to determine the crystal quality. The drawback of this solution is that it is a computer-intensive procedure and it is somewhat dependent on the experimenter. In this paper we explore the possibility of applying a neural network paradigm to recognize the quality of the crystal. Neural networks were introduced in the 1940s and have been applied in general to problems where pattern recognition is required. Neural networks offer exceptionally robust performance in classification problems, even in the presence of noisy input. They provide the accuracy of handwriting- and speech-recognition applications. They are extremely powerful and, unlike traditional techniques for pattern recognition, the individual differences in diffraction patterns such as intensity and contrast are insignificant to the classification accuracy. This flexibility makes neural networks the ideal choice for identifying good from poor protein crystals based on their diffraction patterns.

In this paper we explore the use of a cascade-correlation neural network to monitor protein diffraction patterns. This technique is extremely fast and can be used in conjunction with automated crystal-mounting systems to allow for high-throughput protein crystallography studies.

2. Neural networks and cascade correlation

As the name implies, neural networks take a cue from the human brain by emulating its structure. Work on neural networks (Haykin, 1999; Bishop, 1995) began in the 1940s by McCulloch and Pitts and was followed by the advent of Frank Rosenblatt's *Perceptron* (Rojas, 1996; Lawrence *et al.*, 1996; Haykin, 1994).

The neuron is the basic structural unit of a neural network. In the brain, a neuron receives electrical impulses from numerous sources. If there are enough agonist signals, the neuron triggers all of its outputs. A neural network neuron functions similarly. A neuron receives any number of inputs and weights the inputs based on their importance. Just as in a real neuron, the weighted inputs are summed and the output, defined by a threshold function (*e.g.* a step function), is sent to every neuron downstream. In a neural network, weights and threshold function parameters are selected to provide a desired output, *e.g.* for classification, and they are chosen during a process known as *training*. A barrage of positive inputs will provide a positive output and *vice versa*. The original *Perceptron* (Rojas, 1996) received two inputs and gave a single output. Although this system worked well for simple problems, Minsky demonstrated in 1969 that non-linear classifications, such as exclusive-or (XOR) logic, were impossible (Haykin, 1994).

It was not until the 1980s that training algorithms for multi-layered networks were introduced to solve this problem, restoring the usefulness of neural networks. A multi-layered network consists of numerous neurons, which are arranged into levels as shown in Fig. 1. The first layer receives external inputs and is aptly named the input layer. The top layer provides the solution and is called the output layer. Sandwiched between the input and output layers are any number of hidden layers. Multi-layered neural network architecture is a generalization of Rosenblatt's *Perceptron* and allows the selection of events in a multi-dimensional space defined by the input parameters. It is believed that a three-layered network, as indicated in Fig. 1, can accurately classify any non-linear function. Multi-layered networks commonly use more sophisticated threshold functions such as the sigmoid function,

$$f(x) = \frac{1}{1+\exp(-ax)} - 0.5, \tag{1}$$

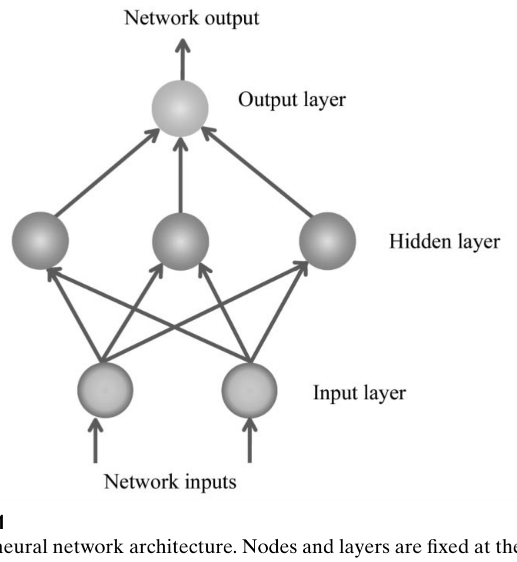

Figure 1
Classic neural network architecture. Nodes and layers are fixed at the creation time and the training process adjusts weights for best performance.

which ranges from −0.5 to 0.5. It is the smoothness of the sigmoid function that prevents any individual output from becoming too large and overpowering the network (Rojas, 1996; Lawrence *et al.*, 1996; Lippmann, 1987).

The power of a neural network is contained in its ability to *remember* and provide classifications based on past data. Past input samples are remembered through the network's weights. The efficiency of a neural network is determined by how it *learns*. The most widely used teaching technique for neural networks is *via* supervised learning. In the supervised-learning algorithm a training data set whose classifications are known is shown to the network one at a time. Each time, the weights are adjusted to provide the desired output with the given inputs. Back-propagation, radial-basis and delta-rule training algorithms are among the most popular and versatile (Rojas, 1996).

Back propagation was one of the first training algorithms developed. It is widely used for its simplicity; however, it is far from being the most efficient. To train a back-propagation network, every weight in the network must be initialized by a small random number. As the network is trained, the randomness of the initial weights guarantees that each neuron will have a different weight. The entire data set used for training is presented to the network one at the time. For every training sample, the desired output is compared with the actual output, and the weights of each neuron are altered based on the amount of error it contributed. The error of a neural network is defined as the discrepancy between its current output and the expected response. Functionally the error depends on the weights. After many iterations, or epochs, the weights reach values that offer minimal output error. This process can take many CPU cycles. Teaching the XOR classification to a simple network consisting of two input, two hidden and one output neuron using back-propagation can take over 500000 epochs to reach an acceptable error level. Fahlman & Lebiere (1991) identified two likely culprits responsible for this time-consuming training curve: the step-size problem and the moving-target problem.

The back-propagation algorithm minimizes the error of the network after each interaction, adjusting the weights to find a minimum for the error. This is done simply by correcting the weights in a first approximation by evaluating the correction term as the product of the derivative of the error at the ith interaction and a fixed step size. If the steps are too small, many iterations will be required to converge. If the steps are too large, then it is easy to overshoot the position of the minimum. The ideal step size for a given problem requires detailed high-order-derivative analysis, a task that is not performed by the algorithm (Fahlman & Lebiere, 1991; Werbos, 1990).

The moving-target problem appears because the weights of each neuron are adjusted independently. An advantage of a large network is that each neuron becomes a specialized feature detector; its weights become tuned to identify a specific characteristic of its inputs. As the weights are altered, the role of each neuron becomes increasingly defined. However, back-propagation does not coordinate this development; several neurons may identify a particular feature (*e.g.* feature A) and ignore another (feature B). When the error signal of feature A is eliminated, feature B remains. The neurons may then abandon feature A and begin focusing on feature B. Throughout numerous epochs, the neurons 'dance' between feature A and feature B. It may take several thousand epochs before both feature A and feature B are identified at the same time (Fahlman & Lebiere, 1991; Lawrence *et al.*, 1996).

Developed in 1990 by Fahlman, the cascade-correlation algorithm provides a solution (Fahlman & Lebiere, 1991) to reduce the number of epochs. Cascade-correlation neural networks, shown in Fig. 2, are similar to traditional networks in that the neuron is the most basic unit. The architecture, however, is rather unique. It is assumed that the neural network is not a static structure but dynamically changes as required. An untrained cascade-correlation network is a blank slate; it has no hidden units. A cascade-correlation network's output weights are trained until either the solution is found or progress stagnates. If a single-layered network will suffice, training is complete.

If further training yields no appreciable reduction of error, a hidden neuron is *recruited*. A pool of hidden neurons (usually eight) is created and trained until their error reduction halts. The hidden neuron with the greatest correspondence to the overall error (the one that will affect it the most) is then installed in the network and the others are discarded. The new hidden neuron perturbs the network and significant error reduction is accomplished after each hidden neuron is added. This ingenious design eliminates the moving-target problem by training feature detectors one by one and only accepting the best. The weights of hidden neurons are static; once they are initially trained they are not touched again. The features they identify are permanently cast into the memory of the network.

Preserving the placement of hidden neurons allows cascade correlation to accumulate experience after its initial training session. Few neural network architectures allow this. If a back-propagation network is retrained it *forgets* its initial training. This is not the case for cascade correlation.

The step-size problem contributes greatly to error stagnation as a function of the iteration. When a network takes steps that are too small, the error value reaches an asymptote. Cascade correlation avoids this by using the quick-prop algorithm. Quick-prop makes a bold assumption that the network error is a paraboloid function and evaluates the second derivative (change of slope). With each epoch it takes a different-sized step corresponding to the change of the error slope. If the error is reducing rapidly, quick-prop descends quickly. Once the slope flattens out, quick-prop slows down so that it does not pass the minimum. Quick-prop is like a person searching for a house: if the person knows he is far away, he drives fast; once he approaches the house, he slows down and looks more carefully (Fahlman & Lebiere, 1991). The cascade-correlation algorithm trains substantially

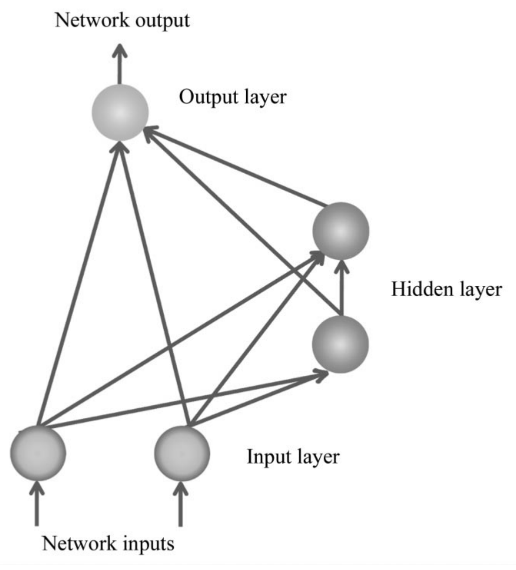

Figure 2
Cascade-correlation neural network architecture.

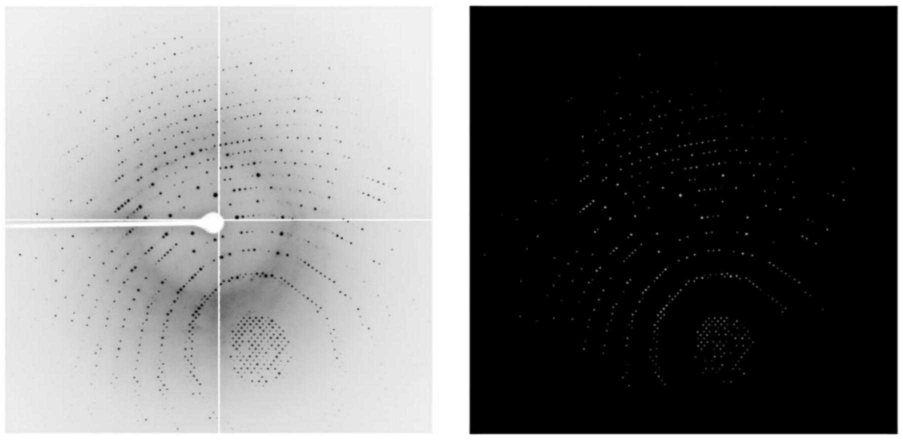

Figure 3
Diffraction pattern resulting from the scattering of X-rays from a good quality protein crystal, and peaks found by software.

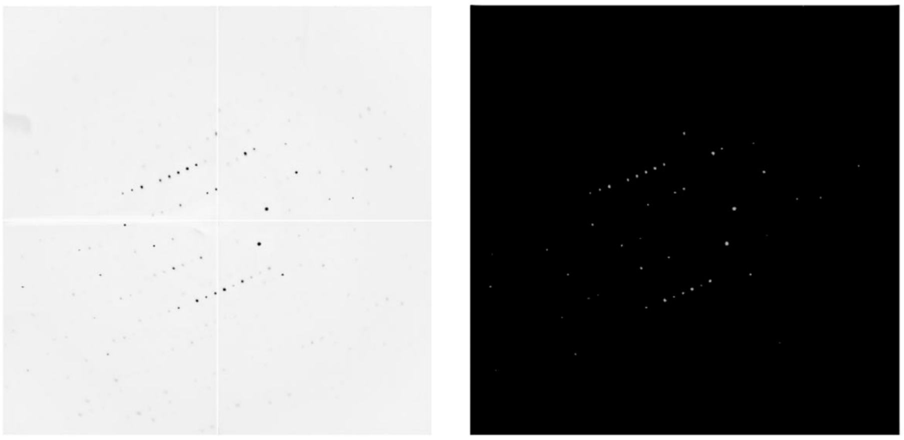

Figure 4
Diffraction pattern resulting from the scattering of X-rays from a poor quality protein crystal, and peaks found by software.

faster than virtually any other existing algorithm (Fahlman & Lebiere, 1991; Thrum *et al.*, 1991; Humpert, 1990).

3. Experimental results and discussion

Our software (*CrySis*) was written to analyze the diffraction patterns from protein crystals. It is written in C++ and uses the open source graphics library SDL and open source examples of the cascade-correlation algorithm.†. A total of 711 diffraction patterns from the X6A beamline at the National Synchrotron Light Source (NSLS) were used in this study. Of these, 500 images were selected to train *CrySis* and the remaining 151 images were used to test the accuracy of the network. Each pattern is a 2048 $\times$ 2048 pixel image of an X-ray diffraction pattern in IMG file format. The images were from 16 crystals of five different proteins and included different derivatives, and data at room temperature and 100 K. The image sets used for training and for tests are somewhat orthogonal in the mathematical sense since they are from different proteins or different crystals.

The network could theoretically be trained to accept $\sim 4 \times 10^6$ pixel images as input, but that would be too much data for a conventional workstation to handle owing to the number of weights and threshold parameters. The complexity of the network required to train such data would be hopelessly inefficient. To gain accurate results in an acceptable time window, the images are pre-processed. This feature-extraction process takes advantage of the characteristics of the diffraction patterns. Good and poor quality protein diffraction patterns differ qualitatively in many ways. Diffraction images of a good crystal have numerous extremely intense peaks as compared with poor crystals, as shown in Figs. 3 and 4, respectively. Peaks are arranged in a different, but meaningful, order for every diffraction pattern. The location of the peaks does not differentiate a viable crystal from a poor one, but the distribution of peaks does. The intensity of the peaks in a viable crystal is at its maximum in the center of the image, and slowly falls off towards the outside of the detector acceptance; the intensity gradient for a poor scatterer is very steep (Dauter, 1997; Otwinowski & Minor, 1997; Pflugrath, 1997).

Based on the observations above, we searched for the minimum set of parameters that could meaningfully classify the images. A choice of coordinates for the analysis was arbitrarily made. The center of the image was chosen as coordinate $(0, 0)$ for the extraction of the parameters used in the analysis. The first implementation relied solely on a frequency distribution of the peaks and the peak value of the ten most intense peaks. This criterion, as expected, did not perform well because it is far from being a unique characterization of the image. It took almost 10 min to train the network, and after almost 4000 epochs the success rate was a dismal 60%. Using only the frequency and peak intensities led a fraction of viable crystals to score less than many poor ones. Some poor crystals have a number of intense peaks, and some good ones have relatively few. Judgement based on such a narrow scope was leading *CrySis* to misidentify the training set. It had to take into account more factors than just frequency and peak intensity. Nevertheless, the neural network was giving sensible results in the sense that we could understand the results. It is often difficult to predict or understand the behavior of a neural network since it is not a linear system.

In an attempt to improve the performance, a simple peak-finding algorithm was implemented. Images were pre-processed by first evaluating the average pixel intensity and its standard deviation. Any group of pixels whose intensity is two standard deviations above the mean and whose size is greater than 5 pixels (single-pixel peaks can be the result of extraneous factors) can be considered a diffraction peak. Next, each image is divided arbitrarily into 14 concentric rings with a radial increment of 256 pixels. We do not calibrate the distance in any units but rather work with raw detector pixels. The number of peaks found in each region is counted to generate three distributions: frequency, percentage and intensity distribution. Frequency and percentage distribution are mathematically defined as dN/dr and

† Lisp and C implementation of Cascade Correlation, Carnegie Mellon University, http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/areas/neural/systems/cascor/.

$(1/N_{\rm tot})\,{\rm d}N/{\rm d}r$, where N is the number of counts in each ring, r is the average radius of the ring and $N_{\rm tot}$ is the total number of peaks in the image. The intensity distribution is the average peak intensity for each ring. The second implementation of *CrySis* bases its decisions on these three distributions. After the image is analyzed, the data (42 floating numbers shown in Fig. 5) are fed into the cascade-correlation neural network. In principle, a more sophisticated principal component analysis (Haykin, 1999) could be applied to the images for feature extraction. For the purposes of crystal quality analysis, however, the parameters chosen above, based on expert image analysis, were favored.

Images used for training are from previously exposed crystals and were analyzed with the *HKL-2000* suite (Otwinowski & Minor, 1997); good sets were used in structural determination. These images were scored from −0.5 (unusable) to +0.5 (viable crystal) in discrete steps of 0.1. The scores are based on two considerations. First we examine the image mosaicity. Low-mosaicity images received high scores and high-mosaicity images received low scores. Roughly speaking, the scoring scale is proportional to the mosaicity. However, it is known that in some cases images with high mosaicity led to determination of protein structures. In these cases the score was set to zero. The *R*-factor was not used to score the images because it depends on the entire data set and in some cases crystals become unusable after being exposed for some time owing to radiation damage. The output of the neural network varies continuously from −0.5 to +0.5 and digitization was not applied to the final result.

To train *CrySis*, each image of the 500-image training set was analyzed and their distributions saved to file. Both the distributions and rates were given to the network. The analysis file was then loaded, and the cascade-correlation training algorithm established the network weights. The training process took on average a mere 170 epochs and 1.3 s to complete. The output of the neural network was then verified against the true value and considered a correct answer if the result was within $\delta = \pm 2^{1/2}\varepsilon$ of the true value, where $\varepsilon = 0.1$ is the step size of the scoring. Every image (100%) found in the training set was successfully classified. *CrySis* was tested with the 151 diffraction patterns not including the training set. In this set, 145 of the 151 (96%) patterns were successfully classified. Clearly, the expanded analysis parameters vastly improved the performance. Most of the missed identifications were from images that were considered good. This is not entirely surprising because there was a bias in our training sample towards bad images (∼60%).

It is entirely possible that in the case of a cascade-correlation neural network not only weights but the hidden layer topology will depend on the sequence that images are presented. We note that the initial weights of any neural network are always initialized using random-number generators (Haykin, 1999; Bishop, 1995). Therefore, the resulting weights will always be different for each training session even if the same sequence of images is presented. The topology of the hidden layer can be different because it is driven by the derivative of the network error, which may depend on the sequence of how the data are presented. Hidden nodes in cascade-correlation neural networks are added to move the network away from a slow converging error. However, what matters in the end is the performance of the neural network. To be sure of this statement, we have performed training sessions where data are presented to the network in many different sequences and have not noticed any changes in the performance or training time, although small variations (1 to 2) in the number of hidden nodes were observed.

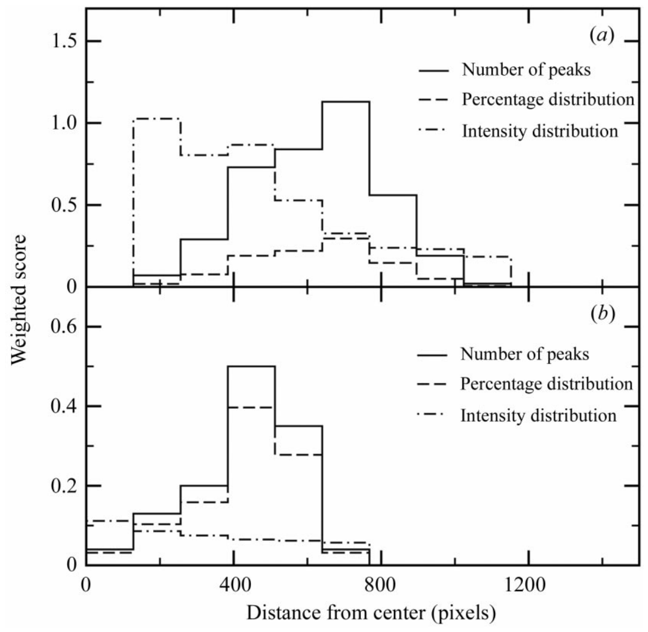

Figure 5
Weighted scores (input parameters) as a function of distance from the center of the image. (*a*) For a crystal considered 'good' and (*b*) for a poor quality crystal.

4. Conclusions and outlook

Cascade-correlation neural networks present several advantages over the more traditional network architectures. They train extremely fast and allow for growth. In the present study we only used 500 sample images for training. As it remembers previous training sessions, one can add new images for further training. One of the dangers of the neural network is that it can develop *tunnel vision* and a tendency to recognize only patterns that are very similar to ones used in the training. If a cascade-correlation network develops this feature, it will need to be retrained from scratch with a thinned number of samples.

The use of neural networks is seen as a tool to aid non-expert crystallographers during data collection. Since the output of the neural network from *CrySis* has a continuous numerical value, it requires an expert crystallographer to define a threshold for an acceptable image quality. It is unavoidable that certain images will fall in gray areas. In this case, direct inspection of the image will be required. However, as mentioned above, cascade-correlation networks allow for growth and therefore corrections to its behavior can be made. Hence, as time evolves one can expect the results given by the network to improve.

In spite of the very encouraging results of this study, more tests are required to definitively prove that a cascade-correlation neural network will be a useful tool for high-throughput protein crystallography. The current feature-extraction component of *CrySis* uses a set of arbitrarily chosen parameters. Whether or not this set of parameters biases the behavior of the neural network needs to be understood. The parameters are, however, based on what an expert would look for in order to evaluate the image quality. Perhaps a more elegant solution to the image pre-processing will make use of a Fourier frequency analysis or Wavelet analysis to extract the relevant input parameters.

AB would like to thank Ms E. Champey from Smithtown High School for support in this work and Mr J. Jakoncic for aid in providing images from the X6A beamline at the National Synchrotron Light Source facility at Brookhaven National Laboratory. We are in debt to Dr M. Siegel, from Carnegie Mellon University, for invaluable comments. This work was performed under DOE Contract number DE-AC02-98CH10886 and NIGMS contract number Y1GM-0080-02.

References

Bishop, C. M. (1995). *Neural Network for Pattern Recognition.* New York: Claredon Press.

Dauter, Z. (1997). *Data Collection Strategy*, *Methods in Enzymology*, Vol. 276, *Macromolecular Crystallography*, Part A, edited by C. W. Carter Jr and R. M. Sweet. New York: Academic Press.

Fahlman, S. & Lebiere, C. (1991). Technical Report CMU-CS-90-100. Carnegie Mellon University, USA.

Haykin, S. (1994). *Neural Networks.* New York: Macmillan College.

Haykin, S. (1999). *Neural Networks – A Comprehensive Foundation.* Englewood Cliffs, NJ: Prentice Hall.

Humpert, B. (1990). *Comput. Phys. Commun.* **58**, 223–256.

Lawrence, S., Giles, C. L. & Tsoi, A. C. (1996). Technical Reports UMIACS-TR-96–22 and CS-TR-3617. Institute for Advanced Computer Studies, University of Maryland, College Park, USA.

Lippmann, R. P. (1987). *IEEE ASSP Mag.* pp. 4–22.

Otwinowski, Z. & Minor, W. (1997). *Processing of X-ray Diffraction Data Collected in Oscillation Mode*, *Methods in Enzymology*, Vol. 276, *Macromolecular Crystallography*, Part A, edited by C. W. Carter Jr and R. M. Sweet, pp. 307–326. New York: Academic Press.

Pflugrath, J. (1997). *Diffraction-Data Processing for Electronic Detectors*, *Methods in Enzymology*, Vol. 276, *Macromolecular Crystallography*, Part A, edited by C. W. Carter Jr and R. M. Sweet, pp. 286–306. New York: Academic Press.

Rojas, R. (1996). *Neural Networks – A Systematic Introduction.* NewYork/Berlin: Springer-Verlag.

Terwilliger, T. (2002). *Interdisciplinary Workshop Promoting Collaboration in High-Throughput X-ray Structure Determination*, Santa Fe, NM, USA, 22–23 March 2002. http://www.solve.lanl.gov/santa_fe_workshop.html.

Thrum, S. B., Bala, J., Bloedorn, E., Bratkom, I., Cestnik, B., Cheng, J., De Jong, K., Dzeroski, S., Fahlman, S. E., Fisher, D., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R. S., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J. & Zhang, J. (1991). Technical Report CMU-CS-91–197. Carnegie Mellon University, USA.

Werbos, P. J. (1990). *Proc. IEEE*, **78**, 1550–1558.

Received 22 May 2003 • Accepted 22 September 2003