Abstract

This thesis investigates the classification of variable stars and the discovery of interesting behaviour in some of them. A secondary aim was to attempt to do this without astronomical expertise (domain knowledge). The data comes from the Macrae dataset which consists of time series observations of the red and blue magnitude of about 60 million stars made observed approximately every night. These observations are generally not equally spaced—since observations occur at different times each object and whose are never entirely—would choose to use these standard time series analysis tools such as Fourier or wavelet series. Time difficult to compute.

We explore the background to the Macrae project and variable stars.
The two datasets that were used in developing the techniques described in this work.

Discovery and Classification of Variable Stars

William Patrick Clarke*

January 2001

* A thesis submitted for the degree of Master of Science of The Australian National University.
I declare that this thesis is my own work, and that all sources used have been acknowledged.

William Clarke
Abstract

This thesis investigates the classification of variable stars and the discovery of interesting behaviour in some of them. A secondary aim was to attempt to do this without astronomical expertise (domain knowledge). The data comes from the MACHO dataset which consists of time series observations of the red and blue magnitudes of about 60 million stars, each observed approximately every night. These observations are generally not equally spaced—since observations occur at different times each night—and some are missing entirely—usually due to weather. So standard time series analysis tools such as Fourier or wavelet series are difficult to compute.

We explain the background to the MACHO project and variable stars. The two datasets that were used in developing the techniques described in this thesis are introduced: the 20k and Wood datasets, subsets of the entire MACHO project database.

We design and implement a binary file format for storing the time series data which is considerably faster to access than the original ASCII format files we had been given.

The very high dimensionality of the data for each star means that it is impractical to compare them directly. We develop a set of features that attempt to summarise the behaviour of each time series, including some frequency estimates based on norms of derivatives. Each star can then be directly compared with other stars based on the set of features.

We use the $k$-means clustering algorithm to cluster the feature-set of one of our datasets and discover that while the features do group similar behaving stars they do not necessarily separate different types of star. We also present the design of a new algorithm, $k$-means-min-$p$, that limits the minimum population of each cluster to a user-defined value.

We develop and test a new generic iterative classification and clustering algorithm that we call boxing. We use this to clearly show that the features do group similar stars. Using extreme-case selection we find an interesting class that was independently found by astronomers. Finally, we use the boxing algorithm together with a boosted generalised additive model classifier and relevance feedback to develop a “star-searcher” tool. This tool is very effective in allowing the user to search for examples of a particular type of variable star.

Despite our success in finding an interesting class of stars with boxing and extreme-case selection, we conclude that domain knowledge is necessary for this sort of work, particularly with interpretation of the results.
Acknowledgements

There are many people I would like to thank for supporting me throughout the development of this thesis, and if they are not mentioned here it is my fault.

Firstly, my supervisor, Markus Hegland, who has been there all the time, never without a helpful suggestion and always available whenever I needed assistance. The MACHO team and Peter Wood must be thanked for the use of their data, especially Tim Axelrod who gave generously of his time and energy to help us in understanding the data. I must acknowledge the assistance and support from my advisors, Steve Roberts and Graham Williams, and the members of the Computer Sciences Laboratory (CSL) Data Mining group, especially Ole Møller Nielsen and Peter Christen.

Special thanks must go to the CSL coke collective, for a cold red can has always been on hand whenever I needed one. The support staff at CSL (especially Michelle and Joe) have made it so easy for me to do my work.

My friends, who have kept me sane.

Finally, my family, who have been unfailing in their support.

For Victoria.

About this document

I must thank the developers of some fine pieces of software which I have used in the course of my research and the writing of this thesis. This document was written in GNU Emacs [25] using the excellent AUC TeX-mode [62], and typeset using \LaTeX\ 2e (courtesy of the \teTeX\ distribution [26]). Plots and all of my research and development work were produced using MATLAB [61]. Other vector graphics were drawn using \fig\ [59]. Version control provided by CVS [22]. The \LaTeX\ packages used include \calc, \ifthen, \geometry, \setspace, \acronym, \cite\, \url, \amsmath, \float, \algorithm\[ic], \graphicx, \[e]\epic, \subfigure, \listings, \varioref and \dropping, and the “dropped” initial font is Yannis Haralambous’ baroque initial font \yinitas; all are available from your local CTAN mirror [20].
# Contents

1 Introduction ............................................. 1
   1.1 The MACHO project .................................. 2
   1.2 Variable stars ...................................... 3
       1.2.1 Previous work on variable stars in the MACHO database 5
   1.3 Datasets ............................................. 7
       1.3.1 Some representative stars .......................... 7
       1.3.2 Locations of stars ................................ 9
       1.3.3 Numbers of observations .......................... 10
       1.3.4 Observation times ................................. 11

2 Data Storage ............................................. 15
   2.1 Original data format .................................. 15
   2.2 Binary data formats .................................. 17
       2.2.1 bmdf: binary MACHO data format .................. 18
       2.2.2 Using bmdf on the Wood and 20k datasets ......... 20

3 Features ................................................ 23
   3.1 Comparing time series ................................. 23
   3.2 MACHO Features ...................................... 25
       3.2.1 Notation ........................................... 25
       3.2.2 Noise and bias ..................................... 26
   3.3 Initial features ...................................... 26
       3.3.1 Average ............................................ 26
       3.3.2 Colour ............................................. 28
       3.3.3 Amplitude ......................................... 31
       3.3.4 Correlation ....................................... 34
   3.4 Derivative-based features ............................. 37
       3.4.1 Average frequency ................................ 38
       3.4.2 Average frequency spread ........................ 39
       3.4.3 Examples of derivative-based features ............ 40
       3.4.4 Time-scale features ............................... 43
       3.4.5 Using the derivative-based features on a sampled signal 44
3.4.6 Applying the derivative based features on MACHO data 45
3.5 Scatter-plot matrices of features .......................... 47

4 Clustering and Classification .................................. 51
4.1 Clustering ...................................................... 51
4.2 Classification ................................................... 52
4.3 k-means ........................................................... 53
   4.3.1 k-means algorithm ........................................ 54
   4.3.2 Change in variance for a reassignment .................. 55
   4.3.3 Applying k-means to the Wood feature-set ............. 56
4.4 k-means with minimum populations ......................... 58
   4.4.1 k-means-min-p algorithm ................................ 59
   4.4.2 Change in variance for a swap .......................... 62
   4.4.3 Comments on the algorithm ............................. 64
   4.4.4 Approximations .......................................... 65
   4.4.5 Implementing k-means-min-p ............................ 69
4.5 The boxing algorithm ......................................... 69
4.6 Boxing: extreme case selection .............................. 71
4.7 Boxing: the star-searcher .................................... 72

5 Conclusions and Further Work ................................. 79

A bmdf implementations ........................................... 81
   A.1 MATLAB implementation ................................... 81
   A.2 C/C++ implementation ...................................... 85
Chapter 1

Introduction

In this thesis we investigate the problems of classifying variable stars and discovering interesting or unusual variable stars based on recordings of their brightness over time. This is particularly interesting because of the inherent difficulty of the problem. The datasets involved can be very large: there are thousands of observations per star and millions of stars in the MACHO project's database. In addition, no standard techniques exist for efficiently comparing time series. The data itself is also quite interesting: two unequally-spaced time series of observations per star with missing data in each time series.

This thesis focuses on three aspects of the general problem: data storage (so we can access the data quickly), finding good features (so we can compare stars) and using clustering and classification techniques to find interesting variable stars. A secondary aim of this thesis was to see if we could find interesting stars without much expertise in astronomical time-series: we wanted to avoid using domain knowledge.

In the remainder of this chapter we introduce the MACHO project, variable stars and star light curves. We conclude with a short exploration into the two datasets—subsets of the MACHO database—we will use for the remainder of this thesis: the Wood and 20k datasets.

In Chapter 2 on page 15 we outline general data storage and access issues, and go into detail about issues with the MACHO data. We describe the binary format we use for storing and accessing the data and compare the performance of that format with the original format we were supplied with.

In Chapter 3 on page 23 we discuss the use of features to aid in our search for interesting behaviour in our datasets. After a brief survey into current literature on comparing time series and finding good features, we develop in detail a set of features for the MACHO datasets: these features
are easy to interpret and simple to compute. In the final part of the chapter we develop two features to describe the rate of variation of variable stars; these features are based on the norms of derivatives.

In Chapter 4 on page 51 we introduce the concepts of clustering and classification, including a brief survey of the state-of-the-art in these techniques. We delve into the detail of the \( k \)-means clustering algorithm and apply it to the Wood dataset. We also develop in some detail an extension called the \( k \)-means-min-\( p \) algorithm. We then introduce a new generic iterative classification and clustering algorithm called "boxing" and apply it in several ways to find interesting classes of variable stars within the Wood dataset.

Finally, in Chapter 5 on page 79 we summarise the main results of this thesis and discuss further work that can be done in this area.

1.1 The MACHO project

The MACHO project is a microlensing survey of about 60 million stars in the galactic bulge, the Large Magellanic Cloud (LMC) and the Small Magellanic Cloud (SMC). Microlensing is the theoretical effect which occurs when a particular form of dark matter (MACHOs: MAssive Compact Halo Objects) pass the line of sight of observed stars, briefly brightening the observation due to a gravitational lens effect [14]. The MACHO database contains about 80 billion photometric measurements [8, p.1539], which is approximately divided up into 50% bulge, 40% LMC and 10% SMC.

The MACHO project observations were made approximately each night from a telescope at Mt Stromlo Observatory in Canberra, Australia. The telescope started observing in June 1992 and recordings ceased in November 1999.

The bulge, LMC and SMC are each divided up into slightly overlapping fields. Each field contains tens of thousands of stars which can be simultaneously observed by the telescope.

A dichroic beam splitter enables simultaneous blue and red imaging; each beam is observed by four 2048 \( \times \) 2048 charge-coupled devices (CCDs). The amplifier of half of one of the red CCDs is not operational, and there are some known CCD defects. Each picture has a 300 second exposure and it takes 67 seconds to read out an entire image.

The resulting data is reduced through a complex series of template matching and error reduction to split the picture into observations for par-
1.2 Variable stars

The very large size of the database and the long observation times mean the database is ideal for identifying variable stars. Variable stars are stars which vary in magnitude in the short term; periodic variable stars are variable stars in which the variation is periodic and the periods can range from hours to hundreds or thousands of days. There is often evidence for multiple periods affecting a star, which makes identification of these periods difficult. The main types of periodic variable stars are discussed below:

Eclipsing binary stars are pseudo-variable stars, which appear as one object but are two stars orbiting very close to each other. They have a particular light-curve signature which has two different “dips” due to the two star’s different brightnesses and each star passing in front of the other. Their periods can range from hundreds of years (although

*More correctly, each “star” in the MACHO database is an “object” since the fitting code may combine multiple stars (or other stellar phenomena) into a single object. We call them stars.
Chapter 1. Introduction

Figure 1.1: The light curve of a Cepheid type star, with an estimated period of about 9.4 days. The bottom plot is a phase plot, where the observations are wrapped around the estimated period; the ends have been extended so the shape of the light curve around zero and one can be seen.

these are not likely to be classed as eclipsing binary in our datasets) down to 11 minutes [28, p.164].

RR Lyrae stars are true variable stars with periods in the range of 0.2 to 0.9 days; some (called RRC) have symmetric signals (the rise is similar to the fall) while others (RRab) have an asymmetric signal.

Cepheids are rare, very bright stars with periods of 1–70 days, usually with an asymmetric signal.

The light curve of a Cepheid type star with an estimated period of about 9.4 days is shown in Figure 1.1. This star comes from the Wood dataset (see §1.3 on page 7).

Long Period Variable stars (LPVs, also known as Mira type variables) are red giant stars with periods in the range of 30 to 1000s of days. The shape and period of the signal can change reasonably significantly between period so these are more accurately called semi-periodic stars.
1.2. Variable stars

Most true variable stars are LPVs but there are many subclasses of these.

The light curves of two LPVs are shown in Figure 1.2. The first one is a very smoothly varying star while the second seems to change character halfway through the observations.

Variable stars are fairly rare and are thought to make up about 1% of the total stellar population. Cook et al. [19] found almost 50000 candidate variable stars out of a total of 9 million stars (about 0.5%) from a subset of the LMC MACHO database. Of these variable candidates about 8000 are RR Lyrae, 1500 are Cepheids, 2200 are eclipsing binaries and 19000 are "red variables" (likely to be Long Period Variables).

1.2.1 Previous work on variable stars in the MACHO database

The large number of stars, large numbers of observations (high dimensionality) and irregular sampling present a problem. The size of the dataset mo-
tivates us to simplify the problem by working with features which describe each star’s behaviour and reduce the dimensionality considerably. Unfortunately, the irregular sampling means that standard methods for Fourier transforms and wavelets cannot be used. We cannot simply interpolate the data on a uniform grid since variable stars may vary with a frequency near or greater than the Nyquist frequency, in which case information is lost when interpolating. Scargle [54, 55] has done a lot of work in extending Fourier analysis to the unevenly sampled time domain but the computations are not simple.

Estimating the periods of the faster varying stars is quite difficult as we only have about one observation each night. However we can take advantage of the fact that the observations are not equally spaced apart, and “fold” the data over an estimated period and see how a given model fits the data. James Reimann’s PhD thesis [49] estimates the frequency of these stars using these period-folding techniques with various fitting techniques to fit the data (Reimann’s code was used to estimate the period of the Cepheid type star shown in Figure 1.1 on page 4). A related paper in [4] gives a detailed analysis when fitting using cubic splines. However, Pelt [46, pp.259–262] demonstrates that the periods found can be quite spurious. These techniques are not so useful with LPVs since the behaviour of LPVs often changes between periods.

A survey of periodic variable stars including LPVs in the early MACHO database is described in [19] and a study of Cepheids in [63]. A series of studies of variable stars in the LMC MACHO database are described in [9].

Wood [64, 65] has done significant work on the theory and observation of Cepheids and long period variable stars and used the MACHO database for his work in [67, 66]. One of the datasets used in this thesis is due to Wood (see §1.3).

All of the work described above was either by astrophysicists or in close collaboration with astrophysicists. Our work differed in this respect: we aimed to find interesting things in the datasets while avoiding using the specialised knowledge that the astrophysicists bring. In the data mining area, Ng, Huang and Hegland [44] attempted to automatically cluster and classify stars from the MACHO database using interpolated Fourier techniques with mixed success. Skinner [58] continued the work, also investigating autoregressive techniques and concluding that the Fourier techniques were more effective; the dataset used in these papers was the 20k dataset described in the next section.

\[\text{\cite{58}}\] Preliminary work was done using the MACHO observations before the survey was completed. Now that observations have finished more complete work can be done.
1.3 Datasets

The two datasets provided to us are called the 20k and Wood datasets. There is no intersection between the datasets.

The 20k dataset consists of 20403 stars (hence "20k") although 20 were excluded because they had no observations. The 20k stars are from two small areas of the LMC: 19698 from part of MACHO field 78 and 705 from MACHO field 77. There was no other selection basis so these should be representative of the population of the LMC.

The Wood dataset consists of stars specially selected from a small area of the LMC (part of MACHO field 77) and sub-selected based on the star's location in colour-magnitude feature-space: only stars redder than the main sequence and brighter than about 1 magnitude from the tip of the first giant branch were selected (see §3.3.2 on page 28 for an explanation of some of these terms). According to astrophysical theory, these stars are much more likely to be variable, with a high proportion of long-period variables. This dataset is described in Wood et al. [67]. The dataset we used was further trimmed by removing stars with low correlations between red and blue observations; the correlation coefficient used is similar to that described in §3.3.4 on page 34. The dataset contains 792 stars.

As an introduction to the datasets, I will show the data for some representative stars from both datasets, then some figures and statistics to illustrate the obvious differences and similarities between the datasets. Firstly I need to introduce some nomenclature. An observation is the data for a particular star at a particular time. This data may contain a red observation and/or a blue observation, each of which consists of a magnitude and an error estimate. A joint observation is an observation where both red and blue were observed: joint observations are useful because we can directly compare the red and blue values. Red-only or blue-only observations are those where the other colour is missing because of a known defect in the recording system or the reduction code was unable to identify that star (see §1.1 on page 2).

1.3.1 Some representative stars

The red observations of four random stars each from the Wood and 20k datasets are displayed in Figure 1.3 on the following page.

Throughout this thesis all star light-curves will have a similar layout to that shown in these figures. Each observation is shown as a point with error-bars to indicate the error for that observation. The average of the
Figure 1.3: The red observations of 4 stars each from the Wood and 20k MACHO datasets.
1.3. Datasets

Figure 1.4: The locations of the stars in the Wood and 20k datasets (right ascension and declination are at J2000: 1 Jan 2000, 12 noon UTC). MACHO field 77 and 78 centres are shown, as are approximate boundaries of the fields. Figure (b) shows the locations of just the field 78 stars in the 20k dataset; the range of (b) is indicated in (a) by a grey dashed rectangle.

Note how the 20k stars are all fairly constant within the ranges of the error-bars, whereas the Wood stars are all clearly variable since the variation is much bigger than the error-bars. Also worth noting are the gaps where there are no observations: this is examined in more detail in §1.3.4 on page 11.

1.3.2 Locations of stars

The locations (right ascension and declination) of the stars in the Wood and 20k datasets are shown in Figure 1.4. Also shown are the centres and approximate boundaries of fields 77 and 78.

This clearly shows the much higher density the 20k dataset has compared to the Wood dataset. Using some crude estimates of density the 20k dataset is approximately 200–460 times more dense than the Wood dataset. The estimates assume the datasets are made up of several rectangles and ignores spherical geometry effects which is probably reasonable given the similar declinations involved. The higher estimate uses two rectangles for
1.3.3 Numbers of observations

Histograms of the number of observations per star, for red-only, blue-only and joint observations for both the Wood and 20k MACHO datasets are shown in Figure 1.5.

Throughout this thesis all histograms (and there are quite a few) will

---

3A “tiling” is a partition of the sky into uniform rectangular pieces (tiles) which is used as an intermediate step in the data reduction process (§1.1 on page 2).
have a similar layout to that shown in these figures. The mean and one standard deviation from the mean are shown with grey dashed and dash-dot vertical lines. If histograms from both the Wood and 20k datasets are shown then to make comparison easier the ranges of values will be the same. If similar measures from the same dataset are shown then they will both have the same vertical range as well as the same value range. The number of bins used is 35 and 75 for the Wood and 20k datasets respectively.

Despite the large number of stars with very low numbers of joint observations, these figures show that both red and blue were recorded for most observations (about 62% of 20k observations and 75% of Wood observations are “joint”). The large number of stars with very low numbers of joint observations are stars which are not observed very much or stars which are consistently observed in one or more of the known system defects: of the 2080 stars in the 20k dataset which have no joint observations, there are only 9 which have both red-only and blue-only observations; 828 are red-only and 1243 are blue-only; all of the 9 have low numbers of observations of at least one of red and blue (less than four).

Most of the stars have small numbers of red- or blue-only observations, except that the 20k blue-only figure—Figure 1.5(f)—shows a curious distribution with two peaks. This is likely due to the red-amplifier defect, which on average would mean one-eighth of observations are blue-only. If we ignore the higher peak in the blue-only figure (probably due again to the red-amplifier defect) then the red- and blue-only distributions are quite similar, suggesting there are similar numbers of defects in the red and blue CCD arrays.

The Wood data seems very similar to the 20k except there are not many stars with low numbers of joint observations. This is likely to have been part of the selection process. Five stars have no joint observations: three of those have only blue observations and one has only red; the last has one red observation and ten blue observations.

Note that, in order to ensure reasonable accuracy for the features in Chapter 3 on page 23, we remove from consideration all stars with less than 200 joint observations. This is discussed in §3.3 on page 26.

### 1.3.4 Observation times

The times of observations in the Wood and 20k datasets range from \(-170.71\) to \(1395.2\) days, a span of about 4.3 years. The epoch (0.0 days) is defined as Julian Day \(-2.449 \times 10^6\) or 12 noon UTC on 12 Jan 1993 (or 10PM standard time where the recordings were made).
Figure 1.6: Unique observation times versus the time until the next unique observation time (with a log axis), for both the Wood and 20k datasets.

At any single observation time there may be thousands of different observations made (one for each star) because of the way the MACHO observations are generated. We call each observation time a “unique” observation time. There are 2001 and 2620 unique observation times in the Wood and 20k datasets respectively.

A plot of unique observation times versus the time until the next unique observation time is shown in Figure 1.6.

Note the sizable gaps between some observation times, especially the gap of 52 days at +320 days. This gap was caused by a fire in an electronics box in the control room which damaged equipment [6, p.699]; other gaps are other down-times or periods of bad weather. This figure also clearly shows the nightly nature of observations with a large number of gaps of around 1, 2, 3 and 4 days.

The large numbers of very small gaps of the order of $10^{-4}$ days (about 9 seconds) are in fact the same observation times; these differences are in the original data we were given and may be due to rounding errors. We have
not attempted to reconcile these differences.

To show the non-equidistant and cyclical nature of the observation times, Figure 1.7 shows observation time in days of the year versus observation time in hours of the day. There are three different times shown here: those times where both Wood and 20k observations occurred, times where just Wood observations occurred, and times where just 20k observations occurred. Also shown are local time sunset and sunrise\(^5\).

Not surprisingly, it is quite clear that the observing time goes through yearly cycles. During summer (Dec–Feb) the nights are short so there is less time for observation. During winter (Jun–Aug) the nights are long and observing can start earlier and finish later; however, during the night the LMC dips too low in the sky to observe which accounts for the gaps around midnight.

\(^5\)Sunset and sunrise times courtesy of [45].
Chapter 1. Introduction

The outliers at around early March (these actually are 3 different observation times at approximately +412.72 days) were observed almost 5 hours before other observations around that time of the year (at about 3PM local time: an unlikely proposition). This suggests that observations that claim to be at these times are in error. There were a total of 759 and 19284 observations made in the Wood and 20k datasets respectively at these times. No attempt was made to remove these data.
Chapter 2

Data Storage

With very large datasets, access to the data can become a bottleneck in the data mining process. We need to optimise the data storage so that the required data can be made available very quickly. In this chapter we present some recent work on this problem and then discuss our approach.

Maniatty and Zaki [42] discuss the requirements for parallel KDD (knowledge discovery in database) systems, including data storage and access. They note that most data mining tools use “flat files” (and this thesis is no exception) but that some work has been done in integrating conventional database management systems (DBMSs) with KDD tools.

Saraee and Tehodoulidis [53] discuss using a temporal DBMS for the data mining of temporal databases, and mention the possibility of using this for scientific applications. Integrating that sort of approach is beyond the scope of this thesis but is worthy of future investigation.

Dunkel and Soparkar [24] investigate data organisation and access with an association rule emphasis. They find that storing data “column-wise”—i.e., grouping all values of the same type but from different observations together—is better than the standard row-wise approach. With our time-series data a full column-wise approach is not sensible: each star should be treated independently; however, a column-wise approach to the data for each star may be appropriate.

2.1 Original data format

The original format we were given from the astronomers was a series of ASCII tables, each file containing observations from a particular star, and each line containing a particular observation. The data for each observation consists of time (in days), red and blue band spectral magnitudes (red and
Figure 2.1: The first few lines of the data for a star in the Wood dataset.

<table>
<thead>
<tr>
<th>HJD-2.449e+06</th>
<th>r (mag)</th>
<th>err_r (mag)</th>
<th>b (mag)</th>
<th>err_b (mag)</th>
<th>obsid</th>
</tr>
</thead>
<tbody>
<tr>
<td>-157.7255</td>
<td>-9.203</td>
<td>0.016</td>
<td>-7.850</td>
<td>0.018</td>
<td>771</td>
</tr>
<tr>
<td>-156.6771</td>
<td>-9.167</td>
<td>0.016</td>
<td>-7.812</td>
<td>0.019</td>
<td>810</td>
</tr>
<tr>
<td>-155.7252</td>
<td>-9.179</td>
<td>0.016</td>
<td>-7.811</td>
<td>0.021</td>
<td>845</td>
</tr>
<tr>
<td>-149.7225</td>
<td>-99.000</td>
<td>-99.000</td>
<td>-7.760</td>
<td>0.019</td>
<td>868</td>
</tr>
<tr>
<td>-147.7280</td>
<td>-9.139</td>
<td>0.017</td>
<td>-7.748</td>
<td>0.025</td>
<td>921</td>
</tr>
<tr>
<td>-145.6797</td>
<td>-9.122</td>
<td>0.016</td>
<td>-7.737</td>
<td>0.021</td>
<td>949</td>
</tr>
</tbody>
</table>
The Wood and 20k ASCII datasets are about 55 MB and 1.25 GB respectively. Accessing this data in the form given to us takes considerable time. A basic load-and-throw-away of the Wood and 20k datasets took about 26 and 640 seconds respectively; the major bulk of this time is spent converting from ASCII to binary.

2.2 Binary data formats

While having an ASCII format makes it easy to read by a human, it is extremely inefficient for reading by a computer since it has to convert to binary.

We wanted a binary format which was space (disk and memory) and time efficient. We did not want to store useless information like missing data, especially since about 25% or 38% of observations have missing data (Wood and 20k respectively). We did not want to use up huge amounts of memory; this means completely loading the file into memory is not possible. We also wanted the data to be fast to load. The data did not need to be portable because it was to be used only on the one computer (so we did not need to worry about floating point representations and endianness). We also wanted to make it easy to get just the “joint” data: where both red and blue were observed at the same time (this would be useful for calculating the correlation; see §3.3.4 on page 34).

The MACHO project used a binary format and associated C++ classes called SodophotSet [50]. Once loaded, it is effectively a three-dimensional array indexed by star number, observation number and colour. It also has a small memory footprint since it is designed to memory-map the binary file. The ASCII data we were given is a standard extraction of SodophotSet data. A SodophotSet does not provide easy access to “joint” data however. It would also be somewhat difficult to use with MATLAB [61], our development tool of choice.

Instead of using SodophotSet, I developed a binary file format that was specialised to handle the MACHO data that we were interested in. It was designed to be a drop-in replacement for the ASCII data we were given.

In order to remove the missing data we have to introduce some additional structure into the data for each star.

One way is to use a variable-length record for each observation, with two tag-bits indicating whether red and/or blue observations were taken. Some testing indicated that although the file would be much smaller this was not going to be much faster to load than ASCII, since each observation
has to be visited in order; we cannot simply “read” the entire structure into memory, the major benefit of having a binary format.

Another way to remove the magic numbers is to separate the data into three blocks: the red-only observations, the blue-only observations and the joint observations. If the user wants all the joint observations only the joint block needs to be loaded, with a single read. If the user wants all the red observations, then both the joint and red-only blocks need to be loaded and then both red blocks can be merged to form all the red observations.

2.2.1 bmfd: binary MACHO data format

The block method for removing missing data as described above is the one used. We call the file format bmfd—binary MACHO data format—and implemented a conversion program (in C) and functions in C and MATLAB for loading the data. A graphical depiction of the bmfd file format is shown in Figure 2.2 with a detailed description below.

This description assumes the use of 32-bit (4 byte) integers and floats. All data in the file is in binary (except the magic number and star-id’s) and is 4-byte aligned.
A bmdf file starts with a header. Firstly is a magic number: this must be the string "bmdf" (integer value 0x626d6466). Then follows an integer indicating the version: this thesis documents version 2; version 1 was the variable-length record format described above. The next integer is the number of stars in the file (nstars).

Following the header is the index. This consists of a tightly packed array of index records (index_t). If we label the stars from 0 to nstars - 1 then the offset (in bytes) from the header to the index record for star s is \( s \times \text{sizeof(index_t)} \). An index record contains a star-id as a 12-byte string and the offsets for the data (data_offset), and the red and blue offsets. The 12-character id length is an artificial limit based on the two datasets we have; if longer is required then 16 or 20 characters should be used (to keep the 4-byte alignment). The id string is only null-terminated if its length is less than 12.

The data_offset is an offset from the end of the index (i.e., \( \text{nstars} \times \text{sizeof(index_t)} \) bytes from the start of the index). Obviously star zero has a data_offset of zero. The other two offsets red_offset and blue_offset are offsets from the start of that star’s data. The end of a star’s blue data is defined as the start of the next star’s joint data (or the end of the file).

The first part of data for a particular star s contains the “joint” block of observations: where both red and blue were observed. It contains

\[
\text{red_offset}(s)/\text{sizeof(joint_t)}
\]

records. Each joint observation consists of 5 floating-point values: time, red, red-error, blue, and blue-error.

The second block of data is the “red-only” block. It contains

\[
(\text{blue_offset}(s) - \text{red_offset}(s))/\text{sizeof(red_t)}
\]

records. A red observation consists of 3 floating-point values: time, red, and red-error.

The third block of data is the “blue-only” block. If \( s < \text{nstars} - 1 \) then the end of the block is indicated by data_offset\((s + 1)\), otherwise the end of the block is the end of the file. A blue observation record is just like a red observation record.

All observations in a single block are sorted by time. This ensures that optimisations can be made to speed loading and displaying of the data. If we want to load all red observations for a particular star then after reading the joint and red blocks the red data from each can be merged in \( O(n) \)
worst-case time (where \( n \) is the number of red observations) instead of the general \( O(n \log_2 n) \) time. Indeed, it can be much better than \( O(n) \): if there are no red-only observations then no work need be done at all.

**Commentary on the binary file format**

Accessing the \texttt{bmdf} file can be extremely memory efficient and is flexible. To get to the data for a particular star one needs to know the number of stars in the dataset—to get to the index for the star, and to the end of the index—and the offsets for the star. At one extreme, we can reopen the data file each time we want to access it, or at the other we can load the entire index into memory (assuming we do not want to load the entire dataset into memory). Our current MATLAB implementation loads the header once and remembers the number of stars and for each access reads the file to get the index.

The 32-bit signed integers used as offsets limit the offsets to \( 2^{31} - 1 \) bytes (2 GB). This limits the size in bytes of a \texttt{bmdf} file to \( \text{sizeof(header\_t)} + \text{nstars} \times \text{sizeof(index\_t)} + 2^{31} - 1 + (\text{star}(\text{nstars} - 1)'s \text{data}) \), although to go beyond 2 GB it may be easier to move to 64-bit offsets. In order to overflow the data offsets the 20k dataset (see \( \S 2.2.2 \)) would need to either increase to about 125000 stars or increase in time to 26 years.

Any other limit (other than the star-id length, described above) such as \texttt{nstars}, \texttt{red\_offset} or \texttt{blue\_offset} are unlikely to overflow.

No provision has been made for random access via star-id. If this was required a separate index could be created creating a mapping from star-id to star number, possibly using a hash-table or tree or even using a specialised system taking advantage of the structure within a star-id.

Appendix A on page 81 has listings of MATLAB and C code which implement the loading of \texttt{bmdf} files.

We did not use the column-wise approach advocated by Dunkel and Soparkar [24] because it is never necessary to load only part of a block and MATLAB is very efficient in extracting columns from matrices.

### 2.2.2 Using \texttt{bmdf} on the Wood and 20k datasets

The Wood and 20k \texttt{bmdf} files are about 15 MB and 333 MB respectively, just over a quarter of the size of the ASCII datasets.

A basic load-and-throw-away of the Wood and 20k \texttt{bmdf} data took about 0.3 and 9 seconds respectively, between 70 and 80 times faster than the ASCII data. The different ratios for loading depend on the cache size of the computer and the file size.
2.2. **Binary data formats**

A table summarising the statistics of file sizes and access times is shown in Table 2.1.

This comparison with an ASCII format is unfair but illustrative of the problem with ASCII files. A fairer comparison would be with the native `MACHO` project format SodaphotSet [50].

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ASCII File size (MB)</th>
<th>Access time (s)</th>
<th>bmdf File size (MB)</th>
<th>Access time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wood 20k</td>
<td>55</td>
<td>26</td>
<td>15</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>1277</td>
<td>640</td>
<td>333</td>
<td>9</td>
</tr>
</tbody>
</table>

**Table 2.1:** Summary of file sizes and access times for the ASCII and bmdf file types.

3.1 **Comparing time series**

We are interested in series in medicine, biology and astrophysics where we need to make meaningful comparisons among them.

There has been considerable work on the study of astronomical time series, including the detection of periodic signals in E.2.1 to page 123 in the `Sobolev` book, an excellent book containing selected papers regarding the study of astronomical time series. In particular, `Huenneke et al.` [30] discusses various methods of frequency estimation from classical periods of time, while `Gray` [41] provides a survey in which the use of multiple applications of period estimation of various stars. Various techniques are reviewed, including the "stochastic" method. `Feder et al.` introduce the `CLEAN algorithm`. Many of the techniques are computationally expensive.

`Fitch` [46] presents a semi-parametric approach to the periodicity detection problem which is reasonably computationally efficient and also possesses the advantage: since periodicity detection finds epochs, period.

In the next section on time series from `Sobolev`, the reader may notice...
Chapter 2. Data Storage

2.2.2 Using f today on the scientific 23k dataset

The dataset contains a large number of scientific data (23k) from various experiments. Each data point is a record of the results of one of the 23k experiments.

A basic statistical analysis of the data showed that about 70% of the data points are between 50% and 80% of the maximum value. The distribution of the data points depends on the specific type of the experiment and the dataset.
Chapter 3

Features

Features are measurements of the behaviour of objects with which we can compare these objects. The MACHO data already consists of many features, including the location and the red and blue magnitude observations of each star. We could compare these “time-series” directly, or transform these very high-dimensional data into a small number of features.

We present the general problem of comparing time-series in the next section, including a brief survey of other work in the area, and discuss our approach to this problem. In the remainder of this chapter we develop our set of features.

3.1 Comparing time series

We are interested in ways to compare many very long astronomical time series (generally involving non-equidistant sampling).

There has been considerable work on the study of astronomical time series, including the MACHO-specific work summarised in §1.2.1 on page 5. Subba Rao et al. [60] edited an excellent book containing collected papers regarding the study of astronomical time series. In particular, Hannan et al. [30] discusses various methods of frequency estimation from a statistical point of view, while Cuypers [21] provides a survey of work on the specific application of period analysis of variable stars. Various techniques are presented, including the “string length” methods, Fourier methods and the CLEAN algorithm. Many of the techniques are computationally expensive. Pelt [46] presents a non-parametric approach to the periodicity detection problem which is reasonably computationally efficient and also presents an example where periodicity detection finds a spurious period.

In the more general time series area, Agrawal et al. [3] present an
envelope-matching algorithm, using specialised R+ trees for efficient access. Keogh and Pazzani [38, 39] transform each initial time series into piece-wise linear components and use relevance feedback to optimise the search for similar time series. They also present a hierarchical classification algorithm which progressively groups the most similar sequences; this is highly computationally expensive.

This thesis uses a feature-based approach, where instead of directly comparing the time series we transform each time series into a short vector of features which attempt to summarise the overall behaviour of the time series. These can then be compared directly using standard clustering and/or classification methods (see Chapter 4 on page 51).

The general problem of finding good features is discussed in Ripley [52, pp.327–332]. Ripley states,

\[ \text{W}e \text{ consider the problem of what features should be included when designing our classifier. We should make clear at the outset that this is an impossible problem; there may be no substitute for trying them all and seeing how well the resulting classifier works. However, this may be computationally impracticable...} \]

Indeed this possible approach is computationally impracticable for our data. Instead we tailor our features to try to model interesting and separate characteristics of our time series.

Astrophysicists already use a feature-based approach in classifying stars. They use average colour and magnitude to separate different sorts of stars [7]. They use weighted residual sum-of-squares (WRSS) to find variable star candidates [49, p.4]. In the search for MACHOs Alcock et al. [6] use features describing the closeness of fit to a microlensing model as part of their selection criteria for candidate microlensing events. In this thesis we use similar sorts of features and use a data mining approach for the comparisons.

We want features which are computationally simple to calculate because of the large number of stars and observations. We would also like interpretable features so that astrophysicists can use them to understand the star’s behaviour.

This thesis uses the general dataset (20k) and the specialised dataset (Wood) and develops computationally simple features to attempt to cluster and classify the variable stars within the datasets. The 20k dataset is used in developing the initial features to compare with the Wood dataset, but is not used for the later parts (derivative-based frequency features and clustering...
and classification). This is because the population of LPVs in the 20k dataset is probably very small (compare the areas of the Wood and 20k datasets in Figure 1.4(a) on page 9) and would be drowned out by the normal (constant) stars.

3.2 Macho Features

In order to compare stars, we need to reduce the amount of information from the thousands of data points per star to a short vector of features. These features should be time-invariant, computationally simple to calculate (O(N) to ensure scalability) and preferably interpretable. The last allows experts to make judgements based on their knowledge of the physics of the problem, without having to learn new paradigms, and also allows non-experts to gain some grasp on the situation.

The remainder of this chapter will focus on features developed to investigate the 20k and Wood Macho datasets. In §3.2.1 we introduce some notation to describe the time series we will be working with. In §3.3 on the next page we discuss the initial features such as average and amplitude. The estimation of frequency based on derivatives is discussed in §3.4 on page 37. Finally, §3.5 on page 47 shows scatter-plot matrices summarising the features applied to the Wood and 20k datasets.

3.2.1 Notation

Due to the complication of missing values, it is simpler to consider the data for each star to be two separate time series of magnitudes and errors: red: \( \{(X_R^{(i)}, E_R(i)), i = 1, 2, \ldots, N_R\} \) and blue: \( \{(X_B^{(i)}, E_B^{(i)}), i = 1, 2, \ldots, N_B\} \). \( X_R \) and \( X_B \) are the magnitudes and \( E_R \) and \( E_B \) are the errors. The time series \( X \) is assumed to be the sum of a deterministic signal and random errors

\[
X^{(i)} = x(i) + e^{(i)}. \tag{3.1}
\]

The error \( E \) is assumed to have a similar scale to \( e \) (see §3.2.2).

There are cases where we wish to compare the two time series \( X_R \) and \( X_B \) directly. In these cases we will use a joint time series

\[
\left\{ \begin{array}{l} X_R^{(i)}, X_B^{(i)} \end{array} \right\}, i = 1, 2, \ldots, N_J
\]

which contains those observations where both red and blue magnitudes were recorded. When it is clear that we are comparing a joint time series, \( X_R \)
and $X_B$ will refer to the red and blue time series subsets from $\bar{X}_j$.

Throughout the remainder of this chapter, we will refer to a generic continuous signal as $x(t)$.

### 3.2.2 Noise and bias

The noise in the measurements can be considered to be independent over time and is also approximately normally distributed. Averages over time will reduce the effect of this noise.

The systematic bias that is evident in each star’s light curves means that each feature will contain components of these biases. Direct comparisons of the values of these features for different stars will therefore include the biases of those star’s observations. We assume that these biases are either not very large or do not effect the ordering of stars in each feature.

### 3.3 Initial features

Known statistical summarisations of continuous signals will be used to motivate the features we use. We then transform the continuous feature to make it appropriate for our sampled data; there are many ways to do this, and for each feature we mention some possibilities and discuss the reasons for a particular feature being used. Each feature is illustrated with histograms and other diagrams showing the distribution for the datasets we were using.

In order to get more accurate results I filtered the datasets by removing from consideration all stars with less than 200 joint observations. This ensures that each star has at least 200 observations, and also ensures that the correlation (see §3.3.4 on page 34) is accurate. This selection removed 30 stars from the Wood dataset and 3274 from the 20k dataset. All of the following work in this thesis is with these filtered datasets.

#### 3.3.1 Average

The obvious first feature to use is the average value. For a continuous signal $x$ the average (or mean) value is defined as

$$m(x) = \frac{1}{\bar{t}^{(N)} - \bar{t}^{(1)}} \int_{\bar{t}^{(1)}}^{\bar{t}^{(N)}} x(t) \, dt \quad (3.3)$$
for which we can make approximations given our irregularly sampled signal $X$. The usual definition of the average value of a time series is

$$m_X = \frac{1}{N} \sum_{i=1}^{N} X^{(i)} \quad (3.4)$$

which has minimum variance for a uniform grid of $t^{(i)}$.

An alternative is to directly approximate the integral in (3.3) using the trapezoidal rule

$$\int_{t^{(1)}}^{t^{(N)}} x(t) \, dt \approx \sum_{i=1}^{N-1} \frac{1}{2} \left( t^{(i+1)} - t^{(i)} \right) \left( X^{(i)} + X^{(i+1)} \right) \quad (3.5)$$

which would give a “trapezoidal average” of

$$m_{X_{\text{trap}}} = \frac{1}{t^{(N)} - t^{(1)}} \sum_{i=1}^{N-1} \frac{1}{2} \left( t^{(i+1)} - t^{(i)} \right) \left( X^{(i)} + X^{(i+1)} \right). \quad (3.6)$$

Another alternative to approximate the average is the weighted average

$$w_X = \frac{\sum_{i=1}^{N} X^{(i)}}{\sum_{i=1}^{N} \frac{1}{E^{(i)}}} \quad (3.7)$$

and a trapezoidal version of this is

$$w_{X_{\text{trap}}} = \frac{\sum_{i=1}^{N-1} \frac{1}{2} \left( t^{(i+1)} - t^{(i)} \right) \left( X^{(i)} - \frac{1}{E^{(i)}} + X^{(i+1)} - \frac{1}{E^{(i+1)}} \right)}{\sum_{i=1}^{N-1} \frac{1}{2} \left( t^{(i+1)} - t^{(i)} \right) \left( \frac{1}{E^{(i)}} + \frac{1}{E^{(i+1)}} \right)}. \quad (3.8)$$

The trapezoidal approximations attempt to deal with the numerical error introduced with the non-uniform sampling by attenuating the effect of many samples taken close together in time and favouring those taken further apart. This reduces bias at the expense of variance.

The weighted averages attempt to deal with random error by penalising less precise observations and favouring those observed with low noise. This relies on good estimates of error or precision. The weighted error reduces variance at the expense of bias.

The average minimises the sum of squared distance. The median minimises the sum of absolute distances and is much less sensitive to outliers than the mean-based average. This is widely used in astronomical literature but is not used here.

For this thesis only plain average (3.4) and weighted average (3.7) were tried. The resulting features were so similar that only the average $m_X$ is used in the remainder of this thesis. It would be interesting to compare the
two trapezoidal approximations with the two simpler averages.

The relationship between the *apparent* brightness of a star (which the average is trying to measure) and the *absolute* brightness of that star is dependent on the distance between the observer and what lies between the observer and the star. Both the Wood and 20k datasets are from the Large Magellanic Cloud (LMC) for which the distance is well known and the depth of the LMC is small compared with the distance to the LMC—i.e., the distances from the observer to all of the stars in the datasets are approximately the same. In addition the amount of dust between the observer and the LMC is measurable and small [12]. This means that there is a known relationship between the apparent brightness of the star and its absolute brightness.

An estimate for the error in the average can be made (assuming independent and zero mean noise)

$$
\varepsilon_X = \sqrt{\frac{\sum_{i=1}^{N} E(i)^2}{N}}
$$

but since the number of observations is usually very large this tended to be very small (even with $\sqrt{N}$) and so was not used as a feature.

### Applying the average feature to the MACHO datasets

Histograms of the red and blue averages for both the Wood and 20k MACHO datasets are in Figure 3.1.

The 20k red average is clearly bimodal, whereas the blue is less so.

These figures show how the Wood dataset is a specific sample of the wider population based on the averages, whereas the 20k dataset is a representative sample of the dataset (and should behave approximately the same as the overall population): this is more clearly shown in the colour-magnitude diagrams in §3.3.2.

With the red-versus-blue figures, note how, for the Wood dataset, the red average is consistently higher (more negative) than the blue average. The 20k dataset stars are fairly evenly spread around the 1:1 line, with a divergence away from the 1:1 line as they get brighter: brighter stars tend to be more red or more blue: this is emphasised in the next section on colour.

### 3.3.2 Colour

Astronomers define "colour" as the ratio of two fluxes for different wavelengths, or through two different filters. Since magnitude is the logarithm of
3.3. Initial features

Figure 3.1: Histograms of the red and blue averages for both the Wood and 20k MACHO datasets. Also shown is red versus blue for both datasets.

flux, then colour is the difference of two magnitudes. We define our feature colour as the difference between the blue and red averages

$$\text{col}_{BR} = m_B - m_R.$$  

The colour is directly related to temperature, with hotter stars being bluer (colour is more negative).

Interestingly, astronomers view colour without inverting the axes as they do for magnitude (which is brighter when more negative). This means that in graphs bluer (hotter) stars are further to the left (or down) and redder (cooler) stars are further to the right (or up).

Histograms for the Wood and 20k colour features are shown in Figure 3.2 on the following page.

Also shown are “colour-magnitude” diagrams, which are colour versus blue. These diagrams show a great deal of structure which astronomers use to identify the type of each star. As stars evolve, their characteristics change: their location in the colour-magnitude diagram often gives an idea to how old a star is.

The 20k colour-magnitude diagram (Figure 3.2(d) on the next page)
Figure 3.2: Histograms of the feature "colour" for the Wood and 20k MACHO datasets. Also shown are "colour-magnitude" diagrams (see text for details).

has a very similar structure to that in the MACHO project 9-million star colour magnitude diagram [7, p.2196]: this supports the 20k dataset as a representative sample of the overall population. The "main sequence" is the nearly vertical line (constant colour) at around colour = −0.5. The main sequence should continue down below blue average = −3; the reason it does not is because the MACHO observations cannot resolve stars accurately that are that dim. The red giant branch (RGB, or first giant branch, FGB) is the other main linear cluster; the RGB "tip" is the end of the RGB at about (colour, average) = (0.75, −6.5). The Asymptotic Giant Branch (AGB) starts within the RGB but continues up and to the right.

The Wood dataset are clearly stars selected primarily from the AGB; these stars are bright, red, "highly-evolved" and are often variable [7, p.2207].
3.3.3 Amplitude

The amplitude of a continuous signal is often calculated by averaging the distance squared from the mean:

\[ s(x)^2 = \frac{1}{t(N) - t(1)} \int_{t(1)}^{t(N)} (x(t) - m(x))^2 \, dt \]  \hspace{1cm} (3.11)

which can be rewritten as

\[ = \frac{1}{t(N) - t(1)} \int_{t(1)}^{t(N)} x(t)^2 \, dt - m(x)^2. \]  \hspace{1cm} (3.12)

The corresponding amplitude of a time series is

\[ s_X^2 = \frac{1}{N - 1} \sum_{i=1}^{N} (X^{(i)} - m_X)^2. \]  \hspace{1cm} (3.13)

Alternatively, we could use the trapezoidal rule from (3.5) on page 27 to approximate (3.11)

\[ s_{X,\text{trap}}^2 = \frac{1}{t(N) - t(1)} \sum_{i=1}^{N-1} \frac{1}{2} (t^{(i+1)} - t^{(i)}) \left( X^{(i)}^2 + X^{(i+1)}^2 \right) - m_{X,\text{trap}}^2 \]  \hspace{1cm} (3.14)

but this was not done.

The feature we use is the plain time series amplitude from (3.13).

Histograms of the red amplitudes for both the Wood and 20k datasets are shown in Figure 3.3 on the next page. The blue amplitude distributions are similar and are not shown.

It is worth noting that the largest Wood amplitude is larger than the largest 20k amplitude (for both red and blue). Both Wood and 20k amplitude distributions are clustered around zero with a few very large values, suggesting that a logarithmic transformation may be useful. The 20k red amplitude is clearly bimodal: these two peaks correspond to the peaks in the colour-magnitude diagram in Figure 3.2(d) (the smaller right peak are generally the stars in the main sequence).

Also shown is red amplitude versus blue amplitude: the axes are log-log to spread out the amplitudes. In the main line of the Wood dataset the blue amplitudes are consistently higher than the red amplitudes whereas in the 20k dataset they are usually more similar—this contrasts with the averages in which the Wood red average were consistently higher than the blue average (see §3.3.1 on page 26). There is an interesting cluster of a few stars in the Wood dataset which have much higher blue amplitude than red.
Figure 3.3: The behaviour of the amplitude feature with histograms of the red amplitude, red amplitude versus blue amplitude, and red amplitude versus red averages, for both the Wood and 20k MACHO datasets. The blue amplitude feature behaves similarly.

amplitude: these stars have a few outliers in the blue observations which completely overload the amplitude; a more robust amplitude estimator may be more appropriate (for example, the median absolute deviation).

The 20k red amplitude versus red average plot (Figure 3.3(f)) shows an interesting and counterintuitive correspondence: dimmer stars in general have higher amplitude, but these are not like the stars in the Wood dataset which are much brighter. This can be explained by the effect of the Poisson-like behaviour in the photon counts when measuring the flux: if a star is essentially constant with flux $\mu$ (as most of the stars in the 20k dataset are) then its measured flux $Y$ is a Poisson random variable, with $Y = \mu + \eta = \mu \left(1 + \frac{\eta}{\mu}\right)$ where $\eta$ is the Poisson error. The expected value of $Y$ is $E[Y] = \mu$, and the variance of $Y$ is $E[(Y - \mu)^2] = \mu = E[\eta^2]$, so $\eta \approx O(\sqrt{\mu})$. However, what we are measuring is the magnitudes, which is the logarithm of the flux; so $\log Y = \log \mu + \log \left(1 + \frac{\eta}{\mu}\right) \approx \log \mu + 1 + \frac{\eta}{\mu}$ where the approximation relies on $Y \gg 0$. So the observed magnitude has
3.3. Initial features

(a) Wood red log relative amplitude histogram  
(b) Wood red lra vs blue lra  
(c) Wood red log relative amplitude versus red average

(d) 20k red log relative amplitude histogram  
(e) 20k red lra vs blue lra  
(f) 20k red log relative amplitude versus red average

Figure 3.4: The behaviour of log-relative-amplitude (lra) with histograms of the red lra, red lra versus blue lra, and red lra versus red average, for both the Wood and 20k MACHO datasets.

a random error of the order of \( \frac{n}{\mu} \approx O\left(\frac{1}{\sqrt{\mu}}\right) \). For constant stars then, lower magnitudes will have a higher amplitude due entirely to this error.

Log relative amplitude

The apparent relationship between amplitude and average suggested that normalising amplitude by average might be useful. The logarithmic distribution of the resulting “relative amplitude” then suggested “log relative amplitude” (lra); this is defined as:

\[
\text{lra}_X = \ln \left( \frac{s_X}{m_X} \right). \tag{3.15}
\]

The negative is required since \( m_X \) is negative.

The behaviour of the log-relative-amplitude feature for the Wood and 20k datasets is shown in Figure 3.4. The blue lra distributions are similar and are not shown.

The 20k red log-relative-amplitude histogram clearly shows the bimodal
distribution which was also evident in Figure 3.3(d) on page 32. The right peak consists primarily of the constant Poisson-noise affected stars described above.

The red-versus-blue graphs of amplitude and log-relative-amplitude are very similar, suggesting not much gain is to be had by using this transformation. About the only difference is that the largest Wood log-relative-amplitudes are smaller than the largest 20k log-relative-amplitudes.

The 20k red Ira versus red average plot (Figure 3.4(f) on the previous page) shows a small number of stars above the main line. These are bright and have high relative amplitude. These stars are in a similar region of feature-space to the Wood stars, and are primarily long-period variables.

Due to the flatter distribution of the log-relative-amplitude compared to the plain amplitude, the Ira was used in the clustering and classification work described in the next chapter.

Since the red and blue log-relative-amplitudes are quite similar, a transformation inspired by the “colour” average transformation was used to produce the log-relative-amplitude-colour (Irac)

\[ \text{Irac}_{BR} = \text{Ira}_B - \text{Ira}_R. \] (3.16)

The “colour” in this context refers to the difference between blue and red features. It is also used in the frequency colour and relative frequency spread colour described in §3.4.5 on page 44. The distributions of the Ira for both the Wood and 20k datasets are shown in the scatter-plot matrices in §3.5 on page 47.

### 3.3.4 Correlation

If we wish to compare the behaviour of two signals at the same time, averaged over time, then the usual measure is the correlation. The correlation of two continuous signals \(x_1\) and \(x_2\) is

\[ \rho(x_1, x_2) = \frac{1}{s(x_1)s(x_2)} \frac{1}{(N - 1)} \int_{t(1)}^{t(N)} (x_1(t) - m(x_1))(x_2(t) - m(x_2)) \, dt \] (3.17)

which ranges from -1 (negatively correlated) through zero (not correlated) to 1 (completely correlated). The corresponding correlation of a joint time series \(X_J\) is

\[ \rho_j = \frac{1}{(N_j - 1)s_{RB}} \sum_{i=1}^{N_j} \left( X_R^{(i)} - m_R \right) \left( X_B^{(i)} - m_B \right) \] (3.18)
where \( \left( \frac{X_R^{(i)}}{X_J^{(i)}} \right) = \bar{X}_j^{(i)} \). We use this as our correlation feature.

One could also make a trapezoidal approximation to (3.17) but this was not attempted.

Stars with high correlation and high log relative amplitude could be reasonably expected to be clearly variable. The high Ira selection could choose stars in which noise made a significant contribution to the signal; these stars are likely to have low correlation since we assume that the noise is uncorrelated across frequency (colour) as well as across time.

The correlation feature for both the Wood and 20k datasets is shown in Figure 3.5 on the next page as histograms and against red log-relative-amplitude. There are several items of note in these figures.

Firstly, the bulk of the Wood dataset is highly positively correlated, suggesting that it is full of variable stars. The bulk of the 20k dataset is between 0 and 0.3, with a peak around 0.15. This positive peak is logical: if the dataset were wholly constant stars, then we would expect a zero peak; any additional variation in stars is probably positively correlated which shifts the peak to the right. Since the minimum correlation is \(-0.4\) a cut-off of \(+0.4\) might be a good place to select variable stars.

Secondly, the correlation versus red log-relative-amplitude shows interesting structure. The main bulk of the 20k dataset consists of fairly uncorrelated stars, which also cover a large range of amplitudes.

The sequence of stars starting at the middle bottom and curving up to the top right is of interest. The bulk of the Wood dataset seems to come from this sequence as well. If we assume that all of these are variable stars, then the behaviour of this line can be explained by the effect of additive noise; if the noise has a variance of \(\sigma^2\) and a star has an amplitude of \(s_R\) then the correlation equals \(\frac{\bar{X}_J^{(i)} - \sigma^2}{s_R^2}\). Intuitively, high-amplitude variable stars are likely to have high correlation, since their variation will overwhelm the effect of noise; as the amplitude decreases relative to the noise, the noise will tend to reduce the correlation, so the line moves to the left.

**Phase-shifted stars**

Some stars are clearly variable but have an interesting correlated behaviour: they have a “phase difference” in their variation. This phase difference corresponds to the temperature fluctuations within the star as it oscillates. An example of such a star from the Wood database is shown in Figure 3.6 on page 37 (note that this star was found by chance while viewing light-curves).

The error-bars in the colour are estimated as \(E_C^{(i)} = \sqrt{E_R^{(i)2} + E_B^{(i)2}}\).

The phase shift is clearly visible in the red-versus-blue portion of the
Figure 3.5: The behaviour of the correlation feature for both the Wood and 20k MACHO datasets. See text for commentary.

plot: a zero phase shift would result in a straight line, a non-zero phase shift results in a rotated ellipse and a “90-degree” phase shift would result in a circle (or horizontal ellipse, depending on relative strengths of red and blue). Also note that there is evidence in the figure of a much longer period influencing the star.

A measure of the actual phase shift may be of some use; this could be defined as the time $\tau$ of the maximum cross-correlation

$$R_{x_1,x_2}(\tau) = \frac{1}{t(N) - t(1)} \int_{t(1)}^{t(N)} (x_1(t) - m(x_1))(x_2(t - \tau) - m(x_2)) \, dt.$$  

(3.19)

This would be difficult to estimate with our data, due to the non-equidistant
3.4 Derivative-based features

The above features do not give any information about how fast a variable star oscillates, only an indication if it oscillates at all (using amplitude and
correlation). In this section we define a few ways to estimate the frequency of variation, and also how regular that variation is. The general method is to start with a definition in the frequency domain and transform that into the time domain so it can be easily calculated. The transformations generally require some assumptions, but we show some examples where the assumptions are violated but the results make sense.

### 3.4.1 Average frequency

We define the average squared frequency $\bar{\omega}^2$ of a continuous signal $x(t)$ (with Fourier transform $\hat{x}(\omega)$) to be

$$\bar{\omega}^2 = \frac{\int_{-\infty}^{+\infty} \omega^2 |\hat{x}(\omega)|^2 d\omega}{\int_{-\infty}^{+\infty} |\hat{x}(\omega)|^2 d\omega}. \quad (3.20)$$

This makes sense as a definition of average frequency since for real signals negative frequency is equivalent to positive frequency (ignoring phase) and squared frequencies are more tractable than absolute values: see the transformation in the next few paragraphs.

For continuous square integrable functions $x(t)$, the 2-norm (energy) is

$$\|x\|^2 = \int_{-\infty}^{+\infty} x(t)^2 dt = \frac{1}{2\pi} \int_{-\infty}^{+\infty} |\hat{x}(\omega)|^2 d\omega \quad (3.21)$$

where the right-hand equality comes from Parseval’s relation.

The 2-norm of the derivative of $x(t)$: $x'(t) = \frac{dx}{dt}$ is

$$\|x'\|^2 = \int_{-\infty}^{+\infty} x'(t)^2 dt = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \omega^2 |\hat{x}(\omega)|^2 d\omega \quad (3.22)$$

since $\mathcal{F}[x'(t)] = i\omega \hat{x}(\omega)$.

Now we can rewrite (3.20) as

$$\bar{\omega}^2 = \frac{\|x'\|^2}{\|x\|^2}. \quad (3.23)$$

This is a useful basis for a feature, since it is phase-invariant and only dependent on the energies of the signal and its derivative. We use this as a feature with other norms as well, using the transformation from (3.20) as motivation for the feature despite using non-square-integrable functions in the examples in §3.4.3 on page 40. We call $\bar{\omega}$ the “average frequency”.

The calculation of the norms and the derivatives for the discrete, unevenly-sampled signal we get in the MACHO dataset we are using is discussed...
3.4. Derivative-based features

in §3.4.5 on page 44.

3.4.2 Average frequency spread

We define the average squared deviation from the average squared frequency $\overline{\delta \omega}^2$ as

$$
\overline{\delta \omega}^2 = \frac{\int_{-\infty}^{+\infty} (\omega^2 - \overline{\omega}^2)^2 |\hat{x}(\omega)|^2 \, d\omega}{\int_{-\infty}^{+\infty} |\hat{x}(\omega)|^2 \, d\omega}.
$$

This takes its motivation from the definition of amplitude (see §3.3.3 on page 31) and the average squared frequency (3.20).

Again, assuming $x(t)$ is square integrable, there are two obvious ways to transform this equation to the time domain. The first method moves to the time domain before expanding the square:

$$
\overline{\delta \omega}^2 = \frac{\int_{-\infty}^{+\infty} |(\omega^2 - \overline{\omega}^2)\hat{x}(\omega)|^2 \, d\omega}{\int_{-\infty}^{+\infty} |\hat{x}(\omega)|^2 \, d\omega}
$$

(3.25)

$$
= \frac{\int_{-\infty}^{+\infty} (-x''(t) - \overline{\omega}^2 x(t))^2 \, dt}{\int_{-\infty}^{+\infty} x(t)^2 \, dt}, \quad x \in \mathcal{L}_2
$$

(3.26)

$$
= \frac{\|x''\|^2}{\|x\|^2} + \frac{\|x'\|^4}{\|x\|^4} + 2 \frac{\|x''\|^2}{\|x\|^4} \frac{\langle x'', x \rangle}{\|x\|^4}
$$

(3.27)

$$
= \frac{\|x''\|^2}{\|x\|^2} + \frac{\|x'\|^4}{\|x\|^4} - 2 \frac{\|x'\|^4}{\|x\|^4}
$$

(3.28)

where the scalar product $\langle a, b \rangle = \int_{-\infty}^{+\infty} a(t) b(t) \, dt$. Using this as a feature would require calculating the scalar product of the signal and its second derivative, which is difficult if using numerical methods to estimate the second derivative since the sampling points may not overlap. With regular sampling this method may be usable since the second derivative may have the same sampling points as the original signal (except for the boundaries).

If we expand out the square before moving to the time domain, then we get

$$
\overline{\delta \omega}^2 = \frac{\int_{-\infty}^{+\infty} (\omega^4 + \overline{\omega}^4 - 2 \overline{\omega}^2 \omega^2) |\hat{x}(\omega)|^2 \, d\omega}{\int_{-\infty}^{+\infty} |\hat{x}(\omega)|^2 \, d\omega}
$$

(3.29)

$$
= \frac{\int_{-\infty}^{+\infty} x''(t)^2 \, dt}{\int_{-\infty}^{+\infty} x(t)^2 \, dt} + \overline{\omega}^4 - 2 \overline{\omega}^2 \frac{\int_{-\infty}^{+\infty} x'(t)^2 \, dt}{\int_{-\infty}^{+\infty} x(t)^2 \, dt}, \quad x \in \mathcal{L}_2
$$

(3.30)

$$
= \frac{\|x''\|^2}{\|x\|^2} + \frac{\|x'\|^4}{\|x\|^4} - 2 \frac{\|x'\|^4}{\|x\|^4}
$$

(3.31)

$$
= \frac{\|x''\|^2}{\|x\|^2} - \frac{\|x'\|^4}{\|x\|^4}
$$

(3.32)
which has the desirable property of not comparing signals in the time domain*. We use this expansion as a feature for our datasets, with the norm approximations discussed in §3.4.5 on page 44.

We call $\delta \omega^2$ the “average frequency spread”; this (and the “average frequency”) is justified in the next section.

### 3.4.3 Examples of derivative-based features

All our examples are sums of sinusoids, based on the generic signal $x_n(t)$

$$x_n(t) = \sum_{k=1}^{n} a_k \sin (\omega_k t + \phi_k)$$  

(3.33)

where $\omega_k$ is the frequency (in radians), $\phi_k$ is the phase and $a_k$ is the amplitude of the $k$-th sinusoid. Depending on the ratios of the $\omega_k$’s this signal is either periodic or quasiperiodic.

Since these signals are not square-integrable, for these examples we use this 2-norm (power)

$$\|x\|^2 = \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{+T} x(t)^2 \, dt.$$  

(3.34)

The first example is with a single sinusoid

$$x_1(t) = a_1 \sin (\omega_1 t + \phi_1).$$  

(3.35)

Following the definition of $\bar{w}$ from (3.23) on page 38

$$\bar{w}_1 = \left[ \frac{\|x_1'\|}{\|x_1\|} \right]$$  

(3.36)

$$= \left[ \frac{\lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{+T} x_1'(t)^2 \, dt}{\lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{+T} x_1(t)^2 \, dt} \right]$$  

(3.37)

$$= \omega_1 \frac{\lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{+T} a_1^2 \omega_1^2 \cos^2 (\omega_1 t + \phi_1) \, dt}{\lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{+T} a_1^2 \sin^2 (\omega_1 t + \phi_1) \, dt}$$  

(3.38)

$$= \omega_1 \frac{\lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{+T} \cos^2 (\omega_1 t + \phi_1) \, dt}{\lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{+T} \sin^2 (\omega_1 t + \phi_1) \, dt}$$  

(3.39)

$$= \omega_1.$$  

(3.40)

*The equivalence of (3.28) on the preceding page and (3.32) on the previous page implies that $\|x'\|^2 = -\langle x'', x \rangle$ which can be easily shown using integration by parts.
3.4. Derivative-based features

Following the definition of $J_\omega^2$ (3.29) on page 39 for the example signal

\[
J_\omega^2 = \sqrt{\frac{\|x''_1\|^2}{\|x'_1\|^2} - \frac{\|x'_1\|^4}{\|x_1\|^4}} 
\]

(3.41)

\[
= \sqrt{\omega_1^4 - \omega_1^4} 
\]

(3.42)

\[
= 0. 
\]

(3.43)

This example shows that the derivative-based features reproduce exactly the original parameters of the signal (namely a single pure sinusoid of frequency $\omega_1$) while ignoring phase and amplitude.

For a slightly more complicated example we use the sum of two sinusoids

\[
x_2(t) = a_1 \sin (\omega_1 t + \varphi_1) + a_2 \sin (\omega_2 t + \varphi_2). 
\]

(3.44)

It is tedious but trivial to show that we get the following values for the features

\[
\overline{\omega}_2 = \sqrt{\frac{a_1^2 \omega_1^2 + a_2^2 \omega_2^2}{a_1^2 + a_2^2}} 
\]

(3.45)

\[
J_\omega^2 = \frac{a_1 a_2}{a_1^2 + a_2^2} (\omega_1^2 - \omega_2^2). 
\]

(3.46)

In this example amplitudes are relevant but phases are not.

One can interpret $\overline{\omega}_2^2$ as the weighted mean of the squares of the frequencies, where the weightings are the square of the amplitude (inverse of the “errors” in the weighted average in (3.7) on page 27).

Alternatively, we can rewrite these as

\[
\overline{\omega}_2^2 = \lambda_2 \omega_1^2 + (1 - \lambda_2) \omega_2^2 
\]

(3.47)

\[
J_\omega^2 = \lambda_2 (1 - \lambda_2) (\omega_1^2 - \omega_2^2)^2 
\]

(3.48)

where

\[
\lambda_2 = \frac{a_1^2}{a_1^2 + a_2^2}. 
\]

(3.49)

If both sinusoids have the same amplitude, then the values reduce to

\[
\overline{\omega}_{2,a_1=a_2} = \sqrt{\omega_1^2 + \omega_2^2} 
\]

(3.50)

\[
J_{\omega,2,a_1=a_2} = \frac{\omega_1^2 - \omega_2^2}{2}. 
\]

(3.51)

Here $\overline{\omega}_{2,a_1=a_2}$ is the quadratic mean of the frequencies, and $J_{\omega,2,a_1=a_2}$ is the
standard deviation of the square of the frequencies.

If we add a third sinusoidal term to the signal

\[ x_3(t) = a_1 \sin (\omega_1 t + \phi_1) + a_2 \sin (\omega_2 t + \phi_2) + a_3 \sin (\omega_3 t + \phi_3) \]  \hfill (3.52)

then we end up with the following values for the features

\[
\bar{\omega}_3^2 = \frac{a_1^2 \omega_1^2 + a_2^2 \omega_2^2 + a_3^2 \omega_3^2}{a_1^2 + a_2^2 + a_3^2},
\]  \hfill (3.53)

\[
\overline{\delta \omega_3^2} = \frac{a_1^2 a_2^2 (\omega_1^2 - \omega_2^2)^2 + a_2^2 a_3^2 (\omega_2^2 - \omega_3^2)^2 + a_3^2 a_3^2 (\omega_3^2 - \omega_3^2)^2}{(a_1^2 + a_2^2 + a_3^2)^2}.
\]  \hfill (3.54)

We can rewrite these as

\[
\bar{\omega}_3^2 = \lambda_3 \omega_1^2 + \mu_3 \omega_2^2 + \nu_3 \omega_3^2, \quad \overline{\delta \omega_3^2} = \lambda_3 \mu_3 (\omega_1^2 - \omega_2^2)^2 + \lambda_3 \nu_3 (\omega_1^2 - \omega_3^2)^2 + \mu_3 \nu_3 (\omega_2^2 - \omega_3^2)^2
\]  \hfill (3.55) \hfill (3.56)

where

\[
\lambda_3 = \frac{a_1^2}{a_1^2 + a_2^2 + a_3^2}, \quad \mu_3 = \frac{a_2^2}{a_1^2 + a_2^2 + a_3^2}, \quad \nu_3 = \frac{a_3^2}{a_1^2 + a_2^2 + a_3^2}
\]  \hfill (3.57)

and \( \lambda_3 + \mu_3 + \nu_3 = 1 \).

If all three signals have the same amplitude, we get

\[
\bar{\omega}_{3,a_1=a_2=a_3}^2 = \sqrt{\frac{\omega_1^2 + \omega_2^2 + \omega_3^2}{3}}
\]  \hfill (3.58)

\[
\overline{\delta \omega_{3,a_1=a_2=a_3}^2} = \sqrt{\frac{\omega_1^2 + \omega_2^2 + \omega_3^2 - 2\omega_1^2 \omega_2^2 - 2\omega_1^2 \omega_3^2 - 2\omega_2^2 \omega_3^2}{3}}
\]  \hfill (3.59)

Again, \( \bar{\omega}_{3,a_1=a_2=a_3} \) is the quadratic mean of the frequencies, and \( \overline{\delta \omega_{3,a_1=a_2=a_3}^2} \) is the standard deviation of the square of the frequencies.

Based on these examples I conjecture that, for \( x_n(t) \) defined in (3.33) on page 40,

\[
\overline{\omega_n^2} = \sum_{k=1}^{n} \lambda_k \omega_k^2\]

\[
\overline{\delta \omega_n^2} = \sum_{k=1}^{n-1} \sum_{j=k+1}^{n} \lambda_k \lambda_j (\omega_k^2 - \omega_j^2)^2
\]  \hfill (3.60) \hfill (3.61)

where

\[
\lambda_k = \frac{a_k^2}{\sum_{j=1}^{n} a_j^2}.
\]  \hfill (3.62)
3.4. Derivative-based features

Instead of the general quasiperiodic signal as defined in (3.33) we could use the general periodic signal

\[ x_m(t) = \sum_{k=1}^{m} a_k \sin (k\omega_0 t + \phi_k) \]  

(3.63)

where \( \omega_0 \) is the base frequency. Then we would get the following values for the features

\[ \bar{\omega}_m^2 = \omega_0^2 \sum_{k=1}^{m} k^2 \lambda_k \]  

(3.64)

\[ \bar{\delta \omega}_m^2 = \omega_0^2 \sum_{k=1}^{m-1} \sum_{j=k+1}^{m} \lambda_k \lambda_j (k^2 - j^2)^2 \]  

(3.65)

where \( \lambda_k \) is as defined in (3.62).

3.4.4 Time-scale features

Instead of viewing the \( \bar{\omega} \) and \( \bar{\delta \omega} \) features as frequencies, we could transform them into time-scale features. This would be useful because it is easier to understand a period of 100 days than a frequency of 0.01 cycles per day (or 0.06 radians per day). One way to transform them into the corresponding period or “time-scale” features is:

\[ \bar{\tau} = \frac{2\pi}{\bar{\omega}} \]  

(3.66)

\[ \bar{\delta \tau}^2 = \frac{\bar{\delta \omega}^2}{\bar{\omega}^2} \]  

(3.67)

which can be rewritten in terms of norms as

\[ \bar{\tau}_x = 2\pi \frac{||x||}{||x'||} \]  

(3.68)

\[ \bar{\delta \tau}^2_x = \sqrt{\frac{||x''||^2 ||x||^2}{||x'||^4} - 1}. \]  

(3.69)

We call \( \bar{\tau}_x \) the “time-scale” and \( \bar{\delta \tau}^2_x \) the “delta-time-scale”\(^\dagger\).

If we use the same quasiperiodic examples as the previous section, but replace \( \omega \) with \( \frac{2\pi}{\tau} \) we end up with:

\[ \bar{\tau}_1 = \tau \]  

(3.70)

\(^\dagger\)Correction to a previous paper: in [18, §2.2] we had \( \bar{\delta \tau}^2 = \frac{2\pi}{\bar{\delta \omega}^2} \) which behaves contrary to what we would want from the feature: as \( \bar{\delta \omega}^2 \) approaches zero (i.e., the signal becomes one pure frequency) then \( \bar{\delta \tau}^2 \) would approach infinity, when it should approach zero.
$$\overline{\delta \tau^2}_1 = 0$$  \hspace{1cm} (3.71)

$$\overline{\tau}_2^2 = \left( \frac{\lambda_2}{\tau_1^2} + \frac{1 - \lambda_2}{\tau_2^2} \right)^{-1}$$  \hspace{1cm} (3.72)

$$\overline{\delta \tau^2}_2 = \frac{\lambda_2 (1 - \lambda_2)}{\left( \frac{\lambda_2}{\tau_1^2} + \frac{1 - \lambda_2}{\tau_2^2} \right)^2}$$  \hspace{1cm} (3.73)

$$\overline{\tau}_3^2 = \left( \frac{\lambda_3}{\tau_1^2} + \frac{\mu_3}{\tau_2^2} + \frac{\nu_3}{\tau_3^2} \right)^{-1}$$  \hspace{1cm} (3.74)

$$\overline{\delta \tau^2}_3 = \frac{\lambda_3 \mu_3 \left( \frac{1}{\tau_1^2} - \frac{1}{\tau_2^2} \right)^2 + \lambda_3 \nu_3 \left( \frac{1}{\tau_1^2} - \frac{1}{\tau_3^2} \right)^2 + \mu_3 \nu_3 \left( \frac{1}{\tau_2^2} - \frac{1}{\tau_3^2} \right)^2}{\left( \frac{\lambda_2}{\tau_1^2} + \frac{\nu_3}{\tau_2^2} + \frac{\mu_3}{\tau_3^2} \right)^2}.$$  \hspace{1cm} (3.75)

If we use the general periodic signal as defined in (3.63) on the preceding page, and replacing $\omega_0$ with $\frac{2\pi}{\tau_0}$ then we get the following for the time-scale features

$$\overline{\tau}_m = \frac{\tau_0}{\sum_{k=1}^{m} k^2 \lambda_k}$$  \hspace{1cm} (3.76)

$$\overline{\delta \tau^2}_m = \sqrt{\frac{\sum_{k=1}^{m-1} \sum_{j=k+1}^{m} \lambda_k \lambda_j (k^2 - j^2)^2}{\sum_{k=1}^{m} k^2 \lambda_k}}.$$  \hspace{1cm} (3.77)

Note that $\overline{\delta \tau^2}_m$ is independent of the base frequency.

One could interpret $\overline{\tau}$ as the average period, and $\overline{\delta \tau^2}$ as the deviation from that period.

We use the frequency-based features in the remainder of this thesis since the time-scale features do not seem to have such a nice interpretation in the examples.

### 3.4.5 Using the derivative-based features on a sampled signal

Since we are working with a time series, which we assume is an irregular sampling of the continuous model $x$ we can only approximate the true norms and derivatives of the signal.

For the norm of the signal we make the assumption that we have enough “periods” of the signal for the average of the sampled signal to be close to the true average, as well as close enough samples for the norm to be accurate.

We define this semi-norm of a continuous signal $x(t)$

$$|x|^2 = \frac{1}{t^{(N)} - t^{(1)}} \int_{t^{(1)}}^{t^{(N)}} (x(t) - m(x))^2 \, dt$$  \hspace{1cm} (3.78)
which removes the contribution from the mean. The integral is approximated for a time series $X$ using the trapezoidal method from (3.5) on page 27, which expands to

$$|X|^2_{\text{trap}} = \frac{\sum_{i=1}^{N} \frac{1}{2} (t^{(i+1)} - t^{(i)}) \left( (X^{(i)} - m_X)^2 + (X^{(i+1)} - m_X)^2 \right)}{t^N - t^1}.$$  

(3.79)

The derivatives of the signal are approximated using a numerical difference method described in [10], which works here because the sampling is usually close to equidistant. This required the use of a smoothing parameter $s$, which is the number of points to average over; this was chosen after extensive calculations of the norms for all the Wood data for the original signal and first and second derivatives with $s$ varying from 1 to 16. When comparing the behaviour of all the stars, $s = 8$ seemed to provide the least noisy results without oversmoothing and is used for the rest of this thesis. Adaptively choosing the smoothing parameter would be worthwhile investigating, since good choices seem to depend strongly upon the signal and the number of derivatives required.

The norms of the derivatives are then calculated using the semi-norm in (3.79).

### 3.4.6 Applying the derivative based features on MA-CHO data

Due to the very large number of uncorrelated stars in the 20k data, it was not used for frequency calculations. Histograms of the calculated features of the red band of the Wood dataset are shown in Figure 3.7 on the next page, as well as frequency versus frequency spread. The blue band is not shown because they are very similar.

The average frequency has a curious distribution. The time-scale variant from (3.66) on page 43 ranges from about 55 to 750 days; this suggests that the average frequency is not very accurate as an estimate of the frequency, especially for the faster varying stars. However, the feature does seem to separate the stars reasonably well since “lower average frequency” stars are much more likely to be clearly Long Period Variables than not, and vice versa.

The frequency spread appears to have a bimodal distribution. The delta-time-scale from (3.67) on page 43 ranges from about 0.6 to 17, with most stars clustered between one and three.

Since frequency spread and frequency seem to be correlated, the fourth
plot shows relative frequency spread versus frequency, where relative frequency spread is defined as

$$\text{rdf}_x = \frac{\delta \omega_x^2}{\overline{\omega}_x^2}. \quad \text{(3.80)}$$

In this plot there is evidence for a quadratic relationship between frequency and frequency spread. This is interesting because we defined “delta-time-scale” $\overline{\delta \tau^2}$ as $\delta \omega^2 / \overline{\omega}^2$ in (3.67) on page 43; this means that the distribution of delta-time-scale is reasonably independent for varying frequency. One possible reason for this is if all the variable stars varied periodically then
3.5 \textit{Scatter-plot matrices of features} \\

the delta-time-scale is independent of base frequency: see (3.77) on page 44.

Since the blue and red frequencies, and the blue and red relative-frequency-spreads are so similar, like the log-relative-amplitude in §3.3.3 on page 31 we apply a "colour" transformation to produce the frequency colour (freqc) and relative-frequency-spread colour (rdfc)

\[
\text{freqc}_{BR} = \overline{\omega}_B - \overline{\omega}_R \\
\text{rdfc}_{BR} = \text{rdf}_B - \text{rdf}_R. 
\]

These are not particularly easy to interpret, and have very high peaked distributions. The stars on the edges of these distributions may be interesting (see §4.6 on page 71). The distributions of these features are shown in the Wood scatter-plot matrix in §3.5.

3.5 \textbf{Scatter-plot matrices of features}

The features described in the previous two sections were applied to the Wood and 20k datasets. Subsets of those features are shown as scatter-plot matrices in Figure 3.8 on the next page and Figure 3.9 on page 49.
Figure 3.8: Scatter plot matrix of 9 selected features of the Wood MACHO dataset.
Figure 3.9: Scatter plot matrix of 5 selected features of the 20k MACHO dataset.
Chapter 4

Clustering and Classification

Clustering and classification refer to groups of machine learning algorithms. Classification is "supervised", which means it requires previously labelled data, and then tries to build a model which can predict the label of new examples. Clustering is unsupervised and attempts to divide the set of examples into groups which are more or less similar.

In this chapter we will introduce clustering (§4.1) and classification (§4.2 on the following page) and discuss some of the standard clustering and classification techniques. Then we will discuss in detail the k-means clustering algorithm in §4.3 on page 53, and discuss the results when applying k-means to the features of the Wood dataset in §4.3.3 on page 56. We describe in detail the design of a k-means clustering variant called "k-means-min-p" (§4.4 on page 58) which adds an additional parameter which specifies the minimum number of points in each cluster.

In §4.5 on page 69 we then introduce a generic iterative classification and clustering algorithm which we term "boxing". We then apply the boxing algorithm to the Wood dataset in several ways to demonstrate the effectiveness of the features: in §4.6 on page 71 we select examples from the stars on the boundaries of feature-space, and in §4.7 on page 72 we use an additive model predictor to order and select stars for comparison.

4.1 Clustering

Clustering applies to a group of algorithms which are designed to group similar items together, where the meaning of "similar" has to be specified. For a recent and comprehensive survey of the methods see Jain et al. [36]. There are two main types of clustering algorithms: partitioning and hierarchical methods.

Partitioning methods divide the examples into a number of groups, at-
tempting to minimise some criterion about the partitioning. Examples of clustering algorithms include $k$-means, $k$-medoids and ISODATA. The $k$-means method is described in detail in §4.3. Recently developed methods to cope with very large datasets include BIRCH [68], CLIQUE [2] and PROCLUS [1].

Most of these methods take an initial partitioning and iteratively improve the partitioning by reassigning points to other clusters, or possibly creating new clusters or removing clusters. Generating good initial partitions is a difficult problem in itself and is studied in [5, 27, 15]. Hierarchical methods produce a tree, for which each leaf is an example and each node combines several (usually two) leaves or nodes into a cluster. If we start from the bottom of the tree we have $N$ clusters (for each of the $N$ examples) and as we move up the tree the closest leaves and nodes are progressively joined—where “closest” depends on the particular algorithm. Some examples of hierarchical methods include “single-link”, “complete-link” and “group-average” methods. Recently developed methods include Chameleon [37] and OPTICS [11]. We do not consider hierarchical clustering.

An aim of clustering is to try to understand the groups of objects within the data, and to get indicative examples of those groups. The result of a clustering is also often used to label each example so that a classifier can be built. This is the subject of the next section.

4.2 Classification

Given a feature vector of values, classification algorithms produce a model which predicts what a particular value—the label—will be. Sometimes the model produced is itself interesting and understandable (e.g., decision trees) but other times the model is usually intractable and the use is only for prediction (e.g., neural networks). Ripley [52] covers many of these algorithms; we only present some of the more important ones.

Decision trees build a tree divisively by splitting the examples such that some measure of impurity with regard to the distribution of label values is reduced. Standard algorithms are CART [16] and C4.5 [48].

Generalised additive models [33] are an extension to multiple linear regression, where each dependent variable $X_i$ is modelled with a general function $f_i$; the $f_i$'s may be completely different types of functions for each $i$. The model is of the form

$$y = \beta_0 + \sum_{i=1}^{p} f_i(x_i)$$  \hspace{1cm} (4.1)
where $\beta_0$ is a constant offset, and each of the $f_i$'s model the dependence output variable $y$ has on $x_i$.

The generalised additive model approach is particularly well suited for data mining since it is highly scalable and parallelisable [43, 35].

Boosting [29] is a general methodology which is designed to improve the performance of simple classifiers. Essentially it involves iteratively classifying, testing and re-weighting our labelled examples, where incorrectly classified examples are given higher weighting for the next iteration. The boosted classifier is defined as the weighted votes of each of the classifiers built during the iterative process. Boosting can provide dramatic increases in performance for some classifiers.

We use a boosted generalised additive model classifier in the star-searcher in §4.7 on page 72.

4.3 k-means

The clustering algorithm k-means attempts to find a partitioning of the $N$ given points into a fixed number ($K$) of clusters such that the within-cluster variance is minimised [41].

Each point $p$ has a location $X_p$ which we represent as a vector in $d$-dimensional space.

A partition can be given in two forms: a set of points $V_l$ for each cluster $l$, or a cluster $l = B_p$ for each point $p$. We use both interchangeably to simplify notation (see the next two equations).

The cluster centre is defined as the average of all points in the cluster

$$C_l = \frac{\sum_{p \in V_l} X_p}{Q_l}$$  \hspace{1cm} (4.2)

where $Q_l$ is the number of points in cluster $l$.

The variance $E$ is defined as the sum of the squares of the distances from each point to its cluster centre

$$E = \sum_{p=1}^{N} \|X_p - C_{B_p}\|^2$$  \hspace{1cm} (4.3)

where the norm is the 2-norm. One reason for using the sum-of-squared-distance is that it means the change in variance for moving one point to a different cluster can be calculated in constant time.
4.3.1 \(k\)-means algorithm

Essentially, the \(k\)-means algorithm iteratively tries to reassign all points to a better cluster, until no reassignments occur. This procedure will find a local minimum of \(E\). How well it does depends highly on the initial partition given to it and the ordering of the points.

The change in variance if we reassign point \(p\) from cluster \(l\) to \(k\) is [31, p.85]

\[
\delta E(p, l, k) = \| X_p - C_k \|^2 \frac{Q_k}{Q_k + 1} - \| X_p - C_l \|^2 \frac{Q_l}{Q_l - 1}.
\] (4.4)

This is derived in §4.3.2. Note that this can be calculated in constant time (ignoring the dimension); indeed, the second half of (4.4) is independent of \(k\) and so can be calculated once, before calculating \(\delta E\) for all \(j \neq l\) (in Algorithm 4.1).

Given a point \(p\) in cluster \(l\), then we need to find a cluster \(k\) for which the change in variance for moving \(p\) to cluster \(k\) is the most negative. This search algorithm (\(k\)-means-find-reassign) is shown in Algorithm 4.1. If it does not find any better cluster, the result is \(l\).

\begin{algorithm}
1: \(\delta E_{\min} \leftarrow 0\)
2: \(k \leftarrow l\)
3: for all clusters \(j \neq l\) do
4: \(\delta E(p, l, j)\)
5: if \(\delta E < \delta E_{\min}\) then
6: \(\delta E_{\min} \leftarrow \delta E\)
7: \(k \leftarrow j\)
8: end if
9: end for
\end{algorithm}

The \(k\)-means algorithm is shown in Algorithm 4.2 [31, pp.85–86]. It takes an initial partitioning \(B\) and the result is the final partitioning.

\begin{algorithm}
1: initialise \(C_l\) and \(Q_l\) for all clusters
2: repeat \{outer loop:\}
3: for all points \(p\) do \{inner loop:\}
4: \(l \leftarrow B_p\)
5: \(k \leftarrow k\)-means-find-reassign\((p, l)\)
6: if \(k \neq l\) then
7: \(B_p \leftarrow k\)
8: adjust centres \(C_l\) and \(C_k\) and cluster sizes \(Q_l\) and \(Q_k\)
9: end if
10: end for
11: until no reassignments occur in the inner loop
\end{algorithm}
Hartigan and Wong [32] describe a variant of the k-means algorithm which remembers the next best possible cluster for each point and maintains a “live set” of clusters and uses this information to make the algorithm more efficient.

4.3.2 Change in variance for a reassignment

We can rewrite the variance $E$ as the sum of variances of all clusters $E(l)$, where

$$E(l) = \sum_{p \in V_l} \|X_p - C_l\|^2.$$ \hfill (4.5)

If we reassign point $p$ from cluster $l$ to $k$ then we need to adjust the cluster memberships, sizes and centres. We represent the adjusted clusters by a subscripted $-p$ or $+p$: e.g., $l_{-p}$ and $k_{+p}$.

The cluster memberships $\mathcal{V}$ involve removing and adding $p$: $\mathcal{V}_{l_{-p}} = \mathcal{V}_l \setminus \{p\}$, $\mathcal{V}_{k_{+p}} = \mathcal{V}_k \cup \{p\}$. Obviously, $Q_{l_{-p}} = Q_l - 1$ and $Q_{k_{+p}} = Q_k + 1$. The cluster centres can also be calculated:

$$C_{l_{-p}} = \frac{C_l Q_l - X_p}{Q_l - 1} = C_l - \frac{X_p - C_l}{Q_l - 1}.$$ \hfill (4.6)

$$C_{k_{+p}} = \frac{C_k Q_k + X_p}{Q_k + 1} = C_k + \frac{X_p - C_k}{Q_k + 1}.$$ \hfill (4.7)

The change in overall variance $E$ is entirely composed of the changes in variances for the clusters $l$ and $k$

$$\delta E = E(\mathcal{V}_{l_{-p}}) + E(\mathcal{V}_{k_{+p}}) - E(\mathcal{V}_l) - E(\mathcal{V}_k).$$ \hfill (4.8)

The variance in cluster $l_{-p}$ is

$$E(\mathcal{V}_{l_{-p}}) = \sum_{n \in \mathcal{V}_{l_{-p}}} \|X_n - C_{l_{-p}}\|^2$$ \hfill (4.9)

substitute (4.6):

$$= \sum_{n \in \mathcal{V}_{l_{-p}}} \left\|X_n - C_l + \frac{X_p - C_l}{Q_l - 1}\right\|^2$$ \hfill (4.10)

expand $\mathcal{V}_{l_{-p}} = \mathcal{V}_l \setminus \{p\}$:

$$= \sum_{n \in \mathcal{V}_l} \left\|X_n - C_l + \frac{X_p - C_l}{Q_l - 1}\right\|^2 - \left\|X_p - C_l + \frac{X_p - C_l}{Q_l - 1}\right\|^2$$ \hfill (4.11)
expand $\|a + b\|^2 = \|a\|^2 + \|b\|^2 + 2 \langle a, b \rangle$, where $\langle a, b \rangle$ is the scalar product:

$$
\begin{align*}
\sum_{n \in V_l} \left( \|X_n - C_l\|^2 + \left\| \frac{X_p - C_l}{Q_l} \right\|^2 + 2 \left\langle X_n - C_l, \frac{X_p - C_l}{Q_l} \right\rangle \right) \\
- \frac{Q_l^2}{(Q_l - 1)^2} \|X_p - C_l\|^2
\end{align*}
$$

(4.12)

$$
\begin{align*}
\sum_{n \in V_l} \|X_n - C_l\|^2 + \frac{Q_l}{(Q_l - 1)^2} \|X_p - C_l\|^2 \\
+ 2 \left\langle \sum_{n \in V_l} (X_n - C_l), \frac{X_p - C_l}{Q_l - 1} \right\rangle - \frac{Q_l^2}{(Q_l - 1)^2} \|X_p - C_l\|^2
\end{align*}
$$

(4.13)

and since $\sum_{n \in V_l} (X_n - C_l) = 0$:

$$
E(l_{-p}) = E(l) - \frac{Q_l}{Q_l - 1} \|X_p - C_l\|^2.
$$

(4.14)

Using a similar argument we get the following for $E_{k+p}$:

$$
E(k_{+p}) = E(k) + \frac{Q_k}{Q_k + 1} \|X_p - C_k\|^2.
$$

(4.15)

The change in overall variance is then

$$
\delta E = \frac{Q_k}{Q_k + 1} \|X_p - C_k\|^2 - \frac{Q_l}{Q_l - 1} \|X_p - C_l\|^2
$$

(4.16)

as in (4.4) on page 54.

4.3.3 Applying $k$-means to the Wood feature-set

SGI's MineSet program (version 2.6) [57] contains an implementation of the $k$-means algorithm.

MineSet also has an “iterative” $k$-means algorithm which effectively runs $k$-means for all $K = K_{\text{min}}, \ldots, K_{\text{max}}$ where $K_{\text{min}}$ and $K_{\text{max}}$ are user-supplied parameters. For the $K + 1$-th clustering it uses the $K$-th partitioning as an initial partitioning, with the biggest cluster—cluster $l$ with largest $E(l)$—split into two. The variance $E_K$ for each clustering is calculated from (4.3) on page 53, which always decreases for increasing $K$. The iterative method chooses the $K$ for which $E_K$ is closest to $E_{K_{\text{min}}} - \xi (E_{K_{\text{min}}} - E_{K_{\text{max}}})$. $\xi$ is the user-supplied “choice point” ranging from 0 to 1 (0 chooses $K_{\text{min}}, 1$ chooses...
Figure 4.1: The clusters of each star from k-means clustering with $K = 5$ on the Wood feature-set. Axes shown are red Ira and red freq.

$K_{\text{max}}$).

We used the MineSet iterative k-means algorithm on the Wood feature-set. With a first attempt with the default values of $K_{\text{min}} = 1$, $K_{\text{max}} = 10$ and $\xi = 0.5$, the “optimal” $K$ was two. The resulting clusters were not very interesting so we chose $K_{\text{max}} = 15$ and $\xi = 0.75$. This came up with an “optimal” $K$ of five. Each star was labelled with its closest cluster.

Figure 4.1 shows the cluster for each star with red log-relative-amplitude plotted against red average frequency.

After viewing the light curves of randomly selected stars from each cluster, it seemed that each cluster in general contained stars which were similar, but they also contained stars which were quite different. The light curves of three examples each from two of the most separated clusters (clusters 3 and 4) are shown in Figure 4.2 on the next page.

Increasing the numbers of clusters did not seem to help, either, suggesting there was a reasonable amount of overlap of star types within the features or the number of clusters was too small. It is possible, however,
that we did not increase the numbers of clusters enough.

Another problem observed with using clustering algorithms is with data that does not group into natural clusters, such as the Wood feature-set. Each of the Wood features are generally continuous with one main mode: compare the Wood scatter-plot matrix in Figure 3.8 on page 48 with the 20k feature-set scatter-plot matrix in Figure 3.9 on page 49 in which there are two clear clusters. With these sorts of data, clustering algorithms tend to “force” the data into artificial and not particularly interesting clusters. \( k \)-means also produces convex clusters; other clustering algorithms, particularly hierarchical methods like Chameleon [37], can produce irregularly-shaped clusters which may be more appropriate for these data.

4.4 \( k \)-means with minimum populations

The \( k \)-means algorithm has the problem that it may find a partitioning that contains clusters with very few points. If we are fairly certain of the number and the sizes of the clusters that ought to be within our data then we may
want to force each cluster in the k-means algorithm to have a minimum number of points (minimum population). The k-means-min-\(p\) algorithm described here is an extension to k-means which does exactly that*.

Standard k-means iteratively attempts to move each point from one cluster to another, seeing if that reduces the variance. With minimum population, if a point belongs to a cluster that has minimum population it cannot be removed. Instead we look for a point in another cluster with which to swap cluster assignments such that the variance is reduced.

In a similar theme Banfield and Bassill [13] describe an algorithm which can be used as a k-means replacement but also considers all possible “swops”† between all points in all other clusters. This search is very expensive; k-means-min-\(p\) attempts to limit this search by reordering clusters and other heuristics.

### 4.4.1 k-means-min-\(p\) algorithm

We present a new algorithm called k-means-min-\(p\) which is an extension of k-means but limits the minimum population of each cluster to a user-defined value \(P\). It attempts to minimise the search for swaps by reordering clusters—and their points—by their size during an iteration, but retains the property that each point will be checked exactly once in each iteration. We use the same notation as with the k-means algorithm (§4.3 on page 53). We call a cluster that contains exactly \(P\) points a minimal cluster.

The idea is to avoid checking for swaps, since that is more expensive than checking for reassignments:

- the cost to find the best reassignment of point \(p\) is \(O(K)\) (more precisely \(O(K - 1)\));

- the cost to find the best swap for point \(p\) is \(O(KN)\) (more precisely \(O((K - 1)(N - Q_l))\), where \(l\) is the cluster that \(p\) currently belongs to).

If we can (a), reduce the number of checks for swaps by reducing the number of minimal clusters, and (b), reduce the number of points to check for a swap, then this should speed up the algorithm.

We can do (a) by trying to reassign points from the largest clusters first. These points may be assigned to smaller clusters, increasing their size. Hopefully this way we can avoid many unnecessary checks for swaps.

*This idea was suggested by Michael Powell [47].
†The word “swop” is a variant of “swap”.
We can do (b) by approximations only: for example see the circle property in §4.4.4 on page 67. The algorithm also attempts to do this by firstly only swapping with points in other minimal clusters.

If there is a minimal cluster \( l \), with a point \( p \) that can be swapped with another point \( o \) in cluster \( k \) that is not minimal then it is likely that if we check \( o \) first it would be assigned to the minimal cluster \( l \); now that cluster \( l \) is not minimal, point \( p \) can be safely reassigned (possibly to cluster \( k \)). This case is examined further in §4.4.4 on page 68.

For the case above, if \( k \) was also minimal then we reach the true swapping case: in this case we search through all other minimal clusters for points to swap with. This should be much less than all other clusters (unless \( P \times K \) is nearly \( N \)).

If we want to visit points in order of cluster size, then to check each point exactly once per iteration requires storing lists of points which have already been checked in this iteration. This is because clusters change size during an iteration and may be reordered.

In order to do this we have two lists of clusters: \( \mathcal{L} \) which contains the clusters that still have points that have yet to be checked in this iteration, and \( \mathcal{M} \) which contains the clusters in which all points have been checked. Both lists are sorted in decreasing order of cluster size. We define \( \mathcal{L}.\operatorname{insertsorted}(l) \) as the sorted insertion of cluster \( l \) into cluster list \( \mathcal{L} \).

Each cluster \( l \) has two lists which contain the points in that cluster: \( \mathcal{U}_l \) contains the points that have yet to be checked in this iteration and \( \mathcal{V}_l \) contains the points that have already been checked. The lists are stored in no particular order. We define \( \mathcal{U}_l.\operatorname{insert}(p) \) as the insertion of point \( p \) into point list \( \mathcal{U}_l \).

We also define \( g \leftarrow \mathcal{G}.\operatorname{pop}() \) as the removal of the first element of \( \mathcal{G} \) (where \( \mathcal{G} \) is either of the above types of lists).

Each cluster also has size (or population) \( Q_l \) and centre \( C_l \). The location of a point \( p \) is \( X_p \).

Given a point \( p \) in cluster \( l \) which is not minimal (i.e., \( Q_l > P \)) then the search for the best cluster to which point \( p \) should belong to is the same as for standard \( k \)-means, so we can use Algorithm 4.1 on page 54. To cope with the book-keeping of reordering points and clusters is somewhat more complicated and this is shown in Algorithm 4.3.

The algorithm starts with point \( p \) having already been removed from \( \mathcal{U}_l \) and \( l \) having already been removed from \( \mathcal{L} \). The first line uses the standard \( k \)-means-find-reassign algorithm to find the best cluster to which point \( p \) could be reassigned. If the current cluster is the best one, then lines 8–9 firstly puts \( p \) into the checked list \( \mathcal{V}_l \) and then reinserts \( l \) into \( \mathcal{L} \) or \( \mathcal{M} \).
Algorithm 4.3 $k$-means-min-$p$-reassign($p, l$)

1: $k \leftarrow k$-means-find-reassign($p, l$)
2: if $k \neq l$ then {assign point to $k$}
3: find and remove cluster $k$ from $\mathcal{L}$ or $\mathcal{M}$.
4: $\mathcal{V}_k.$ insert($p$)
5: recalculate centres and sizes of $k$ and $l$.
6: reinsert($k$), reinsert($l$)
7: else {$k = l$; no reassignment necessary;}
8: $\mathcal{V}_l.$ insert($p$)
9: reinsert($l$)
10: end if

depending on whether the cluster has any unchecked points left. If there is
another cluster $k$ which is better, then lines 3–6 deal with the housekeeping
for adding $p$ to cluster $k$.

Algorithm 4.4 is the helper algorithm which reinserts $l$ into $\mathcal{L}$ or $\mathcal{M}$
depending on whether the cluster has any unchecked points.

Algorithm 4.4 reinsert($l$)

1: if $\mathcal{U}_l = \emptyset$ then
2: $\mathcal{M}.$ insertsorted($l$)
3: else
4: $\mathcal{L}.$ insertsorted($l$)
5: end if

If $l$ is minimal, however, then we need to look for a swap point such
that the change in variance is negative. There are many possible ways to
do this, depending on whether we want to find the best swap point or any
swap point and whether we want to limit the search. Some of these options
are described in §4.4.3 on page 64.

The change in variance if we swap $p$ and $o$ between clusters $l$ and $k$
respectively is

$$
\delta E(p, l, o, k) = \|X_p - C_k\|^2 + \|X_o - C_l\|^2 - \|X_p - C_l\|^2
- \|X_o - C_k\|^2 - \|X_p - X_o\|^2 \left(\frac{1}{Q_l} + \frac{1}{Q_k}\right).
$$

We derive this in §4.4.2 on the next page. Notice that this equation is
symmetric with regards to all factors, so swapping $p$ and $o$ is the same as
swapping $o$ and $p$ (which is logical).

The $k$-means-min-$p$-swap algorithm searches for a point $o$ with which
to swap cluster assignments with $p$; it is shown in Algorithm 4.5 on the
following page. At this point in an iteration, all points in non-minimal
clusters have been checked.
Algorithm 4.5 k-means-min-p-swap(p, l)

1: search through all unchecked points not in cluster l and find the best swap (cluster k, point o).
2: if there is a swap o which improves the variance then
3: \( U_k.\) insert(o), \( V_k.\) insert(p)
4: recalculate centres of k and l
5: reinsert(l), reinsert(k)
6: else {there is no swap which improves the variance:}
7: \( V_l.\) insert(p)
8: reinsert(l)
9: end if

The first line in this algorithm implies a particular approximation for the search for a good swap since it only searches through unchecked points: this means it only searches through other minimal clusters. This approximation is discussed in §4.4.3 on page 64 and other approximations are discussed in §4.4.4. The housekeeping is as for k-means-min-p-reassign except that if a swap does occur then we also need to deal with inserting o into \( U_k \) (since o has not really been checked) and reinserting k.

The overall k-means-min-p algorithm is shown in Algorithm 4.6. It is given an initial cluster assignment set of lists \( V_l \) (abbreviated as \( V \)). The result is the new cluster assignment.

The first few lines (lines 1–5) initialise the algorithm by calculating the size and centre of each cluster and populating the sorted cluster list \( M \).

At the start/end of the outer loop (and the end of each inner loop) all the \( U_l \)’s are empty and the \( V_l \)’s hold the cluster assignments. To restart the inner loop requires reassigning the checked lists to the unchecked lists. This is what lines 7–12 do.

At each iteration of the inner loop we find an unchecked point p from a cluster l which still contains unchecked points (lines 14–15). Depending on the size of cluster l we then try to find a reassignment or a swap for p (lines 16–20).

4.4.2 Change in variance for a swap

Due to the symmetric nature of a swap, only one half of the swap will be dealt with in the derivation.

The adjusted cluster l is represented like this: \( l_{p+o} \). The cluster membership \( V_l \) involves adding and removing a point: \( V_{l_{p+o}} = V_l \setminus \{ p \} \cup \{ o \} \). Obviously, the cluster size is unchanged. The new cluster centre is

\[
C_{l_{p+o}} = \frac{C_lQ_l - X_p + X_o}{Q_l} = C_l - \frac{X_p - X_o}{Q_l}. \tag{4.18}
\]


Algorithm 4.6 $V \leftarrow k$-means-min-$p(V)$

Require: $Q_l \geq P, \forall l$
Ensure: $Q_l \geq P, \forall l$

1: $M \leftarrow \emptyset$
2: for all clusters $l$ do \{initialise:\}
3: calculate cluster size $Q_l$ and centre $C_l$
4: $M$.insertsorted($l$)
5: end for
6: repeat \{outer loop:\}
7: for all clusters $l$ do \{initialise:\}
8: $\mathcal{U}_l \leftarrow \mathcal{V}_l$
9: $\mathcal{V}_l \leftarrow \emptyset$
10: end for
11: $\mathcal{L} \leftarrow M$
12: $M \leftarrow \emptyset$
13: while $\mathcal{L} \neq \emptyset$ do \{inner loop:\}
14: $l \leftarrow \mathcal{L}$.pop()
15: $p \leftarrow \mathcal{U}_l$.pop()
16: if $Q_l > P$ then
17: $k$-means-min-$p$-reassign($p, l$)
18: else \{$Q_l = P$\}
19: $k$-means-min-$p$-swap($p, l$)
20: end if
21: end while
22: until the inner loop does no reassignments and/or swaps

In this derivation we use the same method as in (4.9) to (4.14) on pages 55–56.

The change in overall variance is

\[
\delta E = E(l_{p+o}) + E(k_{o+1p}) - E(l) - E(k). \tag{4.19}
\]

The variance of cluster $l_{p+o}$ is

\[
E(l_{p+o}) = \sum_{n \in \mathcal{V}_{l_{p+o}}} \| X_n - C_{l_{p+o}} \|^2 \tag{4.20}
\]

\[
= \sum_{n \in \mathcal{V}_{l_{p+o}}} \left\| X_n - C_l + \frac{X_p - X_o}{Q_l} \right\|^2 \tag{4.21}
\]

\[
= \sum_{n \in \mathcal{V}_l} \left| X_n - C_l + \frac{X_p - X_o}{Q_l} \right|^2 - \left| X_p - C_l + \frac{X_p - X_o}{Q_l} \right|^2 \tag{4.22}
\]

\[
+ \left| X_o - C_l + \frac{X_p - X_o}{Q_l} \right|^2 \tag{4.23}
\]

\[= \Sigma - \Upsilon + \Phi. \]
\[
\Sigma = \sum_{n \in V_l} ||X_n - C_l||^2 + Q_l \left( \frac{X_p - X_o}{Q_l} \right)^2 \\
+ 2 \left( \sum_{n \in V_l} (X_n - C_l), \frac{X_p - X_o}{Q_l} \right)
\]
\[= E(l) + \frac{1}{Q_l} ||X_p - X_o||^2; \tag{4.25}\]
\[
\Upsilon = ||X_p - C_l||^2 + \left| \frac{X_p - X_o}{Q_l} \right|^2 + 2 \left( X_p - C_l, \frac{X_p - X_o}{Q_l} \right); \tag{4.26}
\]
\[
\Phi = ||X_o - C_l||^2 + \left| \frac{X_p - X_o}{Q_l} \right|^2 + 2 \left( X_o - C_l, \frac{X_p - X_o}{Q_l} \right); \tag{4.27}
\]
\[
\Phi - \Upsilon = ||X_o - C_l||^2 - ||X_p - C_l||^2 + 2 \left( X_o - X_p, \frac{X_p - X_o}{Q_l} \right) \]
\[= ||X_o - C_l||^2 - ||X_p - C_l||^2 - \frac{2}{Q_l} ||X_p - X_o||^2; \tag{4.29}\]
\[
E(l_{-p+o}) = (4.25) + (4.29) \]
\[= E(l) + ||X_o - C_l||^2 - ||X_p - C_l||^2 - \frac{1}{Q_l} ||X_p - X_o||^2. \tag{4.30}\]

Similarly, for cluster \(k_{-o+p}\),
\[
E(k_{-o+p}) = E(k) + ||X_p - C_k||^2 - ||X_o - C_k||^2 - \frac{1}{Q_k} ||X_o - X_p||^2. \tag{4.32}\]

The change in overall variance is then
\[
\delta E = ||X_o - C_l||^2 - ||X_p - C_l||^2 + ||X_p - C_k||^2 - ||X_o - C_k||^2 \\
- \left( \frac{1}{Q_l} + \frac{1}{Q_k} \right) ||X_p - X_o||^2 \tag{4.33}\]
as in (4.17) on page 61.

This can be verified by repeated use of the change in variance for a reassignment from (4.4) on page 54; this is left as an exercise for the reader.

### 4.4.3 Comments on the algorithm

Given \(p\), if \(o\) is the the best swap out of minimal clusters, is it possible for there to be a point \(q\) in another minimal cluster which can swapped with \(o\) or \(p\) (after they have been swapped)? The answer is yes: put \(l, o, p, q, k\) in a line; \(p\) and \(q\) belong to \(l\) and \(o\) belongs to \(k\). If we start with \(p\), then the best swap is with \(o\). If we then look at \(q\), it is better to swap with \(p\). However, since a swap has occurred the algorithm will re-iterate, and hence will look at all three points again. In line 3 of Algorithm 4.5 on page 62 we mark \(o\) as unchecked to emphasise the exactly-once per iteration idea, although
this may generate unnecessary extra work. If we use the heuristics described below then where \( o \) goes would depend on whether it was previously checked or not.

The only problem that can occur is if we get no reassignments in an iteration and we get down to the minimal clusters. It may make sense at this point to search through all points. The reason we do not search through all points all the time is because of the cost involved. Also worth noting is that as the algorithm currently stands, once we have reached the last unchecked minimal cluster in an iteration then there is no work to do, since there are no unchecked points in another cluster.

Another alternative is not to try to find the best swap point but to try to find any swap point which reduces the variance; this is investigated next.

### 4.4.4 Approximations

Instead of trying to find the best swap point (which is very costly) we could try to find any swap point which reduces the variance (i.e., \( \delta E < 0 \)). If a swap occurs, the algorithm will re-iterate and so should continue towards a minima of \( E \). If we have heuristics which limit the search space at the expense of accuracy then we may not reach a true minima.

An example of such a heuristic is to search only other minimal clusters for a swap point (as the \( k \)-means-min-\( p \) algorithm does). This relies on \( PK \ll N \)—so that most clusters are not minimal—and the reordering of clusters in the algorithm—so that points in non-minimal clusters that would be good swap points with points in minimal clusters are reassigned to those minimal clusters (and hence make them non-minimal).

Other insights into the boundary of a swap are interesting. By expanding out the squared norms and combining the scalar products, we can rewrite (4.17) on page 61 as

\[
\delta E(p, l, o, k) = 2 \langle X_p - X_o, C_l - C_k \rangle - \|X_p - X_o\|^2 \left( \frac{1}{Q_l} + \frac{1}{Q_k} \right). \tag{4.34}
\]

If we set \( \delta E(p, l, o, k) < 0 \), then we get

\[
2 \langle X_p - X_o, C_l - C_k \rangle < \|X_p - X_o\|^2 \left( \frac{1}{Q_l} + \frac{1}{Q_k} \right) \tag{4.35}
\]

as the boundary. Since \( ||\cdot||^2 \geq 0 \), it is sufficient for

\[
\langle X_p - X_o, C_l - C_k \rangle < 0. \tag{4.36}
\]
This can be interpreted as so: if the vectors $X_p - X_o$ and $C_l - C_k$ point in approximately the opposite direction (not orthogonal and not in the same direction) then they make up a good swap. If they do not point in approximately the opposite direction, then the distance between $X_p$ and $X_o$ will be required to determine whether it is a good swap.

Another interpretation is to draw a line that goes through $l$ and $k$ ($lk$), and a line through $p$ which intersects $lk$ at right angles ($plk$). Any point $o$ which is in cluster $k$ and lies on the "right" side of $plk$ satisfies this inequality. The "right" side is the side of the line which is in the direction from $k$ towards $l$. This is illustrated in Figure 4.3 for three other clusters $i$, $j$ and $k$.

We are interested in point $p$ which is in cluster $l$. Cluster $k$ is the same distance from $p$ as $l$ is; hence $plk$ then goes through the midpoint between $l$ and $k$. Cluster $j$ is closer to $p$ than $l$ is; hence $plj$ goes through $lj$ closer to $j$ than $l$. As the cluster $j$ moves closer between $p$ and $l$ the boundary gets closer to $j$, until the boundary crosses to the other side of $j$: this is investigated below. Cluster $i$ is on the right side of $l$, and so the boundary goes below $l$; these clusters are much less likely to contain good swap points for $p$. 

Figure 4.3: Boundaries of the location of a point $o$ in clusters $i$, $j$ or $k$, such that the variance approximation for a swap for point $p$ (which is in cluster $l$) is negative. See text for details.
4.4. *k*-means with minimum populations

![Diagram of circle property](image)

Figure 4.4: A graphical illustration of the circle property applied to point $p$ in cluster $l$, with another cluster $k$. See text for details.

**Circle property**

Alternatively, we can rewrite (4.36) on page 65 like this:

$$\langle X_p - C_k, C_l - C_k \rangle + \langle C_k - X_o, C_l - C_k \rangle < 0. \quad (4.37)$$

The strict inequality $\langle X_p - C_k, C_l - C_k \rangle < 0$ describes the condition for $C_k$ to be strictly within the circle (or (hyper)sphere) defined by $X_p$ and $C_l$ as the end-points of the diameter of the circle. If this strict inequality holds, then $\langle C_k - X_o, C_l - C_k \rangle \leq 0$ is sufficient for (4.37) to hold. This describes a linear constraint on $X_o$, which includes $C_k$ on the boundary. Hence if we can find an $X_o$ in cluster $k$ such that $\langle C_k - X_o, C_l - C_k \rangle \leq 0$ then it is a good swap. $C_k$ is the average of all points in the cluster, so it is within the convex hull of the cluster; if every point in cluster $k$ did not satisfy this constraint, then $C_k$ would not be on the boundary (because $C_k$ is within the convex hull of the cluster). Hence there must exist a point $o$ within cluster $k$ which does satisfy the constraint, and hence is a good swap with $p$.

The circle property is graphically illustrated in Figure 4.4.

We are interested in point $p$ which is in cluster $l$. The centre of cluster $k$ is shown which is inside the inscribed circle. The line $kkl$ shows the boundary where a point $o$ would satisfy the equation $\langle C_k - X_o, C_l - C_k \rangle \leq 0$, where the arrows indicate the direction in which the inequality is satisfied. It is

---

\[ ^{\text{This derivation courtesy of Mike Powell [47].}} \]
not possible for all points in cluster \( k \) to be on the left side of line \( k_{kl} \): hence there must be at least one point in cluster \( k \) which is a good swap for \( p \). The line \( p_{kl} \) is the more accurate approximation from (4.36) on page 65.

This means that an alternative heuristic for searching for a good swap is to find a cluster \( k \), the centre of which is within the circle between \( X_p \) and \( C_l \). If we find such a cluster, then we know that there must exist a point in that cluster which is a good swap.

If we do not find such a cluster, we do not know if there is a good swap. Since the circle property is an approximation, it only ensures that there is a good swap in a cluster if the centre of that cluster is within the inscribed circle. As the cluster centre moves away from the circle, then portions of the inequality increase and the boundary of the approximation solution moves further away from and does not include the cluster centre. The true variance contains the term \(-\|X_p - X_o\|^2 \left(\frac{1}{Q_l} + \frac{1}{Q_k}\right)\) which can only reduce the variance. So clusters with centres near but not within the circle are still likely to contain good swap points.

Hence we could order all other clusters by their distance from the average of \( X_p \) and \( C_l \), and search the closer clusters first. Assuming \( K \ll N \) (a reasonable assumption) then this should not be too costly (in comparison to the swap search). We could stop the search at the first point which reduces the variance, or continue searching for a preset number of clusters or points. We may also want to stop searching once we reach clusters which are a preset number of diameters away from the circle, to avoid what is probably unnecessary work.

Reordering

The \( k \)-means-min-\( p \) algorithm relies on reordering of clusters based on size, with the intuition being that if a point \( o \) in a large cluster \( k \) is a good swap point for a point \( p \) in a minimal cluster \( l \), then \( o \) is likely to be assigned to \( l \) without removing \( p \). We investigate that intuition here.

The cluster \( l \) is minimal, i.e., \( Q_l = P \) and cluster \( k \) is large, i.e., \( Q_k > P \). We know that \( \delta E(p, l, o, k) < 0 \) where \( \delta E(p, l, o, k) \) is defined in (4.17) on page 61. If we were to consider reassigning \( o \) to \( l \) then from (4.4) on page 54

\[
\delta E(o, k, l) = \alpha \|X_o - C_l\|^2 - \beta \|X_o - C_k\|^2
\]  

(4.38)

where \( \alpha = \frac{Q_l}{Q_{l+1}} < 1 \) and \( \beta = \frac{Q_k}{Q_{k-1}} > 1 \). The effect of \( \alpha \) and \( \beta \) is to effectively move cluster \( l \) closer to and cluster \( k \) further away from \( o \). This means that \( \delta E(o, k, l) \) will be less than the corresponding terms in \( \delta E(p, l, o, k) \):
4.5. The boxing algorithm

\[ \|X_o - C_i\|^2 - \|X_o - C_k\|^2 \] and is more likely to be negative.

This is not very strong because it depends on the relative distances and sizes of the points and clusters.

If \(o\) is the best swap point for \(p\) then it is likely that \(\delta E(p, l, o, k) \ll 0\). This makes it even more likely that \(\delta E(o, k, l) < 0\). Whether \(l\) is the best cluster for reassigning \(o\) depends on the locations and sizes of the other clusters.

4.4.5 Implementing \(k\)-means-min-\(p\)

Due to time constraints, and the limitations of clustering in general (and in particular with the Wood feature-set: see §4.3.3 on page 56) the \(k\)-means-min-\(p\) algorithm has not been implemented. Some things to note when implementing it are listed below.

It would be interesting to compare the resulting clusterings with those resulting from plain \(k\)-means, especially if \(p\) is set so small (e.g., one) so that swaps are never needed. The reordering of clusters and points means the search path would be different to plain \(k\)-means so the local minimum reached may be different. The additional book-keeping may make the algorithm too slow in comparison to \(k\)-means.

The various approximations described in §4.4.4 on page 65 should be used to try to limit the search for swaps. Experiments should be performed to determine the usefulness of any of the heuristics.

4.5 The boxing algorithm

When looking for new classes of long period variable stars astronomers generally look through piles of light-curves until they find something that may be a new class of star and then they try to find other examples of that class. This is a very tedious process. We would like to use the power of the features and provide tools which will reduce this manual labour.

The \(k\)-means clustering of the Wood feature-set (§4.3.3 on page 56) showed that while the features of the Wood dataset were reasonable at grouping similar stars together, they were not totally successful (at least in the \(k\)-means sense) at separating quite different sorts of stars. We would like to take advantage of the similarity power within the feature-set to find interesting classes of variable stars.

Deciding whether a star is an example of a particular class of star is difficult (especially if this class is new and there are not many other examples to compare with) and is often best left to the astronomer (or user). We can
use the features to find examples for the user to decide whether or not they are of a particular class. If we do this repeatedly we can build up a set of labelled stars; this set can be used for other purposes, such as building a classifier model.

We call this overall process “boxing” since we iteratively pick out potential examples and then put them in a “box”: this labels each of those examples.

The intuitive idea behind the boxing algorithm is to repeatedly choose “interesting” examples (patterns) from the dataset and classify them according to previously selected (and labelled) examples. Clearly, there are three difficulties here: the ability to choose interesting examples, the classification, and deciding whether the interesting example is deserving of a new class.

Given a set of unboxed patterns \( P \), a set of boxed patterns \( B \) (set of pairs \( (p, l) \), where \( p \) is a pattern and \( l \) is a label), a method of choosing a pattern from the unboxed pattern set \( p \leftarrow \text{select}(P) \) and a method of comparing a pattern with the boxed patterns \( l \leftarrow \text{compare}(p, B) \), the boxing method is a repeated application of Algorithm 4.7. The unspecified parts of the algorithm are described in detail below.

\[ \text{Algorithm 4.7 } (P, B) \leftarrow \text{box}(P, B, \text{select}, \text{compare}) \]

1. \( p \leftarrow \text{select}(P) \)
2. \( l \leftarrow \text{compare}(p, B) \)
3. \( B \leftarrow B \cup \{(p, l)\} \)
4. \( P \leftarrow P \setminus \{p\} \)

Essentially, each iteration of the boxing process selects a pattern \( p \) from the set of unboxed patterns \( P \), compares that pattern with the previously boxed patterns \( B \) and decides what label \( p \) should be given.

The selection algorithm \textit{select} picks a pattern from the unboxed pattern set. Assuming the pattern set contains different classes of patterns, and the comparison algorithm is reasonable at comparing these classes, a good selection algorithm could choose patterns which are “typical” of those different classes (e.g., near the mean, in a \( k \)-means sense).

The comparison algorithm \textit{compare} is used to compare a particular pattern with the already boxed patterns. The result is a label identifying with which class that particular pattern belongs. The comparison algorithm can be thought of as a front-end to a similarity measure which compares two patterns. The comparison algorithm then chooses the class that contains the most patterns which are most similar. If none of the previously boxed patterns are similar enough to the test pattern, then the pattern is put in
4.6. Boxing: extreme case selection

The selection algorithm may take into account the boxed patterns $B$ in order to make its selection. For example, the star-searcher described in §4.7 on the following page uses a generalised additive model code to model the boxed patterns (based on their features), and then predicts the class probability of the unboxed patterns which are then ordered for comparison.

The comparison algorithm (or similarity measure) must clearly have some “expert” knowledge about the dataset before being able to make good judgements regarding the similarity of two patterns. Automating this is a difficult task, and in our experiments we only use an interactive approach from viewing light-curves. Potentially one could initiate the process using an expert human until the classified base is large enough, such that standard automated methods can be used. The standard automated methods could then use a larger set of explicit information (such as Fourier series or wavelet coefficient features) than the human.

Boxing is an application of both clustering and classification: clustering because the comparison process labels examples, and classification because the selection process generally builds a model in order to find other examples.

In the next two sections we describe two different experiments in finding interesting variables stars in the Wood dataset using the boxing algorithm. In §4.6 we select stars on the edge of feature space—extreme cases—and compare the results with selecting stars randomly, while with the star-searcher (§4.7 on the following page) we use a boosted generalised additive model classifier and predictor for selection.

4.6 Boxing: extreme case selection

For this first experiment, we wanted to see how the boxing algorithm worked and also how well the Wood features separated different and grouped similar types of stars. We used two selection algorithms: one based on a star’s location in feature-space, and one random. In both cases, the comparison was done by viewing light curves by a non-expert human who had extensively investigated the dataset.

We reasoned that stars located near the boundaries of feature-space (what we called “extreme cases”) would be more interesting, and may exhibit more extreme behaviour§. They would hence be better examples of their respective classes, particularly where there is a smooth variation be-

§This is due to a suggestion by John Rice [51].
tween two classes (as seems to be the case in the Wood dataset).

Astrophysicists believe that stars change in behaviour over their lifetime, and many of these features reflect this variation. Hence a particular class of star may have a large amplitude (say), and this amplitude may change through its lifetime. It is important to be able to distinguish between these sub-classes (i.e., differentiate young and old stars). Extreme-case selection may find old or young stars, making it easier to decide that they are different.

The extreme-case selection in this example was manual, based on each star's locations in various two-dimensional plots of particular features, notably red log relative amplitude (red lra), red average frequency (red freq), and red relative frequency spread (red rdf). In earlier investigations, these features seemed to separate the stars best.

During the extreme-case selection and boxing process 59 stars were extracted and classified, producing a total of 10 boxes (or classes). Some examples of these stars are shown in Figure 4.5. Box number 9 is especially interesting, since it corresponds to a new class of variable star previously discovered from the same dataset by astronomer Peter Wood [67].

As a justification of the extreme-case selection process, see Figure 4.6 which shows that these features are good at grouping similar sorts of stars.

During the random selection and boxing process 30 stars were extracted and classified, producing a total of 9 boxes. Some of the cleanest examples of these stars are shown in Figure 4.7 on page 74. During the process, it was noted that it became increasingly difficult to make a decision regarding which class the example pattern belonged to since there were usually two potential matches in different classes. It was presumably easier to decide when using the extreme-case selection because it chose stars which were more extreme in behaviour.

4.7 Boxing: the star-searcher

The star-searcher is an application of the boxing algorithm as described in §4.5 on page 69 with the selection being done by a boosting generalised additive model code (see §4.2 on page 52 and [17]). The comparison is still done interactively by the user, but instead of choosing multiple classes the user simply has to decide whether a star is in or not in the class of interest (i.e., $l$ is either one or zero).

An algorithmic description of the star-searcher is shown in Algorithm 4.8 on page 74 and is described in detail below.

---

*This is joint work with Fergus McGlynn, Markus Hegland and Margaret Kahn [34].
4.7. Boxing: the star-searcher

Figure 4.5: Six stars belonging to three boxes, where selection was by extreme cases.

Figure 4.6: Two depictions of the extreme-case boxing process with respect to two particular features: red frequency and red log relative amplitude. Each point in the left plot is a star in the Wood dataset; the +’s are those that were selected for boxing using the extreme-case method. The right plot shows just those stars that were boxed identified by their box label.
Figure 4.7: Six stars belonging to three boxes, where selection was random.

Algorithm 4.8 $\mathcal{B} \leftarrow \text{star-searcher}(\mathcal{P}, \mathcal{B})$

1: repeat
2: $M \leftarrow \text{build-model}(\mathcal{B}, \mathcal{P})$
3: $C \leftarrow \text{predict}(M, \mathcal{P})$
4: $S \leftarrow \text{sort-by}(\mathcal{P}, C)$
5: display($S$)
6: $\mathcal{B} \leftarrow \text{label}(S)$
7: until enough examples have been found

The star-searcher requires an initial list of labelled stars, or can display a random selection of stars (not shown in this algorithm).

The build-model algorithm builds our boosted additive model classifier which takes the list of boxed stars $\mathcal{B}$ and their feature-set and builds a model $M$.

The whole set of patterns (dataset) is then predicted based on the model $M$ (predict) with the label for a star—ranging from zero to one—representing the probability that it is in the class.

The list of stars is then sorted by the predicted probability (sort-by) and the light curves of the top 50 stars are displayed to the user in this
4.7. Boxing: the star-searcher

Figure 4.8: Screenshot of the first iteration of the star-searcher. The first two stars were part of the labelled list (another two were labelled as not in the class).

order for further comparison (display). Unlike the original boxing algorithm, this list also contains the previously labelled stars so the user can see the positively predicted boxed stars. The star-searcher then allows the user to choose whether any of the stars shown are in or not in the class of interest. Previously boxed stars can be unboxed or relabelled if the user chooses. After selecting the labels for some stars (label) the additive model is regenerated and the process continues.

The star-searcher often builds a good enough model in a few iterations such the top few (if not more) patterns are correctly classified and often finds new examples of the class from the unboxed patterns.

Some screen-shots of a prototype star-searcher at work are shown in Figures 4.8–4.10.

The star-searcher was initiated with four boxed stars, two labelled as being in the class (“positively labelled”) and two not in the class. The first selection seen by the user (Figure 4.8) shows that the two positively labelled stars are the highest in the order chosen by the model (this is not always the case).

The user then labels several stars from the top 50 list as being in the
Figure 4.9: Screenshot of the second iteration of the star-searcher.

Figure 4.10: Screenshot of the third iteration of the star-searcher.
class or not in the class—in this case we are after stars which are clear long-period variables with an additional low amplitude higher frequency superimposed. Examples that are high on the selection list and are not in the class of interest are important to label so the modeller can correct its mistakes. Examples that are lower on the list and are clear examples of the class are also important to select although they are harder to find (the population of stars not in a class is generally much larger than the population of stars in a class).

Following the labelling of stars, the additive model is regenerated. The next selection of stars is shown in Figure 4.9. This shows the model is getting better at selecting the right sort of star but is still getting some wrong.

Another iteration of positive and negative labelling (mostly negative) and we get the selection shown in Figure 4.10. The top six stars were already chosen by the user as being part of the class but the next three were chosen by the model.

The star-searcher is an effective way of searching for and finding examples of a particular class of stars, even with the very basic features described in Chapter 3 on page 23. We suspect it would do even better with more complex features such as the discrete Fourier transform, wavelets [56] or specialised "shape" features.

The star-searcher is an example of the effectiveness of relevance feedback in assisting the generation of good models. In related work Keogh and Pazzani use relevance feedback to refine queries in their time-series comparison tool [38, 39].
Chapter 4. Clustering and Classification
Chapter 5

Conclusions and Further Work

This thesis has presented an approach to the discovery and classification of interesting variable stars from a time-series astronomical database. We have designed an efficient file format for accessing the time-series data. We have developed a set of features which summarise the behaviour of each time series, and seem to separate the types of stars reasonably well. We also presented a tool which uses user input (or relevance feedback) to narrow the search for classes of stars.

One of the purposes of this work was to try to find interesting variable stars from the datasets without much astronomical expertise; i.e., whether it was possible to do without domain knowledge in the data mining process. While we did find interesting classes of stars in the Wood dataset (e.g., the extreme-case selection boxing method in §4.6 on page 71) we could not interpret our results without domain knowledge. It also became much more difficult to decide whether something would be interesting, since “interesting” depends on the beholder. However, we believe our star-searcher tool will be very useful for astronomers in searching for interesting variable stars.

The MACHO project has made the data available on the web [40]. It would be worthwhile to integrate this into our tools, along with some caching, so that more data could be used. The data available is purely the observational data; ideally we would like an adaptive collaborative database of features so that they do not need to be developed and calculated independently.

It would be interesting to compare the features we have used with their alternatives, such as weighted and/or trapezoidal approximations and medians. More robust versions of the derivative-based frequency estimates would be useful, as they are currently very sensitive to noise. In addition we should make the derivative smoothing parameter selection process
adaptive and automated.

More complex features such as Fourier series, wavelets (see Scargle [56]) and other "shape" features ought to assist in differentiating stars. However we may lose interpretability with these more complex features.

The $k$-means-min-$p$ algorithm looks promising and needs to be implemented and compared with $k$-means. Other clustering techniques should be used on the Wood feature-set to see if they work better than our previous experience.

Classifying constant stars is quite different to classifying variable stars: constant stars are generally classified based on their locations in colour-magnitude space, whereas variable stars are further classified based on their light curves and periods. Also, there are good techniques for estimating the periods of faster varying stars which do not work so well for LPVs and vice versa (period folding versus derivative-based "average frequency"). Using a tree-based approach to classifying stars would be likely to help here.

As the Wood dataset contains mostly LPVs it is probably suited to direct time-series comparison such as those described in Keogh and Pazzani [38, 39]. Faster-varying stars are not likely to be suited to this unless we use pre-folded data. Keogh and Pazzani already use weights as part of the piece-wise linear representation of a time-series; the conversion process would need to be extended to using the estimates of error given for each observation (both as part of finding a piece-wise linear fit and for the weight of that piece).
Appendix A

bmdf implementations

This appendix shows two implementations to read bmdf (binary MACHO data format) files, as described in §2.2.1 on page 18.

A.1 MATLAB implementation

Firstly, we have the MATLAB [61] code. The constructor bmdf (Listing A.1 on the following page) creates a bmdf object (in MATLAB version 5) from the given filename which contains three members: the filename, the file id, and the number of stars.

The constructor makes use of a helper function init_bmdf (Listing A.2 on the next page) which initialises some global variables which specify the size of various blocks within the bmdf file.

Once we have finished with the bmdf object, we must close the file associated with it; this is shown in Listing A.3 on page 83.

A small function max (Listing A.4 on page 83) returns the number of stars in the bmdf object. It is poorly named since star numbers range from 0 to max(mf) - 1.

The function data (Listing A.5 on page 83) reads the data for a particular star from a bmdf object. Optional parameters specify if we want only the joint data, or we want the data sorted.

The data function makes use of the index function (Listing A.6 on page 84) which reads the index record for a given star and converts the values into something easier to use.
Listing A.1: The constructor of a `bmdf` object: opens the specified file.

```matlab
function mf = bmdf(a)
% mf = open_bmdf(a)
% opens "Binary Macho Data Format" file filename
% reads and checks the magic number
% and reads the number of stars
% the result is an object mf
if isa(a, 'bmdf')
    mf = a;
    return
end
% set up some globals (if not already):
init_bmdf
global BMDF_MAGIC_LEN BMDF_MAGIC BMDF_VERSION
mf.filename = a;
mf.fid = fopen(a, 'r');
[magic_number, count] = fread(mf.fid, BMDF_MAGIC_LEN, 'char');
[vers, count] = fread(mf.fid, 1, 'int32');
if ((char(magic_number) == BMDF_MAGIC) | (vers == BMDF_VERSION))
    error('bad.magic_number');
end
[mf.number.stars, count] = fread(mf.fid, 1, 'int32');
mf = class(mf, 'bmdf');
```

Listing A.2: Global initialiser of `bmdf` definitions.

```matlab
function init_bmdf
% sets up some initial variables as globals, for use in
% manipulating bmdf files
global BMDF_INIT
if BMDF_INIT
    return
end
BMDF_INIT = 1;
global BMDF_MAGIC_LEN BMDF_MAGIC BMDF_VERSION BMDF_SID_LEN
BMDF_MAGIC_LEN = 4;
BMDF_MAGIC = 'bmdf';
BMDF_VERSION = 2;
BMDF_SID_LEN = 12;
global BMDF_INDEX_OFFSET BMDF_SIZEOF_INDEX_REC
BMDF_INDEX_OFFSET = 12;
BMDF_SIZEOF_INDEX_REC = 24;
global BMDF_SIZEOF_JOINT BMDF_SIZEOF_SINGLE
BMDF_SIZEOF_JOINT = 20;
BMDF_SIZEOF_SINGLE = 12;
```
Listing A.3: Close a bmdf object.

```matlab
function close(mf)
    % close(mf)
    % closes bmdf object mf. It is no longer usable.
    fclose(mf.fid);
end
```

Listing A.4: A helper function to get the number of stars in a bmdf object.

```matlab
function m = max(mf)
    % max(mf)
    % returns the number of stars in mf
    m = mf.number.stars;
end
```

Listing A.5: Extracts data from a bmdf object.

```matlab
function [red, blue] = data(mf, star_number, joint, dosort)
    % [red, blue] = data(mf, star_number, joint, dosort)
    % reads data of star_number from bmdf file mf. If joint
    % is given and is non-zero, only elements which have
    % joint results (both red and blue) are extracted. If
    % dosort is given and is non-zero, the result is sorted
    % by day (otherwise the result may not be sorted).
    [star_id, star_data_offset, num_joint, num_red, num_blue] = index(mf, star_number);
    fseek(mf.fid, star_data_offset, 'bof');
    if (nargin < 4)
        dosort = 0;
    end
    if (nargin < 3)
        joint = 0;
    end
    [joint_vals, count] = fread(mf.fid, [5, num_joint], 'float');
    joint_vals = joint_vals';
    red = joint_vals (:,[1 2 3]);
    blue = joint_vals (:,[1 4 5]);
    if (joint)
        return; % no need to sort
    end
    [red_vals, count] = fread(mf.fid, [3, num_red], 'float');
    red = [red; red_vals '];
    [blue_vals, count] = fread(mf.fid, [3, num_blue], 'float');
    blue = [blue; blue_vals '];
    if (dosort)
        red = sortrows(red, 1);
    end
    blue = sortrows(blue, 1);
end
```
Listing A.6: Extracts index from a bmdf object.

```matlab
function [star_id, star_data_offset, num_joint, num_red, num_blue] = index(mf, star_number)
    if star_number >= mf.number_stars
        error('bad star_number')
    end
    offset = BMDF_INDEX_OFFSET + star_number * BMDF_SIZEOF_INDEX_REC;
    data_offset = BMDF_INDEX_OFFSET + mf.number_stars * BMDF_SIZEOF_INDEX_REC;
    fseek(mf.fid, offset, 'bof');
    [star_id, count] = fread(mf.fid, BMDF_SID_LEN, 'char');
    star_id = char(star_id);
    [star_data_offsets, count] = fread(mf.fid, 3, 'int32');
    star_data_offset = star_data_offsets(1) + data_offset;
    if star_number == (mf.number_stars - 1)
        fseek(mf.fid, 0, 'eof');
        end_data_offset = ftell(mf.fid);
    else
        fseek(mf.fid, BMDF_SID_LEN, 'cof');
        [end_data_offset, count] = fread(mf.fid, 1, 'int32');
        end_data_offset = end_data_offset + data_offset;
    end
    num_joint = star_data_offsets(2) / BMDF_SIZEOF_JOINT;
    num_red = (star_data_offsets(3) - star_data_offsets(2)) / BMDF_SIZEOF_SINGLE;
    num_blue = (end_data_offset - star_data_offset - star_data_offsets(3)) / BMDF_SIZEOF_SINGLE;
```
A.2 C/C++ implementation

The plain C header file `bmdf.h` in Listing A.7 describes the structures within a `bmdf` file.

The C++ header file `bmdf.hh` in Listing A.8 on the next page has the definition of a C++ class `bmdf` which provides methods for extracting the data from a `bmdf` file.

The C++ program `bmdfcat` in Listing A.9 on page 87 uses the `bmdf` class to print out the data for particular star.

Listing A.7: Plain C header file `bmdf.h` of structures in `bmdf` file.

```c
/*
 /lib 31mar98
 macho bin file format def
 */
#ifndef bmdf
#define bmdf

#define SID_LEN 12
#define MAGIC_LEN 4
#define MAGIC_NUMBER "bmdf"
#define MAGIC_VERSION 2

typedef struct header_s {
    char magic_number[MAGIC_LEN];
    int version;
    int number_stars;
} header_t;

typedef struct indexrec_s {
    char star_id[SID_LEN];
    int data_offset, red_offset, blue_offset;
} indexrec_t;

typedef struct data_value_s {
    float value, error;
} data_value_t;

typedef struct jointobs_s {
    float julian_day;
    data_value_t red, blue;
} jointobs_t;

typedef struct singleobs_s {
    float julian_day;
    data_value_t value;
} singleobs_t;
#endif
```
Listing A.8: C++ header file bmdf.hh of bmdf class.

```cpp
/*
*/
#ifndef bmdf.hh
#define bmdf.hh
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include "bmdf.h"

class bmdf
{
 public:
  int fid, nstars, d_off;
  static const int i_off = sizeof(header_t);
  bmdf(const char *const fname) {
    fid = open(fname, O_RDONLY); if (fid == -1) return;
    header_t h; read(fid, &h, sizeof(h));
    if ((strncmp(h.magic_number, MAGICIVUMBER, MAGIC_LEN) != 0) || (h.
      version != MAGIC_VERSION)) { fid = -1; return; }
    nstars = h.number_stars;
    d_off = i_off + sizeof(index_rec_t) * nstars;
  }
  ~bmdf() { if (fid != -1) close(fid); }
  void index(int star, index_rec_t &idx, int &end_offset) {
    lseek(fid, i_off + sizeof(index_rec_t) * star, SEEK_SET);
    read(fid, &idx, sizeof(idx));
    if (star == nstars - 1) { end_offset = lseek(fid, 0, SEEK_END); } else {
      index_rec_t eidx; read(fid, &eidx, sizeof(eidx));
      end_offset = eidx.data_offset + d_off;
    }
  }
  void index2nums(index_rec_t idx, int end_offset, int &num_joint, int &num_red, int
    &num_blue) {
    num_joint = idx.red_offset / sizeof(joint_obs_t);
    num_red = (idx.blue_offset - idx.red_offset) / sizeof(single_obs_t);
    num_blue = (end_offset - idx.data_offset - d_off - idx.blue_offset) / sizeof(
      single_obs_t);
  }
  void data(index_rec_t idx, int end_offset, joint_obs_t *joint_r, single_obs_t *red,
    single_obs_t *blue) {
    lseek(fid, idx.data_offset + d_off, SEEK_SET);
    read(fid, joint_r, idx.red_offset);
    read(fid, red, idx.blue_offset - idx.red_offset);
    read(fid, blue, end_offset - idx.data_offset - d_off - idx.blue_offset);
  }
};
#endif
```
A.2. C/C++ implementation

Listing A.9: C++ program bmdfcat.cc using bmdf class.

```c++
/*
//lib 20001128
prints out a single star’s data.
shows how to use bmdf class.
*/

#include "bmdf.hh"
#include <stdio.h>

int main(int argc, char *argv[]) {
    if (argc != 3) return(-1);
    int star_number;
    sscanf(argv[1], "%d", &star_number);
    bmdf mf(argv[2]);
    if ((mf.fid == -1) || (star_number >= mf.nstars)) return(-1);
    index_rec_t idx;
    int end_offset;
    mf.index(star_number, idx, end_offset);
    int num_joint, num_red, num_blue;
    mf.index2nums(idx, end_offset, num_joint, num_red, num_blue);
    joint_obs_t * joint = new joint_obs_t[num_joint];
    single_obs_t * red = new single_obs_t[num_red];
    single_obs_t * blue = new single_obs_t[num_blue];
    mf.data(idx, end_offset , joint , red , blue);
    printf("joint: \\
") ;
    for(int i=0;i<num_joint;++i) {
        joint_obs_t & co = joint[i];
        printf("%f\t%f\t%f\t%f\n", co.julian_day, co.red.value, co.red.error, co.
        blue.value, co.blue.error);
    }
    printf("red-only: \\
") ;
    for(int i=0;i<num_red;++i) {
        single_obs_t & so = red[i];
        printf("%f\t%f\n", so.julian_day, so.value.value, so.value.error);
    }
    printf("blue-only: \\
") ;
    for(int i=0;i<num_blue;++i) {
        single_obs_t & so = blue[i];
        printf("%f\t%f\n", so.julian_day, so.value.value, so.value.error);
    }
    delete [] joint; delete [] red; delete [] blue;
    return(0);
}
```
Bibliography


[56] J. D. Scargle. Wavelet methods in astronomical time series analysis. In Subba Rao et al. [60], chapter 14, pages 226–248. (cited on pages 77 and 80)


