Rapid Data Classification via Kohonen Self-Organising Maps

A thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

by

David Roger Brett

Theoretical Astrophysics Group
Department of Physics and Astronomy
University of Leicester

March 2005
For Anna Francesca Vives Riera
Acknowledgements

Firstly I would like to acknowledge PPARC for the studentship which has financially enabled me to do the work for this thesis. Both of my supervisors, Pete Wheatley and Richard West, have been a great deal of help throughout my time at Leicester...even if we didn't always agree on the best way to do things (I think I was right about 10% of the time, which I reckon is pretty good!).

Thanks to all those who have shared an office with me over the last few years (Owen, Yohann, Sam, Rosie, Paul and and to the rest of the Theory Group) and have had to listen to me complaining about the monitor/window/printer/radio/data1/potato clock. I hope the odd occasions when I've been able to help you with your C coding troubles has made up for this. I am especially grateful for the suggestions provided by Martin Beer with respect to chapter 5 and to Dan Rolfe for software provided for the same chapter.

I should say at this point that I probably would not have completed my first year, never mind the three (and a bit) it has taken me, without Dan, Owen and Antje. For those who know Dan you may find this next statement surprising. Dan kept me sane in my first few months in Leicester; without his kindly provided cups of tea and plates of chips (which he almost burned the kitchen down to cook...the chips, not the tea) I would have gone stark-raving bonkers in that student accommodation. Owen and Antje, thanks for the nights observing at the telescope, the times you invited me over for whatever that word game was called and listening to me waffle on about whatever was bothering me at the time, it was much appreciated. Oh, and Owen, I will get around to writing that code
cracker at some point!

Ah, the mid-point of the acknowledgements, the point at which the slightly more tongue-in-cheek acknowledgements can be slotted. Bongo, Jiggles, Flenser, Posh, Napi, Tinkerbell, Bazza, Le Biggles, Jem Jem, Slixgurnt, Rich’s mate whose name escapes me but who was “a bit good”, Buster Gut, Warlord and a host of others who have joined me online to let a few virtual missiles fly. The games were good fun, I hope we’ll find time to play them again in the future sometime (don’t worry Dan, we’ll provide the motion-sickness pills for you).

What acknowledgements would be complete without reference to my Mum and Dad? Thanks for listening to me when I was down, and putting up with me when I yammered on about things that made no sense to you!

Lastly (and by absolutely no means least!) I would like to thank my Anna. Without your encouragement and gentle patience I think that I would still be struggling my way through ¡los capitulos del diablo! Et vull.
Declaration

I hereby declare that this thesis has not been submitted in full or in part for any other degree at this or any other university. I confirm that in those sections which were written in collaboration, the majority of the work was performed by me.

The following paper has resulted from the work presented here:

Chapter 4
The Automated Classification of Astronomical Lightcurves using Kohonen Self-Organising Maps

David Brett
March 2005
Abstract

Rapid Data Classification via Kohonen Self-Organising Maps

I here present my version of the Kohonen Self-Organising Map (KSOM) as applied to the classification, or rather clustering, of astronomical data. The main body of this work is concerned with the grouping of period-folded stellar lightcurves and clustering based on the lightcurve shape alone. It has been found that the algorithm is an extremely stable grouping mechanism for data of low (3σ signal to noise) to good quality. With further analysis of the results it is possible to locate underpopulated samples of data that exist within the data. This can be successfully achieved for samples of 1%, or less, total population. Additionally the same algorithm has been applied to the extraction of planetary transit lightcurves from those of eclipsing binaries (chapter 5), and to the grouping of X-ray/optical data from the XMM-Subaru deep-field observations (chapter 6). In both cases the algorithm has shown itself to be quite capable of performing such tasks and as such I propose that it could become a very useful astronomical tool. In summary I also present a few ideas for further refinement of the results presented by the KSOM and how these may be used in future study.
Contents

1 Introduction ................................................................. 1

1.1 Data Storage in a Nutshell: Papyrus to the Petabyte Age .......... 1

1.1.1 Large Data-sets ......................................................... 3

1.2 Astronomical Data Analysis:

Chewing Through Every Last Byte ........................................ 5

1.2.1 Classification: A Form of Analysis ................................. 6

1.3 Automated Classification ............................................... 7

1.3.1 Supervised Learning .................................................... 8

1.3.2 Unsupervised Learning ............................................... 9

1.4 Previous Work on Automated Classifiers .............................. 10

1.4.1 Principal Component Analysis (PCA) ............................... 12

1.4.2 Parameterisation ....................................................... 13

1.4.3 Cluster Analysis ....................................................... 13

1.4.4 Genetic Algorithms (GAs) ......................................... 19

1.4.5 Learning Vector Quantisation (LVQ) ............................... 21

1.4.6 Summary .............................................................. 24
CONTENTS

5 Application to Exo-Planet Transit Lightcurves

5.1 The Task ................................................................. 105

5.2 Methods of Planet Finding ........................................ 106

5.2.1 Pulsar Timing ..................................................... 106

5.2.2 The Doppler Method .......................................... 107

5.2.3 Gravitational Lensing ......................................... 109

5.2.4 Direct Imaging ................................................... 110

5.2.5 Transits ............................................................ 110

5.2.6 Future Techniques for Planet Observation ............ 117

5.3 Transit Separation Using the KSOM ......................... 118

5.3.1 The Simulated Data ............................................ 119

5.3.2 Generating the Simulated Data ......................... 119

5.3.3 Utilising the Simulated Data ............................. 120

5.3.4 Results with the Simulated Data ..................... 121

5.3.5 Utilising Real OGLE Data .............................. 131

5.4 Conclusions .......................................................... 136

6 Alternative Applications and Future Directions ............. 137

6.1 Alternative Applications .......................................... 137

6.1.1 Analysis of the XMM-Subaru Dataset .................. 138

6.2 Future Directions .................................................. 150

6.2.1 Hierarchical Network Analysis ......................... 150

6.2.2 Pre-Trained Analysis ...................................... 151

6.2.3 The Floating-Point Map ................................... 152
Chapter 1

Introduction

1.1 Data Storage in a Nutshell: Papyrus to the Petabyte Age

Around 2500BC papyrus, or paper, technology was developed and the rapid storage of information could begin (Fielden 2002). It was the true beginning of the data-storage era for the western hemisphere. There would be many people who, in these first years of easy data storage, would contribute, gather together and submit to hardcopy the information that had up until then remained largely distributed by word of mouth. This data-gathering continued and the libraries of the ancient times began to form. By 280BC the library of Alexandria, the largest library of its time, and for over a thousand years hence (Fielden 2002), reached its storage peak with over 700000 documents adorning its shelves (Parsons 1967).

In more modern times such places as The British Library or the American Library of Congress have become the main storage places for the written word and other non-electronic forms of data storage. The British Library alone houses over 625km of storage shelving (The British Library 2004), and this currently grows at over 12km per year.

With the advent of the world-wide web around 1990 the real distributed data-storage
explosion could begin. In 1998 the web search engine Google was accessing around 1 billion web pages (Fielden 2002), a quick check on any web browser today shows that that number has since grown to over 4 billion (4,285,199,774 as of September 2004 using Google.com English language search).

It is now the year 2005 and the total amount of written and electronically stored data in the world is believed to be around one exabyte, or \(10^{18}\) bytes, (Sullivan 2004). It is an almost incomprehendable amount of information and it is forever growing, be it in the form of scientific data, pictures, sounds or the written word. If the current data rate continues then by 2010, applying Moore’s Law, which states that storage density roughly doubles every 1.5 years, and assuming that people will be able to fill that space (as has always been the case in the past) then we will be gathering 16 times more data per second than we do now.

Within astronomy today we can already gather of the order of terabytes \((10^{12}\) bytes\) of data for single telescopes each year and, in the case of the large particle colliders such as that at CERN, petabytes \((10^{15}\) bytes\). To put that in to perspective: the longest place name in Britain is GORSAFAWDDACHAIDRAIGODANHEDDOGLEDDOLONPENRHYNAREURDRAETHCEREDIGION, each letter is 1 byte long, there are 66 letters therefore the word is 66 bytes long. You can fit the name roughly 25 times on to one page of this thesis, and this thesis contains around 150 pages. Therefore the thesis you are reading, were it full with only text, would be around 250000 bytes in size, or rather 250 kilobytes. Now, if you stacked copies of this thesis one on top of the other so that they could reach the Moon, given that the thesis is around 0.02m thick and the distance to the Moon is around \(3 \times 10^8\)m, you would have an erection that contained only the same amount of raw data as one year’s experimentation at CERN or a few years observations at a survey telescope.
1.1.1 Large Data-sets

With the increasing sensitivity, storage and processing power of modern technical systems it is inevitable that the amount of data gathered will increase by greater and greater amounts each year. However, an unfortunate disparity in the rate at which we can store data and the rate at which we can usefully process it means that new and more inventive data-reduction techniques must be realised. This is a problem faced by all of the large data-gathering projects, which within astronomy include the Wide Angle Search for Planets (WASP), Robotic Optical Tranient Search Experiment (ROTSE), Optical Gravitational Lensing Experiment (OGLE), the Visible and Infrared Survey Telescope for Astronomy (VISTA) and the All-Sky Automated Survey (ASAS). The technical details of one such project, run by the WASP collaboration, is outlined below.

The SuperWASP Telescope

An example of a high data-rate modern instrument is the SuperWASP telescope (Street et al. 2002) which has recently begun operations on La Palma.

This survey instrument consists of five $2048^2$ pixel CCDs with telephoto lenses attached, mounted to a robotically guided mount. This set-up allows the imaging of an 8° by 8° field of view with each exposure for each camera. To put this in to context the 'wide-field camera' of the INT (Isaac Newton Telescope) images an area roughly 0.5° across, or the size of the full moon.

A prototype system, WASP0, was in operation over the last few years and has produced promising results. It has shown that it is possible to perform 1% photometry on stars down to $11^{th}$ magnitude and, with 30 second exposures, make detections down to $15^{th}$ magnitude. The full-size SuperWASP telescope allows photometry of this accuracy to be performed but with several hundred square degrees worth of information being gathered at each exposure. This in turn relates to a raw data gathering rate of the order of 10TB per year. Table 1.1 shows where this estimate comes from.
Table 1.1. Data volume calculations for the WASP telescope

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame duration (s)</td>
<td>10</td>
</tr>
<tr>
<td>Dead Time (s)</td>
<td>3</td>
</tr>
<tr>
<td>Frames per minute</td>
<td>4.61</td>
</tr>
<tr>
<td>Frames per hour</td>
<td>276.9</td>
</tr>
<tr>
<td>CCD pixels (on one edge)</td>
<td>2048</td>
</tr>
<tr>
<td>Bits per pixel</td>
<td>16</td>
</tr>
<tr>
<td>Data volume per frame (bytes)</td>
<td>8388608</td>
</tr>
<tr>
<td>Data rate (MB hour$^{-1}$ CCD$^{-1}$)</td>
<td>2215.3</td>
</tr>
<tr>
<td>Hours per night of observing</td>
<td>10</td>
</tr>
<tr>
<td>Frame rate (Frames day$^{-1}$ CCD$^{-1}$)</td>
<td>21.6</td>
</tr>
<tr>
<td>Observing efficiency</td>
<td>0.66</td>
</tr>
<tr>
<td>Nights per year</td>
<td>240.9</td>
</tr>
<tr>
<td>Frames per year(per CCD)</td>
<td>667107</td>
</tr>
<tr>
<td>Stars per frame</td>
<td>14000</td>
</tr>
<tr>
<td>Brightness measurement rate (stars CCD$^{-1}$ night$^{-1}$)</td>
<td>38769230</td>
</tr>
<tr>
<td>Bytes per record</td>
<td>32</td>
</tr>
<tr>
<td>Data volume (GB year$^{-1}$ CCD$^{-1}$)</td>
<td>5211</td>
</tr>
<tr>
<td>Compression ratio for raw images</td>
<td>2.5</td>
</tr>
<tr>
<td>Archive storage (TB CCD$^{-1}$ year$^{-1}$)</td>
<td>2.02</td>
</tr>
<tr>
<td>Results storage (GB CCD$^{-1}$ year$^{-1}$)</td>
<td>298.3</td>
</tr>
</tbody>
</table>
Being able to gather these vast amounts of data is all well and good, but unless we can analyse it it is as good as useless.

1.2 Astronomical Data Analysis:

Chewing Through Every Last Byte

A consequence of all this stored data is a need to have automated information retrieval systems, such as the Virtual Observatory (Szalay 2001), which can quickly and efficiently locate the relevant, or requested, data and present it to the scientist so that they might perform the data analysis. Such information portals are being designed and implemented so as to create an easy single interface to any and all data pertaining to the astronomical arena. With such huge quantities of data coming from a variety of sub-fields within astronomy, be it radio astronomy, high energy astrophysics or the time-domain study of extra-solar bodies, it seems in the best interest of the scientist and their studies to have this information accessible from one point, allowing the often tedious and time consuming task of data gathering to be done by the software. Archive once, retrieve many times is the key idea here.

However, as stated previously, these data are of no use to us if we cannot analyse it for scientific gain.

Data analysis may be many things, it could be classification of an identified object, identification of the object itself, the measurement of parameters of the object (such as flux received in photometry, size of the object imaged in radio, energy involved in a particle collision, the atomic/molecular lines visible in spectroscopy, or the colour of an object in different wavebands) or even the process of searching for patterns in those parameters (much like colour-magnitude diagrams of star clusters, or the Hertzsprung-Russel diagram, Zeilik et al. 1992).
CHAPTER 1. INTRODUCTION

1.2.1 Classification: A Form of Analysis

In this thesis I will cover work on automated classifier systems, concentrating mainly on those systems used to classify photometric lightcurves, but with the potential to perform any number of data classifying, or parameter grouping, tasks.

Classification of objects is useful as it allows us to not only investigate the populations of various types of phenomena, but also because knowledge of what an object is allows us to make informed decisions as to what type of further scientific investigation may be pertinent. For instance, if we were trying to calculate the mass density coefficient for the universe, $\Omega_M$, then it is helpful to understand the distribution of galaxies at various red-shifts (Verde et al. 2002). This requires that we know which objects we are looking at are galaxies and so we need to identify and classify these objects as such.

As for the latter point, follow-up investigations, this becomes important when we may wish to clarify a classification or identify which objects within a field we have observed we believe are of a certain type (ie: searching for planetary transits within the Pegasus field (Kane et al. 2004)) and so may require alternative, but often costly, methods of follow-up observation. An example of this is the follow-up spectroscopic observation of the transiting planet HD209458b to determine information about its atmospheric composition (Brown et al. 2000) which required the use of the very impressive, but also very expensive, 10m Keck telescopes.

Lightcurve Analysis

There are many stages to the full analysis of any astronomical data. In the instance of lightcurve analysis, of which this thesis is concerned mainly with the fifth point below, the following stages may be taken:

Firstly the data gathered by the instrument must be stored and checked for errors, beit something simple like “the shutter failed to open on the instrument”, or something more complicated such as the flat-fielding or removal of image abberations
in an ultra-wide field image.

Secondly the relevant data must be extracted. This may take the form of source/object identification and extraction in an optical field.

Thirdly the measurable parameters for these objects must be calculated from the data and stored in some manner.

Fourthly these parameters may be used to calculate further information about the object. Such as the objects type or classification, for example in variable star analysis one may choose to look at a number of photometric observations of the source star, calculate the star's periodogram and thus attempt to extract a main oscillation period for the star's variability.

Fifthly that period can then be used to create a phase-folded lightcurve. This lightcurve may then be analysed to see what type of variability, and therefore what class of object, is being observed.

Sixthly further study of this object could then be undertaken, given that the objects class is now known, at further wavelengths (possibly using a VO, as mentioned above in the main section 1.2) or using alternative methods or even simply further observation.

And the list could continue to any number of steps of analysis. So it becomes rapidly obvious that reducing any number of those steps to automated procedures carried out by a computer should mean that a lot more real science gets done, leaving the monotony of bulk data classification and reduction to the computer.

1.3 Automated Classification

In response to the ever-increasing data-set sizes that modern instruments produce, and the limited number of man-hours that can be expended on the rudimentary data analysis,
it is necessary to turn to automated techniques to deal with the more repetitive data-handling tasks. Automated analysis may take the form of a simple script looping over several commands or it may be as complex as an image-recognition and automated classification software suite. At either extreme it is all about removing repetitive tasks from the operator and placing them in the trust of the computer. The human operator's time is spent instead on encoding and preparing the automated technique so that time may be saved later in the data-processing stages. Consequently, more time can be spent on the scientific analysis of the resulting classifications.

Classification and identification are both based on finding patterns in parameters. Just as a person learns to recognise items or words through traits they associate with the item (a smell, taste, sound, texture or appearance), so artificial systems, such as a computer program, may learn to recognise sets of parameters in data.

To summarise, when we refer to automated analysis techniques we are referring to those methods that do not rely solely on a skilled individual or group, but on the software or algorithm that attempts to mimic the abilities of those experts. These techniques fall broadly under two categories, supervised learning and unsupervised learning.

### 1.3.1 Supervised Learning

A supervised learning scheme (within the context of classification) is one where the resultant classifications or groups derived by the scheme are verified by an external source, such as an experienced human operator, or by having knowledge of the dataset classifications a priori. Using comparisons between the output of the supervised learning scheme and the knowledge held about the desired output from the scheme we can place a value on how well the current configuration is classifying the dataset. Subsequently, if the data are not classified sufficiently within set tolerances the configuration of the learning scheme can be modified and the test run again. This can be repeated iteratively until the errors made by the method drop below a pre-defined threshold. Once this supervised
training period is completed the classifier should be able to classify new data without further outside influence.

The disadvantage in using supervised learning is that the inherent structure within the data-set is not being investigated directly, rather the ability of the classifier to separate out the pre-defined classifications deemed to occur in the data. By extension this means that a classification algorithm trained in this manner will not be sensitive to new classes of object within the data. Instead, it classifies these objects as one of the pre-determined types, or placing all unrecognised classifications in a single "no matches" group.

1.3.2 Unsupervised Learning

A more fully automated process involves unsupervised learning. This relies on the classifying algorithm to find the underlying groupings and structure within the data and thus lends itself more completely to fully extracting the variation in the data presented. Unlike supervised learning the output results are not confirmed or measured by a human operator, or pre-defined set of "correct" answers, but rather follow some criterion for convergence. This may be, for example, a set number of learning iterations, a measure of the variance across the output within a defined tolerance, or simply that all the data has been run once through the algorithm. In theory this means that previously unknown groupings within the data should become apparent once run through the algorithm. However, it also means that there is ample chance for the learning mechanism to focus on consistent irregularities in the data (such as that which would be produced by poor pipeline processing and data reduction, but that in a supervised regime would simply be brushed aside and ignored), or to over analyse the data and for the groupings to start becoming noise dominated and therefore overly segmented.
1.4 Previous Work on Automated Classifiers

Classifications within a dataset are features that can be dependent on the parameters chosen to form the classifications. If a poor, or insufficiently descriptive, set of parameters is chosen then the resultant groupings, or classifications, may seem non-sensicle or scientifically unhelpful.

Within the context of astronomy, automated classification, supervised or unsupervised, is a field which is rapidly developing. For large all-sky surveys the ability to distinguish the many thousands of detected galaxies from the many millions of stars is extremely important, and more so to divide those galaxies into the various morphologies, be they elliptical, spiral or irregular. Work to this end has been performed using self-organising maps (Mähönen & Hakala 1995), artificial neural networks (de la Calleja & Fuentes 2004), image-derived parameter based classification (Lotz et al. 2004) and shapelet decomposition (Kelly & McKay 2004). A number of papers have also been published concerning the classification of spectroscopic data, or rather the classification of stellar spectra with respect to the Harvard sequence (OBAFGKM and the sub-types). These have used a variety of techniques from component analysis to create suitable parameters for classification (Deeming 1964, Singh et al. 1998) to the analysis of these parameters using artificial neural networks (ANNs) and genetic algorithms (GAs). In summary the automated analysis of stellar spectra and source distinction appear to have well established communities.

Conversely there are seemingly few people involved in the development of automated techniques required to analyse and classify the vast quantities of time-series photometric data, such as that produced by the WASP, ROTSE or ASAS collaborations. Although impressive work has been carried out with respect to temporal variability classification (Belokurov et al. 2003, Belokurov et al. 2004), this has often been concerned only with finding a single type of variability and not with the full analysis of the dataset. Within Belokurov’s papers classification techniques based on artificial neural networks
(see chapter 2) have been exploited in order to extract single types of variability, specifically microlensing events (see section 5.2.3). It is perhaps a consequence of technology only now providing us with the mountains of time-domain data that require the use of automated analysis techniques that more work hasn’t been done previously in this area.

Covered below is a brief summary of some of the various techniques available for use in astronomical classification work.

**PCA:** Principal Component Analysis is the process of reducing the number of parameters, $N$, required to describe a dataset by analysis of the lines of minimum variance through the $N$ (or less) dimensional parameter space. For example we may create a best-fit straight line to a set of points in a 2D space, we could call this line the principle component. It is very useful for reducing the number, and hence complexity, of a dataset for further analysis.

**Parameterisation:** Parameterisation is the process of reducing a complex dataset to a number of descriptive parameters.

**Cluster Analysis:** A form of classification analysis which is used to group similar objects together in a dataset in to clusters. The end result should be a set of interlinked data, those closely linked are more alike than those further apart.

**GA:** Genetic Algorithms. A form of analysis that works much the same as evolution in nature. A set of templates will change based on how alike they are to some form of input data. In addition to this, and what makes the GA different to other techniques, is that a small amount of random “mutation” is mixed in to the templates. Sometimes this means the template will better fit the data, other times it does not. A useful method for solving complex problems quickly, but also susceptible to non-convergent answers.

**LVQ:** Learning Vector Quantisation. A type of supervised learning that works so as to minimise the differences between an input dataset and a set of parameter
CHAPTER 1. INTRODUCTION

templates. This is achieved via constructive learning (the data are blended positively with the template) on correct classification and destructive learning (the data are blended negatively with the template) on incorrect classification.

This discussion shows that there are strengths and weaknesses in using the different techniques, some techniques, such as PCA (Principal Component Analysis), are best used as a tool to reduce the complexity of a dataset and others, such as the MST (Minimal Spanning Tree, a type of cluster analysis), are good choices when the dataset you wish to analyse has relatively well defined groupings and possibly unknown classifications.

1.4.1 Principal Component Analysis (PCA)

This is a commonly used technique in astronomical data analysis. Its primary aim is to reduce the number of parameters the analyst is investigating by attempting to find trends within the parameter space. This trend may be represented by a straight line within a 2D, or higher, parameter space (if the points within this space lie close enough to a straight line) or to a plane in 3D or higher space and so on. For instance if we had the graph of $x = y$ on the standard (x,y) axes then we could reduce this to a single line following $x = y$ and all the points on that line to a distance down the new axis; this, of course, is a simple example but it is this general idea that is applied to Principal Component Analysis (PCA). The method attempts, by minimisation of the linear combination of variables, to find the principal components (or axes) of any nD data-set and thus reduce the number of parameters being dealt with so that groupings or trends may be located in a more manageable, i.e. more suitable for visualisation, data-space. In other words the principal component attempts to account for as much of the variability in the data as possible, the second component as much of the remaining variability and so on.

This method is an excellent starting point for reduction of the data-set complexity, reducing the data to features which characterise them. Therefore making the resulting, reduced parameter data, easier and more efficient to sort. An example of this can be seen
in the stellar spectral classification work of Singh et al. (1998). In this work the relatively high resolution data (the spectra themselves) are reduced from around 700 parameters to just 20 using PCA (resulting in a 99.5% reconstruction of the original data from the 20 PCs). This pre-processed data are then fed in to an artificial neural network code (ANN) where the data are then classified in to the various spectral classifications.

1.4.2 Parameterisation

Much like PCA this method uses the idea of reducing what could be a rather complex set of data to a reduced set of parameters (indeed it could be said that PCA is a form of parameterisation). These parameters may then be compared for similarities and groupings determined. A basic form of parameterisation may be to reduce a set of periodic lightcurves to their main periods. These may then be compared and separated by the scientist in to different classifications of periodic variable based on different period ranges. Of course this is a very simplistic parameterisation and in a real dataset would be of limited benefit. However, the reduction of a complex dataset, such as a folded lightcurve which may contain 64 points, to a small set of parameters, the period and/or the first 6 Fourier components based on power, yields an object that is much simpler to deal with. And of course a reduction in the number of parameters to investigate should mean a dataset that is quicker to analyse and thus more useful for classifying algorithms which deal with large amounts of data.

1.4.3 Cluster Analysis

Cluster analysis is, as it sounds, the analysis of data in terms of the distances between data-set members, or rather of the clusters that are apparent in the data. By looking at the distance between any two points within an n-space you may determine how similar they are (on a relative scale) compared to other pairs of data within the n-space.
Single Linkage Hierarchical Clustering

One form of cluster analysis, called single linkage hierarchical clustering (Murtagh & Heck 1987), works by finding the least dissimilar pairing of data and pairing them together. This pair is then agglomerated to form a new object (that object being a mixture of the two members of the pair) and the differences between this new object and all the other members of the data-set calculated. We call this new object a cluster. This process repeats, each iteration creating either a new closest-match pair, absorbing another member in to one of the clusters which has formed or agglomerating two of the previously created clusters together. This process continues until there are no more members of the data-set remaining and we obtain an interlinked set of clusters. When these interlinkings are viewed we are left with a hierarchical structure much like the sketch shown in Fig.1.1 (we refer to this type of diagram as a dendrogram).

This in turn gives us a result whereby the apparent clusterings in the data may be extracted. However, this does not indicate to us clearly where two objects just happen to be the least dissimilar rather than the objects actually being similar and thus belonging to the same groups. Therefore locating sensible cut-offs for any cluster, or the possible overlap between two or more clusters, may be difficult to judge. It is, however, a reason-
ably efficient algorithm as at each iteration only $n - 1$ new comparisons (where $n$ is the number of data groups, single or agglomerated, which remain), between the groups/data that currently exist and the last-updated group need be made (as it is only in the initial stage that the $n!$ comparisons must be made between all data-set members). For this reason with smaller data-sets this is a suitably fast algorithm, however as the data-sets get larger it becomes less efficient than the more modern methods (such as the LVQ method discussed below) which may make comparisons not with other members of the data-set but with a smaller set of simulated, or template, data.

The advantage of this method is the ease with which the data groupings may be viewed and sub-divided. With distinct and individual links the group extraction process is simply a matter of choosing where to break the dendridic connections on the dendrogram.

Nearest Neighbour Chain Algorithm

There is another form of cluster analysis that works on a similar principle. Known as the Nearest Neighbour Chain algorithm, or NN Chain algorithm, this also works by arbitrarily choosing a data-set member and then finding the nearest data-set members to it and agglomerating them together in to a single group. However, unlike the single linkage hierarchical clustering method the agglomerations are only formed when a certain condition is met. This occurs when two points within the data space are mutually nearest neighbours (Reciprocal Nearest Neighbours or RNNs) and so the NN chain ends. This means that rather than updates of the distances between points being done for every nearest neighbour the update is only calculated once NN chains end and is therefore done less than with a single linkage method.

The method is explained more clearly if one imagines a line of points at (0), (2), (3), (4) and (7). If the first point, that at position (0), is the first point we look at then its NN will be the point at (2), its NN the one at (3) and the NN chain continues to (4), given that (4) is no further away from (3) than (2) is we may continue the chain to here. However, at this point the nearest neighbour of (4) is (3) and not (7). Therefore we have an RNN
FIGURE 1.2. A sketch to demonstrate the Nearest Neighbour Chain clustering algorithm. Here showing the first groups in the sequence.
and the chain ends. Thus the first 4 points can be formed into a group and the NN chain algorithm can continue starting from point (7). This continues, as per the single linkage algorithm, until only one point, or group, remains. The process is illustrated in figure 1.2. If we were to look at examples with higher dimensionality we could simply use the Euclidean distance between the points to calculate the NNs.

**Minimal Spanning Tree (MST)**

Yet another form of cluster analysis utilises the Minimal Spanning Tree (or MST) approach. This once again begins by taking an arbitrary point in the data-set and locating the least dissimilar neighbour within the remaining members of the data-set. These two points are then linked and said to form a sub-graph of what will become the MST (which will, upon completion, include all members of the data-set). This sub-graph then has all its data-set members compared to all the remaining (non-members of the MST sub-graph) data and the least dissimilar neighbour to any member is then connected to the sub-graph. The process is then repeated until the data-set members are connected in the one sub-graph and the MST is completed, as in figure 1.3.

This particular method of cluster analysis would appear to lend itself more easily to fully automated classifying as the MST may now be analysed for outlier links, i.e. links where the distances between the two points of connection are a set amount greater than the mean and so it would appear we have two distinct groups. An excellent demonstration of the behaviour of an MST given certain data can be found in an example given by Kohonen (Kohonen 1990).

**Single-Pass Overlapping Cluster Algorithm**

The clustering methods discussed so far have all involved hierarchical structuring of the data to form one large interlinked cluster, the links forming on various levels of similarity. However, clustering algorithms also work via partitioning, or non-hierarchical,
Figure 1.3. A sketch to demonstrate the Minimal Spanning Tree approach to point clustering. The dotted lines on the lowest plot show where there are particularly long linkages in the tree and therefore possible group edges.
CHAPTER 1. INTRODUCTION

means. One such non-hierarchical method is called the single-pass overlapping cluster algorithm. Unlike the previous algorithms that have made use of the smallest dissimilarities between two objects in the data-set this algorithm has a set tolerance, \( t \), above which the two data-members may not be considered to be of the same group. The algorithm begins by taking the first object in the data-set and assigning it to cluster 1. The second object in the data-set is then compared to the first, if the difference between the two objects is below a certain threshold, \( t \), then the object is assigned again to cluster 1 and the cluster is altered to be an amalgamation of the two cluster members (usually the mean of the two objects’ data points), this process is then repeated with subsequent data-set members. Should an object not fit within the tolerance to deem it one of the cluster 1 members then a new group is begun using the current object, or data-set member, as the cluster template. This process is then repeated for each object in the data-set until all the objects have been analysed and, within the set tolerance, all the inherent clusters in the data have been found. Figure 1.4 gives an idea of how this process might occur if presented with coloured dots to group.

This is a very suitable method for finding the natural groupings in the data, given that a suitable \( t \) is chosen \textit{a priori}, as the clusters that form are naturally distinct from each other and therefore a clean classification is possible. However a further step may be necessary to distinguish which of the formed clusters are similar and thus, if too rigid a \( t \) has been chosen, which clusters may in fact belong to the same classification. With any dataset, where the contents are not known \textit{a priori}, choosing a suitable \( t \) first time could be a very tricky choice.

1.4.4 Genetic Algorithms (GAs)

Genetic algorithms (GAs) are a form of machine learning adapted from the way in which a biological system evolves to fit a situation or solve a problem. Unlike other methods which may involve either a hard cut-off boundary to define an object as being of a certain class, or methods where the learning is accomplished by the blending of some training
Figure 1.4. A sketch to demonstrate the Single-Pass Overlapping Cluster algorithm. Here the pink, green and blue colours are too dissimilar to fall under the same group. However the various shades of each colour are similar enough to form single groups. As the data are analysed the definitions (shades) of the groups alter to reflect the data.
data with a set of templates (see the clustering algorithms section above and the LVQ discussion below), the genetic algorithm works through a combination of blending and mutation.

As in nature it is the "genes" of the GA which adjust based on the environmental, or input, data. The basic idea behind the GA is that the genes, represented by a set, or string, of parameters, which best-match the input data are more likely to survive each training iteration than those genes which don't closely match the data. Figure 1.5 represents these parameters as a set of binary values illustrated as a binary waveform. In addition to this "survival of the fittest" method there is also the introduction of small amounts of mutation to parts of the surviving genes.

All in all there are several stages to each learning iteration: each gene is rated by some kind of fitness function which determines the likelihood of that gene's survival, the surviving genes are used to generate new genes based on these likelihoods (it is possible for more than one copy of a single gene to occur), the surviving genes are then randomly spliced together to form a further set of genes and it is these genes that may be subject to minor mutations. The mutations take form as an altered parameter within the gene, within figure 1.5 these mutations are indicated by the red boxed sections of each gene.

As each generation of genes is run through this training process the gene pool as a whole should move closer to a convergence of genes and environmental input.

This form of machine learning has been used by Carlos Prieto in a prototype automated stellar spectra classification program (Prieto 2003).

1.4.5 Learning Vector Quantisation (LVQ)

Learning Vector Quantisation, or LVQ, is another method which works under a supervised regime.

The LVQ classification space (or network) is made up of a number of classification points, which we shall call nodes, each of which corresponds to a certain classification of
CHAPTER 1. INTRODUCTION

Figure 1.5. This figure shows how through the combination of two "genes", here represented by a binary waveform, and a little mutation a better-match to the data can be created than was previously available from the 2 genes alone. In this example the parts of each gene highlighted by the dashed lines have been spliced together. The two mutations, indicated by the red boxes, show how a better-match may be created (mutation 2), and also how the mutation may fail to produce a better-match (mutation 1).
object. The classification is defined by a number of parameters that together make up the node, we shall call these parameters weights. These classifications may be set initially based on some object template or arbitrarily (though this latter method may result in a network that is slower to converge).

The training of the network proceeds via comparison of a set of training data (which will be classified a priori) with the various nodes on the network. By comparison of the weights on each node to those parameters that make up each training data (where the weights on each node will map directly to the parameters on each training data) the closest match node to the current data is found. If this best-match node is defined to be of the same classification as the data best-matching it then the node is updated to more closely resemble the data presented to it. This process results in a node that more closely resembles the data classified as a certain type within the training dataset. However, if the best-match node is not of the same classification as the training data then the best-match node is altered so as to be less like the data that best-matched it.

To put it mathematically, if the best-match is correct then the following update occurs to the network node:

\[ W_c(t + 1) = W_c(t) + \alpha(t)(x - W_c(t)) \]  \hspace{1cm} (1.1)

where \( W_c \) is the best-fit, or closest match, node vector, \( \alpha(t) \) some learning parameter between 0 and 1 that decreases with time and \( x \) the training data. And if the best-match is incorrect then the following update occurs instead:

\[ W_c(t + 1) = W_c(t) - \alpha(t)(x - W_c(t)) \]  \hspace{1cm} (1.2)

All the other nodes remain unaffected. In this way the algorithm alters the network nodes to best-fit a given set of classifications and the nodes themselves have no topological relationship. In a perfect system this would mean that all data of a specific type fall on a single node and so the classifications are easily extractable. However, it is a supervised
method that relies on the full set of classifications being known \emph{a priori} and so it does not lend itself well to locating data types not previously known to be in the data. Should these previously unseen data-types occur then they would simply fall on whatever nodes appeared most similar to them and be classified as something else; in a best-case scenario all data of the unknown type would fall on one node and be distinguishable from the data that should truly be there via further post-network analysis. It is also sensitive to errors introduced by incorrect classifications being used in training, however, one would hope that, in the case of training, a good unambiguous dataset would used.

LVQ has been used to distinguish between astronomical objects such as quasars, active galaxies, stars and normal galaxies in a variety of wavebands (Zhang & Zhao 2003). Their results found that LVQ was an effective method for separating AGNs from stars and normal galaxies given multi-wavelength data, managing an accuracy of around 88.5% for all objects. However, individually it was not as impressive, often confusing BL Lacs with quasars and AGNs and, to a lesser extent, galaxies with stars and AGN. This of course may simply have been due to the small sample size, and often low accuracy, of the non-stellar objects’ data.

1.4.6 Summary

To summarise, there are both strengths and weaknesses in the various techniques. Some techniques, such as PCA, are best used as a tool to reduce the complexity of a dataset and others, such as the various clustering algorithms, are good choices when the dataset you wish to analyse has relatively well defined groupings and requires very quick analysis. However, none of these methods present themselves as good possible classifiers (or rather clustering algorithms) suitable for the identification of previously unknown data-types which may also be underpopulated with respect to the other classifications. This would require an unsupervised clustering algorithm capable of handling “low impact”, underpopulated, data. The best option so far for such conditions possibly being the MST method.
However, there is a method suitable for the clustering of underpopulated samples, or those samples which may contain unknown classifications of data. We can use something called the Kohonen Self-Organising Map (or KSOM). It is this which is discussed in the next chapter, with a discussion of varying sample populations in section 4.3.2.
Chapter 2

Kohonen Maps and Artificial Neural Networks

2.1 What is a Neural Network?

An artificial neural network is an artificial device, usually computer software, which is designed to mimic some of the behaviour of the human mind. They are typically used to classify objects or calculate answers to problems where a precise answer, or classification, may be difficult to come by using analytical means, but where an accurate approximation or "most likely" solution to the question is suitable.

When presented with the task of describing such an object, whose appearance and behaviour does not clearly mimic that of items we see around us in our everyday lives, it can be exceedingly tricky to find a balance between over-simplification and, though closer to the true nature of the object, over-complication. A neural network may be described in the following, rather unspecific, manner:

"an abstract object which alters itself based on input stimuli from an external environment so as to better represent that environment and the objects defined within it"

Perhaps, however, a more useful initial description should revolve around a more
tangible object, something most of us will have had some experience with in secondary school art class.

Imagine if you will a lump of modelling putty. The putty is pliable, workable and, through input, action and comparison can be crafted in to alternative creations that do not appear as random, or homogeneous, as the initial lump, or ball, of putty. It may be shaped in a variety of ways so that its appearance is altered to mimic any number of objects, be it a bird, a car or a human face.

An artificial neural network (or ANN) is, in essence, very similar to this. Much like a lump of putty its structure is created by slowly altering its initial configuration so as to mimic inputs provided to it. However, whereas with a lump of putty we may use a picture of an animal as the creative input that helps us create the final configuration of the putty (a bird sculpture for instance), so with the ANN we will use a set of pre-chosen parameters to train its final configuration and it is these which the ANN is altered to mimic. As an extension of this the ANN can be used for many more tasks than simply classification. Any task which requires an output which fits some set of rules, such as solving a set of complex equations, or finding a lowest energy path of an object in a gravitational field (P.Hakala private communication), could be attacked using an ANN.

Once something resembles a known object it may be used to make comparisons to other objects so as to determine whether the new object is of the same type as the model. For instance, a young child may be presented with objects which resemble a horse, a dog and a cat and can use these to group further objects it is presented with in to “dog”, “cat” and “horse” classifications. An ANN is very much like this, having been presented with several objects it will compare them to the objects it “knows about” and, if a close match is found, may declare the object it is currently inspecting as “an object I know about of type X”. If no close match is found it may declare “an object I do not recognise” or “an object not of type X, Y or Z”, much like the child in the above situation may not be able to recognise an object shaped like a dolphin and would declare it “not one of these”.

This is in essence the purpose of a fully constructed and configured (or trained) ANN.
It is a tool for making comparisons of various types of input data with configurations of data it is aware of so as to determine type classifications for those input data.

In many cases these comparisons may be binary decisions, it either is or it isn't an object of a certain type. This type of network has been used successfully within astronomy for the extraction of microlensing lightcurves from other sources of variation (Belokurov et al. 2003) and for finding transients in massive variability surveys (Belokurov et al. 2004).

However, we may wish, as above, to identify a number of different objects in a data sample. In this case we want to have several template object representations within the network with which to make comparisons to the inputs the network receives. Were we to return to the putty analogy one more time we would be required to break the putty into numerous pieces in order to model the various templates. An ANN works in much the same way. However where we would have individual putty pieces we now have distinct sets of parameters. Each of these individual parameter sets has the ability to mimic an input stimulus (an object) and thus create the potential to distinguish one more classification in any input data. Within ANN terminology these templates are referred to as neurons or nodes, and the parameters that form them are known as weights. These weights, in the case of animal classification, may be such values as height, weight, number of limbs or average running speed. Within astronomy the node weights may hold parameters such as optical magnitude, X-ray to optical brightness, redshift or flux at various phase offsets (if dealing with periodic lightcurves of variable stars, as presented through the majority of this thesis).

Where an ANN differs from the distinct set of templates, as described by the putty analogy, is in the way these templates, or nodes, are linked together in some manner to form, as a whole, a network of nodes. Hence the term Artificial Neural Network. The way in which the nodes are related may vary between network implementations, but it is often the case that topologically close nodes in network space are similar, or near, in that their parameter sets, or weights, contain similar values. We may refer to the distance
between nodes as either a topological distance within the network space, or as a measure of the distance between their data as calculated by taking some measure related to the Euclidean distance between two points in space, where those two points' positions are described by the weights on the node as if they were a position vector.

This finally gives us a mechanism which, when presented with input data, can learn to mimic and therefore recognise that data and which can form a very powerful classifying technique. As stated by Russel and Norvig (Russel & Norvig 1995) a neural network, and by extension its computational counterpart (an Artificial Neural Network, or ANN), is:

...composed of a number of nodes, or units, connected by links. Each link has a numeric weight associated with it. Weights are the primary means of long-term storage in neural networks, and learning usually takes place by updating the weights. Some of the units are connected to the external environment, and can be designated as input or output units. The weights are modified so as to try to bring the network's input/output more into line with that of the environment providing the inputs.

It is for this last point that the ANN is seen as an excellent tool to use when trying to classify, or sort, objects in datasets in to their component types. It is a type of what is known as machine learning.

2.1.1 Types of Artificial Neural Network

Here follows a very brief summary of the basis of artificial neural networks used in science, including astronomy, today. However, before the discussion is begun it is worth defining a few technical terms which relate to various inputs, outputs and states of the ANN.

The following attributes are given by the symbols indicated in the proceeding section:

Weights: The components of the neurons, described as $W$ or $W_i$. In the case of a weight relating to the connection between two neurons we may subscript both.
I.e: \( W_{j,i} \) relates to the weight \( j \) from a previous neuron, which influences weight \( i \) on another neuron. It is these which are updated so as to alter the behaviour of the network.

**Input:** The vector \( \mathbf{I} \), constructed of various weights, which describes the input to a neuron. This input may be constructed from the outputs of other neurons, or from the trial data itself (see below).

**Output:** The vector, constructed of various weights, which describes the output of a neuron. Described as \( O_i \), where \( i \) are the individual parameters. It may also refer to a scalar quantity in terms of a network’s final output decision.

**Trial Data:** The data, broken down into various parameters and described as \( T_i \).

**Activation Function:** Denoted as \( f() \). It is the function used to calculate when the various inputs should cause the neuron to switch on, or the manner in which the neuron outputs. This may be a simple sine function for example. It would usually be a function of both the weights, \( W_{j,i} \), and the input, \( I_j \) for example.

**The Perceptron**

The basic component of standard ANNs, commonly referred to as a perceptron, is not in itself a network but rather a condition-dependent switch. It is to artificial neural networks as a transistor is to modern electronics. With a number of inputs going to a single output this type of neural network is very good at making simple logical decisions, such as the basic boolean logic operators NOT, AND and OR. The perceptron works simply by having some activation level over which it outputs a 1 and under which it outputs 0. This activation level is determined by an activation function, \( f() \) (figure 2.1). As the inputs, \( T_i \), are totalled the related output level, using the activation function, is calculated. If the output value is greater than the activation level (or threshold) then the perceptron “fires”.

For the basic logic operators we could construct them thus:
Figure 2.1. An example of activation functions for the perceptron.
AND: Each input is worth 1.0, if the total input is greater than 1.5 then the perceptron fires.

OR: Each input is worth 1.0, if the total input is greater than 0.5 then the perceptron fires.

NOT: The input is worth 1.0, if the total input is less than 0.5 then the perceptron fires. This is much like using a different activation function that would behave in the same way as for AND and OR, but negates the input and output.

It can, however, fail on more complex problems. As an example consider the logical operator XOR. With XOR there is no "one line" of cut-off that can be drawn to arrive at an answer. If both or neither of the inputs fire then the output must be zero (no firing of the perceptron), but otherwise the perceptron is required to fire. This is a cumulatively non-linear activation function, the output cannot be calculated from a simple summation of the inputs, and so the single perceptron cannot handle it. For this reason there are only a limited number of problems where a single perceptron model is useful. However, when multiple perceptrons, or neurons, are used together in networks they form the building blocks of what can be very powerful tools. As the complexity of a network is built and the outputs of one neuron are fed in to another we may say that we have different layers to the network. Each new neuron output which feeds to the input of another neuron being said to go to the next layer. In effect we are "stacking" the neurons to create more complex networks, each time we place another set of neurons in a new level we create a new layer. Those layers that we can see, such as the input and the final output neurons are named simply input and output layers. Those intervening layers, over which we have no direct access, be it through data or the output result, are called hidden layers.

Feed-forward Networks

The feed-forward (also known as the multi-layer) network is not as susceptible to failure as a single perceptron when presented with the more complex problems. A feed-forward
network consists of an input layer and an output layer, but also hidden layers which lie between the two. These hidden layers effectively allow the combination of outputs from any two, or more, perceptrons into the output. This in turn means that more complex problems can be tackled by feed-forward networks.

As an example let's look at the XOR problem discussed previously. If we have a layered network with a layout as in figure 2.2 then we can create an XOR function.

In a more complicated example, when we are handling less "clean" problems such as the classification or extraction of a certain lightcurve type from other types, we will want to update the firing threshold values of the various neurons (which may additionally be referred to as nodes) contained within the network so as to better handle the extraction process. In a feed-forward network this is done by comparison of the results achieved by the network when presented with known data. If the network output agrees with the known answer then nothing is done, but if there is an error then the threshold values are updated so as to reduce the error. The tricky part here is knowing how to divide the "blame" for the error between the various weights that have produced the answer. As there is no feedback in the feed-forward network this is a non trivial problem and can result in a network that does not converge to produce correct answers (Russel & Norvig 1995). Instead any changes must be made manually in a "blind" trial and error manner. A more capable type of multilayer network is the back propagation, or feedback, network.

Feedback Networks

With a similar layout to a feed-forward network the feed-back, or back propagation, network has one important difference: the output weight values are compared to the hidden layer and subsequently the hidden layer node weights are compared to the input values. The comparison of the output and hidden layer allows for an error to be calculated between the activation level of the hidden layer nodes and the output they generate. This
Figure 2.2. An example of how an XOR feed-forward network may work.
allows the updating of the hidden layer's weights and subsequently the updating of the input layer's weights via back-propagation of the error.

The error between the output and trial data are:

\[ E_i = T_i - O_i \quad (2.1) \]

And the weights on the hidden layer are updated thus:

\[ W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times E_i \times f'(\cdot) \quad (2.2) \]

where \( \alpha \) is some learning parameter between 0 and 1, \( W_{j,i} \) are the weights relating to hidden node (\( a_j \)) outputs \( j \) going to output node weight \( i \) and \( f'(\cdot) \) is the value of the derivative of the activation function. The gradient of the activation function becomes relevant here as a steep part of the function may mean that there is a change of output state of the node over a very small range of input and so changes of state around this point, and how clearly the weights distinguish this, becomes more relevant.

This gives the update for the weights on the hidden layer to output nodes. For the updates on the input to hidden layer weights a very similar route is followed, except this time the error, \( E \), is calculated from a fraction of the error calculated previously.

If we say that:

\[ \Delta_i = E_i f'(\cdot) \quad (2.3) \]

and that this is an error relating to the weights between the output and hidden layer, and further that:

\[ \Delta_j = f'(\cdot) \sum_i W_{j,i} \Delta_i \quad (2.4) \]
where $\Delta_j$, a measure of the error contributed by the weights on the hidden layers, is used to update the weights in the following manner:

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times I_k \times \Delta_j$$  \hspace{1cm} (2.5)

where the subscript $k$ relates to the input to hidden layer nodes and $I_k$ the input parameters to the various hidden nodes.

The nomenclature becomes clearer when presented with a diagram such as figure 2.3.

This type of network, when presented with a known dataset is a very capable tool for specific pattern recognition. Within astronomy it has been used to extract microlensing lightcurves from other types of variation (Belokurov et al. 2003) and to estimate photometric redshifts from spectroscopic data (Firth et al. 2002). In both cases the networks were found to perform admirably, as well or better than previously used methods. However they do require a rather large known dataset to train with, which either means the generation of a large simulated dataset beforehand or a large amount of a priori classified data. These types of network are a supervised method and therefore cannot self-organise themselves into a state that can cope well with previously unknown data. To do that we may use the self-organising map.

**The Self-Organising Map**

This type of ANN, called the Kohonen Self-Organising Map (or KSOM, Kohonen 1990) after its creator, is constructed from a set of interconnected nodes, or neurons, each with a number of weights. These weights correspond to the parameters used to classify the dataset and it is these that alter in value so as to mimic the dataset and thus the network is said to learn and adapt to the dataset presented to it. In this way they can be used to identify known patterns in data or even to discover and recognise new patterns. In the case of singly periodic lightcurves, as in this thesis, the weights represent the normalised magnitude of photometric lightcurves at different phase offsets. How the
Figure 2.3. An example of how a feed-back or feed-forward network may appear with one hidden layer. $I_k$ denotes the inputs, $a_j$ the hidden layer and $O_i$ the output.
nodes are arranged within the network is a matter for the network implementer to decide
but in the case of Kohonen Self-Organising maps (section 2.3) they are simply arranged
in a grid structure, usually hexagonally or square tiled.

This ANN implementation is slightly different from a conventional implementation
as we are dealing with a contiguous map-space rather than discrete network neurons.
As the map is made up of nodes arranged in a grid structure we are not changing the
geometric manner in which the nodes are connected but rather the weights upon the
nodes, and thus the underlying map topology. Nodes adjacent in map space can be
thought of as being more similar to each other than those placed $X$ nodes apart. In this
way the network has areas upon it which represent certain ranges of weight values.

Ultimately they are nothing magical or fantastical that can simply solve all problems,
but they are a very useful new tool that if implemented correctly can save a lot of time
with tedious or laborious tasks.

2.2 Uses of Kohonen’s Method Across the Sciences

Kohonen himself has used his network to group various sets of parameters as demonstra-
tions of the method’s breadth of applicability. His interests appear to lie in the fields of
speech recognition and text analysis, and thus he has used the method to create “semantic
maps” separating the nouns, verbs and adverbs of a set of sentences and to group Finnish
phonetics in to a phoneme map.

As the basis of the method is to group sets of similar parameters together it may be
used for almost any situation for which a set of parameters can be defined. Among the
wealth of problems suggested by Kohonen as suitable for analysis using his method are
the following (Kohonen 1990):

- Statistical pattern recognition, especially recognition of speech;

- Control of robot arms, and other problems in robotics;
- Control of industrial processes, especially diffusion processes in the production of semiconductor substrates;

- Adaptive devices for various telecommunications tasks;

- Image compression;

- Radar classification of sea-ice;

- Optimisation problems;

- Sentence understanding;

- Classification of insect courtship songs.

In addition to these suggestions the map has also shown itself to be a valuable tool in the analysis of the astronomical literature (Poincot et al. 1998) and in discrimination between galaxies and stars in images (Mähönen & Hakala 1995, Miller & Coe 1996).

VizieR (Poincot et al. 1998) uses the Kohonen map to organise astronomical papers in to groupings based on the keywords present. Once this is done the user is able to select papers based on a subject area by clicking within a user interface which depicts the map and its associated areas. These areas may be such topics as the ISM, stars or galaxies; and these areas may have sub-areas within them for more explicit subject areas such as dust, binaries or galaxy evolution. Once the user has selected the topic they require they may view any of a list of papers generated from the CDS and ADS astronomical catalogues.

These last two examples are particularly interesting as they not only show the algorithm’s applicability to astronomy based problems, but to two very distinct and different astronomical problems.
2.3  The Kohonen Self-Organising Map Algorithm in Depth

2.3.1  An Introduction to the Algorithm

The algorithm I have developed utilises a form of artificial neural network, namely the self-organising map scheme due to Kohonen (1990). I am motivated in this choice by two desirable properties of Kohonen maps: simplicity of implementation and their ability to learn in an unsupervised régime. As mentioned in section 1.3.2 unsupervised learning allows a network to be trained without the requirement for a pre-classified training set, and free from any biases which might be introduced by a human trainer. The lack of pre-defined types also means the network is capable of grouping previously unknown types should they occur.

There are basically two sections to the algorithm, firstly the pattern matching, or shape recognition, to find the best-fit node on the network to represent the current member of the training dataset and secondly the blending and updating of the network with the data. Thus altering the network to better represent the training set. The following section details these stages along with the various network parameters and geometry used in the algorithm.

Network Geometry

The self-organizing map consists of a set of \( M \) neurons (or nodes). Associated with each neuron \( j \) is a reference vector, \( w_j \).

The neurons are arranged in an \( N \)-dimensional space. Each dimension is cyclic, so for example if we were looking at a 2-dimensional map it would be considered to have the topology of a torus. Figures 2.4 and 2.5 show how a 2-dimensional network of 16 nodes may appear. Figure 2.4 showing the overall structure contained within the network, with each node adjacent to the next and with the weights contained upon one of the nodes shown. The distance between nodes on the network is indicated by the various
colours on the nodes. The distance between nodes increases with decreasing intensity of pink, the furthest node is indicated in green. Further clarification of the cyclic nature of the map space is available by viewing figures 2.5 and 2.6, which show the distances between nodes with the wrap-around nodes redrawn and the contents of the nodes after one lightcurve has been run through the network, respectively.

Typically $N$ is chosen to be 1, 2 or 3; for the tests presented in this thesis, and most other work, $N$ is chosen to be 2. The primary reason for choosing to work mainly in 2-dimensions is simply for ease of illustration. However, I found that in all tests the data separated as well in 2D as they appeared to do in 3D. Should sufficient separation not occur in the chosen number of dimensions then a higher number may be selected to allow more freedom of movement within the network data space. This is, of course, at the expense of algorithm execution time as this scales linearly with the number of nodes. Each neuron $j$ therefore has a coordinate $r_j$ in this $N$-dimensional space (hereafter referred to as map space).

Further discussion of optimum network dimensionality can be found in section 4.1.3

An ideal network, once taught, will have structure within it relating to the objects it has been attempting to classify or separate. In the case mentioned in this thesis I am mainly trying to classify photometric lightcurves and so in the networks I create different areas will relate to different lightcurve shapes (see figure 2.7).

**Shape Recognition**

When presented with a stimulus in the form of an input vector, $x_i$, each neuron "fires" with a strength which is related to the similarity between the input and the reference vector. The reference vector of the most strongly firing neuron in the map therefore represents the closest match to the input vector.

Every template lightcurve is constructed from a reference vector made up of a set of weights. Each of these weights, $w_i$, has a value, which in the case I use relate to nor-
Figure 2.4. This figure shows how a 2-dimensional network of 16 nodes may appear. The highlighted box shows how the weights may appear on the nodes. Figure 2.6 shows this throughout a real network. The distance between nodes increases with decreasing intensity of pink, the furthest node is indicated in green.
Figure 2.5. This figure shows the distances between nodes with the wrap-around nodes redrawn for further clarity.
FIGURE 2.6. A 10x10 network's appearance before (above) and after (below) the blending of a lightcurve's features. The wrap-around can be seen with the most intense reinforcement of the lightcurve's shape around one node.
Figure 2.7. An example of the nodes on a 10x10 2-dimensional network using ten ROTSE lightcurves and an $\alpha$ learning parameter of 0.1. The top diagram shows the appearance of the nodes' weights after the first iteration and the lower diagram the appearance after the final (twentieth) iteration.
malised magnitudes, which can be altered during the learning process. When viewed as a whole these weights may make up a shape such as that of a lightcurve. For a network to make any sensible inferences as to the type of shape, or lightcurve, it is attempting to classify it must be able to match the shape with another. In an ANN we will almost certainly never have two completely alike shapes, it must therefore look for the most-similar shape to the lightcurve. This constitutes the first stage in the network learning process. The lightcurves I have been looking at do not necessarily all begin at the same point in phase, they are simply folded on the best-fit period. This means that there may be two lightcurves of the same type, but out of phase with each other, that will appear without alteration to be completely different. Therefore to properly perform the shape-recognition stage of the learning process we must search for the best-fit node at all phase positions, or offsets. In other words the current lightcurve must be shifted in phase between 0 and 1 (a shift equal to the phase difference between weights is sufficient, and indeed efficient, for this with a large enough number of weights per node) and then compared to all the nodes. The winning, or best-fit, node is that which produces the minimum value for the weight difference vector (calculated from the Euclidean distance in Cartesian space between the node weight vector and the lightcurve normalised magnitude vector). Once this best-fit node has been located the lightcurve may be mixed with the ANN and the network “learns”.

**Network Learning**

In the implementation of the algorithm used here the reference vectors $\mathbf{w}_j$ of the neurons can be thought to represent template lightcurves, and the input vectors $\mathbf{x}_i$ the measured, magnitude normalised, lightcurves of the astronomical objects I wish to classify.

The reference vectors of the neurons are initially assigned random values. The goal of training the network is to adjust the neuron reference vectors in such a manner that members of the input set that are “similar” are placed more closely together in map space than members that are dissimilar. In that way we may later extract areas of the map which
should represent data of the same morphological type.

The network is trained in an iterative fashion by exposing it to a training set. Each member of the training set is assigned a coordinate in map space corresponding to the neuron in the map that most closely matches the member. The reference vectors in the best-matching neuron, and to a lesser extent those neurons surrounding it, are updated by blending them with the input data. The algorithm continues until all members of the training set have been assigned a node and have updated the network reference vectors accordingly. In the second and subsequent iterations the network is repeatedly exposed to the same training set, and the algorithm proceeds in this manner until some completion criterion is met.

The blending process is a key element of the algorithm, as it is this that allows the network to "learn" by adapting the contents of the neuron reference vectors to match the incoming data. The learning process is competitive; features in the reference vectors which lead to good matches with input set members tend to be reinforced. This reinforcement effect is stronger if the feature is shared by many members of the training set. The idea behind this is that similar lightcurve shapes will group around the same nodes in the network and therefore reinforce their presence in that area; if the blending affects more than one node then we allow lightcurves which are similar, but sufficiently different so as to best-fit different nodes, to affect one area of the network and thus similar lightcurve shapes begin to group together into clusters. It is, if you like, the idea that in mixing all the lightcurves of a similar shape in small amounts we may create a compromise form which is a best-fit for a large selection of slight variations of one lightcurve shape. Thus as the network learns the coordinates of the members of the training set in map space will change, and training set members which resemble each other will tend to self-organise into clusters within the map space.
CHAPTER 2. KOHONEN MAPS AND ARTIFICIAL NEURAL NETWORKS

2.3.2 Network Parameters

The instantaneous rate at which the network learns is influenced by both the learning rate coefficient $\alpha$, which controls the degree to which reference vectors are blended with the input data, and $\sigma$ which determines the effective size of the blending neighbourhood. In a typical application of the Kohonen scheme both $\alpha$ and $\sigma$ are chosen to be monotonically decreasing functions of time, $t$. The combined effect of such a choice is that the large-scale structure in the map tends to form early in the training phase, with the finer details crystallising in the later stages of learning.

The $\alpha$ Parameter

$\alpha$ may be seen as the ratio by which the best-fit node is blended with the training data. As we wish to train the network in such a way that the trained map is stable we should reduce either, or both, the $\alpha$ parameter and $\sigma$ neighbourhood distance (discussed below). $\alpha$ can be defined to behave in any of three ways: firstly to remain at a constant value throughout the learning phase, secondly to decrease linearly to zero during the learning and thirdly to decrease with some other function.

I found that altering the initial value of $\alpha$ and the way in which it behaved affected the map more strongly than other parameters and thus it is this which is discussed in more depth in section 4.2.

The $\sigma$ Parameter

During the blending process the current training dataset member is blended strongly with its best-fit node and less strongly with those nodes topologically close to the best-fit node. The strength with which those surrounding nodes are blended is dependent upon the choice of blending function, and the power with which this function blends is related to the neighbourhood parameter, $\sigma$. 
The initial and final choices of $\sigma$ primarily affect what could be called the crystallisation size of the map. As the map becomes more structured through network learning different areas change so as to represent different types of data, the larger, and thus more general, these areas are the larger the "crystals" of data types appear to be. Thus we may refer to the "crystallisation size" of the map.

In order for large scale structure, and therefore order, to form in the map we want to include the whole map more strongly in the initial iterations. Thus the initial $\sigma$ is chosen to be half the map width (for non-regular map sizes you would choose the size of the smallest dimension so as to avoid strong overlap due to the cyclic boundary conditions). The final $\sigma$ is chosen such that the crystallisation area is small enough for intra-cluster differences to resolve themselves without affecting too strongly the overall structure of the map. This whole process of shrinking $\sigma$ means that there is a chance for similar but different data types to separate, and thus for the map space to reflect more subtle differences between the data types. This small difference resolution can be helpful in the cluster definition and extraction stages as discussed in section 3.3.

The Number of Iterations

In Kohonen's original algorithm one iteration is defined as having one object run once through the network (Kohonen 1990), two iterations are deemed to have completed once the first and second members of the classifying dataset have been run through, and so on. This process is repeated and the first, and subsequent, data run through the network repeatedly until a set number of iterations are completed. In this format there may be hundreds of thousands of iterations during the training process. The definition of an iteration, as stated for the work throughout this thesis however, is as stated before; one iteration corresponds to the entire training set being run through the network best-match and blending process once, two iterations to the entire set twice, and so on.

An alternative and more automated approach may be to request that learning stops when the network is sufficiently stable or when the mean difference between the dataset
and the best-fit nodes falls below a threshold level.

Network Size

One further point of consideration, which may fairly be seen as a weakness of the static size of the Kohonen SOM when compared to more volatile networks (such as the growing cell structure, Fritzke 1991), is that to easily extract the classifications/clusters later we should ideally have a greater number of nodes than classifiable data types. The intention being that different classifications of data will fall on different nodes.

In reality the number of classification groups will not be known. Indeed this is a reason to use a self-organising map algorithm as it allows the creation of a program fully capable of finding what may normally be discarded as "not of any grouping" using conventional methods.

Further discussion of the network parameter choices investigated in my use of the KSOM can be found in section 3.2.
Chapter 3

My Algorithm

3.1 Description

The algorithm and code developed for this thesis is a slight modification of the Kohonen SOM algorithm discussed in the previous chapter.

Unlike Kohonen’s implementation the iterations in this version of the algorithm are sequential once all the data has been run through the network process, rather than each iteration being considered as complete when one data has been sent through the network code.

In addition to this the code has been written and modified such that should later experimentation be desired with multiple, or hierarchical, networks then transference of data between networks, plus generation and addition of new data in to the network, should be quite manageable. Indeed there is code already implemented to handle such tasks. Although it has not been thoroughly tested due to lack of need and current redundancy. Should it be desired or required it would only be a short project to test and remove any of the minor bugs that may exist in the network/data transportation code.

As this is an e-science research project it is prudent to spend a little time discussing those aspects of the research which are more focused on the internal handling of the data
(with respect to the code) and how this has evolved to handle the requirements of the thesis research.

### 3.1.1 Evolution Of The Data Structures Present In The Code

In any program where a large amount of data are to be handled it is important to create a system which does so in a manner that should be both efficient, in terms of access and in terms of program execution, and relatively easy to understand. Often it is the case that a piece of ultra-efficient code is not necessarily the easiest to maintain. A further point of consideration is adaptability; it was the intention of this coder to create a code which could later be used for a variety of clustering/classification problems and as such the algorithm data-structures and routines have been designed with this in mind.

Initially the data to be analysed (in this case phase-folded lightcurve data) was read in and placed into one long array, with another array containing the length of each sub-array (one sub-array for each member of the dataset) within the 1D data-array. The array was kept like this and not formed from a 2D array as each dataset member had a varying number of data points, thus fixation of the secondary dimensional size would have been inefficient in terms of memory usage. From this lightcurve data the binned data are created and empty bins interpolated across in order to remove these blank spots (and thus problems later in the code in the lightcurve best-matching and blending functions). This interpolated data, which is used in the actual network algorithm, was stored again in a large 1D array. In this case there are a user defined number of points per lightcurve, or rather weights per node, and so we need not store the size of each lightcurve as before. The data were stored like this as it was no more complicated to do it this way and use pointer arithmetic than it was to call the individual elements in a 2D array. It also meant that pointer incrementing was an easy option if desired later on. As mentioned previously the code was designed to be easily alterable at a later date.

The network structure itself was also stored in a 1D array. This was done as I felt
it important to preserve the ability to explore networks of greater than 3 dimensions as for highly complex dataset variations this may be neccessary to truly extract the clustering behaviour. But also because it was no more difficult to create the few pieces of code that would deal with 1, 2 or 3D networks than it was to create code for higher dimensionalities, indeed they are the same, unaltered, sections of code. It is far more worthwhile taking the time to write an nD code using pointer offsets and a 1D array than it is to quickly cobble together highly obtuse code which can deal with a set number of dimensions and declares the arrays of a set dimensionality based on the number of dimensions requested. By simply creating a lookup table of Euclidean distances between any 2 nodes on the network, and by offsetting this table to calculate the various distances between nodes for different starting points, it is actually not particularly difficult to deal with nD systems. It is also quick as it only needs to be calculated once at the beginning of program execution and the power lookup table (relating to the strength of the current neighbourhood function for that particular iteration, given that the neighbourhood distance decrease with each new iteration) calculated once per iteration. The only limitation on size really being the memory free in the computer.

Further Refinements

Later on, due to wishes to make the code more expandable and adaptable for further tasks, the data structure was changed for both the interpolated lightcurve data and the network nodes themselves. The 1D complete array structure of the network was retained for the same reason it was initially created, although it was now encapsulated within a network structure which could be part of a doubly linked-list structure, should multi-tier networks be desired at a later date, with the network parameters also encapsulated within the bounds of these network structure elements. The lightcurve data to be handled by the network was placed in a similar structure. However each lightcurve’s data are stored in an individual lightcurve structure element, along with links to the next lightcurve data element, the previous and any associated lightcurve elements such as the lightcurve
internal designation or object name. The lightcurves were split up in this fashion so that more efficient use of memory could be made (in terms of using many small chunks rather than one large chunk which may fail on allocation) and also so that lightcurves could be moved between networks or removed entirely from the network processing code (they are simply moved or removed from the doubly link-list and the previous and next data entries in the list linked). This should allow the usage of hierarchical networks should they ever be desired; or simply just the removal of a bad data.

**Handling Null Data**

The final update to how the data are handled relates to the interpolation routine. Whereas before a linear interpolation was created across the data-absent parts of the binned lightcurve data, it now weights each part of the binned lightcurve data separately (in effect a per-weight weighting for the weights on the nodes). This stops the effect of data creation brought about by the interpolation in to bins resulting in, in cases of extreme data gaps, a highly false representation of the underlying behaviour of the data to be analysed. With the per-weight weighting technique these empty bins (weights with no data values associated) may be set to a weighting of zero and so are not considered in the best-fit node finding routine, nor do they affect the network blending part of the algorithm.

Although in these cases the weightings were either 0.0 for an empty bin or 1.0 for data-containing bin it is quite possible to have a weighting for each point based on the scatter or error within that bin. For the purposes used here this was unnecessary and so the simple binary weighting format was kept.

**3.2 Parameter choices/development**

As discussed in the previous chapter there are several parameters associated with the KSOM network. $\alpha$, the learning parameter, $\sigma$, the network neighbourhood function
size, the network size and dimensionality and the number of iterations requested for the training process. There is also the choice of network layout, in this case either a square lattice structure or a hexagonal lattice. For all tests run on data sets in this thesis the square layout is chosen as the hexagonal layout appears to behave no better, is slower to calculate (in terms of the offset lookup table being more complex) and becomes very non-trivial in any number of dimensions greater than 2. Finally the number of weights, or parameters, per node is variable. The choice of value here being dependent, in the case of lightcurve morphologies, on how many points it takes to necessarily describe any full lightcurve oscillation. For instance if the size of the smallest dip is likely to be 0.02 in phase then we would want at least 50 weights per node, preferably at least double this, in order to pick up this shape for analysis (of course assuming that we are analysing the full lightcurve phase range). However it is also important not to have excessive numbers of weights per node as the execution time will scale linearly with the number of weights per node, and will scale as the square of the number of weights per node if the best-fit statistic is being calculated at every phase offset, as is the case with the lightcurves here. Where discrete parameters are being analysed then there is only need to have as many weights per node as there are analysable parameters and, obviously, no need to search at every offset (see section 3.2, Phase Sequential Weights) as this is meaningless when dealing with distinct parameters.

The alpha learning parameter can behave in several ways. It can remain fixed at whatever value it is initialised with until the end of training. Or it can decay in some predefined way, in this case either linearly between $\alpha_0$ and zero, or in an exponential fashion from $\alpha_0$ to roughly $10^{-3}$ or some other suitably small (near-zero) value.

The sigma parameter affects how large the neighbourhood function’s one sigma distance is. This is true for any form of the neighbourhood function (which may be abbreviated to NHF), though in this case the NHF of a Gaussian and the Laplacian of the Gaussian were investigated. My initial investigations lead me to try a Gaussian form with a hard cut-off point. In other words the blending did not occur across the whole map but
rather within a limited (σ) radius of the best-fit node. I quickly found that this could produce undesirable results and discontinuities in the map. Following this the algorithm was changed to incorporate all the nodes in the blend and σ became simply a reference distance to track a set number of standard deviations, σ_{SD} from the Gaussian centre.

One other form of neighbourhood function was looked at, the Laplacian Gaussian, ζ (eqn 3.1), as I felt that this may provide some way to "push away" those data not similar to the best-fit node and therefore clarify the clusters (fig 3.2 shows the Laplacian Gaussian form and the negative "repulsive" section of the function). I found that although this method would work it could produce effects in the map more due to the switching point (where the function goes negative) than the shape of the lightcurve upon the best-fit node. Therefore a plain Gaussian form was used as it is easy to understand and control.

\[
ζ = (1.0 - (r/σ)^2)e^{-(r/σ)^2}
\]  

(3.1)

The neighbourhood distance may decrease from any size to any size (in terms of the 1σ distance being a number of nodes on the network, this being the unit of measurement used when talking of distances within the network, as map space distance is a measure of the number of nodes traversed).

A range of initial and final 1σ sizes have been tried. It appears to be that if the final σ is too large then the data are not allowed to spread out so fully on the map by the final iteration and thus the full range of changes within the data may not become apparent, however if the final σ is too small then the area of the map being strongly affected by the current dataset member will be effectively limited to a single node and so the larger scale structure on the map (which produces the groupings, but not the finer changes within those groupings) may become disrupted. The final version of the algorithm has an initial 1σ distance equal to half the width of the map (in other words the maximum radius that will fit within the map-space without wrap-around interference within 1σ) and a final σ neighbourhood of 0.5 units, or nodes, allows sufficient spreading of the clusters upon the map for the datasets tested so far. Figure 3.3 is an example of a 34 iteration run of a
Figure 3.1. The Lap-Gaussian functional form showing the negative dip beyond 25 on the horizontal axis.
20x20 network with a fixed \( \alpha \), using a larger than usual final \( \sigma \). For comparison figure 3.2 is a run performed with a standard smaller final \( \sigma \).

It has also been investigated whether linear changes in the area-of-effect of the NHF kernel behave differently, or better, than those changes brought about by linearly decreasing neighbourhood distance. It was found that although there are very mild differences between the two methods (see figure 3.4) that it was the aforementioned final \( \sigma \) value which dominates the ability of the network to group. There is some concern, however, that for very fine differences between two groupings that the decreasing area-of-effect method may cause too rapid a drop in the strongly blended nodes in the final stages of learning possibly causing two very similar groups to remain unseparated. This effect, of course, increases with increasing dimensionality and so I decided to remain with a simple linearly decreasing neighbourhood. Figure 3.5 show the evolution of a decreasing area-of-effect network run.

The network size chosen depends on the number of data to be run through the algorithm. It would be inefficient to choose to have more network nodes than dataset members. Although I have found that the size chosen for the network is, like other parameters, not strongly influential on the network's ability to separate out the various data types. However, in follow-up work, namely the cluster-edge definition and identification (section 3.3) I have found that it is helpful to choose the network size such that there are a minimum of around 5 to 10 data per-node on average. In this way the underlying structure of the network is enforced enough that the number density map is sufficiently populated and that the features of the data are strongly reinforced on the map. The greatest effect related to the number of nodes being the algorithm execution time which scales linearly with the number of nodes on the network.

One other choice when considering the network parameters is the dimensionality of the network itself. The code allows, within the capabilities of the computer’s memory, any number of network dimensionality to be chosen. However tests have only been done using 1,2 and 3D networks as these are the only ones which we can easily visualise. In
Figure 3.2. An example of what occurs when a lower ($\sigma_{\text{final}} = 0.25\text{nodes}$) final $\sigma$ is chosen. As can be seen the groupings have spread more evenly during the training process and there is time for more refined differences to emerge.
FIGURE 3.3. An example of what occurs when too large a final \( \sigma \) is chosen. As can be seen the groupings do not spread well during training and the distinctions between similar data do not have a chance to separate smoothly during the learning process.
Figure 3.4. A comparison of linearly and area decreasing neighbourhoods using the ROTSE dataset (discussed in section 4.4). It can be seen from the lower plots that the areas between groupings (in red) are slightly wider for a decreasing area-of-effect than those for a linearly decreasing $\sigma$. 
Figure 3.5. An example of the network evolution using a decreasing area-of-effect NHF rather than a linearly decreasing NHF $1\sigma$ distance. As can be seen the separation of the differing data-types is very anisotropic through the training, suggesting that a truly stable and descriptive network map may not have chance to form.
general 2D networks have been used for their ease of cluster identification, being as they are like geographical relief maps in appearance. 3D maps, using OpenGL visualisation, were created but it was found that the addition of the extra dimension added nothing to the ability to separate the various data groups and merely added to the complication of interpreting the final maps.

However, the dimensionality of the network, though not a parameter which greatly affects the network’s ability to cluster, does appear to have an optimum value. Although the dimensionality of the template lightcurves used in the training process is dependent on the number of parameters, or phase-offsets, chosen to classify on, this is not the dimensionality of the behaviour within the data itself. For this value we must first consider how many effects are contributing to the shape or behaviour of the data. For example, in the case of a top hat function of unity height the parameter is simply the width of the top-hat portion of the function (assuming, as is the case in this instance, that the starting point of the rise in the function occurs at the same point in each data) and so the behaviour of the data-set is described as one dimensional. Knowing this, as would never likely be the case in a true dataset given noise and more complex behaviour, we can say that the optimum network dimensionality for these data would be 1D. The groupings will still occur in a 2D or higher network but they will take the form of Peano curves within the map space (see section 4.1.3 for examples). The 1D network would be optimal in that it would likely require the fewest number of nodes and would therefore execute most swiftly. It is for the sake of ease of representation and with which we can extract the information from the trained map that a 2D map is chosen.

Finally we come to the choice of the number of network iterations to perform during the training process. In this version of the algorithm I simply choose a set number of iterations for the learning phase. Although I have found that the number of iterations to achieve clustering is not greatly important, and that it is the choice of behaviour of \( \sigma \) and \( \alpha \) that are more relevant here, I have found that requesting a modest number (in this case 34) of iterations encourages better and more understandable behaviour in the network.
statistics (section 4.2) when analysing the data.

A Nuance of Phase-Sequential Weights

In a dataset such as the periodic lightcurves, investigated here in later chapters, where we are investigating the morphology, or shape, of the phase-folded lightcurve itself we end up with a set of related network weights. Instead of having $x$ weights we must individually compare we have a set of $x$ weights each of which are related to the next by a phase distance. In this case there is the possibility that we will not be comparing any two lightcurves and the network at the same relative position in phase. Because of this during the calculation of the best-fit node the data lightcurve must be compared at every phase offset with the templates on the network. Simply put the lightcurve is shunted along (in a cyclic manner so that anything shunted to a phase of 1.1, say, now represents that at phase 0.1. Fig 3.6) and then compared to all of the nodes (templates), shunted to the next phase-offset and compared again. This process is then repeated until it has been compared with every node at every phase-offset. This of course adds to the algorithm execution time (which will now scale as the number of weights squared rather than linearly with the number of nodes) but, as demonstrated by figure 3.7, it can save on mis-groupings of the data.

3.3 Visualisation/Exploitation

Once the network has gone through the training process it contains the grouped/clustered data stored in a FITS format file. However, to extract any useful information from this set of numbers requires the use of some form of cluster identification process. An initial method to achieve this is to look at the number counts of lightcurve data that fall on each node and plot this as a data density map. This is useful as it shows where there are high-numbers of data in one part of the network, and hence a grouping, and if there are very clear distinctions in the data classifications then this should show a very clear
Figure 3.6. An example of phase-offsetting a lightcurve. Here the lightcurve is phase-offset by 0.1.
Figure 3.7. Here we can see that it is possible for an incorrect best-fit to be calculated if phase-offsetting is not taken into account. Here we have (top) a reference lightcurve and two comparison lightcurves (middle and bottom); one of the two comparison lightcurves is a better match to the reference than the other. Without phase-offsetting the middle lightcurve would be best-matched with the first, however the real best-match is lightcurve three which is simply the original lightcurve offset in phase.
boundary, or low data number-count, between different groupings on the map. However, if there are two groups whose data members touch, in map space, then there may be no clearly distinguishable drop in the number density between the two groups. And this is where the number density map alone could fail as a post-network analysis tool. Figure 3.8 shows an example of this taken from a test done using the ROTSE periodic lightcurve dataset (discussed in section 4.4). It is apparent here that the low ratio of number of data to number of nodes produces a plot rather lacking clarity, however the application of some gentle smoothing produces a more informative plot (Fig. 3.8 lower).

Another way to visualise those data-sets whose groupings are known \textit{a priori} is to track these types and plot as a scatter diagram the network as an NxM grid. Within this grid there are only the discrete node coordinates where a network-run lightcurve may lie. If all the lightcurves that lie on that node are plotted at those precise coordinates then all we will see is the one lightcurve that was drawn last in the plotting routine. To account for this a randomised intra-bin offset is added to the lightcurve coordinate, in effect scattering the data coordinates so that we can get an idea of not only the types of lightcurve that have fallen on any one node, but also the number that have fallen on that node. This representation is a good indicator of how well the network training process has evolved to separate out the various lightcurves types. It does of course assume that the lightcurve classifications given to the data beforehand are accurate. Though it is also important to note that, even with the data-types known \textit{a priori}, the network code is not privy to such information and so the network structure is in no way biased towards these classifications. This form of plot is demonstrated in figure 3.9, again using the ROTSE dataset discussed in section 4.4. As discussed in section 4.4 as these clusterings are based around the shape of the phase-folded lightcurves the Cepheid, RRc Lyrae and \(\delta\)-Scuti type stars will group together. A brief discussion of the physical reasons for this can be found in appendix A.1.

However, with any dataset where the network is truly to be of use the data classifications are unknown and so any groupings found must originate from the information we
Figure 3.8. The top plot shows the raw number density map of the ROTSE dataset discussed in section 4.4, the lower plot shows the same result but with smoothing applied. The darker values show areas of lower density, with the colours blue to green to red show increasing densities.
Figure 3.9. This figure demonstrates the scatter plot of the ROTSE dataset discussed in section 4.4 once training is complete.
can extract from the map itself. To extract this information it's more useful to look at the underlying structure in the network itself rather than the positioning of the data on the network map. A process to look at the underlying structure is called the U-Matrix (Ultsch & Siemon 1990). This simply looks at the underlying structure surrounding the focus node and calculates a mean vector difference between the node and its surroundings. In this way if a node lies on the edge of a cluster boundary, and therefore at a point where more rapid changes in the underlying structure of the network are occurring, a larger value of the U-Matrix will occur. As a whole this leads to a visual map which can indicate the position of cluster boundaries and also the smooth changes within a cluster. Figure 3.10 demonstrates this plotting mechanism using the same data-set and results as used in previous plots. It is almost the inverse of the number density plot with the higher values present where the most change, and therefore fewest data, is occurring. It is also less susceptible to small number statistics, unlike the number density plots, as the values are determined by the underlying structure of the network itself which in turn is formed from the smooth blending of the lightcurves in the smooth continuous form of the neighbourhood function (a Gaussian form say as discussed in section 3.2). Using both of these measurements together can help in the detection of low-number groupings. In this case we often have small groupings that have been “pushed” in to the valleys in the network structure that form due to the strong reinforcement of the high-number groupings and so they make some impression in the U-Matrix values, albeit a small effect, and also a slightly noticeable effect in the number density map. Together they produce two slightly positive effects in what are usually mutually exclusive maps. This in turn when creating a point by point multiplication of the two maps can lead to the hi-lighting of any lesser groupings on the map and to their detection. It has been shown that this is possible for data-types of the order 0.5% population. This is discussed further in section 4.3.2.
Figure 3.10. This figure demonstrates the U-Matrix plot of the ROTSE dataset discussed in section 4.4 once training is complete. It indicates where areas of the underlying network structure are similar (low differences) showing where groupings appear to be, and where they are more dissimilar (high differences visible as "valleys" in the U-Matrix).
Chapter 4

Testing My Algorithm

4.1 A Quick Sanity Check

Before the network is tested using more complex and variable shapes it is sensible to perform a "sanity check" using extremely simple and easily distinguishable shapes to ensure the network is functioning as expected.

4.1.1 Noise Check

Firstly, however, one alternative check that should be, and has been, made: does the network find order where there is none? To test this the data that is read in to be run through the network training process is initialised to random values (generated by the Numerical Recipes' \texttt{ran1()} routine). This random scatter is then processed by the network, in the same manner as any other dataset, and the results checked via the U-Matrix analysis for underlying structure. Figure 4.1 shows the U-Matrix result of running such an experiment and figure 4.2 the associated scatter plots for the zeroth and final, thirty-fourth, iteration. It is obvious from the U-Matrix plot that the differences across the entire network do not indicate that any clear groups are emergent (as would be expected
CHAPTER 4. TESTING MY ALGORITHM

from a set of random numbers), a visual inspection of the scatter plot confirms this. Figure 4.1 shows the evolution of the map U-Matrix during the randomised test.

Now this shows that the network won't create a result from nothing, however it doesn't actually show it is capable of finding groupings when they do exist within the data. A suitable first test of this capability, as mentioned previously, is to run several extremely simple and distinct shapes through the algorithm and confirm that they do indeed separate.

4.1.2 Checking The Basics

Two shapes that should be easily distinguishable are a simple flat profile and a top-hat function (where half the weights are high and the other half low). For these basic checks no noise is initially added to the shapes generated. This will mean that in a set of lightcurves that there are precisely 2 distinct lightcurves and for this reason we would expect only two nodes on the network to appear populated. As the first lightcurve is blended with the network it makes an area of the network more like itself and so the next lightcurve of the same type (remembering that they are exactly the same) will fall on the same node. As this process repeats during training for the two types we will unsurprisingly end up with two poles on the network. Figures 4.3 and 4.4 show the U-Matrix and scatter-plot respectively for this basic check. As can be seen the data do fall on the two distinct nodes and the network is strongly dipoled (as further demonstrated by the U-Matrix plot).

4.1.3 The Effects of Differences Within the Data

Although the final appearance of the network is not strongly affected by the choice of network parameter behaviour (as shown in section 3.2) its final appearance can be affected by the dataset itself. For instance a taxa which is comprised solely of a top-hat function whose duty-cycle is high for 0.5 in phase and low for the rest has less variation than a
Figure 4.1. The random network run, iterations 0, 1 and 2. Below these are the mid-learning iterations 9, 17 and 26. The final two plots show the state of the network in the closing stages of learning, iterations 30 and 34 of a 34 iteration run.
FIGURE 4.2. The scatter-plot results of the noise-test performed in figure 4.1. Here showing the zeroth and thirty-fourth iterations.
CHAPTER 4. TESTING MY ALGORITHM

Figure 4.3. The U-Matrix plot results of the simple algorithm check. No noise is applied to either of the two shapes. Here showing the zeroth and final (thirty-fourth) iterations.

Figure 4.4. The scatter-plot results of the simple algorithm check. No noise is applied to either of the two shapes. Here showing the zeroth and final (thirty-fourth) iterations.

taxa whose high-state has a range of values. As we will see the size, in map-space, of any taxa on the final trained network is partly dependent on the range of variation within that taxa and partly on the number present within it. A larger variation will spread to
CHAPTER 4. TESTING MY ALGORITHM

fill a larger area of the map, whilst less variable taxa will remain confined to a smaller grouping. Figure 4.6 (discussed in more detail below) shows this effect, one data-type has a varying parameter, the other is constant. There are equal numbers of each data, but the one with the greater variation fills the larger area of the map.

Also the manner in which the data spreads itself throughout map-space appears to be reliant on the dimensionality of the variations in the data itself. For instance in a taxa comprising a square-wave shape (or top-hat function), with variations in the length of the wave's high-state, we may say that the variations are 1-dimensional (simply the value between minimum high-state length and maximum). If these data are then used to train a 2D network then the data appears to fall on and fill the network as best it can. It is a 1D variation filling a 2D space as shown in figure 4.5, and as can be seen in figure 4.7. In figure 4.5 the top sketch shows how a 1D line with 5 points may move to fill a 2D space (in this case a square), as we add more points to the line it may fill the space more completely, as indicated by the lower sketch. If we were to continue this process then with an infinite number of points on the line we could describe any point in the 2D space using a 1D coordinate (the coordinate, or distance, along the line). Thus the line is 1D, but it fills a 2D space. For this reason we may say that the curve has a dimensionality somewhere between 1 and 2 (a non-integer dimensionality) and it is this that is known as a Peano Curve (Kohonen 1990). If we now look again at figure 4.7 we can see that running data that is effectively 1D in behaviour through the network appears to exhibit this behaviour.

In comparison the flat data has remained, as expected, in its singular grouping. An inspection of the trained network's structure (Fig. 4.6 second panel) shows that the data with the range of behaviour, though it fills a larger area of the network, does not impress its structure as strongly on any one point of the network as the data with the single set of behaviour does.
Figure 4.5. The top sketch shows how a 1D line with 5 points may move to fill a 2D space (in this case a square), as more points are added the line may fill the space more completely, as indicated by the lower sketch.
CHAPTER 4. TESTING MY ALGORITHM

Figure 4.6. The U-Matrix plot results of the simple ranged algorithm check. No noise is applied to either of the two shapes. Here showing the zeroth and final (thirty-fourth) iterations.

Figure 4.7. The scatter-plot results of the simple ranged algorithm check. No noise is applied to either of the two shapes. Here showing the zeroth and final (thirty-fourth) iterations.
CHAPTER 4. TESTING MY ALGORITHM

4.2 Results with a Simulated Dataset

Now the algorithm has been tested with basic data it is sensible to move on to more complex and realistic simulated datasets. In this regard I have simulated lightcurves with the following basic shapes (see Figure 4.8):

1. a single skewed Gaussian peak (representing stars of the RR Lyrae ab-type family);
2. a single symmetric Gaussian peak (representing stars of the RR Lyrae c-type, δ Scuti and Cepheid families);
3. constant brightness with two broad Gaussian dips of equal width but differing depth (representing eclipsing close binary stars);
4. as 3 but with narrower dips (representing well-separated eclipsing binaries).

These functions are not intended to represent accurately the real lightcurves of such systems, but are merely intended to be representative of the appropriate range of behaviour.

For each synthetic lightcurve the characteristic features of its class (the width of peaks and dips, and the relative depths of the dips) were chosen randomly from within ranges chosen to yield profiles representative of those observed by ROTSE and ASAS (Akerlof et al. 2000, Pojmanski 1997, Pojmanski 2002).

Noise is added to the synthetic lightcurves to represent photometric measurement errors. The distribution of signal-to-noise ratio in the set of synthetic lightcurves is chosen to represent a source population with a cumulative distribution in flux with a slope of \(-1\) (a uniformly distributed source population would have a steeper slope of \(-3/2\)), and a signal-to-noise power of \(10\sigma\) at 16th magnitude, which is representative of the performance of current ultra-wide field survey instruments such as WASP (Street et al. 2002). In terms of magnitudes our synthetic data have the following mean brightness distribution.
CHAPTER 4. TESTING MY ALGORITHM

Figure 4.8. Samples of the four classes of simulated lightcurves, representing type-c RR Lyrae, δ Scuti and Cepheid types (top right), type-ab RR Lyrae (top left), short-period (bottom left) and long-period eclipsing binary (bottom right).
CHAPTER 4. TESTING MY ALGORITHM

Figure 4.9. The lightcurve brightness distribution for the simulated datasets.

\[ \log N(< m_v) \propto \frac{2}{5} m_v \quad (4.1) \]

The actual magnitude distribution is presented in Fig. 4.9.

Figure 4.10 shows the evolution of a typical network. In this example the \( \sigma \) parameter is initially set to half the size of the network and is then reduced linearly to one quarter the size of a single node (i.e. the one-sigma full-width of the smoothing Gaussian is reduced from the full size of the network to half the size of a single node), which for the remainder of this section will be referred to as a neuron. The \( \alpha \) parameter is held constant at 0.01 throughout. The full run consisted of 34 iterations as this appeared to be sufficiently long to fully display the network statistic behaviour and the formation of the map. Figure 4.10 shows every third iteration.
Figure 4.10. Evolution of the network trained with $\alpha(t) = \alpha_0 = 0.01$. Panels show (from left to right, top to bottom), iterations 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34. The four classes of synthetic lightcurve (described in Section 4.2) are denoted by (i) squares, (ii) crosses, (iii) diamonds and (iv) stars. Note that for the purposes of visualisation the map space coordinates are dithered within the area occupied by each neuron. The lower plots show examples of lightcurves who, though of different types, appear close in map-space (as expected the double peaked lightcurve, third panel, has a low signal to noise).
CHAPTER 4. TESTING MY ALGORITHM

It can be seen in Fig. 4.10 that the 5000 lightcurves initially form two tight clusters. This is because the synthetic lightcurves naturally divide into two basic types (single and double peaked) and because the large smoothing kernel does not allow the network to form sufficient structure to resolve the lightcurve sub-types. The lightcurves are tightly clustered because the large smoothing length forces the clusters to strongly repel one another. Weak cross-shaped structures are also apparent, and are a natural consequence of the wrapping of the Gaussian kernel in both dimensions. As the network is evolved, and the kernel size is decreased, the network becomes increasingly free to develop substructure which better represents the range of shapes of the synthetic lightcurves. Between iterations 13 and 16 one of the initial clusters splits into two (the contact and detached eclipsing binaries), and between iterations 13 and 22 the other initial cluster also gradually splits (the symmetric and skewed Gaussian shapes). By iteration 22 the map well represents the four types of lightcurve. As the kernel size continues to decrease the clusters repel each other less strongly and begin to spread to occupy more of the available neurons. This is because sharper differences in the template lightcurves are now allowed between neighbouring neurons. The increasing size of the clusters allow them to better describe the range of sub-structure within each type of lightcurve. Smooth variations of lightcurve properties are seen across the individual clusters. Finally, as the smoothing kernel decreases below the size of an individual neuron, neighbouring neurons approach decoupling and it is possible for the clusters to make contact. At this point the network has evolved to the point that all of its neurons are representing subtly different lightcurve shapes, and learning is complete for this dataset.

The learning behaviour of the maps motivates the tracking of three measures which allow me to quantify the degree of re-organisation which is taking place at a given time, and the quality of the match between the template lightcurves and the training set members assigned to them.

The first of these measures (\( \bar{S} \)) is the value of \( S_{i, \text{min}} \) averaged over all members \( i \) of the training set. The statistic \( S_{i, \text{min}} \) can be regarded as a measure of the goodness-
Figure 4.11. The average "goodness of fit", $\bar{S}$. In the leftmost column the learning parameter $\alpha$ is held constant throughout the learning phase. In the central column $\alpha$ falls linearly, and in the rightmost column exponentially. The initial value of $\alpha$ assumes the value 0.9 (top row), 0.5 (second row), 0.1 (third row) and 0.01 (bottom row).
Figure 4.12. The number of training set members that move from one iteration to the next, $n_{move}$. Panel order is the same as Figure 4.11.
Figure 4.13. The average Euclidean distance moved by a training set member in map space from one iteration to the next, $\bar{D}$. Panel order is the same as Figure 4.11.
of-fit between a training set member and the template lightcurve of the network neuron $c$ which most closely matches it. By averaging over all members of the training set I can therefore measure how well the template lightcurves of the network as a whole have adjusted to represent the input data. I expect $\bar{S}$ to remain high during early iterations, to show a rapid fall once the network becomes free to represent all the lightcurve sub-types as separate groups, and then to exhibit a slower decline as finer adjustments are made to the template lightcurves.

The second measure ($n_{\text{move}}$) is a count of the number of training set members that have changed location in map space from one iteration to the next. I expect maximum movement at times of maximum learning.

The final measure ($\bar{D}$) is the average Euclidean distance moved by a training set member in map space from one iteration to the next. Again, I expect maximum movement at times of maximum learning.

Figures 4.11, 4.12 and 4.13 show the evolution of $\bar{S}$, $n_{\text{move}}$ and $\bar{D}$ as networks are trained using the synthetic lightcurves described above. Several training runs were made, differing only in the initial value chosen for $\alpha$, and the functional form with which $\alpha$ evolves. Three functional forms for alpha have been chosen: a constant value $\alpha(t) = \alpha_0$, a linear form $\alpha(t) = \alpha_0(1 - t)$, and an exponential form $\alpha(t) = \alpha_0 e^{-\lambda t}$, where the scaling constant $\lambda$ is chosen such that $\alpha = 10^{-3}$ in the last iteration.

Taken together these three figures illustrate that the ability of the map to robustly self-organise is not critically dependent on the choice of either $\alpha_0$ or the functional form of $\alpha(t)$. In particular Fig. 4.11 shows that the template lightcurves adjust to faithfully represent the input data in all cases. All panels in Fig. 4.11 show a monotonic improvement in the fit statistic that begins relatively slowly while $\sigma$ is high, becomes more rapid when $\sigma$ reaches a critical value that allows all types of lightcurve to be represented by their own grouping in the network, and finally slows as the maps approach a steady state at the end of the learning process. The final value of $\bar{S}$ is similar in all cases except in Fig. 4.111. This run refers to an exponential decrease in $\alpha$ from 0.01 to 0.001. It is clear
from this panel that $\alpha$ can drop too fast too soon, thereby "freezing in" structure in the network and prematurely ending learning. For this reason I favour a constant $\alpha(t)$ that allows equal learning at all stages of the learning process. I also do not use $\alpha_0$ values substantially below 0.01 (although lower values may be acceptable provided the map is given sufficient time to evolve into equilibrium).

Figures 4.12 and 4.13 reveal more details of the learning process. Once again, most panels show broadly the same behaviour, with most learning occurring during the middle of the run, and less at the beginning and ends. It is clear, however, that movement of lightcurves within the map is a function of $\alpha$. For $\alpha_0 \lesssim 0.2$ the basic structure of the map forms early and the groupings of lightcurves become fixed in map co-ordinates, even in the cases where $\alpha$ is held constant throughout the learning process. For larger values of $\alpha_0$, however, it is apparent that although large-scale structure does indeed form, that the cluster pattern as a whole continues to migrate rapidly through map space (Figure 4.12a, 4.12b, 4.12d & 4.12e). In a real-world application this mobility and constant restructuring would quite likely prove to be undesirable. For this reason I prefer to choose values of $\alpha_0 \lesssim 0.2$.

The rapid learning apparent in the middle iterations in Figs. 4.11, 4.12 & 4.13 reveals the presence of a critical value of $\sigma$ which allows the network sufficient freedom to well represent all classes of lightcurve. This critical $\sigma$ is likely to take a different value for different datasets (more classes would require more freedom and a smaller $\sigma_{\text{crit}}$) but nevertheless, in a real-world application, one might choose to save computing effort by starting $\sigma_0$ at a value closer to $\sigma_{\text{crit}}$.

There is evidence of some misclassification in the final trained network in Fig. 4.10, for example between skewed and non-skewed Gaussian profiles. This is not unexpected as some members of the skewed Gaussian class will exhibit a very small degree of skewness, and are difficult to distinguish from the non-skewed class members even when inspected by eye. Misclassification is difficult to avoid in any situation where there is a smooth trend in variation of lightcurve shape between two classes, emphasising that
in real-world applications shape-based classification would be used alongside other di-
agnostic attributes. Nevertheless as a whole the algorithm has performed remarkably
well.

4.3 Variations in the Simulated Dataset

Now the behaviour of the various states of a fairly typical dataset has been examined it
seems only sensible to see how the algorithm behaves when presented with less “friendly”
data, or rather less evenly distributed data. This is performed by examining the same
underlying data morphologies with varying amounts of noise applied and by the investi-
gation of the network’s ability to separate out under-populated taxa, or data-types. And
additionally whether we are able, with no a priori knowledge of the classifications in-
volved, to locate these underpopulated clusters via the usual U-Matrix and density maps
(section 3.3).

4.3.1 The Effects of Noise

Depending on the optical quality and photon-gathering power of the telescopes used to
perform the surveys that are responsible for gathering the wealth of lightcurves now
available for analysis, the data we have may have varying signal to noise ratios. To test
the network’s performance with varying levels of signal to noise a number of datasets
were created. The dataset analysed in the previous section, as mentioned previously,
has a signal to noise of 10σ at a simulated 16th magnitude. However, to truly test the
network’s ability to deal with varying levels of noise several datasets were created with
single signal to noise values. In effect the lightcurves created were all set at 16th mag-
nitude and with the same noise level throughout the data. The following signal to noise
values were tested: 1σ, 1.5σ, 2σ, 3σ, 5σ, 7σ and lastly 10σ. Figure 4.14 shows the
scatter plot for a selection of these. It is quickly apparent from both the scatter plots and
CHAPTER 4. TESTING MY ALGORITHM

U-Matrix plots (figure 4.15) that the network can sort data at very low signal to noise. But also, from the U-Matrix plots, that the groupings are clearer around the $3\sigma$ level than below. We would expect no real groupings to occur at $1\sigma$ levels as this is effectively just noise.

As can be seen from these plots at $3\sigma$ the network is quite capable of separating the 5 types of variation with very little overlap (the same types of lightcurve as in section 4.2 with the addition of a very simple box function, visible as the cyan points in figure 4.14). The majority of mis-groupings happening within the singly or doubly peaked "super-groups". It is clear that the box-function is distinct enough from the other singly-peaked data so as not to be confused with it, this is as expected as there is not the blend between skewed and non-skewed Gaussian peaks as with the other data (and as mentioned previously in section 4.2). Beyond $3\sigma$ the network training process very clearly separates out the 5 types of variation.

In summary the KSOM algorithm continues to show itself as a remarkably robust clustering mechanism suitable for the analysis of even quite low-quality data and therefore very applicable to those datasets whose data gathering may be more concerned with time series resolution than photometric accuracy.

4.3.2 Underpopulation of a Taxa

How the algorithm behaves when presented with a data-set containing underpopulated, or indeed overpopulated, taxa is something very relevant to the study of large data-sets. When searching for the classifications and groupings of data within these large data-sets there is always the chance that we will locate some new phenomena, or wish to search for some uncommon occurrence. These will both be represented by data that is relatively uncommon and therefore may not have chance to impress its structure upon the training map.

There are two stages to test the underpopulated taxa finding ability of the algorithm,
Figure 4.14. The scatter-plots for the final state of the network using the various simulated datasets. From the left to right and moving down: $1\sigma$, $1.5\sigma$, $2\sigma$, $3\sigma$, $5\sigma$ and $10\sigma$. Where the light-blue triangles denote the box function, the purple dots the gaussian curve, the black crosses skewed gaussians, the purple crosses the double broad gaussian dips and the dark blue diamonds are the double narrow gaussian curves.
CHAPTER 4. TESTING MY ALGORITHM

Figure 4.15. The U-Matrix plots for the final state of the network using the various simulated datasets. From the left to right and moving down: 1σ, 1.5σ, 2σ, 3σ, 5σ and 10σ.
firstly does it manage to group the underpopulated data together on the map, and sec-
ondly can we locate this grouping given no prior knowledge of the groupings and data
types?

The tests comprise of 5 varied data-sets, each with four different types, or taxa, of
lightcurve. One data-set has taxa with equal populations (as in the previous section’s
tests) and one dataset has one overpopulated taxa. The remaining three data-sets have
one taxa which is underpopulated with respect to the other taxa. All the data-sets are
made up of 5000 lightcurves with the same magnitude and signal-to-noise distribution
as discussed in section 4.2. The taxa with the variable population (the rest of the data
being divided equally between the three remaining taxa) have the following percentages
of overall population: 0.1%, 1%, 2%, 5%, 10%, 25% and 90%. The 25% population is
effectively the same as the data-set used in section 4.2 and is here for comparison only.
The 90% population shows us what may happen when the data are dominated by one
type.

When dominated by one type of data, such as the 90% population, it appears that
the network will alter the majority of itself to reflect this type, as expected, but that it
will still allow the formation of regions relating to the other data types on the network,
albeit on a much reduced scale. Figure 4.16 shows the U-Matrix and scatter plots for this
overpopulated run.

Figures 4.17 and 4.18 show the usual network scatter and U-Matrix plots for the var-
ious underpopulated investigations. It is clear from both visual inspection of the scatter
plots and the appearance of the U-Matrix plots that for populations of 2% and greater
there is sufficient presence from the underpopulated taxa to impress its features upon the
map to a recognisable level, however for the 0.1% and 1% populations the groupings are
much less clear.

Now, if we take only these two very low population runs and look at their U-Matrix
and density plots side by side (Fig. 4.19) we can see that where one has a high value (in
the case of the U-Matrix a place of more rapid change on the network) the other tends
CHAPTER 4. TESTING MY ALGORITHM

Figure 4.16. The U-Matrix and scatter plots for the 25% and 90% population tests. The scatter plots have the same key as figure 4.14.
Figure 4.17. The scatter plots for the various populations of data. From the left to right and top down: 0.1%, 1%, 2%, 5%, 10% and, for comparison, 25%. The scatter plots have the same key as figure 4.14.
CHAPTER 4. TESTING MY ALGORITHM

Figure 4.18. The U-Matrix plots for the various populations of data. From the left to right and top down: 0.1%, 1%, 2%, 5%, 10% and 25%.
CHAPTER 4. TESTING MY ALGORITHM

98

... to have a lower value (where there is rapid change in the network's underlying structure there tend to be fewer data and so the density map has lower values). However, where we have the underpopulated data types they do not have sufficient strength to greatly change the underlying structure of the map, but they do have enough similarity to change one area of the map enough to best-fit themselves to one point on the network, and therefore where there would usually be a very low, or zero, number-count of data on the density map we have a small cluster. If we now convolve the two maps together, that is the U-Matrix and density maps, we are left with a final map, which we will call the Peculiarity map. Figure 4.20 shows the appearance of this mapping for the 0.1% and 1% population runs.

Although this method of a Peculiarity map (or P-Map) is not certain to highlight those groupings lost in the standard mappings it is a good place to start should you wish to search for the rarer types of taxa. Using this method one could define a cut-off peculiarity point above which a manual, or alternative automated, inspection of the data on the peculiar nodes is carried out. As can be seen from the 1% population in figure 4.20 the P-Map does a fine job of extracting the peculiar nodes, it is simply the most peculiar node on the map. However the 0.1% mapping produces a P-Map with around 4 high peculiarity nodes (of which the 0.1% population is one) and so in this case would require further investigation of the 4 nodes to locate the 0.1% population more surely.

These results show that the combination of the KSOM and various data-extraction techniques such as the U-Matrix and P-Map do allow for the extraction of extremely low populations of data and so are very viable techniques in the search for astronomical phonemena in the wealth of data now collected by modern instruments.

4.4 Real-World Test: Results with the ROTSE Dataset

In order to test the classification abilities of the self-organising map using real data I have chosen to train it using lightcurves from the ROTSE experiment. This training set
FIGURE 4.19. The U-Matrix and density maps for the 0.1% and 1% underpopulation tests.
CHAPTER 4. TESTING MY ALGORITHM

Figure 4.20. The scatter-plots and Peculiarity maps (P-Maps) for the 0.1% (left) and 1% (right) underpopulation tests. The outlying cluster of the underpopulated taxa is clearly visible as the dark red point in the 1% P-Map and as one of several "peculiar" points in the 0.1% map.
CHAPTER 4. TESTING MY ALGORITHM

consists of 1206 periodic lightcurves which have been independently pre-classified by other means (Akerlof et al. 2000). The externally derived classifications are not used in training my network, however they do allow me to more definitively assess the ability of the algorithm to group and classify class members after training is complete. This will, of course, act simply as a determination of the classification ability of the KSOM in relation to other techniques. But for the purposes of a general idea of how well the algorithm performs in a real-world test these classifications are sufficient.

Figure 4.21 shows the final state of a network trained using the ROTSE lightcurves. All parameters are set to the same values as the run in Fig. 4.10, in particular $\alpha$ was held at a value of 0.01 throughout. It can be seen that my algorithm has successfully differentiated the contact binaries, the detached eclipsing binaries, and the type-ab RR Lyrae systems. The remaining three classes in the sample, Cepheids, $\delta$ Scuti and type-c RR Lyrae stars, possess very similar lightcurve profiles (indeed the variability is driven by the same underlying physical process, see section A.1 for details), therefore the network has difficulty distinguishing between the classes on shape alone. I can readily separate these groups, however, through the addition of other diagnostic attributes. Figure 4.23 shows the map of Fig. 4.21 collapsed in one dimension and expanded with the period of the variable star. The $\delta$ Scuti, Cepheid and type-c RR Lyrae variables, which formed an overlapping group in Fig. 4.21, have been separated in period with clearly visible cut-off boundaries for each type. Indeed this is to be expected as their ROTSE classifications are based partly on their period; however, it does show that should we wish to use the KSOM as a first-step classifier then it can lead to data which can be sub-classified, based on other parameters, at a later date.
Figure 4.21. The final structure of a network trained using 1206 pre-classified ROTSE lightcurves.
CHAPTER 4. TESTING MY ALGORITHM

Figure 4.22. The final iteration of the ROTSE data map (shown in Fig. 4.21) when processed with the U-Matrix algorithm. The colour scale runs from dark blue (similar reference vectors) to red (dissimilar reference vectors).
Figure 4.23. The ROTSE map of Fig. 4.21 collapsed in one dimension and expanded with the period of the variable star. Here the δ Scuti, Cepheid and type-c RR Lyrae variables, which formed an overlapping group in Fig. 4.21, have been separated in period with clearly defined cut-off boundaries for each type.
Chapter 5

Application to Exo-Planet Transit

Lightcurves

5.1 The Task

There are a number of methods for locating extra-solar planets (also referred to as exoplanets) and discerning information about them. In this chapter I shall outline some of those methods currently available and how we might distinguish a transiting planet's lightcurve from that of a stellar binary system. It is primarily due to the limits of today's telescope hardware that the vast majority of exoplanets located until this point have been of the order of a Jupiter mass (as opposed to Earth mass planets). When future observing missions are run with more precise instrumentation we may then begin looking for those planets closer in size and density to Earth and thus those that may be more suited to supporting life as we know it. But every journey must begin somewhere and so our current goal is to locate and try to understand the larger planets we find around other stars via any method we can.
5.2 Methods of Planet Finding

The following section summarises the main methods used in observing exoplanets. Covered are: The Doppler method, gravitational lensing, pulsar timing, direct imaging and transits. As these exoplanets are currently so difficult to observe all the methods presented have merit. However I will explain how in the near future that it is with the transiting method that the vast majority of Jupiter-mass exoplanets will be found.

5.2.1 Pulsar Timing

There are a number of objects that could be classified as an exoplanet, the primary prerequisite simply being that the object is orbiting a star and is not undergoing thermonuclear reactions in its core (if it were it would be a star, or a brown dwarf, see section A.2). For this reason planet-like satellites orbiting pulsars may also be included in the planetary search. As pulsars are extremely accurate clocks, their pulse timing being effectively constant between pulses (with the exception of jumps due to occasional seismic activity) we may measure the changes in frequency due to the same effect as causes the doppler wobble as mentioned in section 5.2.2 below. As the planet orbits the pulsar it offsets the system’s centre of mass, with respect to the star’s centre of mass, and so the pulsar wobbles on its axis. As it does so it is moved slightly closer and then further away with respect to the Earth. This means that the distance for the light emitted by the star to travel to the observer changes as the planet rotates around the pulsar. Therefore the time between pulses changes a perceptible amount. From this we can calculate the mass of the secondary object and therefore infer as to whether it may be a planet. It is worth noting that with current technology this is the only method sensitive enough to detect Earth-mass exoplanets.

Since the discovery of pulsars in 1967, by Jocelyn Bell-Burnell, scientists searched for evidence of planetary bodies in orbit around them. However, it wasn’t until 1992 that Wolszczan and Frail discovered the first satellite objects orbiting pulsar PSR B1257+12
CHAPTER 5. APPLICATION TO EXO-PLANET TRANSIT LIGHTCURVES

(Wolzszcan & Frail 1992). This was the first discovery of a planet-mass object not orbiting the Sun.

5.2.2 The Doppler Method

This method of exoplanet hunting works in the same manner as is used for finding binary star systems’ orbital parameters. As the planet orbits the star it offsets the system’s centre of mass slightly, with respect to the star’s centre, and so the star appears to “wobble” (see figure 5.1) around the centre of mass. In fact both objects orbit around the common centre of mass of the system, but this is often only slightly offset from the star’s own centre of mass/rotation and so we see the star’s orbit merely as a “wobbling”. This wobble causes a periodic red/blue-shift in the atmospheric lines. By taking spectra of the target star the absorption lines from various elements may be found. Further, by measuring the movement of these absorption lines, and how long they take to repeat their cycle, the orbital period of the planetary object can be calculated and a minimum mass for the exoplanet can be found based on the strength of the doppler shifts. We can only find the minimum mass as this measurement is based upon the amount of oscillation we detect, if we see the system edge-on then we can calculate the true mass of the planet. However if, as would be more likely, we are more unlucky and see the effects of oscillation at an angle then our mass estimate is a minimum, the measured mass, $M_{\text{measured}}$, being related to the angle of inclination of the system with respect to the observer, $i$, by $M_{\text{actual}} \sin i$.

So far the vast majority of exoplanets have been located using this method as it is currently the most sensitive method available. It is, however, very slow without very specialised red-shift measuring apparatus (such as that used by the 2dF survey, R.Cannon 1995). Unlike a sky-survey, which may take photometric measurements of hundreds or thousands of stars in one go and many times per night, and so may get usable lightcurves for those stars in one or two nights of observing, this spectroscopic method allows very few (of the order 5) measurements to be made per year (Cumming 2004). However, this method’s main limiting factor is simply the orbital timescale for the plan-
Figure 5.1. The Doppler induced wobble of a parent star by a planet causes a shift in the observed spectral lines. $dL$ is the change in wavelength of the received light and $V$ is the velocity of the star with respect to the observer.
ets. For example, were aliens trying to observe Jupiter via a radial velocity survey, and beginning like us back in 1995, they would not yet have discovered it!

This is also the method by which the first extra-solar planet around a solar-type star was discovered back in October 1995 (Mayor & Queloz 1995), with follow-up confirmation provided shortly after (Marcy & Butler 1995). Further work has found more than 100 planets via this method and so it has proven very successful. However, due to instrumentation limits it is unlikely that this method will ever be used to find the Earth-mass planets, to quote Geoff Marcy (Marcy & Butler 1998):

*An Earth-mass planet at 1AU induces a reflex velocity of only 10cm s^{-1} in the star, as opposed to our current precision of 3ms^{-1}. I believe that Earth-like planets will remain undetectable by the Doppler technique. The reason is that the photospheric granulation and turbulence in G and K dwarfs seems to result in a fluttering in the net Doppler shift from a star. The slowest rotators exhibit a rock-bottom quiescent photosphere at a level of 2ms^{-1}, similar to the Sun. Many measurements would be needed to beat down that turbulent "noise".*

### 5.2.3 Gravitational Lensing

Another successfully applied method is that using the phenomenon of gravitational lensing. When a massive object passes in front of another object it can distort space in such a way that it acts like a lens in space and causes the amount of light received from the distant object to be magnified. In doing so anything that is in the direct line of sight and further away than the lensing star will appear to get brighter. Therefore if a foreground (lensing) star passes in front of a background star the amount of light received from the background object is focused by the gravitational lens, and this will produce a smooth brightness peak in the lightcurve (see the top sketch in Fig 5.2.3). However, if a planet is present around the lensing star then there will be what is effectively a secondary lens adding to the magnifying power of the star and the smooth appearance of the microlens-
ing lightcurve will be affected, as in Fig 5.2.3 (lower).

The OGLE (Optical Gravitational Lensing Experiment (Udalski et al. 2002)) collaboration is at the forefront of this field currently, having done a number of years of observing whilst searching for MACHO (MAssive Compact Halo Objects) microlensing events. There is also the MOA (Microlensing Observations in Astrophysics) collaboration (Bond 2001). Although only one planet has currently been found via microlensing by both OGLE and MOA (Bond et al. 2004), it has very successfully found planet-candidates via the transit method (discussed below in section 5.2.5).

5.2.4 Direct Imaging

In those systems where there is a low contrast ratio between the parent star and planet it can become possible to image the planetary body directly. A good candidate for this type of planetary search are white dwarfs (WDs), which are the remnants of previously massive (and relatively luminous) stars, and are around 10000 times fainter than their progenitors (Burleigh et al. 2002). There are several benefits to this method; firstly there are around 120 easily probed white dwarfs (within 20pc) and secondly this method probes a different section of parameter space than experiments such as the radial velocity surveys which generally do not target massive stars (the A and B type star progenitors of the WD) and therefore ignore planetary formation frequency in these systems.

This technique is executed by taking images of the star being investigated in two epochs. The first provides us with an idea of those objects apparently close to the WD being investigated, and the second to confirm whether any of those astrometrically close objects have the same proper motion as the WD and are therefore likely companions.

5.2.5 Transits

Lastly there is the photometric method which has been used in the surveys, outlined in previous chapters, such as WASP, TrES (Trans-atlantic Exoplanet Survey) and ROTSE
Figure 5.2. Sketches of a cleanly microlensed lightcurve (top) and a microlensing lightcurve with a planet present around the lensing star (bottom), the planet acts like a defect in the lens. $B$ denotes the relative brightness.
which relies upon photometric variations, or periodic oscillations in the lightcurve of the target stars. When a planet passes in front of its host star with respect to the Earth it will eclipse a small portion of that star's face, when this happens the light emitted by the star will seem to diminish and a lightcurve with a characteristic dip will be created (Fig 5.3). This method of planet hunting has been utilised by the WASP and OGLE transit search experiments. In the case of OGLE this is a rather useful alternate project (producing over 130 transit candidates already) to the lensing search made possible by the vast amounts of photometric data their experiments have gathered whilst searching for lensing events. The lightcurve produced may be affected by many things, such as the size and opacity of the planet's atmosphere, the size of the planet in relation to the star or the angle of inclination of the planet's orbit with respect to the Earth. An additional point, which may call in to question candidates found via this method, is the type of the star around which the transit is found. In more chromospherically active stars, such as K-stars, the large amounts of variation, or surface granulation, caused by this activity can create results which may be difficult to confirm. The transit signal becoming lost within the variation caused by the stellar activity.

The first planet to be observed by this method was HD209458b (Henry et al. 2000, Charbonneau et al. 2000). Later discoveries include a number of OGLE transit candidates and a recent discovery by the TrES collaboration (Alonso et al. 2004). The TrES-1 discovery is exciting as it is only the second planet found via this method which is around a star bright enough to allow a variety of follow-up analyses to be performed (Sozzetti et al. 2004).

One further problem associated with transit detection is that a binary eclipsing star system, whose light is diluted by another star occupying the same pixel or within the PSF (Point Spread Function), may have a lightcurve very similar in appearance and relative flux change to that of a transit. With a clean, low noise, lightcurve there are differences between the two (see figure 5.4), with the stellar binary appearing more triangular in shape and the transit having differences in the edges of the eclipse and a flatter lightcurve.
Figure 5.3. A planetary transit produces a very subtle, but distinctive, lightcurve with steep ingress and egress and a near flat central region. Unfortunately most of the subtle distinctive features are lost with addition of even small amounts of noise.
Figure 5.4. Examples of a high signal-to-noise transiting planet lightcurve (top) and a binary star lightcurve.

during eclipse. However, with increasing noise (figure 5.5) these subtle differences can become lost and so the two types of lightcurve may become confused.
Figure 5.5. An example of a relatively low signal-to-noise transiting planet lightcurve (top) and a binary star lightcurve.
To distinguish between the transits and non-transits can require a large amount of followup work, this can be expensive both in time and money, and so it is desirable to reduce the number of candidates via some means beforehand, this is where methods such as the KSOM become useful. To actually locate those lightcurves which may be of interest and worth a more detailed analysis there are several methods available, of which the two most successful are presented briefly below.

These approaches rely on the identification of the central box-like dip and very subtle characteristic differences in the transiting section of the lightcurve in order to distinguish the exoplanet signals from those created by grazing eclipses or other binary star systems.

The Box-Fitting Technique

The application of the box-fitting algorithm to the search for periodic transits by exoplanets has been investigated by Kovács et al (2002).

The method basically revolves around a \( \chi^2 \) fit of a square-well transit model. Thus the method requires us to explore parameter-space in order to find the square-well transit lightcurve that creates the best fit to the data; however, one free parameter may be removed, and thus the analysis sped up, through the assumption that the average of the signal is zero (Kovács et al. 2002). This effectively means we need only fit the box section of the trial lightcurve as we are assuming the majority of the lightcurve is constant at some level, and thus we can save on computational load. This of course means any further sources of variation are ignored. The parameters may be further reduced by normalisation of the square-well (transit) depth (Kovács et al. 2002).

The Matched Filter

Similar in nature to the box-fitting technique the matched filter works by comparison of a model lightcurve, or filter, with the proposed transit lightcurve. There are a number of filters generated that as a whole fill the parameter space being investigated. In the case of
planetary transits this may involve parameters such as transit depth, total transit duration, duration of transit rise/fall and duration of the full-planet transit. Next, instead of solely a $\chi^2$ comparison of the models, or filters, with the data lightcurve, the best-fit model that this produces is compared to a best-fit constant magnitude. This aids the scientist in seeing how cleanly the transit model actually fits the data and how likely it is that the "signal" identified is simply an artifact of an already noisy lightcurve. Street uses this method to isolate transit candidates in a field around the open cluster NGC 6819 (Street et al. 2003). In 38000 stars observed in this field they found 11 transit-like candidates worthy of follow-up observations. Given the difficulty in what is being undertaken in identifying such small signals these are pleasing first steps and pave the way for current and future survey searches such as WASP (Street et al. 2002).

5.2.6 Future Techniques for Planet Observation

In the not too distant future space probes such as SIM (the Space Interferometry Mission) and TPF (the Terrestrial Planet Finder), part of the NASA Origins program (NASA Origins Program 2004), and the ESA equivalents GAIA and Darwin, will search for even fainter objects than we do currently. In an attempt to find Earth-sized rocky planets these probes are composed, in the case of the TPF, of an array of 3-4m IR telescopes arranged into a large 40m interferometric array, much like the current-day ground-based radio telescope arrays the VLA and the VLBI, that orbit in formation in space. In the case of SIM the observations will be performed across an optical waveband; its main objective to accurately measure the distance to all bright objects via precise parallax measurements (4\(\mu\)as accuracy on position to a 10% accuracy level out to a distance of 25kpc). To quote the NASA Origins site:

"... SIM would be able to see the grass grow in your yard every second, from as far away as 10 kilometres..."

In turn this phenomenal accuracy means that the astrometric amplitude, “wobble”
of the star, can be directly measured and so stars can be surveyed for the presence of orbiting companions and the orbital dynamics they induce.

With such huge baselines and technological advances, and without the effects of the Earth's atmosphere to hinder the photons reaching the detectors, these probes should be able to provide us with much more accurate information about the planets around other stars. With the data these probes will be capable of gathering come a myriad of new possibilities and chances for discovery meaning that the following decades should indeed be an exciting time for the planet-hunter.

5.3 Transit Separation Using the KSOM

Having found transit candidates, we need to do everything we can to separate the most likely transits from possible eclipsing binaries. My idea is to use the KSOM to distinguish these subtle shapes automatically.

This section is concerned with the behaviour of the KSOM algorithm when presented with various datasets containing transit-like lightcurves. As hinted at previously, our motivation for doing this is a hope that any method which can reduce the number of false-positive transit candidates, and yet still find a sufficient fraction of true transits, provides us with science which is more economically followed up in both confirmation telescope time, and hence money, and also in scientific investigation time.

In order to test the algorithm a number of datasets of varying complexity have been presented to it. In the first instance simple simulated transits and stellar eclipses with varying amounts of signal-to-noise have been analysed, this is followed by a more varied set of simulated lightcurves, with a distribution of signal-to-noise, and finally a number of the OGLE exoplanet candidates have been run through the algorithm.

The transit lightcurve simulating software for this section was very kindly provided by Daniel Rolfe.
5.3.1 The Simulated Data

In order to test the ability of the KSOM to distinguish between grazing (and diluted) stellar eclipses and transiting planets a set of simulated transits and eclipses was created to be run through the code. The main parameters which affect the lightcurve shapes, and therefore can result in an eclipse which appears like a transit, are the masses of the objects involved (and as a consequence their radii) and the inclination of the orbital plane with respect to our viewpoint. In order to mimic this behaviour a number of lightcurves have been generated with a range in orbital inclination and in secondary body masses. In addition to this the stellar eclipsing lightcurves were diluted so as to limit their relative reduction in flux due to the eclipse, this is to simulate the effects of another, constant, star which may occupy the same pixel in the sky image as the binary star and hence cause saturation of its lightcurve making it appear similar to a transit. The goal here is to see how well the KSOM separates out the two types of object and whether it can do this as well as the human eye.

Figure 5.4 contains an example of a high \((20\sigma)\) signal-to-noise lightcurve generated for the test and figure 5.5 a lightcurve with lower \((3\sigma)\) signal-to-noise.

5.3.2 Generating the Simulated Data

The arbitrary assumption has been made here that the generated planets are all of the same density as is believed for HD209458B (Sullivan & Sullivan 2003). The relative radii and masses have been scaled accordingly. The atmospheric height is scaled linearly with the planetary radius and the atmospheric scale height is constant throughout the sample both for simplicity and because any effect provided by it is quickly lost within the noise; in a more detailed and accurate study a more completely descriptive set of behaviour for the planetary parameters could be explored.

As stated above the directly varied parameters are those of the orbital inclination, with respect to our viewpoint, and the secondary body, or planetary, mass. The orbital
inclination was varied in 0.5° steps between 90° and 80°, and the secondary body radius in 0.1\(R_J\) steps between 0.5\(R_J\) and 1.9\(R_J\). The planetary density was calculated from values similar to those believed to be the radius and mass of HD209458b: \(M = 0.69M_J\) and \(R = 1.37R_J\) (Sullivan & Sullivan 2003). This gives a rather low density planet, but one that is within the capabilities of today’s telescopes to detect. As an interesting aside it has been shown (Sozzetti et al. 2004) that there does appear to be a trend between planetary mass and radius, but that HD209458b appears as a rather low density outlier on this plot. The orbital period was set to 3.5 days for all data.

The binary systems (or double star systems) have been generated using the same code, but with a secondary body of the same radius and mass as the Sun. In this case the atmospheric height is set to the same as the secondary body radius (zero atmosphere). As the two stars are of the same relative luminosity, and the scale from the central star is between one and zero, an average is taken between the value of the eclipsed and the eclipsing stars. From this the relative dips in the lightcurve for the system is calculated. For these objects only the orbital inclination was varied.

In all cases the central star object is set as \(1R_{\odot}\) and \(1M_{\odot}\).

### 5.3.3 Utilising the Simulated Data

Distinguishing a transit lightcurve from that of a binary star system is, like many astronomical problems, relatively simple at high signal-to-noise, but becomes rapidly more difficult as the data becomes more noise-dominated and the subtle differences between the objects is lost.

To test the KSOM’s ability to separate true transits from stellar eclipses several tests have been performed. Firstly several versions of the generated dataset (which consists of ranges of parameters as outlined above and equal numbers of transits and eclipsing binaries) have been run through the network, each having a different signal-to-noise ratio (though within each dataset the signal-to-noise is constant); secondly the dataset
has been used with a spread in signal-to-noise; thirdly the same dataset has been run through a pre-trained network, the pre-training being done using a low noise dataset; finally the behaviour of the network using the real OGLE transit-candidate and eclipsing binary data has been investigated.

In all of the simulated-data tests presented below only a fraction of the lightcurve is being investigated. Instead of looking at the full 0 to 1 phase plot, only that part of the lightcurve where a dip in the brightness occurs (plus 10% either side in order to provide some background “zero” level) is placed through the algorithm. This allows for the direct comparison of all the lightcurves on their profile, which is both a good test of concept and also not unreasonable (as had the transit candidates been found by some form of box-fitting or matched-filter algorithm then we would have some idea of the size and position of the eclipsing portion of the lightcurve).

### 5.3.4 Results with the Simulated Data

#### Varying Levels of Signal To Noise

An interesting test for these very similar data-types is to investigate at what level of noise the network can no-longer distinguish clearly between transits and eclipses. Not only in terms of whether the network can separate these two types of lightcurve in map-space, but also at what point it becomes possible to distinguish these classifications via the U-Matrix; thus at what point it should be possible to determine the two classifications with no *a priori* knowledge.

If we once again look at the two lightcurves from figure 5.4 we can see that there is a subtle difference between the two shapes. But at what point does this difference become overly difficult to distinguish? Figure 5.6 shows the same two lightcurves at 0.5σ, 1σ, 2σ, 3σ, 5σ and 10σ signal to noise values. We can see here that even at 5σ signal to noise it is tricky to distinguish the two by eye, and at 3σ almost impossible.
Figure 5.6. From the top downwards are the various signal to noise levels (0.5\(\sigma\), 1\(\sigma\), 2\(\sigma\), 3\(\sigma\), 5\(\sigma\) and 10\(\sigma\)) for two of the simulated lightcurves, left the eclipse and right the transit. Any apparent differences in normalisation between the two types is purely due to the plotting routines and outlier points, this does not affect the KSOM algorithm.
Figures 5.7 and 5.8 show the results of a standard \((20 \times 20, \alpha_0 = 0.01, 34 \text{ iteration})\) network run when presented with datasets of varying signal-to-noise. From top left the results are for \(0.1\sigma, 0.5\sigma, 1\sigma, 2\sigma, 3\sigma\) and \(5\sigma\). Although there is some separation visible in the scatter plot for the \(0.5\sigma\) results it is clear from the U-Matrix plot that this is not identifiable from the structure in the network itself and merely a consequence of the near-continuous range in morphological profile between the transits and eclipses. As with the tests in section 4.3.1 it is around the 2 to \(3\sigma\) range that the structure embossed upon the network by the data becomes defined enough to produce a clear U-Matrix, and thus usable result.

As expected there is still some mixing of the two data types, this holds true even for a \(20\sigma\) dataset, but this will always be the case where there is noise present in data with such subtle differences. Indeed it is remarkable, when one views the \(3\sigma\) lightcurves of figure 5.6, that the method is capable of distinguishing the data as well as it does.

**The Distributed Dataset**

As before a fixed \(\alpha\) learning parameter, with \(\alpha_0 = 0.01, 34 \text{ iterations} \) and a \(20 \times 20\) network have been used. There are once again 5000 simulated lightcurves, with equal numbers of transits and eclipses, and with the same noise-distribution as used within the simulated datasets of section 4.2. Within the stellar eclipse data the simulations are drawn from a range of orbital inclinations (the distribution of which can be seen in figure 5.9), from the smallest of grazing eclipses through to full edge-on eclipses; and within the transit data a range of orbital inclinations and planetary masses are generated. Being the main parameters which affect the behaviour of the lightcurves these are the most relevant to produce a test dataset from. The planetary atmosphere may also be varied if desired, in this test it was scaled with planetary radius as any affect it would have would most likely be lost within the noise and would therefore have had no effect.

The results of this test can be seen in figure 5.10. As expected at the higher signal-to-noise extreme it is easier for the KSOM to distinguish between the two object types.
FIGURE 5.7. The scatter-plots for the final state of the network using the datasets with various signal-to-noise levels. From the left to right and moving down: $0.1\sigma$, $0.5\sigma$, $1\sigma$, $2\sigma$, $3\sigma$ and $5\sigma$. 
Figure 5.8. The U-Matrix plots for the final state of the network using the dataset with various signal-to-noise levels. From the left to right and moving down: $0.1\sigma$, $0.5\sigma$, $1\sigma$, $2\sigma$, $3\sigma$ and $5\sigma$. 
and it does a good job. The main eclipse area of the map is visible very clearly in the U-Matrix plot, as are parts of the transit-dominated area. It is worth noting that the KSOM produces two subtly distinct areas for the transits. This (correctly) suggests that the transit distribution I have created is not as continuous as it could be. As with other tests, and with reference to figure 5.8, it appears that this feature becomes evident at around a $3\sigma$ signal to noise ratio.

There are also quite large areas of the map where the two data types overlap, these correspond to those simulated data where the lightcurve has become dominated by noise, or rather that the signal produced by the grazing eclipse is too shallow to over-come the inherent noise in the data and so clearly discerning a signal is not possible.
CHAPTER 5. APPLICATION TO EXO-PLANET TRANSIT LIGHTCURVES

Figure 5.10. The scatter and U-Matrix plots for the simulated transit and eclipses data. On the scatter plot red diamonds denote transits, and blue diamonds eclipses.

The figure shows the scatter and U-Matrix plots for the simulated transit and eclipses data. Red diamonds denote transits, and blue diamonds denote eclipses.
CHAPTER 5. APPLICATION TO EXO-PLANET TRANSIT LIGHTCURVES

One further test was made taking advantage of the knowledge we have a priori about the precise points where the eclipsing portion of the lightcurves begin. In this case we are no-longer obliged to calculate the best-fit at every phase-offset (see section 3.2 for details). In practice we would never be truly sure we had this information, though an estimation of it would be available to us via the transit finding algorithms discussed earlier. One test was performed to check what would occur in this case, figure 5.11 shows the scatter plot of the end result. As with the previous investigation there is still some mixing of transit and eclipse types in parts of the map. This test indicates that it is not essential for the KSOM to know where the transit begins in order to separate the data, but that with the addition of this extra information a map with more separable classification areas may be achievable. Addition of this extra fitting parameter, which we would not normally be certain of, does not hinder the network’s ability in any strong manner; and I would conclude that, if able to, the inclusion of a fixed-phase KSOM analysis might be the best course of action when analysing such subtle differences.

Should more time have been available then a more complete dataset could have been generated with a smoother distribution across the parameter space for the transits. When the data are generated (using the parameter ranges mentioned previously) there is no way to tell whether a transit will occur for that particular parameter set. This in turn means that the dataset generated will favour those transits which occur at an orbital inclination of 90°, as these are the only guaranteed transits. Figure 5.9 shows how the majority of the data are focused within the first few data-types (those at an inclination of 90°). Detection of anomalies in the map due to this feature shows once again how powerful the KSOM algorithm can be.

A Pre-Trained Network

The final test I performed with these data was to see how well the data distinguished themselves if presented with a pre-trained map. This required a very slight alteration of the network initialisation code in order to load a pre-trained network (section 6.2.2 will
Figure 5.11. The scatter plot for the simulated transit and eclipses data using no phase-offsetting. The red crosses denote transits, and the blue crosses eclipses.
talk further about pre-loading the network). To train the networks a range of datasets were used: noise free, $10\sigma$, $5\sigma$ and $3\sigma$. Subsequently a noisy dataset (the same used in the previous section) was run through the KSOM algorithm using the various pre-trained networks for the network initialisation and an $\alpha$ learning parameter of zero (which does not change the underlying network and so basically best-fits data to the trained map). Figure 5.12 shows the range of results using pre-training. If we look at the upper left and upper right plots in figure 5.12, which correspond to the scatter and U-Matrix plots for the noise-free trained network, and then look at the upper centre plot (data analysed with the pretrained network) it is clear, as we would expect, that with the addition of noise the network is finding it more complex to distinguish the two types and so they fall on the edges of the respective classes' areas of influence. As was stated previously (section 2.3.2) the network must be allowed to mould itself in to a full description of the dataset represented to it for it to be effective. There must be sufficient training time for the finer details to crystallise within the map space. In a trained map such as this where such strongly defined shapes have been used there has been no chance for the finer crystallisation, and therefore diversity of shapes within a grouping, to enforce itself on the map. The inverse is also clearly true. In the test performed with the comparatively noisy, $3\sigma$, training data set figure 5.12 (lowest 3 plots) shows that the distributed data are generally falling further from the edges of the two areas of the map. I believe, as stated above, that this is due to the freedom of movement within the map-space given by the noise on the data. In this case the data are more similar to the underlying map areas and so is not constrained to the type-boundaries as in figure 5.12 upper's noise-free result. As with the noise-free results there remain areas on the map where the rate of change of the underlying map structure is high and a noticable number of the distributed data do fall on these points. It appears that the best, or rather most distributed, form of pre-trained map results from a mid-range noise value. In this case the $5\sigma$ map. This appears to provide a map with sufficient underlying structure, yet also sufficient range of lightcurve description, to allow the distributed data set to fill the map-space.
A better pre-trained map solution may involve the use of a more continuous dataset. The results above, as no one value of noise described the distributed data set perfectly, support this point. Unfortunately, time did not allow for the creation of such a dataset and so we are left with the rather quantised set discussed previously and with the 90° orbital inclination focus as shown in figure 5.9 and evidenced by the strong concentration of red points in the centre right of figure 5.12 top left.

5.3.5 Utilising Real OGLE Data

Much like in chapter 4 it is worth testing the network using real data. Unfortunately there isn’t an awful lot of transit data currently available and so we must make do with what we have, in this case the OGLE (Optical Gravitational Lensing Experiment) transit candidate data (Udalski et al. 2002), as derived using DoPhot and available from the OGLE website. Within the 137 transit candidates are three objects strongly believed to be transits, OGLE-TR numbers 32, 56 and 113 (figure 5.13 shows six examples of the OGLE transit-candidate data, including these three, folded on the suspected period). We would hope that the network is capable of separating transits from non-transits, or indeed those 3 objects regarded as strongly transit-like from other possible transits, and that this in turn could be used to find those objects most worthy of further follow-up observations. This in turn reduces the number of objects to choose between for those follow-up observations, or at least to prioritise the follow-up. As stated previously, our primary motivation in all of this is the cost of telescope time required to perform the transit confirmation experiments (such as doppler measurement) to get the orbiting object’s mass and thus confirm or deny whether it is a planet.

Transits and Transit Candidates

An interesting and extremely useful experiment, for lessening the time required to reduce the number of likely transits, is to see if the network can distinguish the confirmed transits
Figure 5.12. From top to bottom these plots show networks pre-trained using noise-free, 10σ, 5σ and 3σ datasets. The left column contains the scatter-plots for the networks trained with these data, the middle column the results using this pre-trained network and the distributed dataset (discussed in the main text) and the right column contains the U-Matrix of the network. Within the scatter plot the red diamonds denote transits, and the blue diamonds eclipses.
Figure 5.13. Examples of the folded OGLE transit-candidate lightcurves. The left side shows, from top to bottom, OGLE-TRs 32, 56 and 113 (the confirmed transits); and the right OGLE-TRs 2, 42 and 96. The data has been folded on the periods provided by the data available via the OGLE website.
from the transit candidates. Here there are 3 OGLE confirmed transits and a further 134 transit candidates. If there are clear patterns here then I would expect the 3 data to fall together on the network. Figure 5.14 is a scatter plot of the results, due to the low number count the network is only $4 \times 4$ in size (though the usual $20 \times 20$ network has been tried and behaved no differently). Although it does appear that there are some nodes with a higher population of data than others there does not appear to be a pattern relating to the confirmed transits and those data which triggered the OGLE transit-candidate flag. This may be due to a number of points: there is not enough self-similar transit data to cause a grouping, the data are not of a high-enough quality or the data are all of what the KSOM groups as one shape and the enforcement on the network of this type fills the map space in a similar manner to that discussed in section 4.1.3. In addition the $1D$ results behaved no better.

Part of the problem with using these data are that neither the precise beginning nor end of the transiting section of the lightcurve is known so that a truer lightcurve transit profile can be extracted, nor are there many “true transits” to be used in identifying/enforcing a “true transits” area on the KSOM map-space. Additionally, where there are transits within the lightcurve there are also a large amount of non-transit affected data points (for example see figure 5.13 and how the transit portions of the lightcurve also have data points at the non-transit lightcurve level). This in turns means that due to the way the network prepares and bins its data that the signal may be lost, or at least greatly reduced, and the analysis then becomes inconclusive due to ill-defined lightcurves. This apparent scatter could either be caused by data of questionable quality, an incorrectly given orbital period or simply that the period given is not to sufficient accuracy to fold the lightcurves accurately. If time were to allow then the last two related points could be checked using periodogram analysis of the data using methods such as $\chi^2$, the L-statistic (Davies 1990) or the analysis of variance method of Alex Schwarzenberg-Czerny (1996).


5.4 Conclusions

It is very tempting, following the utilised data point of concept in the previous section, that the K3Q method is applicable to the reduction of transit candidates into varying levels of likelihood (as very likely, transit through to stellar-like lightcurve determinations), but that to do so requires data of higher quality and higher signal to noise, than is currently available. Observing these transits is not as straightforward as this just to read. We should create a network of actually well isolated events to be analysed by the K3Q.

Whether the K3Q will become a method worth using to determine the transits from other similar starlight is quite open to any answer. All I can say is that with the current low guarantees of availability data, and that this is an open field. We need to create some method to provide adequate criteria for this time.

Figure 5.14. The 4 x 4 node scatter plot of the OGLE transit candidate analysis. It is clear that the three agreed transits, even on such a small network, are not grouping together. The black points on the scatter plot are the transit candidates and the purple the three accepted transits OGLE-TRs 32, 56 and 117.
5.4 Conclusions

It is my opinion, following the simulated data proof of concept in the previous section, that the KSOM method is applicable to the reduction of transit candidates into varying levels of likelihood (ie: very-likely-transit through to stellar-like lightcurve determinations), but that to do so requires data of higher quality, and higher signal to noise, than is currently available to me. With the advent of surveys geared more towards observing these transits, and reducing the data with this goal in mind, there should soon be a stream of suitably well-defined data ready to be analysed by the KSOM.

Whether the KSOM will become a method worth using to determine these transits from other similar forms of variation is a question I cannot currently answer. All I can say is that with the currently low quantities of available data, and classifying on shape alone, that it is not able to provide conclusive answers at this time.
Chapter 6

Alternative Applications and Future Directions

6.1 Alternative Applications

Throughout this work the Kohonen Self-Organising Map has been applied to a variety of time-series related data, more precisely the phased periodic lightcurves of various datasets. However there is no reason to stop there. As mentioned previously in chapter 2 the same basic algorithm has already been applied to a number of very different astronomical tasks such as the VizieR literature search software (Poincot et al. 1998, Ochsenbein et al. 2000) source classification in images (Mähönen & Hakala 1995), gamma-ray burst classification (Rajaniemi & Mähönen 2002) and the separation of galaxies and stars in visual images (Cortiglioni et al. 2001), with excellent results each time.

With these alternate successes in mind it was decided to apply the unmodified network code to an alternative type of astronomical data. To do this a subset of the XMM-Subaru field, a multi-wavelength imaging survey, was used. XMM-Newton is a space based X-ray telescope used for locating high energy sources such as Active Galactic Nuclei (AGN) and other X-ray bright phenomena. The Subaru, ground based, telescope is
A Japanese 8.3m optical telescope. The Subaru/XMM-Newton Deep Survey (or SXDS) is a multi-wavelength study of a 1.3 square degree patch of sky; its purpose to allow for significant progress in the understanding of the contents of the early Universe, the spatial density and evolution of QSOs (Quasi-Stellar Objects) and galaxies, and to gain a better idea of the extragalactic X-ray population (Watson et al. 2001). In order to observe both optical and X-ray data for a number of stars/galaxies a field was observed by both instruments. This field has been examined thoroughly by both XMM-Newton and the Subaru optical telescope and so has non-temporal data relating to both wavebands contained within it and is sufficiently different to the time-series data studied previously to constitute an interesting test. I think that KSOMs may be useful here because the task is to group objects into types based on a large number of measured parameters.

6.1.1 Analysis of the XMM-Subaru Dataset

In this instance, unlike with the periodic lightcurve data which uses some measure of the relative magnitudes at different phase-offsets, the six clustering parameters here are non temporally related, instead relating to single physical parameters of the focus object. These parameters include values such as X-ray to optical brightness, colour ratios and the measured source morphology. Approximately 500 sources in the data are deemed reliable enough (M. Watson, private communication) to be worth using with the algorithm; of these around 470 have complete sets of all six parameters.

The parameters are:

**X-ray to Optical Brightness:** The ratio of X-ray flux to optical flux.

**Optical g-r Colour:** The g'-r' colour of the optical source, where g' and r' are Sloan filter magnitudes.

**Optical r-I Colour:** The r'-i colour of the optical source.
X-Ray “Colour” 1: The X-ray “colour”, more commonly referred to as a hardness ratio (HR2), as determined by a comparison of different energy bands (as per equation 6.1).

X-Ray “Colour” 2: Another X-ray colour, or rather hardness ratio (HR3).

Determined Object Morphology: The morphology of the optical source as determined by the XMM-Subaru team. Either star, galaxy, near-star or likely noise.

A hardness ratio (HR) is defined as:

\[ HR = \frac{H - S}{H + S} \]  

(6.1)

where \( H \) (Hard) and \( S \) (Soft) refer to the energy bands associated with the particular hardness ratio.

The X-ray energy bands used here are defined as (from soft to hard):

**Band 2**: 0.5 \( \rightarrow \) 2.0 keV.

**Band 3**: 2.0 \( \rightarrow \) 4.5 keV.

**Band 4**: 4.5 \( \rightarrow \) 7.5 keV.

-and where \( HR2 \) is calculated from bands 2 and 3, and \( HR3 \) from bands 3 and 4.

Unlike the data used for lightcurve classification the X-ray data analysed here does not require analysis at all phase-offsets, indeed this would be meaningless as the parameters are distinct, and so may be analysed more rapidly.

It has been found, however, that when analysing these data using the KSOM that the quantised nature of the morphology measurement, being as it is limited to star, galaxy, near-star or noise values rather than a spectrum of continuous values, can cause too strong a polarisation of the map. A softening of this affect can be achieved by reducing the range over which the morphology is spread. For instance instead of the full parameter range, which for these data’s parameters are normalised to around 0.0 to 1.0 (as in previous
investigations), the range for the morphology parameter may be reduced to be between 0.3 and 0.7; or the per-weight weighting for the morphology parameter may be reduced from 1.0 (the usual full weighting used for non empty data points) to lessen the affect it has on the structure of the KSOM. Reduction of this weighting to 0.0 will result in a map which will cluster without reference to the perceived source morphology. One point of investigation is whether this strongly quantised morphology parameter affects the KSOM’s ability to self-organise based on all the parameters combined; and also whether without the presence of the morphology parameter the data will group in to any structure which will reflect the measured morphology value. If this does occur then we can say that the other parameters alone are sufficient to describe the object type.

Figure 6.1 shows just this. Here the clustering has been performed using no morphology parameter in the training. As a lone scatter plot this does not appear to produce anything clearly interesting. However, if the map of the most common type of object to fall in each node is plotted up (based on the morphology parameter and where different colours denote different morphological types) then there are clearly areas of the map which favour one type over another. This, assuming the given morphology values are accurate, shows that the objects are grouping to some degree by object morphology without knowledge of the measured object morphology. I say “assuming the given morphology values are accurate” as the morphology parameter is an ordinal number. Its value is biased because it represents a pre-classification derived by analysis of the PSF and not the physical measure (or “extent measurement”) taken directly from the physically observed data. In this manner it can unfairly influence the suggested classifications of the objects (as summarised later in table 6.1.1) as we are not relying purely on the continuum of results the KSOM analysis would normally provide.

The 2D Map Results

Due to the low number density of the data on the map, the initial parameters of which were chosen so as to compare directly to the ROTSE and simulated lightcurve KSOMs
Figure 6.1. A plot of data-type density on the network, the colouration determines the most common data-type on that node (star, near star, galaxy or no data (white)). The yellow and black data-types (star and galaxy) are clearly stronger in different parts of the map. The red sections are near-stars. Over plotted is the scatter-plot so that other data-types that fall on the node can be seen.
of section 4.4 and section 4.2, the number density plot alone is not a great indicator of clustering within the data (see figure 6.2). Instead the underlying structure, made apparent by the the U-Matrix plot, of the KSOM and a smoothed density map give us more useful information. It is clearer in these plots that there are areas where true groupings, or clusterings, appear to be occurring and the underlying gradient of differences across the map is lower. It is also interesting to note that in one area of the map there are rapid changes in the structure (figure 6.2, red areas of the U-Matrix plot) and therefore no clear groupings.

The 1D Map Results

When dealing with a dataset, such as this one, where the parameters that the network is grouping on are distinct and not immediately related (in contrast the magnitudes of stars at different phase-offsets, as in the lightcurve work, could be said to be related) it can be beneficial and easier to understand the results if we observe the results in 1D space. By doing this the underlying structure of the map may be observed on a per-parameter basis and plotted in the usual manner, creating something which is more common, and therefore slightly easier to understand, for the scientist. Figure 6.5 (and indeed the plots in figures 6.3 and 6.4) are examples of such plots used to examine the appearance of the network. From the top panel down are the various parameters, discussed above, and their values at each node along the 1D network map.

It is worth quickly taking a look at the plot utilising a full strength morphology parameter (figure 6.4). Here the groupings are very strongly polarised by the morphology parameter, with nodes 20-60 containing those data said to be of one morphology, 80-100 another and the remaining nodes the intervening morphology values (near stars). Although this may appear a pleasing result it is worth remembering that with such a strong influence over the map, and with the morphology value being pre-determined from alternate analysis of the PSF (Point Spread Function) of the data source on the sky image, that it could be said to unfairly suppress the map’s natural behaviour to self-organise based on
Figure 6.2. On the left is the smoothed density plot of the XMM-Subaru analysis, using no morphology parameter, and on the right the unsmoothed plot. The U-Matrix plot for the same data is shown below, some minimums are visible suggesting that the objects that fall in this area are truly similar.
all the parameters combined. As it stands, and as shall be discussed below, the use of a weakened morphology parameter is actually quite useful and does not appear to hinder the KSOM's ability to self-organise.

A Summary of the Results

If the morphology parameter is removed then we do still find groupings within the map. Looking at the first panel of figure 6.5, which shows the number of objects on each node, it is obvious that there are regions of the map space which are more heavily populated than others. The main points being the broad central region between nodes 30 and 80, the small collection around node 10 and the narrow peak on the right side of the plot, around node 90. It is worth remembering that in this particular plot the morphology parameter is unused and therefore that part of the plot is untrained and appears random.

These main areas, given by node references, contain the following behaviours (which are summarised in table 6.1.1):

10: There definitely are a number of objects clustering within a few nodes of the node 10 area of the map and this observation is enforced by the clear features in the parameter charts (figure 6.5). In contrast to the groups between nodes 40 and 90 these objects appear to be bluer in colour and are very strong in HR2 and have a small range, neither strong nor weak, in HR3. They are also reasonably X-ray bright with respect to their optical brightness. Suggested object type: QSO/Borderline AGN.

30: This group becomes more apparent during analysis of the parameters on the network than directly on number count (given that it is directly adjacent to the large 40-80 group discussed below). The objects here appear to be relatively unusual in that they have a low X-ray to optical brightness in comparison to the other objects, but are quite hard in HR3. They appear very similar to the objects in the main large grouping (30-80) in terms of the other X-ray colour and, as with the grouping
Figure 6.3. The results of a 100 node, 1 dimensional, KSOM analysis of the XMM-Subaru data. This KSOM has been trained using a weakened morphology parameter. From top to bottom the 7 windows show: number count of data best-matching that node, X-ray to optical brightness, g-r colour, r-i colour, the morphology, hardness ratio 2 (X-ray colour one) and hardness ratio 3 (X-ray colour two).
Figure 6.4. The results of a 100 node, 1 dimensional, KSOM analysis of the XMM-Subaru data. This KSOM has been trained using a full strength morphology parameter. From top to bottom the 7 windows show: number count of data best-matching that node, X-ray to optical brightness, g-r colour, r-i colour, the morphology, hardness ratio 2 (X-ray colour one) and hardness ratio 3 (X-ray colour two).
Figure 6.5. The results of a 100 node, 1 dimensional, KSOM analysis of the XMM-Subaru data using no morphology parameter. The panel order is the same as that in figure 6.3, with the exception that the morphology parameter on the network has remained untrained and is therefore in its initial random state.
around node 10, more blue than those in the main grouping. Suggested object type: QSO Type 2.

**40-80:** These objects appear, from inspection of the second panel in figure 6.5, X-ray bright relative to their optical brightness. Panel 6 of the same figure also indicates that these objects have a relatively low HR2 (in one X-ray “colour” they are relatively weak), but that they vary from high to relatively low values in HR3. These objects also appear quite red in the optical (panels 3 and 4) with low relative $g-r$ and $r-i$. Suggested object type: Starbursts or Liners.

**90:** This peak looks to be the strongest region on the map in terms of its narrow width, yet strongly enforced features. Much like the central, 40-80, concentration these objects have high X-ray to optical brightness and relatively soft hardness ratios. Although it is worth noting that in both parameters they differ very slightly from the central concentration in that they are very slightly less X-ray bright (relative to their optical brightness) and also very slightly different in X-ray “colour” (HR2 and HR3). This would seem to suggest that they are a distinct and separate object, rather than simply a progression of the parameter behaviour seen in the objects in the 40-80 region. Suggested object type: Normal Galaxies/Unknown.

The same regions are apparent in those plots using the morphology parameter. Figure 6.3 shows this, although there is strong polarisation caused by the full strength morphology plot (figure 6.4), the weakened morphology plot shows a pleasing agreement with that of the zero-morphology investigation and so the morphological type of each grouping can also be gathered.

In summary the results are shown in table 6.1.1 with suggested object types for the various groupings (for which I would like to acknowledge the very kind help of Pete Wheatley and Pam Derry).

The addition of the results with the XMM-Subaru data have shown that the unchanged KSOM algorithm is both capable of finding groupings in periodic lightcurve
<table>
<thead>
<tr>
<th>Grouping Node(s)</th>
<th>Morphology</th>
<th>g-r</th>
<th>r-I</th>
<th>HR2</th>
<th>HR3</th>
<th>$f_x/f_{opt}$</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Star</td>
<td>Blue</td>
<td>Blue</td>
<td>Hard</td>
<td>Medium</td>
<td>Medium</td>
<td>QSO/Borderline AGN</td>
</tr>
<tr>
<td>30</td>
<td>Mix/Near Star</td>
<td>Blue</td>
<td>Blue</td>
<td>Soft</td>
<td>Hard</td>
<td>X-Ray Faint</td>
<td>QSO Type 2</td>
</tr>
<tr>
<td>40-80</td>
<td>Galaxy</td>
<td>Red</td>
<td>Red</td>
<td>Softish</td>
<td>A Range</td>
<td>X-Ray Bright</td>
<td>Starbursts or Liners</td>
</tr>
<tr>
<td>90</td>
<td>Mix/Near Star</td>
<td>Grey</td>
<td>Grey</td>
<td>Medium</td>
<td>Soft</td>
<td>Medium</td>
<td>Normal Galaxies/Unknown</td>
</tr>
</tbody>
</table>

Table 6.1. A summary of the groupings found in the 1D XMM-Subaru results. The right-most column contains the suggested object type for this grouping.
data and data which involves distinct non-temporal parameters. These are pleasing results and serve as an independent verification of previous statements that the method is applicable to a variety of data analysis problems within astronomy and physics.

6.2 Future Directions

As has been shown here and in the work of others that the Kohonen self-organising map is an extremely versatile algorithm and, with the correct choice of clustering parameters, there is no reason why it could not be applied to further fields within or outside of astronomy.

Within the further analysis of stellar lightcurves (and other areas for that matter) we could also apply hierarchical network training or, with very large datasets, the use of partial pre-training of the network.

6.2.1 Hierarchical Network Analysis

In section 4.4 the results of the KSOM algorithm’s analysis of the ROTSE dataset was discussed. It was clear that the set of parameters, the normalised magnitude at various phase positions, chosen to classify the lightcurves presented to the network was inadequate to extract every sub-class of objects. Leaving us with one group consisting of the Cepheid, δ-Scuti and type-c RR Lyrae stars (the physical reasons why are discussed briefly in section A.1). As stated, should we wish to further separate this group then subsequent analysis using the period of the variable as the classifying parameter would allow us to separate out the various classifications. But what if the relationship between groups was less clear, or rather we didn’t believe it to rely on only one easily quantifiable parameter. In this case we may wish to use a multi-level analysis of the data. A first pass analysis may be done using the morphology of the lightcurves, as done in this thesis, with a second analysis of the data within any one group done using a different set of
parameters, the lightcurve’s period for example.

As the code currently stands the transfer of one datum, or record, from a network to another is possible due to the way the data handling is structured (section 3.1.1). And so the implementation of this method of analysis should require minimum coding by any interested party, simply the initialisation of a new network structure, the creation of a new data-list structure and the transfer of the record from one network to another (all of which is pre-written code, though in the latter case, due to current lack of necessity, is unused).

6.2.2 Pre-Trained Analysis

Should we wish to use a pre-trained network configuration to simply find where additional data falls on the network then we can, following training of a “blank” network by a completely descriptive (or a dataset we at least believe to be completely descriptive) set of data, simply reload the final trained network’s state as the initialised network structure. This is then followed by a zero learning (by, for example, setting the learning parameter $\alpha$ to zero) algorithm run using the new data and will give classification results based on the pre-trained network’s structure. Should the algorithm be placed in a pipeline where execution speed is important the user may wish to classify objects using a subset of the full dataset and then classify the rest using the pre-trained network. Occasional re-trainings of the underlying network may be required so as to incorporate any possible outlying data types (locatable using the P-Map, discussed in section 4.3.2, or by eye) to include in the next re-training. Whether to include these outliers may depend on a measure of the quality of the data, too much noise may result in a map that has increasing areas denoting noise as the number of re-training iterations increases. However, should careful re-training be applied the map should converge on a configuration which is fully descriptive of the dataset presented to it. A first stage in this process was taken in order to perform one of the tests in chapter 5. As the code stands it can now load in a network from a previous run as a command-line option. However, it cannot yet cope with differ-
ing network sizes, or number of weights per node, between the requested parameters for analysis and those used to create the pre-trained map. This is not too important a factor, but it may be a point that in the future the user may wish to use a larger or smaller map than that used to create the pre-trained map (perhaps for finer definition across the map), in this case an interpolation/averaging function for the new map would be used. Alternatively the parameters for the network run utilising the pre-trained network could load all of its parameters from the pre-trained network output file. I don't see either of these two options as essential, in fact I feel that they could remove some of the inherent power in the map's ability to separate objects, but should a user wish to enable these options then the coding involved is quite trivial.

6.2.3 The Floating-Point Map

As it stands the Kohonen scheme works to produce a classification of object based on some arbitrary coordinate within a map space. This coordinate is quantized based on the nodes that construct the map. However, on any one node there may fall two or more types of data, this is acceptable under normal circumstances, and expected. With any dataset containing some noise there are likely to be regions of uncertain classification. However this does not necessarily mean that the differences in the various data-types falling on the node absolutely cannot be determined. There are two approaches that may be taken, both of which I feel would be useful only on those data that fall on decision boundaries (the high-value parts of the U-Matrix) and are therefore open to type mixing.

Node-centre Vector Method

Any data which falls upon a certain node within the map space is only ever a best-fit, or rather least worst-fit, to that node. It will never under normal conditions absolutely match the node it falls on. This means that there is some difference between the node and the data and we may say that, under a continuous map space, the data falls not on
the node centre, but on some point between the node centre and surrounding nodes. This amount is quantified during the calculation of the best-fit statistic (section 2.3.1) and relates to the Euclidean distance between the node's vector of weights and the data's vector of weights (or parameters). So what the best-fit value, $B_F$ say, really means is the distance between the node centre and the data is $B_F$ and that this is less than for all other nodes. But it is still likely, due to the topological ordering of the map, that the nodes surrounding the best-fit node are also close matches to the data. Therefore I propose that it may be possible to work out the "force of attraction" between the $N + 1$ best-fit nodes, where $N$ is the dimensionality of the map, and the only nodes considered are the $3^N - 1$ nodes surrounding the best-fit and the best-fit node itself. In effect "centroiding" the best-fit position using the $3^N - 1$ best-fit nodes. Figure 6.6 gives an illustration of the idea suggested here. In effect this would create a continuous map space rather than the currently quantised one defined by the nodes themselves. Though it would still not be a map-space that absolutely described the data falling on it, its contiguous nature would allow for finer differences to be resolved. It would be much like running an extremely high resolution map, but without the enormous computational overhead this would produce. In practice this could produce a map with less quantized type boundaries. It may be possible to analyse these boundaries using a type of polygon definition based on the strongest differences between neighbouring data defining a normal to the polygon's edge; although this is merely conjecture as I unfortunately do not have time to investigate this idea.

**Histogram Method**

The second method, and a less computationally intense one, is to plot a histogram of the best-fit values between a node and the data that fall on it. It may occur that a certain type of data are falling on one node as it can't either enforce itself strongly enough on the network to carve its own niche, or there is insufficient freedom of movement across the network to allow it to occupy its own node (be that through insufficient network size
Figure 6.6. A sketch of how the implementer may begin analysing the best-fit vectors in order to calculate a floating-point position on the map. Here the 3 numbers denote the three best-fit nodes. The green cross the data’s true position on the network and the arrows the “forces” pulling on the data (the arrow lengths bare no relation to the magnitude of the forces, they are simply there to illustrate the direction of the force).
Figure 6.7. An illustration of the best-fit histograms. What is important to note here is the purple, underpopulated, group of data (bottom right area of the scatter-plot) which can be seen in the histogram as a non-continuous, or divergent, set of peaks. Values for the numbers in each bin are unimportant here, it is the relative amounts at different best-fit proximities to the node which are important. A small, 5x5, network has been used to illustrate the point.

or through overly strong network training parameters; though as shown previously the network isn’t strongly affected by the choice of parameters). In this case there may be a divergent set of best-fit values for the node: one set very close to the node as they correspond to the true best-fit data, and the other slightly less well fit corresponding to the “interlopers”. Figure 6.7 shows the output of the 0.1% underpopulated dataset (used in section 4.3.2) alongside a set of histograms relating to the value of the best-fit statistic for the data on each node. The purple underpopulated data-type can be seen in the lower portion of the scatter-plot having failed to impress its structure on the map. The divergence in the histogram plot for this node can be seen clearly on that node (as well as some others relating to other type mixes) indicating that there are likely two distinct types falling on the node. In practice the separation of the two peaks may allow us to extract outliers and where two (or more) types have fallen on one node.
6.3 Conclusion

The Kohonen Self-Organising Map algorithm has proven itself to be a remarkably robust and flexible method of data analysis and there are many further uses it could be put to than those shown in this thesis. The suggestions for future work outlined above, though perhaps subtracting from some of the algorithm’s elegance, do suggest that the “vanilla” KSOM has a strong future as a base for further classification methods, or as a strong beginning stage in a classification pipeline.

I firmly believe that with the correct interpretation of the parameters required to pass to the algorithm that this method can be applied to almost any classification task within astronomy.
Appendix A

Science Notes

A.1 Similarities in Cepheid, δ-Scuti and Type-c RR Lyrae Lightcurves

As mentioned in section 4.4 there are types of singly periodic variables which exhibit the same morphological form of lightcurve as each other. In this instance the δ-Scuti, Cepheid and type-c RR Lyrae variables. Varying primarily in period rather than lightcurve shape they are grouped together by the Self-Organising Map which in the first instance is grouping the dataset on lightcurve shape alone. Their similarity in shape arises from the cause of their variability being due to the same underlying physical process, in this case stellar pulsations (Strohmeier 1972). Radial pulsations in a star are caused by an imbalance between the forces holding the star in centrally, gravity, and pushing it outwards radially, radiation pressure. When a star is said to be in hydrostatic equilibrium both of these forces are in harmony and the star remains at the same radius and thus its variation due to pulsation is zero. However, if there is an imbalance in the two forces then an oscillation can occur as both the gravitational and radiation pressures fight to return to the equilibrium state. If, for example, there is a cooling of the star which reduces the radiation pressure exerted within it then gravity will start to cause the star to collapse in
on itself. As it does so the gases become more compressed and heat up; in doing this the radiation pressure increases and pushes back on the infalling stellar envelope causing it to re-expand. As each process, both infall and expansion, have some momentum behind them they both cause the star to pass beyond that point where the two forces are in equilibrium and the process repeats. Hence the variation repeats periodically and we get the characteristic lightcurve.

A.2 Brown Dwarfs

Brown dwarfs are, in two words, failed stars. At masses of between 0.013 and 0.08\(M_\odot\) (Zeilik et al. 1992) they are not massive enough for their core temperatures to remain high enough to sustain hydrogen burning, however they may have undergone limited amounts of deuterium burning at some point in their lives. If they were any less massive then they would be no deuterium burning and they would be classed as gas giant planets (although this value is arbitrary and not an absolute definition of a planet), and any more massive and they would undergo prolonged thermonuclear reactions within their cores and would be true stars (red dwarfs for the lowest masses). In holding with the well known astronomy saying “burn bright, die young”, brown dwarfs are exceptionally long lived. Most all brown dwarfs that have ever existed should still exist and so by this argument they should be very common. For this reason, and due to their low masses, we should be careful not to confuse a small brown dwarf with a large exoplanet. Within transit surveys there is the additional problem of brown dwarfs having similar radii (Baraffe et al. 2003) to gas giant planets and so to confirm whether they are indeed planets would require follow-up radial velocity measurements to obtain the mass.
Appendix B

The Code

B.1 Kohonen Network C Code

This section provides a few brief notes about the code complete with a basic program flow diagram. A more complete explanation is available within the main body of the thesis.

The code has been designed to operate in any configuration of regular network the user requests, assuming of course that the memory requirements this places on the executing computer are within its capabilities. Within this network (or map) the space is cyclic in each dimension, if imagining a 2D map this means it would map to the surface of a toroid. It has been tested, in terms of an analysable end-results, with between 1 and 3 dimensional networks; higher dimensionalities, though quite acceptable to the code, are too tricky to visualise and analyse to be of direct use.

It should also work with any dataset where the individual data have been reduced to comparable parameters (and saved in the relevant FITS format). For instance the lightcurve data are reduced to a number of (normalised) magnitude measurements at various phase-offsets, the XMM-Subaru data to a few parameters, such as colour ratios, and if we were looking at phonetics and speech recognition we could reduce the parameter
APPENDIX B. THE CODE

set to the individual phonemes. Where the data may be considered bad it is also possible to reduce the effect this point, or points, will have on the calculation of the best-fit node and blending of the data with the map (see section 3.1.1 for details).

In an attempt to produce a more efficient algorithm the code utilises a simple distance lookup, and subsequently a power lookup, table to calculate the power of the blending function between any two nodes (the lookup table may of course be simply offset in a few arithmetic operations). This table is possible (and information loss-less) due to the quantized nature of the underlying regular node structure.
APPENDIX B. THE CODE

Problem

Start

Read params from file

Read data sizes required from fits file.

Assign relevant sizes to arrays (if possible).

Read in data to initial structures/arrays of structures.

Bin and add weightings for data.

Initialise the network and associated structures.

Create distance lookup tables.

Enter network training routine.

Create power lookup from distance lookup.

Find the best-fit node.

Update the best-fit node.

Update all other nodes based on power lookup table.

Output network status and current classified LC types to FITS file.

Increment iteration count and update training parameters in a manner commensurate with the current iteration.

Iteration # > Iteration MAX?

Dealocate memory.
Close files.

Finish

Exit

Offset current LC data by an amount in order to check all phase offsets (if required).

Calculate the vector distance between the current LC and the weights on each node.

Store the distance if it's a minimum.

Increment phase-offset.
Maximum phase-offset reached?

Yes

Return the node the minimum distance was found for.

No
Bibliography


Bond, I., Udalski, A., Jaroszyński, M., Rattenbury, N., Paczynski, B., Soszyński, I., Wyrzykowski, L., Szymański, M., Kubiak, M., Szewczyk, O., Żebruń, K., Pietrzyński, G., Abe, F., Bennett, D., Eguchi, S., Furuta, Y., Hearnshaw, J., Kamiya,
K., Kilmartin, P., Kurata, Y., Masuda, K., Matsubara, Y., Muraki, Y., Noda, S.,


BIBLIOGRAPHY


Kane, S., Cameron, A., Horne, K., James, D., Lister, T., Pollaco, D., Street, R. & Tsapras, Y. (2004), ‘Results from the wide angle search for planets prototype (wasp0) i: Analysis of the pegasus field’, *MNRAS 353*.


Sozzetti, A., Yong, D., Torres, G., Charbonneau, D., Latham, D., Allende Prieto, C.,
Brown, T., Carney, B. & Laird, J. (2004), ‘High resolution spectroscopy of the

Street, R., D.Pollaco, A.Fitzsimmons, F.P.Keenan, K.D.Horne, S.Kane, A.C.Cameron,
T.A.Lister, C.Haswell, I.Skillen, S.Hodgkin, P.Wheatley, R.West & D.R.Brett

Street, R., Horne, K., Lister, T., Penny, A., Tsapras, Y., Quirrenbach, A., Safizadeh, N.,
Mitchell, D., Cooke, J. & Cameron, A. C. (2003), ‘Searching for planetary transits


Sullivan, D. & Sullivan, T. (2003), ‘Mauna kea high speed photometry of transits of the
extrasolar planet hd209458b’, Baltic Astronomy 12, 145–165.


Szalay, A. (2001), Astronomical data analysis software and systems x, in ‘ASP Confer-
ence Series’, Vol. 238.


Acta Astronomica 52, 1.


Verde, L., Heavens, A., Percival, W., Matarrese, S., Baugh, C., Bland-Hawthorn, J.,
Bridges, T., Cannon, R., Cole, S., Colless, M., Collins, C., Couch, W., Dalton,
G., Propris”, R. D., Driver, S., Efstathiou, G., Ellis, R., Frenk, C., Glazebrook,
K., Jackson, C., Lahav, O., Lewis, I., Lumsden, S., Maddox, S., Madgwick, D.,
2df galaxy redshift survey: the bias of galaxies and the density of the Universe',


