ELSEVIER

Artificial Intelligence in Medicine 24 (2002) 5–24

Artificial Intelligence in Medicine

www.elsevier.com/locate/artmed

Predicting glaucomatous visual field deterioration through short multivariate time series modelling

Stephen Swift, Xiaohui Liu*

Department of Information Systems and Computing, Brunel University, Uxbridge UB8 3PH, Middlesex, UK

Received 4 August 2000; received in revised form 6 March 2001; accepted 31 March 2001

Abstract

In bio-medical domains there are many applications involving the modelling of multivariate time series (MTS) data. One area that has been largely overlooked so far is the particular type of time series where the dataset consists of a large number of variables but with a small number of observations. In this paper, we describe the development of a novel computational method based on genetic algorithms that bypasses the size restrictions of traditional statistical MTS methods, makes no distribution assumptions, and also locates the order and associated parameters as a whole step. We apply this method to the prediction and modelling of glaucomatous visual field deterioration.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Visual field deterioration; Glaucoma; Genetic algorithms; Multivariate time series; Short term forecasting; Model fitting; Vector auto-regressive process

1. Introduction

Much research has gone into the development of ways of analysing multivariate time series (MTS) data in both the statistical and artificial intelligence communities. Statistical MTS modelling methods include the Vector Auto-Regressive Process [18], the Vector Auto-Regressive Moving Average Process [18], and other non-linear and Bayesian approaches [6], while various Artificial Intelligence (AI) methods have been developed for different purposes including dependence detection in MTS of categorical data [20], knowledge-based temporal abstraction [16,22], and forecasting [5,27]. However, one area that has been largely overlooked is the particular type of time series where the dataset consists of a large number of variables but with a small number of observations. There are inherent difficulties in using traditional statistical techniques to model this type of MTS.

* Corresponding author. Tel.: +44-1895-203397; fax: +44-1895-251686.
E-mail addresses: stephen.swift@brunel.ac.uk (S. Swift), xiaohui.liu@brunel.ac.uk (X. Liu).

PII: S0933-3657(01)00095-1

Normal tension glaucoma visual field data is one such MTS. Glaucoma is the name given to a family of eye conditions [13]. The common trait of these conditions is a functional abnormality in the optic nerve, leading to loss of visual field. Ocular hypertension has been known to be a risk factor in glaucoma, i.e. the higher the eye pressure the higher the risk. However, normal tension glaucoma is unusual in that those suffering from the condition have an intra-ocular eye pressure that is normal, thus, reducing the effectiveness of tonometry [15] as a method of screening for this glaucoma variant. The prediction of visual field deterioration in patients who are suffering from glaucoma plays an important role in the management, treatment, and control of the disease progress. For example, if the deterioration is slowing down, it might be appropriate to reduce the medication; or if the deterioration is speeding up, an increase in medication might be needed or surgery might be necessary.

Visual fields are where the retina is divided into a set of points, and the patient is tested to see how sensitive their eyesight is at each point [9]. This level of sensitivity is usually a number between 0 and 60 dB/log units, which is a measure of retinal sensitivity, 0 being no sensitivity and a value above 50 being very high sensitivity. Fig. 1 shows the *Central Threshold 30-2* test which is usually used for normal tension glaucoma where the black squares mark the blind spot. The left hand diagram shows how each of the 76 test points is numbered, and the right hand diagram shows how the points are grouped according to nerve fibre bundles where there are 16 different groups. The different shading is simply to make each group more distinct. Current theory [8,12] states that deterioration of the visual field can be highly correlated if two points lie on the same nerve fibre bundle. The patients within the dataset, we have access to were tested approximately every 6 months for between 5 and 22 years thereby producing an MTS of length between 10 and 44 (for both eyes). In the previous study [24], the visual field data were clearly demonstrated to be multivariate.

We have been researching into the key issues in the modelling and analysis of the short MTS for prediction purposes. In particular, we have been looking into statistical MTS modelling methods [11] since these have the desirable feature of interpretability in that it is relatively easy to understand the internal constructs of the model. This feature is lacking in many modern methods such as neural networks. However, there are difficulties in using

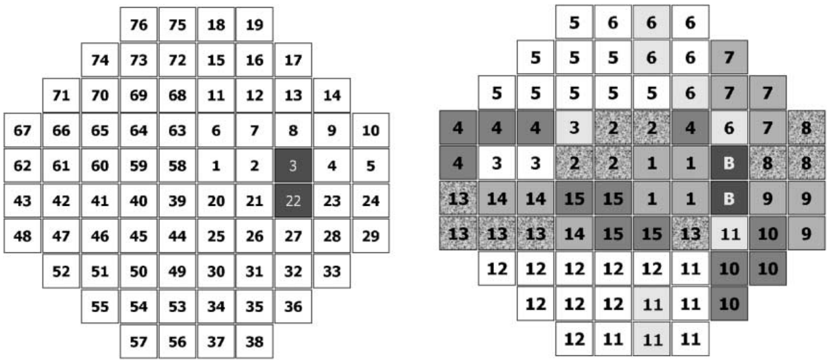

Fig. 1. The normal tension glaucoma data.

traditional statistical methods to model *short* MTS data. In this paper, we present a novel computational method based on genetic algorithms (GA) that can overcome these difficulties, thus, extending the capabilities of statistical modelling methods, and we apply this method to the forecasting and modelling of normal tension glaucoma visual fields.

2. Multivariate time series and the VAR process

MTS data is widely available in different fields including medicine, economics, science, and engineering. An MTS is a series of observations, $x_i(t)$: $(i = 1, \dots, n;\ t = 1, \dots, T)$, made sequentially through time where t indexes the different measurements made at each time point, and i indexes the number of variables in the time series. The vector notation $\underline{x}(t)$ is a shorthand way of referring to the whole set of observations made at time t, i.e. $\underline{x}(t)$ stands for the observations $x_i(t)$ where $1 \leq i \leq n$. A commonly used statistical method for modelling MTS is the Vector Auto-Regressive Process, usually denoted as VAR(p) for a model of order p, as defined in Eq. (1):

$$\underline{x}(t) = \sum_{i=1}^{p} A_i \underline{x}(t-i) + \underline{\varepsilon}(t) \tag{1}$$

where $\underline{x}(t)$ is the next data vector of size n (the number of variables in the model), A_i an $n \times n$ coefficient matrix at time lag i, and $\underline{\varepsilon}(t)$ is an n-dimensional zero mean noise vector at time t (usually Gaussian). The value of each element in A_i is usually a bound real number. The series $x_i(t)$ will be assumed to have zero mean over the entire sample length $t = 1, \dots, T$, i.e. $\sum_{t=1}^{T} x_i(t) = 0$.

The standard statistical methods for fitting a VAR process to a set of data often consist of two steps: order selection (determining a suitable p) and parameter estimation (calculating the matrices A_i from the data). Order selection is commonly performed through the use of information theory-based metrics such as Akaike's Information Criterion (AIC) [2]. Many of these metrics will impose a restriction on the minimum length of a MTS (i.e. setting a lower limit for the series length T), based on the number of degrees of freedom of the model being estimated, namely $T > n(p+1)$.[1] Here, n is the number of variables being modelled, and p is the order of the VAR process. For example, with a MTS involving 10 variables, to find the most appropriate order of a VAR process with a maximum order of five under consideration, T must be at least 52 (the time series length). This restriction is unacceptable for modelling many short time series such as gene expression data produced by DNA array technology or many medical series such as visual field data.

Additionally, the parameter estimation step can experience some difficulties when dealing with a short MTS. The standard methods for parameter estimation include *Maximum Likelihood* (ML) methods, the *Yule–Walker* (YW) equations method, and the *Least Squares* (LS) method [18]. With ML methods, there must be some distribution assumptions on the noise vector. If the MTS values are exclusively within a set interval, an appropriate continuous distribution might be difficult to find, this is the case with dataset

[1] There might be some small variations in the details of implementation, e.g. S-Plus imposes $T > n(p+1)$.

that is the subject of this paper. The YW method can involve matrix inversion, which is computationally expensive with large matrices and can fail if the matrix is singular (especially with shorter time series). The LS method is often used in preference to the YW equations, but it too can involve matrix inversion, and can also impose the degree of freedom restriction mentioned above (this is involved in computing an unbiased estimator for the covariance matrix of the associated noise vector).

Throughout this paper, the notation nVAR(p) will refer to an n variable VAR process of order p where p is an integer greater than zero; we aim to find the best nVAR(p) process for one-step ahead forecasting of the visual field dataset. The *Noise Model* will refer to an nVAR(0) process, i.e. one generated by noise. This model is very easy to construct and simulate, and will be used as a simple benchmark with which to rate other VAR processes against. A VAR process that is designed to model a particular MTS is expected to perform much better than the corresponding *noise model*, since the *noise model* simply assumes that any time series observations were totally randomly generated, with no structure and reliance on previous observations.

An early attempt was made to cope with the challenging problems in the context of predicting glaucoma deterioration [24]. This paper extends that work by presenting an improved version of the algorithm and by testing the method more rigorously. New operators and a seeding strategy are introduced and extensively tested. Additionally, the models are evaluated on *directional accuracy* as well as forecast accuracy.

3. The VARGA method

This section introduces VARGA, a method for finding the order and associated parameter matrices for an nVAR(p) process. VARGA essentially represents a selection of possible nVAR(p) processes to fit the data as a set of p $n \times n$ matrices; a GA-based [14] method is applied to a population of candidate solutions over subsequent generations to improve their suitability to fit the data being modelled. Such a GA can be used to find the parameters and order of a VAR process without making any of the assumptions outlined in Section 2. In addition, VARGA may reduce the length restriction to $T > p$, which makes it possible to model many short MTS. Note that this is the theoretical minimum length an MTS could be, given some constant order p.

The following sections describe the VARGA method, including the representation, fitness function, genetic operators, and algorithms that are essential for any GA implementation.

3.1. Representation

A VAR process is represented with a chromosome consisting of p $n \times n$ matrices. This is shown in Fig. 2. Each matrix element is a bound real number. The possible orders of each VAR process is fixed between one and some upper limit say *MAXORDER*.

This representation differs from that of a conventional GA in that each gene is a matrix, and each matrix has a number of elements. Within Fig. 2, the notation a_{ijk} refers to the element at the jth row and kth column of parameter matrix A_i.

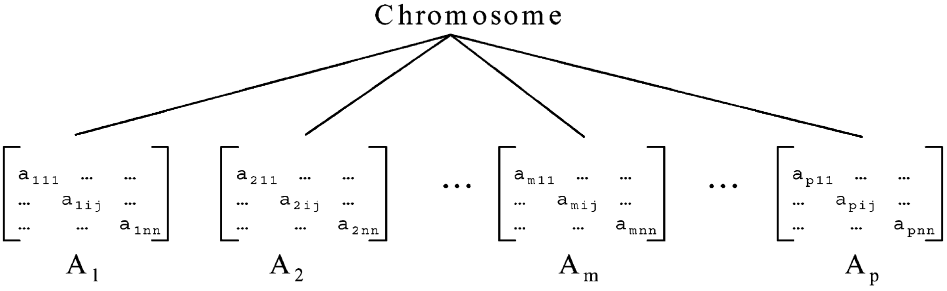

Fig. 2. VARGA representation.

3.2. Fitness

As with a normal GA, VARGA needs a suitable fitness function to evaluate candidate solutions to the problem. This fitness will rate a potential solution against a given dataset using some evaluation criteria. In this paper, we will be searching for a VAR process that is optimised to forecast for a specific dataset. We will concentrate on one step ahead forecasting, although any number of steps could be used. The level of accuracy for VARGA (the fitness function) is defined in Eqs. (2) and (3):

$$\underline{\hat{\varepsilon}}(t) = \underline{x}(t) - \sum_{i=1}^{p} \hat{A}_i \underline{x}(t-i) \tag{2}$$

$$\varepsilon = \sum_{t=T_0}^{T} \sum_{j=1}^{n} |\underline{\hat{\varepsilon}}_j(t)| \tag{3}$$

In Eqs. (2) and (3), $\underline{\hat{\varepsilon}}(t)$ is the estimation of the noise vector, $\hat{A}_i$ the estimation of the ith parameter matrix, ε a scalar that represents the level of noise, and T_0 is defined as MAXORDER + 1. All other variables are defined above. The model with the smallest ε value is deemed the most suitable model for forecasting since it is assumed that the best estimation for any unobserved noise vector is the zero vector. With a GA, it is traditional to maximise the fitness, hence the negative forecast error was used for the fitness function, i.e. $-\varepsilon$.

The value of ε is computed for all possible one step ahead forecasts for a given set of data, starting at the T_0th recorded set of observations. If the order of the VAR process in question was used, then there would be a total of T–p forecasts available for a pth order VAR process. Hence, with a length 20 MTS, a VAR(1) process would be evaluated on 19 forecasts, and a VAR(5) process on 15 forecasts. It can be clearly seen that even if these two VAR processes had equal error at all forecast points, then the VAR(1) process would have 19/15 (1.267) times the forecast error of the VAR(5) process. Hence, the fitness function would be biased for larger order models. It would be a simple step to scale the fitness error by the number of forecasts used to construct it, but this would still mean that smaller order models were evaluated on data that larger order models could not be. By setting the first point that forecast accuracy is evaluated on to be the T_0th observation in the MTS, all order

Table 1
Example fitness function

Data		Forecast
Start	End	
(MAXORDER + 1 − p) 3	(MAXORDER + 0) 5	(MAXORDER + 1) 6
(MAXORDER + 2 − p) 4	(MAXORDER + 1) 6	(MAXORDER + 2) 7
(MAXORDER + 14 − p) 16	(MAXORDER + 13) 18	(MAXORDER + 14) 19
(MAXORDER + 15 − p) 17	(MAXORDER + 14) 19	(MAXORDER + 15) 20

VAR processes are evaluated on the same number of forecasts and the same set of data. Table 1 demonstrates this for a length 20 VAR(3) process with MAXORDER = 5. The forecast difference (the absolute value of the observed minus the predicted) is summed over all of the 15 forecasts and used for the fitness (fitness = negative forecast error).

3.3. The VARGA algorithm

The VARGA algorithm follows a standard GA, but must have enhancements to deal with matrices, variable length chromosomes, and length variation. The function UI(MIN, MAX) is a uniformly distributed random number generator that returns a whole number between the interval (MIN, MAX). The algorithm is as follows (and is shown diagrammatically in Fig. 3).

The initial population of candidate solutions and the genetic operators in steps 4–6 and 8 of the VARGA algorithm are described as follows:

```
1. Create POPULATION chromosomes of size random UI(1, MAXORDER) (the order $p$)
2. Sort the population descending according to fitness (Eq. (3))
3. For $g = 1$ to GENERATIONS do
4.   CROSSOVER the populations genes to list X1, CROSSOVER the populations size
     (order) to list X2
5.   MUTATE X1s and X2s genes
```

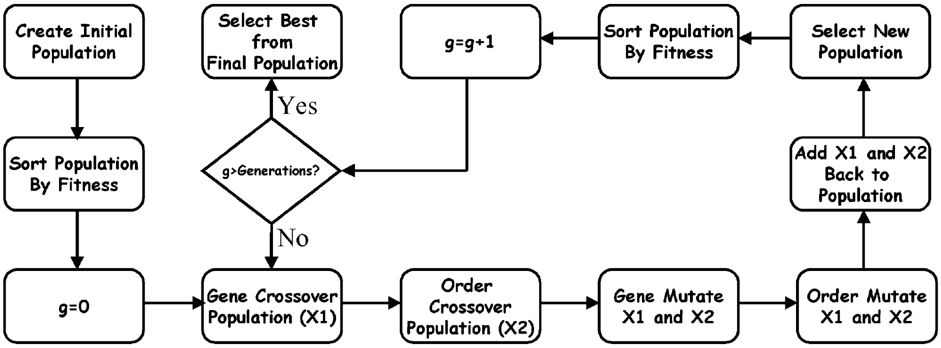
Create Initial Population
Select Best from Final Population
g=g+1
Sort Population By Fitness
Select New Population
Yes
Sort Population By Fitness
g>Generations?
Add X1 and X2 Back to Population
No
g=0
Gene Crossover Population (X1)
Order Crossover Population (X2)
Gene Mutate X1 and X2
Order Mutate X1 and X2

Fig. 3. VARGA algorithm.

```
6.   MUTATE X1s and X2s size (order) and add both lists back to the population
7.   Sort the population in ascending order according to fitness (Eq. (3))
8.   SELECT the new population (size = POPULATION)
9. Next $g$
10. The best VAR process is the chromosome from the final population with the highest
    fitness score (Eq. (3))
```

3.4. The initial population

With a GA, the initial population is usually generated randomly. With VARGA, the initial population was seeded with the *noise model*. Seeding [19,23] is where the initial population for a GA has some or all of its chromosomes set to some predefined representation, as opposed to an entirely random selection. Usually these seeds come from expert knowledge or some other fast approximate search method, e.g. *hill-climbing* [21].

In our implementation, a set of random order (within the limits) individuals whose genes are all zero matrices fill up half of the initial population, and the other half is randomly generated within the specified constraints (see Section 3.8 for the details of all of the VARGA parameters). The justification for this is that experimentation has found that a random VAR(p) process is likely to have a worse fitness when compared with the *noise model* for a given MTS. This is due to the accumulation of forecast errors through there being specified relationships where there should be none. Hence, starting the search from the *noise model* saves VARGA from trying to improve from a very poor starting point. Having the initial population 50% *noise model* and 50% random VAR(p) processes allows some random genes to get mixed in with the zeros of the *noise model*, otherwise the only change from a 100% *noise model* populated initial population would be through *gene mutation*. *Crossover* would simply swap zeros between matrices during the initial generations.

3.5. Crossover

Two crossover methods are used; *Order Crossover*, which acts on whole matrices, and *Gene Crossover*, which acts on the matrix elements. Both methods are designed to be applied to as much of a chromosome as possible, in order to make this stage more efficient.

Order crossover is where matrices from two parents are copied to form children, in a manner similar to uniform crossover [26]. Each member of the current population has a chance of being selected for breeding (*OrderCrossoverRate*). The breeding subset of the population is paired up randomly (*parents*), and two children are produced from each set of parents. Two steps construct these children: firstly, the children are initialised by making a copy of each parent and then secondly for all of the matrices in both children, there is a 50% chance that the two children swap a matrix for a particular order. If the two children are of different sizes, then this is only performed for the matrices that have a corresponding order. The reasoning behind this strategy is that it is assumed that a good parameter matrix for a given order (time lag) may not be any use if moved to a different order (time lag), i.e. it has been optimised for a specific purpose. The following algorithm and Fig. 4 describe this procedure:

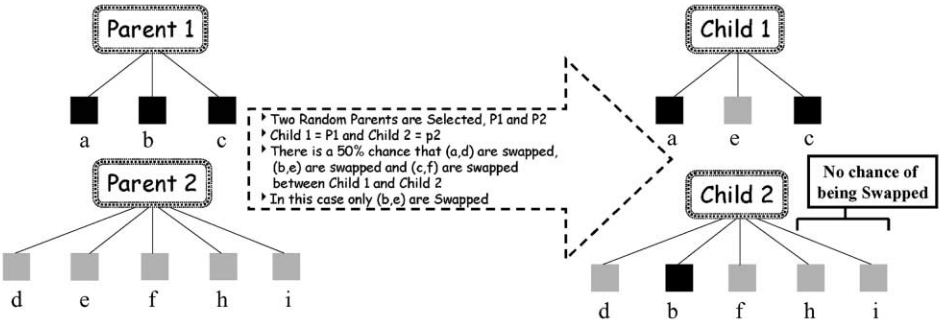
Parent 1
a
b
c
Parent 2
d
e
f
h
i
Two Random Parents are Selected, P1 and P2
Child 1 = P1 and Child 2 = p2
There is a 50% chance that (a,d) are swapped, (b,e) are swapped and (c,f) are swapped between Child 1 and Child 2
In this case only (b,e) are Swapped
Child 1
a
e
c
Child 2
d
b
f
h
i
No chance of being Swapped

Fig. 4. *Order crossover.*

```
1. Given two parents P1 and P2
2. Set MINORDER = Min(Order(P1), Order(P2))
3. Child1 = P1, Child2 = P2
4. For $i = 1$ to MINORDER
5.     If UI(0, 1) = 1 then Swap Matrix $i$ of Child1 and Matrix $i$ of Child2
6. Next $i$
```

where *Min*(x, y) returns the smaller number between x and y and *Order*(z) returns the order of the VAR process represented by chromosome z.

Gene crossover is very similar to *order crossover*. However, matrix elements are swapped, and not whole matrices. If one of the parents is longer than the other then there is a 50% chance that this crossover will be offset by the difference in size. For example, if parent one is of order three and parent two of order five, either elements from matrices 1, 2, and 3 would be uniformly crossed over from both parents, or 1, 2, and 3 of parent one with 3, 4, and 5 of parent two. The procedure is shown in Fig. 5 and described in the algorithm that follows. Within this algorithm *Childx*(i, j, k) is the matrix element a_{ijk} of child number x and the rest of the terms are defined above.

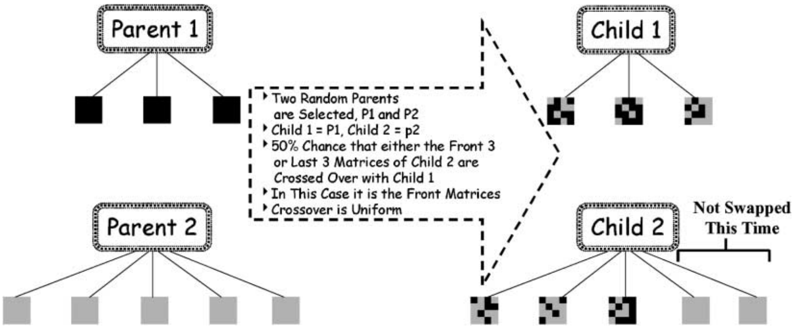
Parent 1
Parent 2
Two Random Parents are Selected, P1 and P2
Child 1 = P1, Child 2 = p2
50% Chance that either the Front 3 or Last 3 Matrices of Child 2 are Crossed Over with Child 1
In This Case it is the Front Matrices
Crossover is Uniform
Child 1
Child 2
Not Swapped This Time

Fig. 5. *Gene crossover.*

```
Given two parents P1 and P2
If Order(P1) > Order(P2) then
  Child2 = P1, Child1 = P2
Else
  Child2 = P2, Child2 = P1
End if
MINORDER = Order(Child1)
Offset = (Order(Child2) − MINORDER) × UI(0, 1)
For $i = 1$ to MINORDER
  For $j = 1$ to $n$
    For $k = 1$ to $n$
      If UI(0, 1) = 1 then
        Temp = Child1($i, j, k$)
        Child1($i, j, k$) = Child2($i$ + Offset, $j, k$)
        Child2($i$ + Offset, $j, k$) = Temp
      End if
    Next $k$
  Next $j$
Next $i$
```

Again each parent has a chance of producing offspring in this way where the chance of a member of the population being chosen for *gene crossover* is *GeneCrossoverRate*.

3.6. Mutation

Two mutation operators are presented. The first mutates the genes, and the second mutates whole matrices, either deleting or adding one.

Gene Mutation: When an individual is deemed to *gene mutate* (steps 6 and 7 in the VARGA algorithm), each element of each matrix (denoted a_{ijk} as in Section 3.1) is given a chance to mutate (*GeneMutationRate*). If a matrix element mutates, then it is changed according to a random Gaussian distribution. Eq. (4) defines this more formally. In this equation, a check is made for each possible matrix element that UR(0.0, 1.0) is less than the *gene mutation* rate parameter (*GeneMutationRate*). The two conditions check that the mutation has not allowed an element to deviate from the limits.

$$a_{ijk} = N(a_{ijk}, \sigma), \quad \forall i \in [1, \mathit{MAXORDER}]j,$$
$$k \in [1, n], \ \ \mathit{where}\ \mathit{UR}(0.0, 1.0) < \mathit{GeneMutationRate} \qquad (4)$$

If $a_{ijk} < \mathit{MINGENE}$ *then* $a_{ijk} = \mathit{MINGENE}$ and *If* $a_{ijk} > \mathit{MAXGENE}$ *then* $a_{ijk} = \mathit{MAXGENE}$

The function UR(MIN, MAX) is a uniformly distributed random number generator that returns a *real* number between the interval (MIN, MAX) (inclusive). MINGENE and MAXGENE are application dependent, and are the minimum and maximum values a gene can take; the sum of these two parameters is usually zero. The standard deviation σ should be chosen so that it is sufficiently small enough so not to cause a mutation to always exceed these two boundaries.

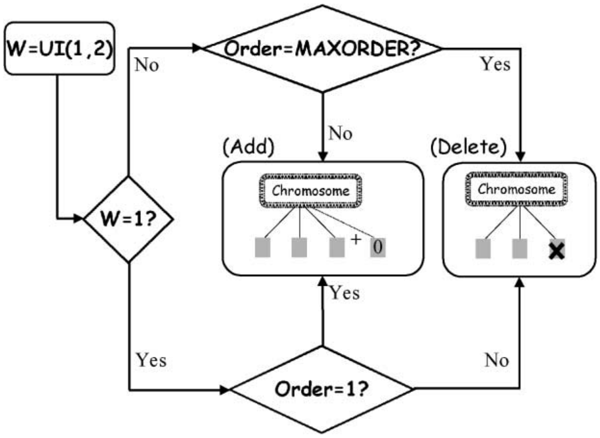

Fig. 6. *Order mutation.*

Order Mutation is responsible for altering the size of a chromosome. Each chromosome (at the appropriate stage in the algorithm (steps 8 and 9)) has a chance of undergoing *order mutation* (*OrderMutationRate*). If it does, then there is a 50% chance that either it increases or decreases in size.

Additions and deletions are always performed on the highest order matrix. Any new matrix that is added consists of zeros, thus, ensuring that the fitness of an individual does not change when its order increases. The order of an individual is not allowed to exceed the specified limits. This procedure is shown in Fig. 6.

3.7. Selection

The selection operator is the *ranked survival* described in [4]. This is the same as the *Roulette Wheel* technique, but the chromosomes rank is used, rather than its fitness. This is because the problem is a minimisation problem; hence the *Roulette Wheel* [14] would favour the worst in a population rather than the best. The chance of surviving is equal to a chromosomes fitness divided by the sum of the populations fitness. Thus, a large forecast error would give a high probability of survival and a low forecast error would give a small chance of surviving—which is exactly the opposite of what is needed, therefore, we base survival on rank. Equal ranking is not dealt with, as it will be very unlikely since the fitness is real. *Elitism* [19] is also used, which is the process of selecting a predefined number of the best chromosomes for survival, before the rest of the next population is selected.

3.8. VARGA parameters

Table 2 details the parameters for the implementation of VARGA presented within this paper. These parameters were found to be the best set through many experiments. For example, the minimum and maximum gene values were chosen from experience with the *Yule–Walker* equations: analysis of many resultant parameter matrices using the visual field data showed that nearly all of the elements fell within the range detailed in Table 2;

Table 2
VARGA parameters

Parameter	Meaning	Value
POPULATION	The size of the population	100
GENERATIONS	The number of generations the algorithm is run for	1000
ELITISM	The number of the fittest individuals guaranteed survival	1
OrderCrossoverRate	The chance of a chromosome being chosen for *Order Crossover*	0.500
GeneCrossoverRate	The chance of a chromosome being chosen for *Gene Crossover*	1.000
OrderMutationRate	The chance of a chromosome having its order mutate	0.050
GeneMutationRate	The chance of a chromosomes gene mutating	0.005
MINGENE	The minimum value a gene can take	−1.250
MAXGENE	The maximum value a gene can take	1.250
MAXORDER	The maximum possible order of a VAR process	5
σ	The standard deviation for Eq. (4)	0.700

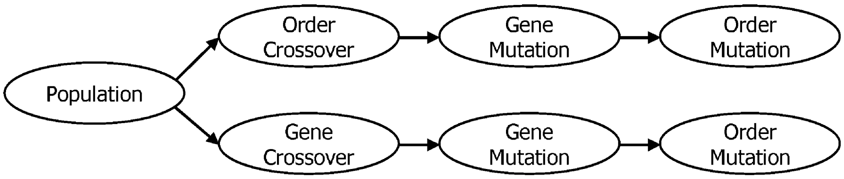

Fig. 7. VARGA population growth through operators.

experience with GAs has shown a large population is needed for crossover operators to be effective and the standard deviation for the *gene mutation* operator was selected so that a mutation is likely to cause only a small perturbation around the genes value.

3.9. Population dynamics

The population grows through the application of genetic operators as in Fig. 7. In this figure, the node at the top of the tree (*Population*) represents the starting size of the population. Two sets of children are produced through the application of *order crossover* and *gene crossover*. The mutation operators are then applied, *order* and then *gene*, to these children, where order mutation is applied after *gene mutation*. The population will increase by a proportion equal to *OrderCrossoverRate* + *GeneCrossoverRate*. The two mutation rates will not affect the increase in the population, since they are only applied to the children. Note that the *selection* operator reduces the population back to the size it was before any of the operators were applied.

4. Evaluation

The proposed algorithm is evaluated against the *Yule–Walker* equations as implemented by S-Plus (version 2000) and the *noise model*. The evaluation will consist of three criteria. The first is the one step ahead forecast accuracy of the methods, where the fitness function described in Section 3.2 is used to evaluate all of the methods. The second evaluates the methods based on the *Weighed–Kappa* [3] metric. This metric indicates a level of agreement between observed and predicted values within an experiment, where the level of agreement is in terms of a positive change, negative change or no change in deterioration. The last evaluation is on the operators that make up VARGA itself. The intention is to demonstrate the effectiveness of these operators. The earlier work on VARGA presented in [24] dealt solely with nerve fibre bundle number *5* for 28 patients. In this paper, another nerve fibre bundle (number *12*—see Fig. 1) has been chosen to evaluate the methods. The reason for this choice is to show that our methods are not just suited to a specific MTS, and that nerve fibre bundle *12* is high dimensional and in a position that is often affected with glaucoma first.

4.1. Forecast accuracy

S-Plus has an easy-to-use function [1] for finding the best-fit VAR(p) process for a given dataset, based on the solution of what is called the *Yule–Walker Equations*. Each patient's visual field results give a model that is rated according to Eq. (3). Since S-Plus uses "*Whittles Recursion*" [28], a limit on the minimum length T of a time series with n variables is constrained by Eq. (5):

$$T > n(p + 1) \tag{5}$$

The records of 280 patients have been made available by *Moorfields Eye Hospital* in London (UK), however, for S-Plus to implement a VAR(0) process (the *noise model*), the number of tests per patient must be at least 10. This follows from Eq. (5), where $p = 0$ (the order) and $n = 9$ (the number of variables). This length restriction cuts the number of usable patient records down to 82. These 82 patients were reduced in numbers again, this time for another set of reasons. As in [24], S-Plus rejected (could not provide a result for) many of the time series due to matrix inversion problems and numerical instability. As a result, only 34 patients' visual field records were usable.

VARGA, the *noise model* and the *Yule–Walker* equations were then run on each of the 34 patients. With the VARGA results, these were performed 10 times, and then the average is taken (GAs are stochastic and so the intention is to show the general performance over several runs of VARGA hence removing the chance of a single experiment producing a fluke result). The forecast accuracy of the *Yule–Walker* equations and the *noise model* was evaluated in the same way as with VARGA, i.e. using the negative forecast error as detailed in Section 3.2. Patients are identified by a number between 0 and 81, even though only 34 patients are used in the experiments. The experimental results are split into two groups, which are described here. Average forecast error (within Figs. 8 and 10) is the value obtained by dividing the forecast error by the number of forecast points. This is defined in Eq. (6).

$$\text{Average forecast error} = \frac{\varepsilon}{n(T - \text{MAXORDER})} \tag{6}$$

where ε is the total forecast errors defined in Eq. (3), n the number of variables in the MTS, i.e. 9, and T is the length of an MTS for a patient. Note that the lower the average forecast error, the better and that the errors are for a one step ahead forecast.

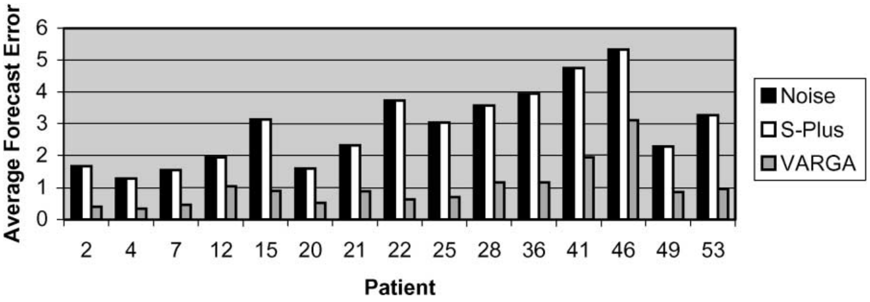

Fig. 8. Forecast error—S-Plus and noise equal.

Cases where the S-Plus order is zero: Fig. 8 shows the results where the order of S-Plus and the *noise model* are equal, i.e. where S-Plus thought that the most likely model to fit the data was a zero order VAR process. The total number of patients for which this is the case is 15 out of the 34. Here VARGA clearly does much better than the *noise model* in all cases, ranging from 170 to 600% better, with an average improvement of 327%. For all of these 15 patients, if we apply Eq. (5), we find that the maximum order can be 1. However, S-Plus decided that a VAR(0) process was suitable, even though it had the option of choosing a VAR(1) process. This inappropriate choice of order is down to the AIC metric (used for order selection by S-Plus) failing to work adequately on a short MTS.

Fig. 9 shows the order VARGA selected for the same patients. Note that the order is fractional in some cases as it has been averaged over the 10 runs. Here the order is between three and five, with the average being about four.

Cases where the S-Plus order is greater than zero: Fig. 10 summarises the results for those patients where S-Plus produced models with an order greater than zero. Here VARGA does better than S-Plus in 18 out of 19 cases, improving on forecasting accuracy between 200 and 450%. In the single case that S-Plus is better than VARGA (patient number 29), S-Plus produces an average forecast error about 70% of the VARGA method.

The order results are summarised in Table 3. The order found by VARGA seems to be on average between 3.5 and 5, suggesting that an order of four would be most suitable for the data. S-Plus, however, chooses lower orders because of size restrictions and problems applying the AIC metric on short time series.

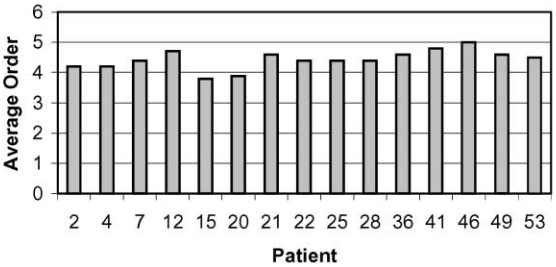

Fig. 9. VARGA order—S-Plus and noise equal.

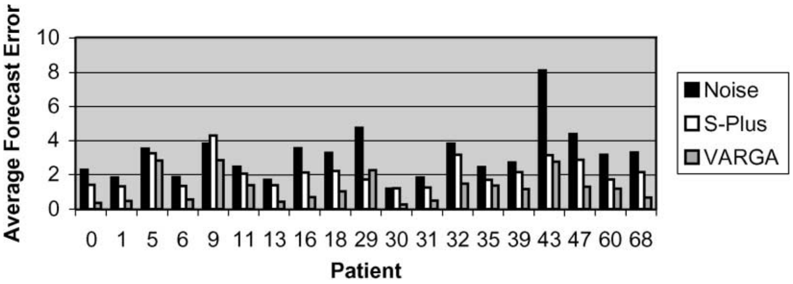

Fig. 10. Forecast error—S-Plus and noise not equal.

Table 3
Order for cases where S-Plus order $\neq 0$

Method	Number of order
Noise	19 of 0
S-Plus	16 of 1, 2 of 2, 1 of 3
VARGA	1 of 3.8, 1 of 4, 1 of 4.2, 2 of 4.3, 2 of 4.4, 3 of 4.5, 2 of 4.6, 3 of 4.8, 4 of 5

All cases together: Table 4 summarises the forecast accuracy for all three methods on all of the 34 patients; this shows that VARGA performs much better than the other two methods. The average sensitivity at each point in the visual field over all of the test results from the 34 patients is 16.907. In Table 4, the average forecast error per point (*By Point Value*) is given for the three the methods. Clearly VARGA is several times better than the other two methods. *By Point Percentage* distributes the error by the average sensitivity and then lists the results as a percentage.

4.2. Weighed–Kappa

The *Weighed–Kappa* metric is used to rate agreement between the classification decisions made by two or more observers. In this case the two observers are the recorded visual field test results and the predicted results from a VAR process. The classification from each observer for each point $x_i(t)$ is divided into three states as in Table 5. Within this table, *direction of change* refers to the shorthand description for whether the change is worsening the condition ($-$), not altering at all (0) or improving ($+$). This classification is based on the change between a point measured at two successive tests, i.e. the direction of change.

Table 4
Average forecast error by point

Measure	Noise	S-Plus	VARGA
By Point Value (BPV)	3.048	2.475	1.144
By Point Percentage (BPV/16.907) × 100 (%)	18.028	14.639	6.766

Table 5
Visual field change classification

Direction of change	Definition
Deterioration (−)	$x_i(t) - x_i(t-1) < -\Omega$
No change (0)	$\lvert x_i(t) - x_i(t-1)\rvert \leq \Omega$
Improvement (+)	$x_i(t) - x_i(t-1) > -\Omega$

The constant Ω gives a margin of error for defining equality, since the visual fields are treated as real numbers (they are integers ranging between 0 and 60). This makes it very unlikly that any pair of observed and predicted points will be equal. In this paper we have choosen $\Omega = 0.5$. A 3 × 3 table of counts is constructed for each forecast where each table element is referred to as $Count_{ij}$. Rows are indexed according to the predicted category and columns by the observed category. The same number of one step ahead forecasts are performed as with the VARGA fitness function, but the differences between forecasts are considered, i.e. the direction of change. This results in a total of $n(T - \text{MAXORDER} - 1) = \sum_{i=0}^{2}\sum_{j=0}^{2}\text{Count}_{ij}$ possible classifications.

Once this table has been completed, the *Weighed–Kappa* value can be computed using the procedure detailed in [3]. This metric was evaluated only for the *Yule–Walker* equations and VARGA since the *noise model* will simply classify any prediction as remaining the same; its forecast is always zero. Hence the *noise model* will have a very low *Weighed–Kappa* and thus not much use in comparison. Table 6 is the suggested interpretation of *Kappa* values to indicate the strength of agreement between two observers [3], which we use as an approximate interpretation for *Weighed–Kappa*. Although the *Weighed–Kappa* values are normally higher than those of *Kappa*, here we are interested in the assessment of the agreement strength between two methods according to their respective categories.

4.2.1. Weighed–Kappa results

Tables 7 and 8 show the results of the evaluation of the *Weighed–Kappa* metric run on the 34 patients for S-Plus and VARGA. Table 7 shows how the 34 sets of results per method can be divided into strength categories according to Table 6. Table 8 shows summary statistics over all of the runs. S-Plus is listed twice, once for when the selected order was zero, and then again for results where the corresponding order was not zero (i.e. not the *noise model*).

Table 7 shows clearly that the majority of the VARGA results strongly agree with the deterioration trends within the data, a total of 26 out of 34, i.e. about 76%.

Table 6
The Kappa guideline

Kappa (κ)	Agreement strength
$0 \leq \kappa \leq 0.2$	Poor
$0.2 < \kappa \leq 0.4$	Fair
$0.4 < \kappa \leq 0.6$	Moderate
$0.6 < \kappa \leq 0.8$	Good
$0.8 \leq \kappa \leq 1.0$	Very Good

Table 7
Weighed–Kappa strength count results by category

Agreement strength	Weighed–Kappa count	
	S-Plus	VARGA
Poor	19 (4)	0
Fair	6 (6)	1
Moderate	8 (8)	7
Good	1 (1)	18
Very Good	0	8

With the S-Plus, the values in parenthesis are those results with an order greater than zero. Hence, it can be clearly seen that there is very poor agreement when S-Plus chooses the *noise model*. However, even when comparing *non-noise model* results, VARGA performs better.

Table 8 shows clearly that the VARGA one step ahead forecasts on average give a good level of agreement with the observed visual fields. The S-Plus results (order $= 0$) are very poor, suggesting very little agreement with the original data's trends. However, the S-Plus results (excluding the results where the order is zero) are much better, again showing that the *noise model* is the poorer choice given a selection of orders. On the other hand, the results for VARGA have a higher average for the *Weighed–Kappa* metric (almost twice as good as the average results for S-Plus), and have a lower standard deviation, indicating a more stable set of results. VARGA would, therefore, be the better choice based on this metric because its forecasts agree strongly with the trends in the original data.

4.3. Operator experiments

The final experiment was to see how effective the various operators are. There are four operators defined in VARGA, and it would be useful to see if some of them can be replaced by combinations of the others. An extensive experiment was designed, executed, and analysed towards this end. VARGA was executed 10 times with each different combination of operators on a number of patients. With the four operators *Gene Mutation*, *Order*

Table 8
Weighed–Kappa strength count results by value

Summary statistic	Weighed–Kappa count		
	S-Plus (order $= 0$)	S-Plus (order > 0)	VARGA[a]
Mean	0.201	0.359	0.684
Standard deviation	0.222	0.173	0.153
Median	0.091	0.380	0.741
Minimum	0.000	0.026	0.219
Maximum	0.616	0.616	0.857

[a] The VARGA results were averaged over the 10 runs, and then summary statistics taken from these patient averages.

Mutation, *Gene Crossover*, and *Order Crossover*, there are 16 possible combinations. However, the case where all of the operators are zero is ignored, because VARGA would simply just run selection on each population, and not add or change any individuals, thus, not increasing the fitness count. An experiment *Run* is identified by a binary digit indicating whether an operator rate has a value as defined in Table 2 (if the digit is one) or is zero. For example, 0101 (or *Run* 5) has operator rates *GeneMutationRate* = 0%, *OrderMutationRate* = 5%, *GeneCrossoverRate* = 0% and *OrderCrossoverRate* = 50%. The experiments were performed on a total of eight patients for a total number of 100,000 fitness function calls. Setting a limit on the fitness function calls is a fairer way of ensuring all of the experiments are fairly matched since each combination of operators result in a different number of offspring being generated. Fixing the number of generations would simply bias the experiments towards the ones that use the most number of operators. Only eight patients (~10% of the total number available) were chosen since each experiment is repeated 10 times; hence it would take a very long time to complete if all the patients had been selected. Therefore, there are 15 sets of operator experiments for 8 patients by 10 repeats per patient which gives $15 \times 8 \times 10 = 1200$ experiments.

The results for these experiments have been summarised in Table 9. The generations average has been listed as an indication of their typical values under the different operator combinations. The algorithm is terminated when the number of fitness function evaluations exceeds 100,000 calls. *Gen.* refers to the number of generations that correspond to 100,000 function evaluations; the figures displayed are averages over the 1200 experiments.

When looking at the results of these experiments, it is immediately clear that the inclusion of *Gene mutation* improves the fitness from approximately −350.000 ± 164.000 to −180.000 ± 175.000, changes the *Order* from 3.0 ± 1.4 to 4.0 ± 0.7, and the *Weighed–Kappa* from 0.006 ± 0.020 to 0.550 ± 0.240. For example, when this operator is not

Table 9
Operator experiment results

Operators			Fitness		Order		Weighed–Kappa	
Run		*Gen.*[a]	Mean	S.D.	Mean	S.D.	Mean	S.D.
0001	(1)	1979	−350.236	164.591	3.1	1.343	0.004	0.010
0010	(2)	999	−348.178	163.949	3.1	1.478	0.004	0.020
0011	(3)	665	−348.112	163.570	3.1	1.322	0.011	0.045
0100	(4)	19982	−350.236	164.591	2.8	1.303	0.004	0.010
0101	(5)	1978	−350.236	164.591	3.2	1.420	0.004	0.010
0110	(6)	999	−346.198	162.427	2.9	1.389	0.008	0.021
0111	(7)	664	−348.304	163.638	3.2	1.406	0.006	0.016
1000	(8)	1322	−194.298	181.108	4.0	0.981	0.517	0.221
1001	(9)	1979	−183.067	179.185	3.6	0.919	0.565	0.241
1010	(10)	999	−181.923	184.869	3.2	0.906	0.560	0.246
1011	(11)	664	−173.251	179.076	3.7	0.941	0.589	0.259
1100	(12)	1066	−193.859	179.828	5.0	0.112	0.510	0.219
1101	(13)	1978	−169.234	174.577	5.0	0.191	0.595	0.242
1110	(14)	999	−169.746	174.301	4.0	0.787	0.609	0.245
1111	(15)	664	−166.865	174.621	4.5	0.693	0.613	0.244

[a] Refers to the number of generations that correspond to 100,000 function evaluations.

included in these experiments (*Runs* 1–7), the fitness is very similar (in fact almost equal to the average of the corresponding *noise models*). This is because the initial populations were set to a selection of random order VAR processes where each matrix element was either zero (the *noise model*) or a random value within the gene element limits (50% chance of one or the other). This demonstrates that a random individual is nearly always worse than the *noise model*, since the final best individual is the *noise model* or very similar to it. The fitness is unlikely to increase beyond this point because selection ensures a zero element population before the operators (that are applied) can make any useful changes. Once this *zero-element-individual* state is achieved (or very close to it), a change is impossible in most cases. *Order mutation* may remove a matrix, which in these cases would not change fitness. If it adds a matrix then it is a zeroed matrix: again resulting in no fitness change. Both *crossover* operators merely arrange genes around, and do not alter their values.

It is also interesting that the standard deviation of the order parameter in runs 12 and 13 drops dramatically from 0.9 (runs 8–11) and 0.7 (runs 14 and 15) to 0.112 and 0.191, respectively. This could be because of the results of combining the *Order mutation* and *Gene crossover* operators; the latter selects a consistent order, and the former a good fitness, hence the combination tries to select both a consistent order and good fitness.

It can be clearly seen that all of the operators acting together are more effective than any other combination. It is worth noting that the best case (run 15) and the worst (non-*noise model*) case (run 8) differ by about 16.4%, the *Weighed–Kappa* values differ by about 18.4%. These are the two cases where either all of the operators are used, or just the gene mutation operator is used. The order column demonstrates a level of agreement between those experiments that have a low forecast error, being between 3 and 5. The standard deviation for the order reduces very significantly, by about 29.4%, between experiments 8 and 15, which indicates that a more consistent order selection can be achieved when more of the genetic operators are used together. Finally, it would be fair to conclude from the observations about the runs 1–7 and from the runs 8 to 15 that the *gene mutation* operator is the most powerful operator, but it will work even better when combined with the other operators.

4.4. Discussion

The following discussion summarises the observations and conclusions draw from the experiments in this paper.

For short MTS, conventional methods for order determination can be unreliable, e.g. the AIC metric. Parameter estimation methods can also fall down, i.e. matrix inversion errors and numerical instability.

Although the performance of the *Yule–Walker* equations is better when the time series is long; VARGA still gives a better set of results for 33 cases out of 34. Further, for those applications involving a short MTS such as the glaucoma visual field data, the *Yule–Walker* equations (in S-Plus) either cannot model them to a sufficiently high degree of accuracy or cannot model them at all.

VARGAs superior performance might be explained in part by the fact that approximately 60% of the resulting parameter matrices are zero. VARGA assumes that any relationships between variables are zero until proved otherwise, i.e. when a zero changes through mutation and the corresponding fitness improves. The standard statistical methods assume that there is some degree of relationship between all variables at all time lags, which appears to be counter-intuitive.

VARGA could have performed so well on the visual field data because of the association between the results of the tests. If a patient has a certain level of visual field deterioration, then the results of the next test would be expected to be similar if not worse, hence the suitability of modelling the data as an MTS.

5. Concluding remarks

Model selection is arguably the most important and most difficult aspect of model building, and yet is the one where there is least help [7,10]. This situation is even worse for modelling short MTS data. In this paper, we have presented a method for learning a Vector Auto-Regressive Process from such a MTS. This is achieved through a real valued matrix-based representation and appropriate crossover and mutation operators for a GA. The results clearly show that the VARGA model provides a better method for fitting a VAR process to a short MTS than the conventional statistical methods.

There are many real-world applications involving the modelling of short MTS, especially in bio-medical domains. Our proposed method has been applied to the effective prediction of glaucomatous visual field deterioration and we have obtained a good understanding of the genetic operators used in the method. Future work will concentrate on the development of more efficient algorithms and their application to other bio-medical problems. One such modelling approach that is currently being studied is the decomposition of high-dimensional short length MTS into a series of smaller dimensional MTS, where there are strong relationships between members of the same sub-group but weak relationships between members of different sub-groups. We have had some promising results with visual field data, simulated data, and chemical process data [25]; the intention is to apply VARGA to these decomposed groups and then evaluate the accuracy of the resultant forecasts.

With the advent of DNA microarray and chip technologies, gene expression can be explored on the "genome scale" [17]. Much of the research in gene expression data analysis has been on the application of clustering algorithms, visualisation methods, and Boolean networks. However, many of the gene expression data collected are essentially short- and high-dimensional MTS. We will apply the VARGA methodology with the aforementioned variable grouping technique to the analysis of gene expression data, especially in the virology domain.

Acknowledgements

We thank our research partners at Moorfields Eye Hospital and the Institute of Ophthalmology for their advice and the MTS visual field data. We are grateful to Allan Tucker for his advice and assistance and would also like to thank the editor, Professor Klaus-Peter Adlassnig, and the three anonymous reviewers for their helpful and constructive comments. This research is funded by Moorfields Eye Hospital, London and the Engineering and Physical Sciences Research Council, UK (GR/M94120).

References

[1] S-Plus 4. Guide to statistics. MathSoft, Seattle, Washington, CA, 1997.

[2] Akaike H. A new look at the statistical model identification. IEEE Trans Automat Control AC-19 1975;6:716–23.

[3] Altman DG. Practical statistics for medical research. London: Chapman & Hall, 1997.

[4] Baker JE. Adaptive selection methods for genetic algorithms. In: Grenfenstett JJ, editor. Proceedings of the First International Conference on Genetic Algorithms. Lawrence Erlbaum Associates, 1985. pp. 101–11.

[5] Baldi P, Brunak S, Frasconi P, Pollastri G, Soda G. Bidirectional dynamics for protein secondary structure prediction. In: Proceedings of the IJCAI-99 Workshop on Neural, Symbolic and Reinforcement Methods for Sequence Learning. 1999. pp. 77–83.

[6] Casdagli M, Eubank S. Nonlinear modeling and forecasting. Reading (MA): Addison-Wesley, 1992.

[7] Chatfield C. Model uncertainty, data mining and statistical inference (with discussion). J R Stat Soc Ser A 1995;158:419–66.

[8] Crabb D, Fitzke F, McNaught A, Hitchings R. A profile of the spatial dependence of pointwise sensitivity across the glaucomatous visual field. Perimetry Update (1996/1997), pp. 301–10.

[9] Haley MJ, editor. The field analyzer primer. San Leandro: Allergan Humphrey, 1987.

[10] Hand DJ. Deconstructing statistical questions (with discussion). J R Stat Soc Ser A 1994;157:317–56.

[11] Hannan EJ. Multiple time series. New York: Wiley, 1970.

[12] Heijl A, Lindgren A, Lindgren G. Inter-point correlations of deviations of threshold values in normal and glaucomatous visual fields. Perimetry Update (1988/1989), pp. 177–83.

[13] Hitchings RA, editor. Glaucoma. London: BMJ Publishing Group, 2000.

[14] Holland JH. Adaptation in natural and artificial systems. Ann Arbor (MI): The University of Michigan Press, 1975.

[15] Hollwich F. Ophthalmology: a short textbook, 2nd edition. Stuttgart: Thieme, 1985.

[16] Lavrac N, Keravnou E, Zupan B, editors. Intelligent data analysis in medicine and pharmacology. Dordrecht, Hingham (MA): Kluwer Academic Publishers, 1997.

[17] Lockhart B, Winzeler E. Genomics, gene expression and DNA arrays. Nature 2000;405:827–36.

[18] Lütkepohl H. Introduction to multiple time series analysis, 2nd edition. Berlin: Springer, 1993.

[19] Michalewicz Z. Genetic algorithms + data structures = evolution programs, 3rd edition. Berlin: Springer, 996.

[20] Oates T, Schmill M, Cohen P. Efficient mining of statistical dependencies. In: Dean T, editor. Proceedings of 16th IJCAI. Morgan Kaufmann; 1999. pp. 794–9.

[21] Russell S, Norvig P. Artificial intelligence: a modern approach. London: Prentice Hall, 1995. pp. 111–2.

[22] Shahar Y, Musen MA. Knowledge-based temporal abstraction in clinical domains. Artif Intell Med 1996;8:267–98.

[23] Sharif AM, Barrett AN. Seeding a genetic population for mesh optimisation and evaluation. In: Koza J, editor. Proceedings of the Genetic Programming 1998 Conference: Late Breaking Papers, 1998. pp. 195–200.

[24] Swift S, Liu X. Modelling and forecasting of glaucomatous visual fields using genetic algorithms. In: Gecco99: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann, 1999. pp. 1731–7.

[25] Swift S, Tucker A, Martin N, Liu X. Grouping multivariate time series variables: applications to chemical process and visual field data. Knowledge-Based Syst. 2001;14:147–54.

[26] Syswerda G. Uniform crossover in genetic algorithms. In: Schaffer JD, editor. Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, 1989. pp. 10–9.

[27] Weigend AS, Garshenfeld NA. Time series prediction. Reading (MA): Addison-Wesley, 1994.

[28] Whittle P. Prediction and regulation, 2nd edition. New York: Blackwell, 1984.