Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Neurocomputing 61 (2004) 217–240
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Single-layer artificial neural networks for gene expression analysis

A. Narayanan[a,*], E.C. Keedwell[a], J. Gamalielsson[b], S. Tatineni[a]

[a] *Bioinformatics Laboratory, School of Engineering & Computer Sciences, University of Exeter, Exeter EX4 3PT, UK*

[b] *Biocomputation Research Group, Department of Computer Science, University of Skövde, Box 408, 54128 Skövde, Sweden*

Received 21 March 2003; accepted 6 October 2003

Abstract

Gene expression datasets are being produced in increasing quantities and made available on the web. Several thousands of genes are usually measured for their mRNA expression levels per sample using Affymetrix gene chips and Stanford microarrays, for instance. Such datasets are normally separated into distinct, objectively measured classes, typically disease states or other objectively measured phenotypes. A major problem for current gene expression analysis is, given the disparity between the number of genes measured (typically, thousands) and number of individuals sampled (typically, dozens), how to identify the handful of genes which, individually or in combination, help classify individuals. Previous approaches when faced with the dimensionality of the problem have tended to use unsupervised or supervised techniques that result in smaller clusters of genes, but clusters by themselves do not yield classification rules. This is especially the case with temporal microarray data, which represents the activity of genes within a cell, tissue or organism over time. The expression levels of some genes at a particular time-point can be controlled by the expression levels of other genes at a previous time-point. It is the extraction of these temporal connections within the data that is of great interest to biomolecular scientists and researchers within the pharmaceutical industry. If these so-called gene networks can be found that explain disease inception and progression, drugs can be designed to target specific genes so that the disease either does not progress or is even eradicated from an individual. In this paper we describe novel experiments using single-layer artificial neural networks for modelling both non-temporal (classificatory) and temporal microarray data.

Keywords: Gene expression analysis; Perceptrons; Temporal gene networks

* Corresponding author. Tel.: +44-1392-264-064; fax: +44-1392-264-067.
E-mail address: a.narayanan@ex.ac.uk (A. Narayanan).

0925-2312/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2003.10.017

1. Introduction

Microarrays and gene chips have made it possible to experimentally measure the expression levels of many individuals' genes simultaneously [19]. Many diseases, because they are caused by problems such as chromosomal imbalance and gene mutations, give rise to abnormal gene expression patterns. These patterns provide a great deal of information about the underlying genetic processes and states of various diseases. If these patterns can be appropriately analysed they can be useful for detecting disease and identifying the extent to which a patient is suffering from the disease (early or late stages), which in turn can help in the management of the disease.

There are two major problems for current gene expression analysis techniques. The first is that of gene dimensionality. For the dataset examined in experiment A, about 7000 genes are measured on an Affymetrix gene chip, and each sample or individual will therefore be associated with 7000 [1] measurements. If there are 100 samples or individuals, the sheer volume of data leads to the need for fast analytical tools. Such techniques include clustering this large number of genes into smaller groups of genes depending on some measure of similarity. Clustering will not be able to identify the possibly small number of individual genes that are directly and perhaps causally related to the classification of disease or independently measured phenotype, but will instead locate this gene with other genes that are similar to that gene in terms of co-expression and co-regulation.

The second problem is that of sparsity. Typically, while thousands of genes are measured, only a few dozen samples are taken. In database terms, there are many more attributes (genes) than records (samples). In the case of the myeloma database, for instance, there are 7129 measurements for each of only 105 individuals. There is a great danger that any technique for identifying relationships between genes and phenotype of records will find spurious relationships, given the vast space of possibilities (the fewer the records, the more unconstrained the possible relationships between gene values and class).

Given these two problems, gene expression analysis (G) can be defined to be concerned with selecting a small subset of relevant genes from the original set of genes (the S problem) as well as combining individual genes in either the original or smaller subsets of genes to identify important classificatory and possibly causal relationships (the C problem). That is, $G = S + C$.

Previous approaches to gene expression analysis have tended to focus on one but not both of these problems. For instance, the extraction of gene regulatory networks from microarray data [14,15,30], where the emphasis was on reverse engineering of 'wiring diagrams' which represent the way in which genes at one timestep influence positively (switch on) and negatively (switch off) genes at the next timestep, is an example of a C problem that assumed the pre-identification of a small number of genes (i.e. assumes a solution to the S problem). Other approaches have included supervised

[1] Not all 7129 measurements on an Affymetrix gene chip measure real genes. The Affymetrix process involves the use of several dozen Affymetrix 'genes' as a control for the accuracy of total measurement. These control genes are excluded from our analysis.

and unsupervised data analysis for the *S* problem (Brazma and Vilo [8] provide an overview of these techniques), graph-theoretic approaches for the *C* problem [51], clustering, cluster analysis and 'gene shaving' for the *S* problem (e.g. [16,20,52]) and Bayesian approaches for the *C* problem (e.g. [4,18]).

Previous artificial neural network (ANN) approaches to gene expression analysis have focused on self-organising maps [42,43], self-organising tree algorithms [21] and associative neural networks [6], all for the *S* problem. Another *S* approach is that of Khan et al. [28], who first 'quality filtered' the measurements for 6567 genes down to 2308 genes and then applied principal component analysis to reduce the genes down to 10 dominant PCA components. These components were then used, through a process of cross-validation, to generate 3750 ANN models which successfully distinguished between the four cancer types of their blue cell tumour database. These ANN models were then analysed to extract 96 genes by measuring the sensitivity of the classification to a change in the expression level of each gene. The 10 dominant PCA components for these 96 genes were then used to 'recalibrate' the ANN models to result in a correct classification of all 63 cases in their database, with multidimensional scaling analysis revealing that the four different cancer types were statistically distinguished. It is not clear why these authors did not try a total ANN approach to the gene reduction problem *S*, or whether the 10 principal components specified are biologically as well as statistically significant, or even whether a simpler method could have achieved the same results. It is also not clear whether their approach can be generalised to other datasets that involve binary or other multi-valued classification, or temporal information.

There is currently no standard method for gene expression analysis and modelling, mainly because there is currently no agreed way to solve the two problems *S* and *C* simultaneously or in harmony. Approaches focusing on *S*, given their dependence on statistical clustering and other mathematical techniques, often fail to identify or distinguish the critical genes which individually or in combination help to classify samples based on their phenotype or class value (the *C* problem). Associations and correlations are often found between genes and/or samples (as for instance with clustering algorithms), thereby solving the *S* problem. However, such clusters do not identify the importance of specific genes for classification or the specific contribution they make with other genes to determining or predicting whether a sample falls within a certain class (the *C* problem). Such clusters can also be very large, even if they contain just 1% of genes actually measured on a microarray (e.g. a cluster of 1% of 30,000 genes is still 300 genes). By reducing the dimensionality of the problem to 300 genes, we have still not achieved the requirement of these techniques, which is to yield a database small enough to be understood by individuals and ultimately tested by experiments in the laboratory. By discovering a very small subset of genes that can completely classify data we provide a better chance of moving from in silico predictions to in vitro tests of the proposed genetic mechanisms.

Surprisingly, there has been very little work on the application of single-layer supervised backpropagation ANNs, or perceptrons, for solving problems *S* and *C* together. The aim of the research described here is to evaluate the application of such networks in the domain of non-temporal (i.e. data which is either intrinsically non-temporal or which has been time-averaged) and temporal (i.e. data for individuals which is repeat-

edly measured over time) gene expression data (see Narayanan et al. [34] for a review of other ANN applications in bioinformatics). Previous approaches in the ANN literature to dimensionality reduction (e.g. [35,37] have used (i) multilayered ANNs, where, because of the presence of intervening layers of nodes between input and output layer, it is difficult to extract the information from the ANNs directly as to which genes combine which other genes to effectively classify samples (extra mathematical and/or statistical techniques such as principal component analysis have to be used), and (ii) datasets which are extremely simple in comparison to gene expression datasets.

2. Classificatory gene expression analysis

2.1. Introduction

Our approach to classification is to adopt single layer ANNs wherever possible to problems S and C. Our research involves the use of discrete-valued data models [22] of gene expression myeloma data. The weights linking input gene nodes to output phenotype/class nodes are examined to determine the influence of each node. Thresholds based on standard deviations are calculated to provide a selection criterion for a reduced number of genes, and these genes are then used in a second iteration of the procedure. The process of training and selection is repeated to produce increasingly smaller sets of genes until a small number of genes is arrived at. This handful of genes can be used to predict and classify new cases. Discrete-valued approaches to gene expression are now widely accepted [2,38,41]. In any case, the Affymetrix method produces a gene expression dataset where each gene is identified as 'present', 'absent' or 'marginal'.[2]

2.2. Multiple myeloma

Multiple myeloma[3] is a type of cancer that affects certain white blood cells called plasma cells and is often referred to as plasma cell dyscrasias. Plasma cells are formed when B-cell lymphocytes mature in response to an infection, producing and releasing immunoglobins (antibodies) that attack and help kill the disease.[4] White blood cells begin their development in bone marrow, the soft, spongy tissue that fills the centre of most of the bones. When cancer involves plasma cells, the body keeps producing more and more of these cells. The unneeded cells, all abnormal and exactly alike, are called myeloma cells. In most cases, myeloma cells collect in a variety of bones, often forming multiple tumours and causing other problems. When this happens, the disease is called multiple myeloma.[5] There is currently little understanding of the genetic causes of the disease. One long-term goal is to come up with quick and effective

[2] See http://www.affymetrix.com/corporate/outreach/educator.affx for further information on how these values are produced.

[3] See http://www.myeloma.org for further information on myeloma.

[4] See MedLinePlus, National Institute of health, National Library of Medicine. http://www.nlm.nih.gov/medlineplus/healthtopics.html

[5] http://www.medifocus.com.

diagnosis of myeloma based on gene expression profiles of individuals that will also identify exactly how far the disease has spread so that appropriate therapy regimes can be initiated. However, it is currently not clear which genes to target for measurement.

2.3. Methods for experiment A

The gene expression profiles of multiple myeloma patients were established with Affymetrix Gene Chip.txt files[6] from 74 diagnosed multiple myeloma patients and 31 normal bone marrow individuals for 7129 genes.[7] The full data set contains almost 290,000 'gene present' values, 450,000 'gene absent' values, and 9700 'gene marginal' values, as identified by Affymetrix. The data set was split into three randomly generated subsets of 35 individuals each, preserving the ratio of normal to myeloma cases in each subset. Subsets A and B contained 10 normal and 25 myeloma cases each, whereas Subset C contained 11 normal and 24 myeloma cases. 'Normal' was represented as 0, and 'myeloma' as 1 on the single output node.[8] 'Present' gene values were coded as 1, 'Marginal' as 0.5 and 'Absent' as 0 after preliminary experiments indicated that this coding technique allowed the single-layer ANN to converge.[9] Standard backpropagation was used with a learning rate of 0.001. Weights were allowed to vary between -1 and $+1$, with random initialisation. Given the architecture (single layer) and representation used (assuming linear separability), the ANN adopted is essentially a perceptron. That is, the output node value is the sum of weighted inputs: $O(x) = \sum_{i=1}^{7129} w_i x_i$, where $O(x)$ is the output value for a sample x and $w_i x_i$ is the weight of the link between each of the 7129 genes making up sample x and the single output node. If the presence of some features x_i tends the output node to be activated, w_i will be positive. If the presence of some features x_i inhibits the output node, w_i will be negative. The transfer function for the output node is sigmoid/logistic (output values between -1 and $+1$). The learning rule is the standard perceptron learning rule: $w_j(t+1) = w_j(t) + \eta(d - y)x$, where connection weight w_j at iteration/epoch $t+1$ is the sum of its weight at iteration/epoch t and the difference between the desired output d and actual output y for an input pattern x, and where this difference is moderated by a step size η that can vary between 0 and 1.

A 'three-fold cross-validation' learning, recalibration and testing regime was then adopted, repeated three times. That is, for the first fold, Subset A was used to train the

[6] More details on the Affymetrix process can be obtained from www.affymetrix.com.

[7] The full original myeloma data files are available online from http://lambertlab.uams.edu/publicdata.htm The full myeloma SNNS data file used in the experiments below is available from http://www.dcs.ex.ac.uk/~anarayan/myeloma/.

[8] A class representation of 1 (myeloma) and -1 (normal) was also tried. However, on testing, none of the test patterns with -1 achieved better than 0 on the output node, indicating that the ANN was having difficulty in adjusting weights to produce both 1 and -1 (probably due to the two-to-one ratio of myeloma to normal cases). We therefore represented normal as 0 in all further experiments. Our purpose was to identify the genes closely related to myeloma (1) rather than normal (0), in any case.

[9] The myeloma dataset contains significantly more 'absent' values than 'present'. When absent values were represented as -1, the ANN could not adjust its weights sufficiently for the present ($+1$) values.

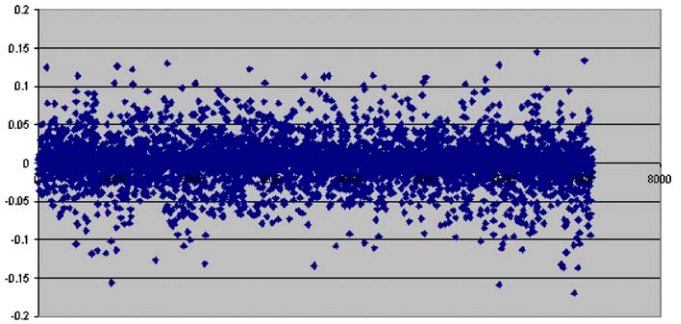

Fig. 1. A scatter diagram for all 7129 genes (x-axis) added across all three runs, ranging in value from -0.16984 to 0.14484 (y-axis). The combined average is 0.001586 and the revised standard deviation is 0.024. There are 1444 genes with 0 weight, 2152 with negative weights, and 3533 with positive weights. Thresholds of ± 2 standard deviations from the mean were calculated, resulting in 481 genes for the second iteration.

neural network (10 000 epochs), Subset B was then used to recalibrate and fine-tune the network (1000 epochs), and Subset C was used for testing purposes after the weights were clamped. This procedure was repeated using Subsets B and C for training and recalibration, respectively, and A for testing (fold 2). Finally C and A were used for training and calibration, followed by B for testing (fold 3). Each subset functioned as a training set, a calibration set and a test set at some stage in the procedure.

During the training and recalibration phases on all three folds, the sum-of-squared-error (SSE) on the output node was well below 0.001 after 11 000 epochs (step size/learning rate of 0.001). There was a sharp reduction in SSE for the first 15% or so of epochs from a value above 10 to below 1, followed by a gradual reduction over the remaining epochs to below 0.001. All three folds resulted in 100% generalisation on the unseen data, using 0.1 and below to signify 'normal' and 0.9 and above to signify 'myeloma' on the output node. The three sets of clamped weights for each gene were then added together so that minor perturbations of weights would cancel each other out, leading to a clearer identification of those genes that were consistently negative or positive (Fig. 1). A combined average and standard deviation were calculated, and only those genes with values exceeding ± 2 standard deviations from the new average were allowed to survive. 481 genes out of the original 7129 genes met or exceeded these thresholds. These 481 gene values were then extracted from the full database, together with the class information of each sample. If the reduction from 7129 genes to 481 appears too severe, a more relaxed standard deviation range can be used. For this dataset, one standard deviation will allow about 10% of genes to survive, whereas 2 standard deviations will permit only about 5% of genes to survive.

The dataset was again split into three subsets, with the three subsets being used for training, recalibration and testing on a three-fold basis as before. This time only 1000 epochs were used for training and 100 for recalibration (learning rate 0.01), and again the ANNs demonstrated 100% generalisation on the three unseen datasets using 0.1 and below to signify normal and 0.9 and above to signify myeloma. A sharp

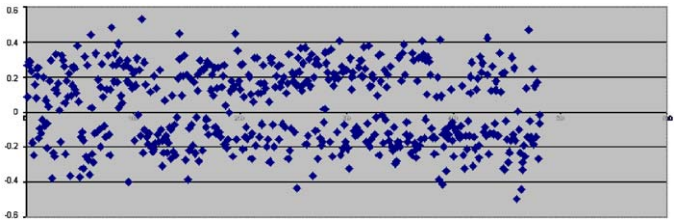

Fig. 2. A scatter diagram of the 481 (x-axis) genes, with weights summed across three folds. A much clearer separation between positive and negative weight genes can now be discerned. The weights now range from -0.49878 to 0.52492 (y-axis), with average 0.026749 and standard deviation of 0.217859. There are no 0 weights, with 234 genes having negative weights and 247 genes having positive weights. However, some genes still tended towards 0 after summation across the three folds. A threshold of ± 1.5 standard deviations from the mean was calculated, resulting in 39 genes.

reduction in SSE was again observed for the first 20% or so of epochs before a more gradual convergence was observed. The SSE figure was again well below 0.01 after training and recalibration on all three folds. The clamped weights of the 481 genes were added together to help identify more strongly those genes which were either negative or positive throughout all training/recalibration regimes (Fig. 2). This time, a more relaxed range of standard deviation (± 1.5 standard deviations from the combined average) was used, resulting in 39 genes.

Finally, the gene values for just these 39 genes were extracted, together with class values for samples, and three new subsets generated. Once again, a three-fold methodology was adopted, with each fold involving one of the datasets for training, the other for recalibration (100 epochs each), and the third for testing. The SSE on the output node was again well below 0.01 after training and recalibration on all three folds (learning rate 0.1). This time, there was dramatic lowering of SSE within the first 30% of epochs (below 1%), followed by a gradual convergence below 0.01 SSE. Again, 100% generalisation on the unseen cases was achieved, this time with 0.2 and below signifying normal and 0.8 and above myeloma. The clamped weights of the three folds were again added together (Fig. 3), and a final, ranked list of these 39 genes was output (Table 1).

2.4. Discussion of results of experiment A

If we examine the weights of the gene connections to the output node in Table 1, gene L00022[10] has the strongest negative value of -1.69196. Given that one of the targets at the output node is 1 (myeloma), this negatively weighted link appears to indicate that the ANN has identified that an important influence on myeloma is the *absence* of this gene. The positive link, on the other hand, between X57809 and the

[10] Affymetrix data are stored using accession codes, where these accession codes are labels consisting of a letter followed by 4 or 5 digits for the full names of genes. The relationship between accession codes and gene names can be found using, for instance, the Entrez system at http://www.ncbi.nlm.nih.gov/Entrez/.

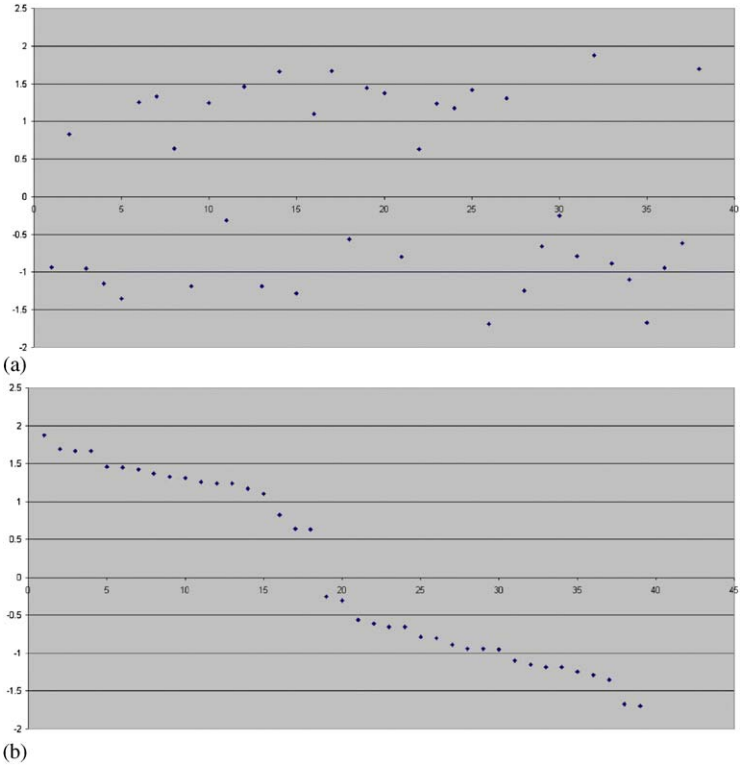

Fig. 3. Scatter diagrams for the final set of 39 genes, with weights added across three folds. Weights now range from −1.69196 to 1.87665, with combined average of 0.080385 and standard deviation of 1.19692. Twenty one of the weights were negative and 18 were positive. Note that two genes tended back to 0 after summation across the three folds. The scatter diagram on top describes the values of all 39 genes in numerical order (x-axis), whereas the diagram below represents the ranked values as given in Table 1.

output node (+1.87665) seems to indicate the importance of the *presence* of this gene on myeloma.

On examining the genes in more detail, it was reassuring to see that many had been linked to cancer in general and myeloma in particular in the medical literature. [11] L00022 is an immunoglobulin (Ig) active heavy chain epsilon-1 gene, constant region, with some historical links to myeloma [27]. L36033 (−1.18266)) is a pre-B cell stimulating factor homologue (stromal cell-derived factor 1 SDF1b gene) modulating the growth and functional activities of immune cells. Its absence therefore can be strongly hypothesised to affect the correct functioning of the immune system. SDF1 has been previously reported to induce intracellular actin polymerisation in lymphocytes [7] as well as act as a chemoattractant for human hematopoietic progenitor cells express-

[11] We used PubMed and Medline to try to identify the biological significance of the genes. These on-line medical literature databases are available through Entrez (see footnote 9).

Table 1
The final 39 genes ranked by weight value, with weights summed across three runs

X57809	1.87665	Rearranged immunoglobulin lambda light chain
M34516	1.69389	Omega light chain protein 14.1 (Ig lambda chain related) gene, exon 3
U78525	1.66582	Translation initiation factor (eIF3)
U09579	1.66234	Melanoma differentiation associated (mda-6)
M55210	1.46093	Laminin B2 chain gene, exon 28
X16416	1.44751	C-abl mRNA encoding p150 protein
HG2383-HT4824	1.42065	Cystathionine Beta Synthase, Alt. Splice 3
X57129	1.3742	H1.2 gene for histone H1
L05779	1.32891	Cytosolic epoxide hydrolase
S71043	1.30777	Alpha 2 immunoglobulin A heavy chain allotype 2
HG4157-HT4427	1.25573	Glycinamide ribonucleotide Synthetase
L77886	1.24156	Tyrosine phosphatase
X87613	1.23623	Skeletal muscle abundant protein
X90392	1.17397	Deoxyribonuclease X
U73167	1.10087	Cosmid clone LUCA14 from 3p21.3
D87683	0.82712	KIAA0243 gene, unknown function
L13329	0.63726	Iduronate-2-sulfhatase (IDS) gene
X81788	0.63134	ICT1 (alias DS-1), novel gene associated with colon carcinoma
Z68228	−0.25491	Plakoglobin
M25897	−0.30939	Platelet factor 4 (PF4)
U81787	−0.56128	Wnt10B (glycoproteins that have been implicated in oncogenesis)
L13943	−0.6129	Glycerol kinase (GK)
D50915	−0.65252	KIAA0125 gene, unknown function
M34996	−0.65549	MHC cell surface glycoprotein (HLA-DQA)
M57466	−0.78742	MHC class II HLA-DP light chain
X62744	−0.8006	RING6 mRNA for HLA class II alpha chain-like product
K02882	−0.887	Germline IgD-chain gene, C-region, second domain of membrane terminus
D87024	−0.93665	Immunoglobulin lambda gene
Z00010	−0.94211	Germ line pseudogene for immunoglobulin kappa light chain leader peptide and variable region
D88270	−0.95079	Immunoglobulin lambda gene
HG3731-HT4001	−1.09965	Immunoglobulin heavy chain
HG2715-HT2811	−1.15114	Tyrosine kinase
L36033	−1.18266	Pre-B cell stimulating factor homologue (SDF1b)
M63928	−1.18442	T cell activation antigen (CD27)
Z74616	−1.24817	mRNA for prepro-alpha2(I) collagen
U64998	−1.28439	Ribonuclease k6 precursor gene
HG3872-HT4142	−1.34684	Immunoglobulin gamma heavy chain, V(6)Djc Regions
U24685	−1.66743	Anti-B cell autoantibody IgM heavy chain variable V--D--J region (VH4) gene
L00022	−1.69196	Immunovlobulin active heavy chain epsilon-1 gene, constant region

Gene names, as provided by Entrez, are provided for information.

ing CD34 [1]. It has also been linked to stem cell mobilisation and myeloma [50]. M63928 (−1.18422)) is an alias for TNFRSF7 (tumour necrosis factor receptor superfamily, member 7 precursor), which is a T cell (CD27L receptor) activation antigen (CD27) [10], and previously linked to myeloma [13]. Again, it can be strongly hypothesised that the absence of tumour necrosis factors will affect the ability of the immune system to deal with tumours. The results reported here add to the evidence that one of the possible reasons for myeloma is that the lack of expression of this receptor gene in tumour cells prevents the necrosis factor from binding to such cells. U24685 (−1.66743) is anti-B cell autoantibody IgM heavy chain variable V–D–J region (VH4) gene with previous links to lymphocytic leukaemia [29]. U64998 (−1.24817) is ribonuclease k6 precursor gene with strong links to host defence systems. Further down the list is D88270 (−0.95079), an immunoglobulin (Ig) lambda gene, another immune system gene the absence of which, according to our results, is strongly linked to myeloma.

With regard to positively weighted genes, U57809 (+1.87665) is rearranged immunoglobulin lambda light chain with some evidence that it is linked to the deletion of DNA [12]. Its positive weight indicates its strong presence with regard to myeloma. M34516 (+1.69389) is omega light chain protein 14.1, the expression of which is restricted to pre-B cells [17]. U78525 (+1.66582) is a translation initiation factor (eIF3) that plays an important role in the synthesis of proteins and cell proliferation [46]. U09579 (+1.66234) is melanoma differentiation associated (mda-6) mRNA, an oncogene not previously associated with myeloma. M55210 (+1.46093) is laminin B2 chain gene, an aid to the promotion of genes but with no previous links to myeloma or cancer. X16416 (+1.44751) is ABL1_HUMAN, a proto-oncogene tyrosine-protein kinase ABL1 encoding p150 protein.

While further clinical and laboratory research is required to determine whether some of these genes represent newly discovered knowledge concerning myeloma, it is clear that our perceptron may be able to capture subtle relationships between input and output which may be beyond the ability of other statistical techniques. Our results (100% correct generalisation on all unseen cases) indicate that our ANN of 39 genes can be used to successfully predict future myeloma cases, even if the reasons for classification (i.e. which genes contribute most to the classification) are not immediately obvious in the weights connecting the input nodes to the output node. However, it is possible to extract some symbolic knowledge from our trained ANNs by simply trying individual genes and various combinations of genes with high weight values. For instance, the following rules for myeloma can be tried using the weight information in Table 1 and their effect tested on the dataset:

(i) *If L00022* (*Ig active heavy chain epsilon*-1 *gene*) *is absent then myeloma* (72 *of the* 74 *myeloma cases captured by this rule*, *and three false positives*).

(ii) *If L00022*(*Ig active heavy chain epsilon*-1 *gene*) *is absent and D88270* (*Ig lambda gene*) *is absent*, *then myeloma* (71 *of the* 74 *cases correctly classified*, *and one false positive*).

Interestingly, if we reverse the ‘polarity’ of these two genes, we have rules for normal cases also.

(iii) *If L00022 is present then normal* (26 *of the* 31 *normal cases correctly classified, with one false negative*).
(iv) *If L00022 is present and D88270 is present then normal* (28 *of the* 31 *normal cases correctly classified, with no false negatives*).

However, these rules are clearly not so good as the combined influence of all 39 genes which, on our three-fold training, recalibration and testing regime, produced 100% predictive accuracy on all unseen cases. We also ran the symbolic machine learning tool See5 [36] on the 39 gene set, using the system's cross-validation facility (which keeps a random 10% of cases for testing purposes across each of 10 runs). The overall error for See5 was 6.6%, although errors were as high as 18% and 10% on some runs. One of the most successful rule sets (i.e. no errors) generated by See5 was

(a) *If U24685 is absent then myeloma.*
(b) *If D50915 is absent then myeloma.*
(c) *If D50195 is present and U24685 is present then normal.*

2.5. Conclusions to experiment A

Our experiments have shown that a standard, one-layer FFBP ANN (perceptron) can be used to predict whether an individual measured over just 39 genes falls in the myeloma class or not. Furthermore, the weights linking genes to output node can be used to identify key genes and to formulate hypothetical rules to test the extent of data coverage. Once we have a small gene set, other well-established rule-generation techniques, such as See5, can be tried on the genes. All such rules must of course be tested clinically or by other experimental means. For instance, one immediate in silico prediction is that a future 'cure' for myeloma may consist of drugs containing adequate and appropriate levels of Ig active heavy chain epsilon-1 (L00022) and Ig lambda (D88270). This prediction must be subject to rigorous clinical (in vitro) testing before being tested on humans. Nevertheless, the ANN can provide useful information and pointers to clinicians and experimentalists.

The application of simple perceptrons to the problem of myeloma dataset reduction S has resulted in smaller sets of genes with increasingly larger weight values which, upon subsequent analysis C, have been demonstrated to be strongly related to these diseases in exactly the way that the weight values would predict (presence and absence). Users have the option of iterating through this process as many times as needed, using whatever weight thresholds they wish to reduce the dimensionality of the problem as well as check on the efficacy of those genes which only just made the previous threshold. The iterative process can be automated in future web tools. It is possible that new discoveries concerning the involvement of specific genes not previously reported in the literature will arise. The process is transparent and, due to the free availability of SNNS, without financial cost. Comparisons with previous data reduction techniques

used for the myeloma data [12] indicate that the use of FFBP ANNs identifies a smaller and more relevant gene set.

Also, our experiments reveal that it would be unwise to reduce several thousand genes to a handful in just one reduction. Our three-fold, multi-reduction approach demonstrates that gene values can vary between folds as well as across reductions. A gene that performs well in one reduction may perform poorly on the next reduction. Figs. 2 and 3 demonstrate that some genes head back towards a summed 0 weight even after surviving a reduction. Further work is required to identify what the optimal number of reductions and appropriate range of standard deviations should be, for different sizes of gene expression data.

3. Temporal gene expression data

3.1. Introduction

It is predicted that within 5 years a doctor will be able to use gene chip results on a palm held computer to determine how a particular patient metabolises certain drugs, over time. However, the major obstacle to this vision is the need to 'reverse engineer' in realistic time an individual's gene network (the description of how one gene affects other genes by either activating them or inhibiting them over a number of timesteps) from the vast amount of temporal gene expression data contained on gene chips.

Temporal microarray data is created by observing gene expression values (or "activation" values) over a number of timesteps, often whilst subjecting the organism to some stimulus. The measurements gained from this study often span thousands of genes (fields, attributes) and only tens of timesteps (records), which makes it relatively unusual in structure and not as amenable to traditional symbolic analysis as other biological databases. The goal of analysing such data is to determine the pattern of excitations and inhibitions between genes that make up a gene network for that organism. A gene network can be inferred by observing the interactions between genes between one timestep and the next. Finding an optimal or even approximate gene network can lead to a better understanding of the regulatory processes occurring in the organism and therefore represent a significant step towards understanding the phenotype of the organism [44]. Drugs designed for a particular individual, given the individual's specific gene regulatory network, can then be used to counter diseases which have a genetic basis.

A number of methods have been tried to generate gene networks from such data: Bayesian networks [33,40], clustering [5,32], statistics [31,45], visualisation [11], weight matrices [47,48], unsupervised neural networks [42], genetic algorithms [3] and supervised learning algorithms [9] have all been used. Simple neural networks, i.e. single

[12] Page, D., Zhan, F., Cussens, J., Waddell, M., Hardin, J., Barlogie, B. and Shaughnessy, J. Jr. (2002). Comparative data mining for microarrays: a case study based on multiple myeloma. Poster presentation at ISMB02, Edmonton, Canada. Presentation slides are available at http://www.biostat.wisc.edu/~mwaddell/publications/. Technical report is available at http://www.cs.wisc.edu/~dpage/.

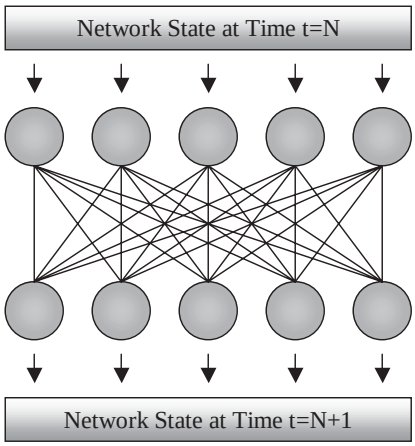

Fig. 4. Gene expression values at Time $t = N$ are input to the network and expression values at Time $t = N + 1$ are used as training targets for the neural network.

layer, feedforward, back-propagation artificial neural networks (ANNs), have not been used apart from our own experiments in this area, to be described below. One of the reasons for this may be that it has not been clear how to represent such data to a FFBP network.

There are two aspects to representation: architectural representation, and data representation [26]. The former deals with issues of layers and learning rules, while the latter deals with data input. With regard to the latter issue, we use Boolean networks [39] that approximate real-world gene expression data, in that gene expression values are restricted to "on" or excitation, and "off" or inhibition, states [22]. Other interpretations of two-valued gene networks are the 'presence' or 'absence' of specific genes, given their values as recorded on the gene chip.

Genes that are on at one timestep can subsequently turn other genes on or off in the next timestep. Any gene that receives no "excitation" in a timestep will switch itself off or can be in a neutral state. These networks therefore model the interactions between genes in a controlled and intelligible manner. The problems associated with extracting networks from what might be considered Boolean data are still not trivial. Gene expression data, including Boolean networks, are problematic because of their dimensionality. They have a large number of attributes (genes) and few examples (time samples): a typical set for the yeast genome, for example, can consist of 6000 genes (attributes) and less than 10 timesteps for each of a handful of samples. In this section we discuss the use of FFBP neural networks to model temporal gene microarray data represented in Boolean format. Algorithms already exist which can perform this task in provably correct terms, e.g. REVEAL [30], but they suffer exponential complexity with respect to the connectivity of the network and the number of genes.

With regard to the issue of architectural representation, again we use the simplest form of FFBP ANN: there are no hidden layers (Fig. 4) and the activation function

used is the simplest of all—the step function. A simple change of activation function, either to the sigmoid or tanh (hyperbolic tangent) function, is all that is required to allow the FFBP ANN to be used on real-world data. The research to be described below focuses on temporal data, that is, data that is assumed to come from one or more organisms over a number of timesteps.

Overall, therefore, our aim in the following experiments is to determine whether single layer FFBP ANNs with a step function are feasible for modelling temporal gene expression data stored in absolute, or Boolean, form. The aim in the first instance is to determine the simplest type of FFBP network possible for dealing with artificially produced temporal gene expression data represented in Boolean form, where rules have been used to generate the data (forward engineering the data) and to see whether it is possible, after successful training of the FFBP ANNs, to reverse engineer back to the original rules by looking at the weight relationships between input and output layers of the ANNs alone. In the real-world, of course, it will not be possible to determine the accuracy of the reverse engineered rules in purely computational terms, because it will not be known what the true causal relationships between genes actually consists of. Nevertheless, the experiments reported here demonstrate that reverse engineering with trained FFBP ANNs leads back to the original rule set which produced the data in the first place and thereby lends confidence to the view that neural networks can play a significant part in reverse engineering from gene expression data. We now describe two experiments, B1 and B2, the former dealing with artificial data and the latter with real-world data.

3.2. Method for experiment B1 using artificial temporal gene expression data

3.2.1. Architecture

The FFBP ANN is single layered and fully connected between input and output layer, with the activation function, the step function, shown here:

$$\left.\begin{array}{l}\sum w_{ij}x_i > 0.5\\ \sum w_{ij}x_i < 0.5\end{array}\right\}\begin{array}{c}1\\ \overline{0}\end{array},$$

where w_{ij} is the weighted value connecting node i to node j and x_i is activation value for node i. There are no bias terms. The architecture learns the input and output connections by virtue of the backpropagation algorithm for the step function which is quite simple: $w_i(t+1)=w_i(t)\pm\eta x_i(t)$, where $w_i(t+1)$ is the weight value at time $t+1$ and η is the learning rate associated with x_i at time t. Due to the fact that the inputs will be either 0 or 1, there must be some graduation to allow the algorithm to learn weights around the threshold (here 0.5) area. Therefore the term η must be included to direct backpropagation in smaller steps towards an optimal solution. The term has been introduced in the experiments below as $\frac{1}{20}$ and means that the weights progress towards the optimum in steps of 0.05.

Training consists of presenting pairs of data (input and target) to the network. The pairs in this instance consist of gene expression values from two consecutive timesteps. Gene values for Time $= t$ are input to the algorithm and the target consists of gene

Table 2
An artificial example of the type of boolean network used in experiments (a "Liang Network")

Time = $t0$			Time = $t1$		
Gene1	Gene 2	Gene 3	Gene1	Gene 2	Gene 3
0	0	0	0	0	0
1	0	0	0	1	0
0	1	0	1	0	0
1	1	0	1	1	1
0	0	1	0	1	1
1	0	1	0	1	1
0	1	1	1	1	1
1	1	1	1	1	1

The left-hand table represents the gene expression values at Time $T = t0$, and the right-hand table, those at Time $T = t1$ (i.e. $T = t+1$). Note that only three genes are involved here. In the actual forward engineering of data, 10 genes are involved. Taken from Liang [30].

values at Time $= t + 1$ (Fig. 4). Once training has been completed, testing consists of re-inputting all input Time $= t$ and comparing it against the target Time $= t + 1$ to compute a percentage accuracy.

3.2.2. *Artificial data generation*

The experiments below have been run on artificial gene expression data 'forward engineered' by a program which creates temporal Boolean data based on 'Liang Networks' [19,30] (Table 2). A Liang network consists of rules which are made of AND and OR operators distributed randomly in a number of random fixed length logical expression, but to a specific k value which determines the maximum number of genes which can affect others in the next timestep. We used only two k values. If $k = 2$, then between two and four genes can be affected at Time $= t1$ by genes at Time $= t0$. When $k = 3$, anything between three and nine genes are involved in the next timestep. The experiments below have been undertaken on artificial data with 10 genes derived from Boolean rules with k values of 2 and 3. The error represents the percentage difference in the output and the target of the Boolean network. Keedwell et al. [26] provide more details.

As can be seen in this artificial example (Table 2), the activations at $T = t0$ represent the full logic table of the three genes. The activations at $T = t1$ represent the activations once the rules have been applied. For instance, the expression values 0 0 1 for gene 1, gene 2 and gene 3, respectively, at $t0$ become 0 1 1 for these three genes at $t1$ (fifth row of Table 2). The 'reverse engineering' task in its basic form is to use the information in the left and right-hand parts of Table 2 (row by row) to extract a candidate set of rules which are consistent with the data. While this may be trivial for three genes (and eight rows), the problem quickly becomes non-trivial, especially when one considers Liang networks consisting of hundreds and possibly thousands of genes. The rules for producing ('forward engineering') the artificial gene expression data contained in Table 2 are as follows (in the format gene expression at $t1$ = gene

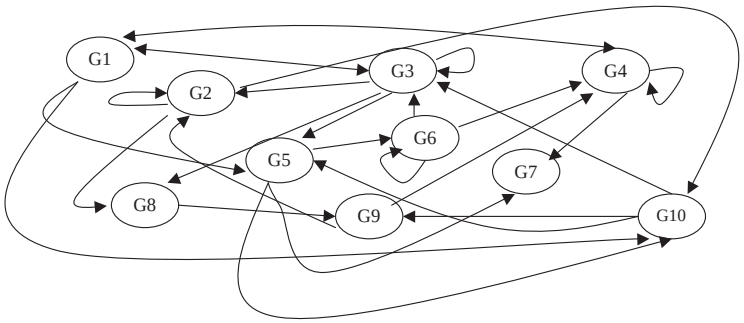

Fig. 5. Gene regulatory "wiring diagram" for $k = 2$ dataset number 3. As can be seen, self-connections are permitted and the total number of connections is quite large. Note that the in degree of each node is 4 or less. G1–G10 represent the 10 genes in the data.

expression at $t0$):

Rule 1: gene 1' = gene 2;

Rule 2: gene 2' = gene 1 OR gene 3

Rule 3: gene 3' = (gene 1 AND gene 2) OR (gene 2 AND gene 3) OR (gene 1 AND gene 3).

For instance, looking at the fifth row of Table 2 again, since gene 2 is 0 at $t0$, gene 1's value at $t1$ is 0 (first rule); since gene 3's value is 1 at $t0$, gene 2's value is 1 at $t1$ (satisfying rule 2), and so on. The data on which these experiments are based are exactly the same except that the rules are randomly distributed and the networks involve 10 genes rather than 3. Altogether, six artificial datasets were produced, with different random rules. Three of the datasets had k value 2 and three had k value 3. Already, 2^{10} (1024) records are needed to specify every possible truth assignment for each dataset. If one were to increase this to 20 genes, this would require 2^{20} (1,048,576) records.

Although Liang networks quickly become intractable to fully enumerate, the point here is to determine whether FFBP ANNs can successfully reverse engineer even constrained, forward engineered Liang networks. If they do not, there is very little chance that FFBP ANNs will be successful for large numbers of genes with sparse data (typically, a real microarray will have data for several hundred and perhaps several thousand genes, for anything between 3 and 50 timesteps) and where the original rules giving rise to the data are not known. When the rules become even more complex, then they can be expressed as a wiring diagram. This method of showing the interactions between genes can also be helpful in allowing biologists to see the overall gene regulatory network. The desired regulatory network for the $k = 2$ dataset number 3 (see below) is shown in Fig. 5. Such wiring diagrams are abstract in the sense that the arrows are neutral in terms of excitation ('switch on') and inhibition ('switch off'), signifying instead that one gene influences another. Also, if there is more than one arrow pointing to a gene, wiring diagrams do not specify whether an 'AND' or 'OR' relation is involved. Nevertheless, such wiring diagrams provide an immediate graphical represen-

Table 3
Percentage success of the step-function neural method over 6 "Liang Networks"

k value	Data 1 (%)	Data 2 (%)	Data 3 (%)
2	98.125	100	99.84
3	100	98.359	99.375

tation of how genes are linked together and affect each other through excitation and inhibition.

3.2.3. Results of experiment B1

Six separate step-function linear models were trained on six 10-gene Liang-type datasets (1024 records each), three with $k=2$ and three with $k=3$ (see Table 3). The linear models seen here were all implemented in C++ using the equations above and the runs were made with a boundary of 0.1% error, or 100 epochs, whichever was achieved sooner.

3.2.4. Discussion of results of experiment B1

As can be seen (Table 3), the results are very close to the optimum for these networks. The average accuracy over the 6 datasets is 99.283%. That is, when all 1024 records are used for training as well as for testing, almost perfect results are obtained. This represents extremely good accuracy over this size and number of datasets. Higher k values do not appear to affect the accuracy. An important aspect, though, is to reveal the knowledge and rules embodied in the networks to see whether the ANNs have indeed reverse engineered successfully back to the original rules which gave rise to the datasets in the first place. What is interesting is that the weights reveal a great deal about the actual distribution of the data—for instance, the most common weights seen are

0.6–0.7: Used for genes which are the sole contributors to output, or genes which are part of OR logic functions. Genes coupled with this weight are capable of turning on units by themselves.

0.05–0.2: Used for genes which do not contribute to the output of that gene. These weights effectively nullify the contribution of that gene.

0.25–0.45: Used for genes which are part of AND statements, where these weight values are not enough to trigger the gene directly but when coupled together with other similarly valued genes will give good results.

These mathematical values were used successfully to extract rules linking genes in the input layer to genes in the output layer and so derive rules from the trained neural networks. The experiments therefore demonstrate that, with constrained Liang datasets, single-layer FFBP ANNs can model temporal Boolean gene expression data and successfully reverse engineer such data to derive rules that are compatible with the data and close in accuracy to the actual rules which underlie the data.

To test this claim further, we trained the model on fewer numbers of records to ensure the ANN was capable of generalising to new data. Table 4 shows the high

Table 4
The percentage accuracy of the step-function neural model when trained on only 100 records and tested on the entire enumeration (1024 records)

k value	Data 1 (%)	Data 2 (%)	Data 3 (%)
2	96.67	97.92	97.98
3	95.53	95.28	97.46

level of accuracy (average 96.81%) which is maintained when training on only 100 randomly chosen records (less than 10% of the available data) and tested on the full enumeration of the Liang network.

3.2.5. Conclusion to experiment B1

Our temporal ANNs are not learning to classify and are therefore not attempting to discriminate linearly between different class values on the basis of the data. Instead, our networks are learning how to predict the next microarray state (the state of data at the next subsequent timepoint, given data from the previous time point). If it is found that microarray states are dependent on earlier states more than one timestep away, the use of recurrent nets may well have to be considered.

Since we use only single layered FFBP ANNs, the connections between all genes at the input layer and individual genes at the output layer have only one minimum, not multiple local minima as with networks with hidden layers. Therefore, given sufficient computational time, the algorithm can be shown to find an optimal answer for each output gene. By using only direct connections between input and output genes, the models described here are multi-perceptrons (i.e. perceptrons with more than one output node). These types of model have a fundamental weakness in that they are unable to satisfactorily process the XOR problem. It is not currently known whether real-world gene regulation has XOR relationships, but if it does then hidden layers will have to be used. As can be seen from the results here, however, the problem of modelling gene expression data and extracting candidate rules from trained networks looks soluble from an ANN viewpoint, using a relatively simple architectural representation. For future work, we intend to reduce the training set to more realistic levels, assess the impact of noise on these networks (microarray expression data are notoriously noisy), and apply single-layer FFBP ANNs to the growing body of temporal microarray data which will become available on the web.

3.3. Experiment B2

3.3.1. Real-world temporal gene expression data

The above experiments have demonstrated the feasibility of applying single-layer FFBP ANNs to artificial temporal data. There is evidence from the experiments above that such networks accurately model the rules and regulatory connections between genes, given forward engineered data. The question now is whether the approach can be applied to real-world data, for which the regulatory connections are not known.

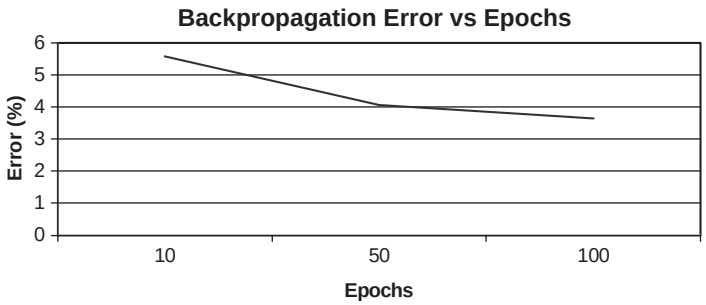

Fig. 6. This graph shows the relationship between the error on the output nodes and epochs during training on the rat spinal cord data [49].

A set of experiments was also completed on a real-world normalised data set known as the Rat Spinal Cord Dataset [49]. This is data derived from microarrays which sampled the Rat Spinal Cord during development and consists of 112 genes and 9 timesteps. The data was converted into 8 pairs and single-layer FFBP ANNs were trained upon this data three times with various parameters changed to give an indication of the accuracy and complexity aspects associated with those parameters.

To run the algorithm on real-world data requires only that the activation function is changed from the step function as described above to the sigmoid function which is used both in standard neural networks and the gene regulatory networks described in Keedwell et al. [26].

3.3.2. *Results of experiment B2*

Fig. 6 shows that the error reduces to around 3%, a respectable result when it comes to real-world data. Also, very small error percentages (the absolute error divided by the average expression level multiplied by 100) are displayed for all the runs, and this shows that the algorithm can consistently discover correct gene networks from real-world data. Future work will consist of analysing the weight matrices of these ANNs to extract regulatory networks and pathways through such networks as well as to check on their biological plausibility. Experiments on this and other larger real-world datasets are ongoing.

3.4. *Conclusion to experiments B1 and B2*

Until very recently, the application of standard FFBP ANNs to gene expression data has been largely ignored, although there are exceptions [22]. As mentioned previously, several ANN-based methods have been discussed but are mainly unsupervised and used for clustering tasks rather than reverse engineering temporal gene expression data. The single-layer ANNs described here have shown that, even without its trademark sigmoid function, the backpropagation algorithm is capable of making an impact in this area. As and when real-world gene expression data is made available publicly in intelligible and standard formats, our research indicates that FFPB ANNs can perform a useful function

in capturing combinations of genes involved in the regulation of other genes. Our conclusion is therefore that ANNs have great potential for solving what was previously considered an intractable problem by bioinformaticians: the reverse engineering problem involving temporal gene expression data. If this is correct, the vision presented at the very beginning of this section, whereby a doctor can be provided with personalised gene regulatory networks involving hundreds and possibly thousands of genes within minutes of analysing temporal gene chip data for separate patients, becomes feasible. The work reported here provides a first and systematic indication that ANNs can analyse temporal data directly without the need to convert temporal data into an average set of measurements over time, where there is a risk of information loss.

4. Conclusion

The experiments reported in this paper represent a first detailed investigation into the application of supervised ANNs for dealing with the *S* and *C* gene expression data analysis. We have shown that such ANNs can solve both *S* and *C*, and therefore the *G* problem, in totality, without recourse to other statistical or mathematical techniques (experiment A). We have also provided preliminary evidence that such ANNs, with a suitable change to the architecture (as many output nodes as input nodes), can deal with temporal gene expression data (experiments B1 and B2).

It could of course be argued that our ANNs are not really backpropagation networks at all, given that they are single layer. In reality, our ANNs are gradient descent networks, but there is the possibility that some gene expression datasets will require hidden layers, in which case our ANNs will need to become multi-layer FFBP networks. We have already pioneered the application of genetic algorithms for the extraction of rules from trained hidden-layer FFBP ANNs [24,25], and if multi-layer ANNs are required in the domain of gene expression analysis we have the tools for mining these networks. Also, it could be claimed that we are performing experiments that could be executed more efficiently with simple linear techniques. Our preliminary application of simple linear regression and other linear methods to the data achieved non-intuitive results, due to the huge sparse space involved. The application of principal component analysis to the data results in arbitrary numbers of components, where the main function is to explain the variance within the data rather than identify key genes important for classification. Also, it is not clear how linear regression, principal component analysis and support vector models can be applied to temporal data, whereas our experiments indicate that one-layer ANNs, with a suitable architectural modification, can solve the *G* problem for both non-temporal and temporal data. Finally, there is a temptation to use complex methods to solve complex problems, when in reality simple techniques can be used to solve such complex tasks. The main result of our experiments is to demonstrate that there is great benefit in keeping gene expression analysis techniques simple, for the sake of ease of repeatability, transparency and generalisabilty.

Finally, there is a need to identify techniques for extracting not just classificatory rules from our trained ANNs but also gene networks, such as the example provided in Fig. 4. Molecular biologists and researchers in the pharmaceutical industry value

these networks, since they provide immediate in silico predictions concerning potential drug targets for knocking out or turning on genes. We stress that all our in silico predictions are just that: in silico. All such predictions must be tested first in vitro and, if successful in the laboratory, in vivo. Current estimates are that one out of a thousand in silico predictions may actually make it to final drug discovery. Currently, there are no known techniques for extracting gene regulatory networks from trained ANNs. Even restricting the number of incoming links to a gene to, say, 5 still leaves the problem of choosing arbitrary combinations of 5 genes from, minimally, 7000 genes per timestep. A recent approach [23] has shown that potentially the union of genetic algorithms and single layer-like neural connections can discover sparse models of gene network connectivity. Any technique that analyses ANN weights and results in a regulatory networks is applicable in this domain.

References

[1] A. Aiuti, I.J. Webb, C. Bleul, T. Springer, J.C. Gutierrez-Ramos, The chemokine SDF-1 is a chemoattractant for human $CD34^+$ hematopoietic progenitor cells and provides a new mechanism to explain the mobilisation of $CD34^+$ progenitors to peripheral blood, J. Exp. Med. 185 (1997) 111–120.

[2] T. Akutsu S. Miyano, S. Kuhara, Identification of genetic networks from a small number of gene expression patterns under the Boolean network model, Pacific Symposium on Biocomputing, Vol. 4, Hawaii, USA, 1999, pp. 17–28.

[3] S. Ando, H. Iba, Identifying the gene regulatory network by real-coded, variable length and multiple-stage GA, 2000, available from http://www.citeseer.nj.nec.com/333262.html.

[4] P. Baldi, A.D. Long, A Bayesian framework for the analysis of microarray expression data: regularised *t*-test and statistical inferences of gene change, Bioinformatics 17 (2001) 509–519.

[5] A. Ben-Dor R. Shamir, Z. Yakhini, Clustering gene expression patterns, J. Comput. Biol. 6 (1999) 281–297.

[6] S. Bicciato, M. Pandin, G.0 Didone, C. Di Bello, Analysis of an associative memory neural network for pattern identification in gene expression data, Workshop on Data Mining in Bioinformatics (BIOKDD01), available from http://www.cs.rpi.edu/∼zaki/BIOKDD01.

[7] C.C. Bleul, R.C. Fuhlbrigge, J.M. Casasnovas, A. Aiuti, T.A. Springer, A highly efficacious lymphocyte chemoattractant, stromal cell-derived factor 1 (SDF-1), J. Exp. Med. 184 (1996) 1101–1110.

[8] A. Brazma, J. Vilo, Gene expression data analysis, FEBS Lett. 480(1) (2000) 17–24, also available at http://industry.ebi.ac.uk/∼vilo/Publications/Febs_Brazma_Vilo_00023893.ps.gz.

[9] A. Califano, G. Stolovitzky, Y. Tu, Analysis of gene expression microarrays for phenotype classification, 2000, available from http://citeseer.nj.nec.com/califano00analysis.html.

[10] D. Camerini, G. Walz, W.A. Loenen, J. Borst, B. Seed, The T cell activation antigen CD27 is a member of the nerve growth factor/tumour necrosis factor receptor gene family, J. Immunol. 147 (9) (1991) 3165–3169.

[11] T. Chen V. Filkov, S.S. Skiena Identifying gene regulatory networks from experimental data, ACM-SIGAT The Third Annual International Conference on Computational Molecular Biology (RECOMB99), Lyon, France, 2000, pp. 94–103.

[12] G. Combriato, H.G. Klobeck, Eur. J. Immunol. 21 (6) (1991) 1513–1522.

[13] F.E. Davies, S.J. Rollinson, A.C. Rawstron, E. Roman, S. Richards, M. Drayson, J.A. Child, G.J. Morgan, High-producer haplotypes of tumour necrosis factor alpha and lymphtoxin alpha are associated with an increased risk of myeloma and have an improved progression-free survival after treatment, J. Clin. Oncol. 18 (15) (2000) 2843–2851.

[14] P. D'haeseleer, S. Liang, R. Somogyi R. Gene expression data analysis and modelling, 1999, available from www.cgl.ucsf.edu/psb/psb99/genetutorial.pdf.

[15] P. D'haeseleer, X. Wen, Mining the gene expression matrix: inferring gene relationships from large scale gene expression data, Bioinformatics 16 (8) (2000) 707–726, available from www.bioinformatics.oupjournals.org.
[16] M.B. Eisen P.T. Spellman P.O. Brown, D. Botstein, Cluster analysis and display of genome-wide expression patterns, Proc. Nat. Acad. Sci. USA 95 (25) (1998) 14863–14868.
[17] R.J. Evans, G.F. Hollis, Genomic structure of the human Ig lambda 1 gene suggests that it may be expressed as an Ig lambda 14.1-like protein or as a canonical B cell Ig lambda light chain: implications for Ig lambda gene evolution, J. Exp. Med. 173 (2) (1991) 305–311.
[18] N. Friedman M. Linial I. Nachman, D. Pe'er, Using Bayesian network to analyze expression data, J. Comput. Biol. 7 (2000) 601–620.
[19] D. Gershon, Microarray technology: an array of opportunities, Nature 416 (2002) 885–891.
[20] T. Hastie, R. Tibshirani, M.B. Eisen, A. Alizadeh, R. Levy, L. Staudt, W.C. Chan, D. Botstein, B. Brown, Gene shaving as a method for identifying distinct sets of genes with similar expression patterns, Genome Biol. 1 (2) (2000), research0003.1–0003.21.
[21] J. Herrero A. Valencia, J. Dopazo, A hierarchical unsupervised growing neural network for clustering gene expression patterns, Bioinformatics 17(2) (2001) 126–136, http://citeseer.nj.nec.com/333262.html.
[22] S.A. Kauffman, The Origins of Order: Self-Organization and Selection in Evolution, Oxford University Press, New York, 1993.
[23] E.C. Keedwell, A. Narayanan, Genetic algorithms for gene expression analysis, Proceedings of the First European Conference on Evolutionary Computation and Bioinformatics, University of Essex, UK, 14–16 April 2003 Springer Verlag, LNCS 2611, 2003, pp. 76–86.
[24] E. Keedwell A. Narayanan, D.A. Savic, Using genetic algorithms to extract rules from trained neural networks, Proceedings of the Genetic and Evolutionary Computing Conference, Vol. 1, 1999, Morgan Kaufmann Publishers, Los Altos, CA, pp. 793.
[25] E. Keedwell A. Narayanan, D.A. Savic, Evolving rules from a neural network trained on continuous data, Proceedings of the Congress on Evolutionary Computation, San Diego, CA, Vol. 1, 2000, pp. 636–639.
[26] E.C. Keedwell, A. Narayanan, D. Savic, Modelling gene regulatory data using artificial neural networks, Proceedings of the International Joint Conference on Neural Networks (IJCNN '02), Honolulu, HI, USA, 2002, pp. 183–189.
[27] J.H. Kenten, H.V. Molgaard, M. Houghton, R.B. Derbyshire, J. Viney, L.O. Bell, H.J. Gould, Cloning and sequence determination of the gene for the human immunoglobulin epsilon chain expressed in a myeloma cell line, Proc. Nat. Acad. Sci. USA 79 (21) (1982) 6661–6665.
[28] J. Khan, J.S. Wei, M. Ringner, L.H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C.R. Antonescu, C. Peterson, P.S. Meltzer, Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks, Nat. Med. 7 (6) (2001) 673–679.
[29] T.J. Kipps, Immunoglobulin genes in chronic lymphocytic leukemia, Blood Cells 19 (3) (1993) 15–25.
[30] S. Liang, S. Fuhrman, R. Somogyi, REVEAL, a general reverse engineering algorithm for inference of genetic network architectures, Pacific Symposium on Biocomputing, Vol. 3, Hawaii, USA, 1998, pp. 18–29.
[31] Y. Maki, D. Tominaga, M. Okamoto, S. Watanabe, Y. Eguchi, Development of a system for the inference of large scale genetic networks, Proceedings of Pacific Symposium on Biocomputing, Vol. 6, Hawaii, USA, 2001, pp. 446–458, available from www-smi.stanford.edu/projects/helix/psb01/.
[32] G.S. Michaels, D.B. Carr, Cluster analysis and data visualisation of large-scale gene expression data, Pacific Symposium on Biocomputing, Vol. 3, Hawaii, USA, 1998, pp. 42–53.
[33] K. Murphy, S. Mian, Modelling gene expression data using dynamic Bayesian networks, Technical Report, Computer Science Division, University of California, Berkeley, CA, available from http://citeseer.nj.nec.com/murphy99modelling.html.
[34] A. Narayanan, E.C. Keedwell, B. Olsson, Artificial intelligence techniques for bioinformatics, Appl. Bioinform. 1 (4) (2002) 191–222.
[35] S.J. Perantonis, V. Virvilis, Dimensionality reduction using a novel neural network based feature extraction method, Neural Process. Lett. 10 (3) (1999) 243–252.
[36] J.R. Quinlan, Data mining tools See5 and C5.0, 2000, http://www.rulequest.com/see5-info.html.

[37] D.W. Ruck, S.K. Rogers, M. Kabrisky, Feature selection using a multiplayer perceptron, Neural Network Comput. 2 (1990) 40–48.

[38] I. Shmulevich, W. Zhang, Binary analysis and optimisation-based normalization of gene expression data, Bioinformatics 18 (4) (2002) 555–565.

[39] A. Silvescu, V. Honavar, Temporal boolean network models of genetic networks and their inference from gene expression time series, Complex Syst. 13 (2001) 54–70.

[40] P. Spirtes, C. Glymour, R. Scheines, S. Kauffmann, V. Aimale, F. Wimberly, Constructing Bayesian network models of gene expression networks from microarray data, in: Proceedings of the Atlantic Symposium on Computational Biology, Genome Information Systems & Technology, Duke University, NC, USA, 2000.

[41] Z. Szallasi, S. Liang, Modeling the normal and neoplastic cell cycle with 'realistic Boolean genetic networks': their application for understanding carcinogenesis and assessing therapeutic strategies. Pacific Symposium on Biocomputing, Vol. 3, Hawaii, USA, 1998, pp. 66–76.

[42] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E.S. Lander, T.R. Golub, Interpreting patterns of gene expression with self-organising maps: methods and applications to hematopoietic differentiation, Proc. Nat. Aca. Sci. USA 96 (1999) 2907–2912.

[43] P. Törönen, M. Kolehmainen, G. Wong, E. Castrén, Analysis of gene expression data using self-organising maps, FEBS Lett. 451 (1999) 142–146.

[44] J. Tsang, Gene expression, DNA arrays, and genetic networks, 1999, available from http://citeseer.nj.nec.com/tsang99gene.html.

[45] E.P. van Someren, L.F.A Wessels, M.J.T. Reinders, E. Backer, Robust genetic network modeling by adding noisy data, in: Proceedings of the 2001 IEEE–EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP01), Baltimore, MD, June 2001.

[46] M.H. Verlhac, R.H. Chen, P. Hanachi, J.W. Hershey, R. Derynck, EMBO J 16 (22) (1997) 6812–6822.

[47] M. Wahde, J. Hertz, Coarse-grained reverse engineering of genetic regulatory network, Biosystems 55 (2000) 129–136.

[48] D.C. Weaver, C.T. Workman, G.D. Stormo, Modelling regulatory networks with weight matrices, Pacific Symposium on Biocomputing, Hawaii, USA, 1999, pp. 112–123.

[49] X. Wen, S. Furhmann, G.S. Michaels, D.B. Carr, S. Smith, J.L. Barker, R. Somogyi, Large-scale temporal gene expression mapping of central nervous system development, Proc. Nat. Acad. Sci. USA 95 (1998) 334–339.

[50] K. Wu, C.-K. Lee, B. Barlogie, M.A.S. Moore, Stromal-derived factor-1 chemokine gene polymorphism: association with stem cell mobilisation and disease characteristics in multiple myeloma patients, 2001, available from http://myeloma.uams.edu/ASH/3073.htm.

[51] Y. Xu, V. Olman, D. Xu, Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees, Bioinformatics 18 (4) (2001) 536–545.

[52] K.Y. Yeung, C. Frayley, A. Murua, Model-based clustering and data transformations for gene expression data, Bioinformatics 17 (10) (2001) 977–987.

Ajit Narayanan is Professor of Artificial Intelligence at the University of Exeter, where he has been located for the last 23 years. His first degree was in Communication Science and Linguistics at the University of Aston (1973) and his second degree was a Ph.D. in Philosophy at the University of Exeter (1976). He took up a lectureship in Computer Science at the University of Exeter soon after the Department was formed (1980). His interests are in artificial intelligence, cognitive science and mind/brain issues. He has written several books and numerous article on AI, and more recently has become Director of Bioinformatics at Exeter. His current research interests focus on the application of AI techniques to bioinformatics problems.

Dr. Edward Keedwell completed his Ph.D. in Computer Science (Bioinformatics) at the University of Exeter in 2003, having previously graduated with degree in Cognitive Science (B.Sc. with Honours) in 1998. Dr. Keedwell's interests are in the application of artificial intelligence and evolutionary computation techniques to optimisation problems in engineering as well as to bioinformatics problems. He has publications in a number of computer science, bioinformatics and engineering journals and proceedings. He is currently a Research Fellow at the University of Exeter.

Jonas Gamalielsson is a Ph.D. student in bioinformatics, working at the University of Skövde, Sweden, where he previously received an M.Sc. in Computer Science with a direction towards artificial intelligence and bioinformatics (2001), and a B.Sc. degree in Computer Science with a direction towards software engineering (2000). His main research interests are data mining algorithms for the analysis of multi-dimensional biological data sets such as microarray gene expression data sets, and the development of algorithms for assessing the interestingness of mined patterns and rules utilising structured prior biological knowledge.

Syam S Tatineni was born and brought up in South India and studied medicine at Manipal Academy of Higher Education in Karnataka, India. He then worked as an intern for a year in Hyderabad, India, before coming to Exeter University for an M.Sc. in Bioinformatics. His M.Sc. research project involved the use of neural networks for analysing gene expression data. After completing his M.Sc., he joined Novartis Pharmaceuticals in Basel, Switzerland, where he is currently working with a team of four bioinformaticians on a comparative genomics project. He is also developing a new genome browser that can be used to view different sections of genomes along with company-derived data.