Classification of Multiwavelength Transients With Machine Learning

under the supervision of Dr Michelle Lochner and Prof Bruce Bassett

Kimeel Sooknunan

Classification of Multiwavelength Transients with Machine Learning

Kimeel Sooknunan

May 2019
Version: Final

Abstract

With the advent of powerful telescopes such as the Square Kilometre Array (SKA), its precursor MeerKAT and the Large Synoptic Survey Telescope (LSST), we are entering a golden era of multiwavelength transient astronomy. The large MeerKAT science project ThunderKAT may dramatically increase the detected number of radio transients. Currently radio transient datasets are still very small, allowing spectroscopic classification of all objects of interest. As the event rate increases, follow-up resources must be prioritised by making use of early classification of the radio data. Machine learning algorithms have proven themselves invaluable in the context of optical astronomy, however it has yet to be applied to radio transients. In the burgeoning era of multi-messenger astronomy, incorporating data from different telescopes such as MeerLICHT, Fermi, LSST and the gravitational wave observatory LIGO could significantly improve classification of events.

Here we present MALT (Machine Learning for Transients): a general machine learning pipeline for multiwavelength transient classification. In order to make use of most machine learning algorithms, "features" must be extracted from complex and often high dimensional datasets. In our approach, we first interpolate the data onto a uniform grid using Gaussian processes, we then perform a wavelet decomposition and finally reduce the dimensionality using principal component analysis. We then classify the light curves with the popular machine learning algorithm random forests.

For the first time, we apply machine learning to the classification of radio transients. Unfortunately publicly available radio transient data is scarce and our dataset consists of just 87 light curves, with several classes only consisting of a single example. However machine learning is often applied to such small datasets by making use of data augmentation. We develop a novel data augmentation technique based on Gaussian processes, able to generate new data statistically consistent with the original.

As the dataset is currently small, three studies were done on the effect of the training set. The classifier was trained on a non-representative training set, achieving an overall accuracy of 77.8% over all 11 classes with the known 87 lightcurves with just eight hours of observations. The expected increase in performance, as more training data are acquired, is shown by training the classifier on a simulated representative training set,

achieving an average accuracy of 95.8% across all 11 classes. Finally, the effectiveness of including multiwavelength data for general transient classification is demonstrated. First the classifier is trained on wavelet features and a contextual feature, achieving an average accuracy of 72.9%. The classifier was then trained on wavelet features and a contextual feature, together with a single optical flux feature. This addition improves the overall accuracy to 94.7%. This work provides a general approach for multiwavelength transient classification and shows that machine learning can be highly effective at classifying the influx of radio transients anticipated with MeerKAT and other radio telescopes.

Contents

List of Figures

List of Tables

Abbreviations

AGN	Active Galactic Nuclei
ANN	Artificial Neural Network
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
FS	Flare Star
GP	Gaussian Process
GRB	Gamma Ray Burst
KN	Kilonova
NB	Naive Bayes
PCA	Principal Component Analysis
RSCVn	RS Canum Venaticorum
SN	Supernova
SVM	Support Vector Machine
SWT	Stationary Wavelet Transform
***t*-SNE**	t-distributed Stochastic Neighbour Embedding
XRB	X-Ray Binary

Acknowledgements

To my parents –

Thank you for always supporting my decisions even when it took me to Cape Town to study a degree you knew nothing about. Thank you for allowing me to pursue my dreams! I could not be where I am now if it weren't for the values of hard work and persistence that you have instilled in me.

To Kim, Nash and Prani –

Thank you for always being the shoulders I can rest on. No matter the problem I was having or what I was going through I knew that after talking it through with you it would be better. I know I can always count on you if I need help with anything! You guys are the only ones that actually know and understand both sides of my life (Cape Town and Joburg). A special shout out to Nash for being the one person I could look up to as an academic role model.

To Jake –

I think our friendship can be summed up in a few key moments. The first was out first trip to Sutherland when I realised that you weren't so bad. Then there was the time in 3^{nd} year when you used my prac as inspiration for one of your art projects. The time in honours when we were both looking at a problem on the board and came up with the same solution at the same time. The time we rocked up to campus with the same clothes completely unintentionally and now when we go shopping together to get the same clothes completely intentionally. I am not sure when we become the best of friends but what I am sure of is that you not persuing your true calling and doing astros is one of the best things that has happened to me.

To Brandon –

I always forget that we only met in NASSP, which I think says everything about the kind of friends we've become. Living in Obz would have been the worst if you weren't my neighbour (which is why I moved out when you left). I am very glad you decided to come to UCT even if it was just for a bit.

To Kevin –

We've been friends for a very long time. I don't think I have met a more positive

person in my life. Thank you for the constant support you have given me over the years! We had been talking about gyming since 2nd year. Last year we finally got the chance to! Without you, I don't think I could've made this drastic lifestyle change, for that I will always be thankful.

To Nicole –

Even though you're constantly not my friend, you still are my first friend I made at UCT. Thank you for reading this thesis a billion times and fixing all the errors!

To Curtly and Ethan –

What would group meetings be without the lunchtime beach trips! Curtly, your outlook on life has always been one I admired. Achtung wouldn't be complete without you. Ethan, thanks for the very needed help!

To Bruce –

Thank you for all the knowledge, advise and wisdom you have shared with me over the last year. You have shown me what it truly means to be an academic.

Finally to Michelle –

Thank you for being the best supervisor I could have asked for. You have consistently gone out of your way to help me in any possible way. This thesis is what it is in no small part due to you. Thank you for always encouraging me when the times got tough. Through your mentorship I believe I have grown to be a much better researcher.

Formal Acknowledgements

This work was partly funded by the National Astrophysics and Space Science Program (NASSP). We also acknowledge support from SKA Africa and the National Research Foundation (NRF) towards this research. Opinions expressed and conclusions arrived at, are those of the authors and are not necessarily to be attributed to the NRF.

Declaration

I, Kimeel Sooknunan, declare that this thesis the work presented in it are my own.

I confirm that –

- This work was done wholly while in candidature for a research degree at the University of Cape Town.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

Cape Town, May 2019

Kimeel Sooknunan

Preface

In the coming years, radio astronomy will enter a new era of deep field surveys with the advent of the Square Kilometre Array[1] (SKA) and its precursors, such as MeerKAT[2] and the Australian Square Kilometre Array Pathfinder[3] (ASKAP). This together with optical telescopes such as MeerLICHT[4] [1], the Large Synoptic Survey Telescope[5] (LSST; [2]) and BlackGEM[6] is propelling astronomy into a new age of multiwavelength observations. Alert streams from telescopes such as Fermi[7] and the gravitational wave observatory LIGO[8] will also enable rapid coordination for multimessenger observations. A prime example of modern multiwavelength astronomy is the MeerLICHT telescope, an optical telescope whose observing schedule is synchronised with that of the (night time) observations of the radio telescope MeerKAT. This creates the unique opportunity of obtaining simultaneous optical and radio observations of transients. Combining these data sources to create an accurate, fast transient classifier necessitates a new general framework for multimessenger machine learning. Such large quantities of simultaneous multimessenger data will require automated methods to filter and classify transients to better select objects for follow-up. Machine learning techniques have been extensively used in optical astronomy but have yet to be employed for radio astronomy.

The main aim of the thesis is to present a general approach for the classification of transients with machine learning using multiwavelength data that could be extended to multimessenger data. The thesis is split into two parts, part I reviews some of the literature on transients and machine learning. Chapter 1 reviews the eleven different variables and transients used in this study. Chapter 2 gives a general overview of machine learning. The start of the chapter outlines the general approach of applying a machine learning algorithm to a problem. We then go on to review some supervised and unsupervised machine learning algorithms. Finally a literature review on machine learning in astronomy is presented.

[1] www.skatelescope.org
[2] www.ska.ac.za/science-engineering/meerkat
[3] www.atnf.csiro.au/projects/askap/index.html
[4] www.meerlicht.uct.ac.za
[5] www.lsst.org
[6] https://astro.ru.nl/blackgem
[7] https://fermi.gsfc.nasa.gov
[8] www.ligo.org

Part II of the thesis presents a general approach for transient classification and an application to radio transient data. Chapter 3 shows how we use some of the algorithms presented in Chapter 2 to create the classification pipeline. We then go on to discuss the process of applying it to radio transient data. First the data used in this study is presented. Finally, this chapter describes how MALT was applied to the data. Chapter 4 presents the results found. The chapter first presents the results of using just radio data and then then moves onto showing the effect of the training set on the results obtained. Finally the results of using both radio and optical data are presented.

Part I

Review

1 Astrophysical Variables and Transients

> *We live in a changing universe and few things are changing faster than our conception of it.*
>
> — **Timothy Ferris**
> (Science Writer)

Variables and Transients are astronomical objects that are observed to have varying brightness (flux). A transient is an object which has a single appreciable increase in flux, after which it fades away. The "cool down" timescale varies for different transients. Variables refer to objects with fluxes that repeatedly vary over different timescales. These timescales range from a few seconds to a few years. In this work we focus on variables and transients that are observed in the radio frequency spectrum.

1.1 MeerKAT and MeerLICHT

The MeerKAT[1] telescope is a precursor to the Square Kilometre Array (SKA). It consists of 64 Offset Gregorian antennas with a diameter of 13.5 meters. One of the MeerKAT key science projects is ThunderKAT[2] (The HUNt for Dynamic and Explosive Radio transients with meerKAT). ThunderKAT's main goal is to find, identify and understand radio transients [3]. They plan on using data from both direct surveys as well as commensal observations, this coupled with access to the data from MeerKATs other larger survey projects will make ThunderKAT the largest GHz-frequency radio transient project carried out to date [3].

While it is difficult to predict the expected number of transients that will be detected with ThunderKAT (as the radio transient sky is relatively unexplored to date), there is evidence to suggest there may be a large population of faint radio transients, perhaps numbering in the hundreds each year, that MeerKAT will be able to observe with its high sensitivity [3].

MeerLICHT[3] [1] is a 0.65 meter optical telescope that has been built to aid the ThunderKAT project. It has the same field of view as MeerKAT and is fully robotic. MeerLICHT will be tethered to MeerKAT i.e. it will always be observing the same patch of sky as

[1] www.ska.ac.za/science-engineering/meerkat
[2] www.thunderkat.uct.ac.za
[3] www.meerlicht.uct.ac.za

MeerKAT. This enables, for the first time ever, simultaneous radio and optical observations of the same patch of sky.

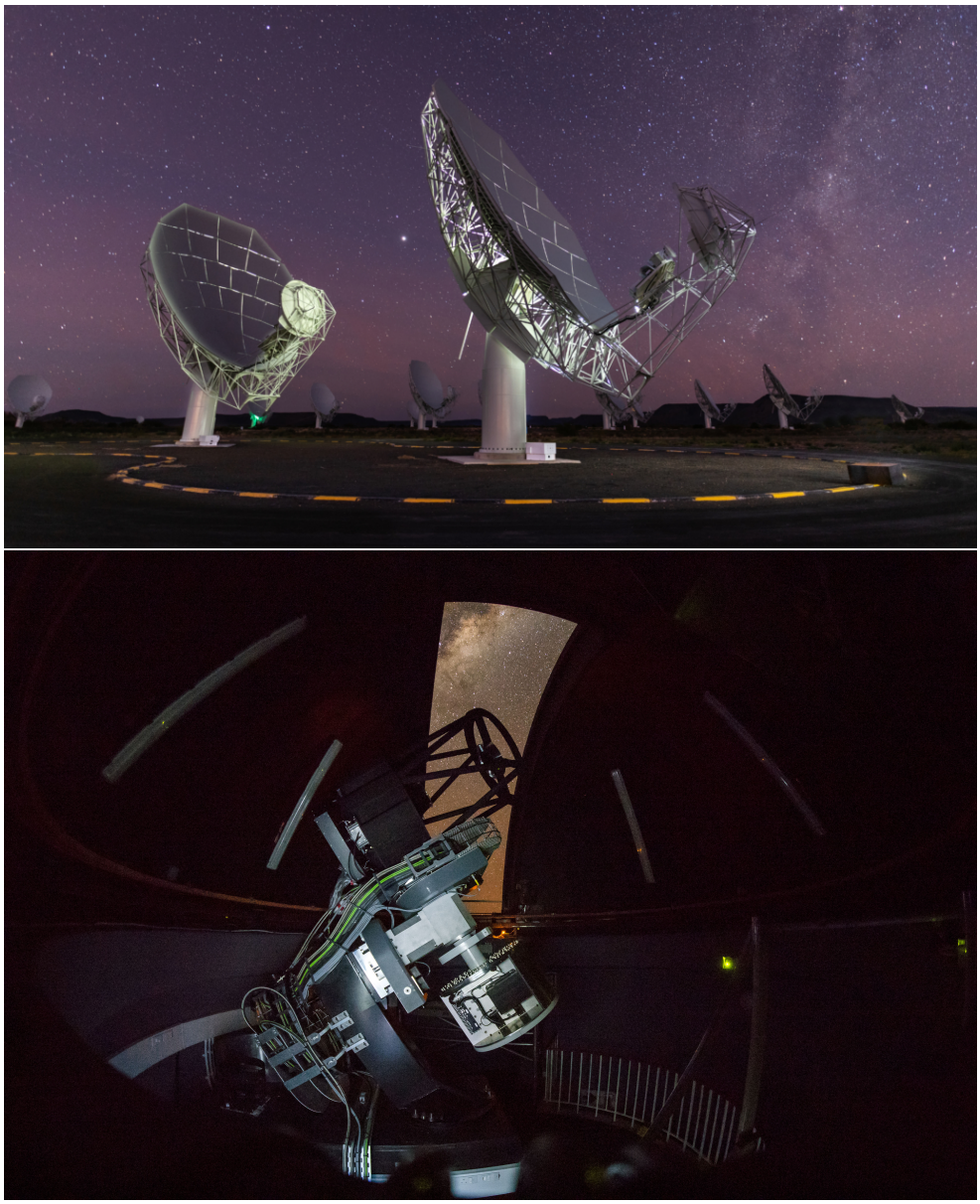


Fig. 1.1. The top and bottom panels show the MeerKAT and MeerLICHT telescopes respectively.

1.2 Radio Emission Mechanisms

Radio emission is measured in Janskys [Jy], a unit of spectral flux density named after Karl Jansky who first observed the radio emission originating from the Milky way. A Jansky is defined as

$$1\ \mathrm{Jy} = 10^{-26}\ \mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{Hz}^{-1}\ ,$$

where `W`, `m` and `Hz` are the SI units Watts, meters and Hertz respectively. There are four main types of radio emission [4], [5], these are:

- **Blackbody radiation:** This is a type of thermal radiation emitted by all objects. It is the result of the object converting internal energy into electromagnetic energy, the hotter the object the more blackbody radiation it releases. Blackbody radiation is emitted across the electromagnetic spectrum with most of it being radiated at shorter wavelengths.

- **Atomic and Molecular transitions:** Atomic transitions between high energy states result in radio emission called the radio recombination lines. Atomic hydrogen also emits in the radio frequencies when it undergoes a spin flip transition giving off the 21cm emission. When molecules such as CO_2, O_2 and H_2 transition between spin states they release photons in the radio regime.

- **Free particle emission:** In very high energy environments such as astrophysical plasmas, free particles are responsible for most of the radio emission. This is through processes such as bremsstrahlung radiation. When particles in plasma collide they release energy in the form of photons, some of these collisions lead to photons being released in the radio wavelengths. Particles can also be accelerated by a magnetic field releasing radiation. If the the particle were travelling at relativistic speeds it would emit synchrotron radiation, if it were travelling at non relativistic speeds it would emit cyclotron radiation.

- **Coherent free particle emission:** All emission mechanisms thus far have been incoherent emission where electrons produce the emission independently. Coherent emission on the other hand occurs when many electrons emit radiation together. An example of this is called plasma emission, plasmas oscillate in a particular mode which releases electromagnetic radiation.

1.3 Types of Variables and Transients

Radio variables and transients are typically divided into two kinds: incoherent synchrotron events and coherent burst events. Incoherent synchrotron events are thought to be caused by a high energy phenomenon, from which a large amount of energy is released over a longer time scale (on the order of minutes or longer). When in a steady state, this energy release is limited to a brightness temperature of $T \leq 10^{12}$K [6]. Coherent bursts occur on much shorter time scales (on the order of seconds) and can have brightness temperatures up to 10^{30}K. This type of emission can only be observed using a special mode on radio telescopes [6]. Some examples of coherent events are pulsars[7], [8] and fast radio bursts (FRBs; [9], [10]). For this study, we will only consider incoherent synchrotron events.

In this section we present an overview of the 11 types of variables and transients that form the base classes for the dataset used in this thesis. First the galactic objects are presented, in order these are algol, x-ray binaries, flare stars, RS canum venaticorum, nova and magnetars. Extra-galactic objects are then presented, in order these are supernova, gamma ray burst, kilonova, tidal disruption events and active galactic nuclei. This introductory material is derived from Percy et al [11] unless stated otherwise.

1.3.1 Algol

Algol binaries are star systems whose brightness varies periodically or semi-periodically due to one star eclipsing the other. This class of eclipsing binaries gets its name from the β Persei system which was given the name Algol [12]. Algols form part of a larger group of semi-detached binary systems where one star is a main sequence star[4][5] and its companion is a massive cool, faint star that lies above the main sequence. One star is distorted with a filled Roche lobe[6] while the other is spherical and well within its Roche lobe. Semi-detached binaries are known to have an accretion disc around one of the stars. This can be seen in Figure 1.2.

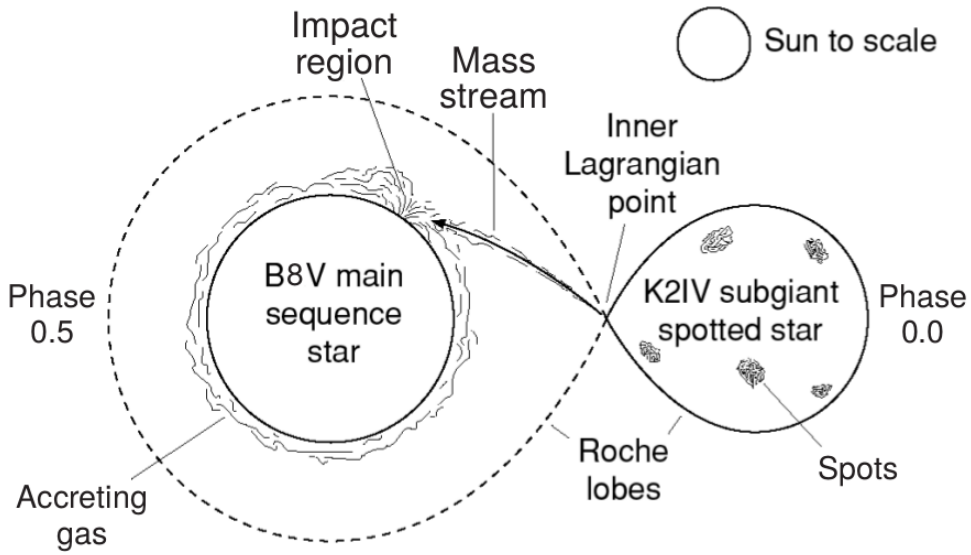


Fig. 1.2. Model of the original Algol system (β Persei). The Roche lobes of each of the stars are shown. The main sequence star can also be seen accreting matter from the smaller distorted star. The sun is shown for scale. Diagram from [11].

The accretion disk gives off both radio and X-ray emission. The accretion disk may also be variable if the mass transfer between the stars is periodic. The size of the accretion

[4] A main sequence star is one that lies on a specific band on the Hertzsprung-Russell (HR) diagram. These stars are all still burning hydrogen in their cores.

[5] HR diagrams are sometimes called colour-magnitude diagrams. It is a scatter plot of a stars temperature/colour against its luminosity/absolute magnitude.

[6] The Roche Lobe is the unique tear drop shaped area around a star in binary system in which the matter orbiting the star is gravitationally bound.

disk varies depending on the system, some known to block out one of the stars in the system. The variation seen in Algol systems are mainly due to one star eclipsing the other, hence the time scale of the variation will depend on the orbital periods of the stars. These stars are in very close orbits hence Algols vary on short timescales. An example Algol radio light curve is shown in Figure 1.3.

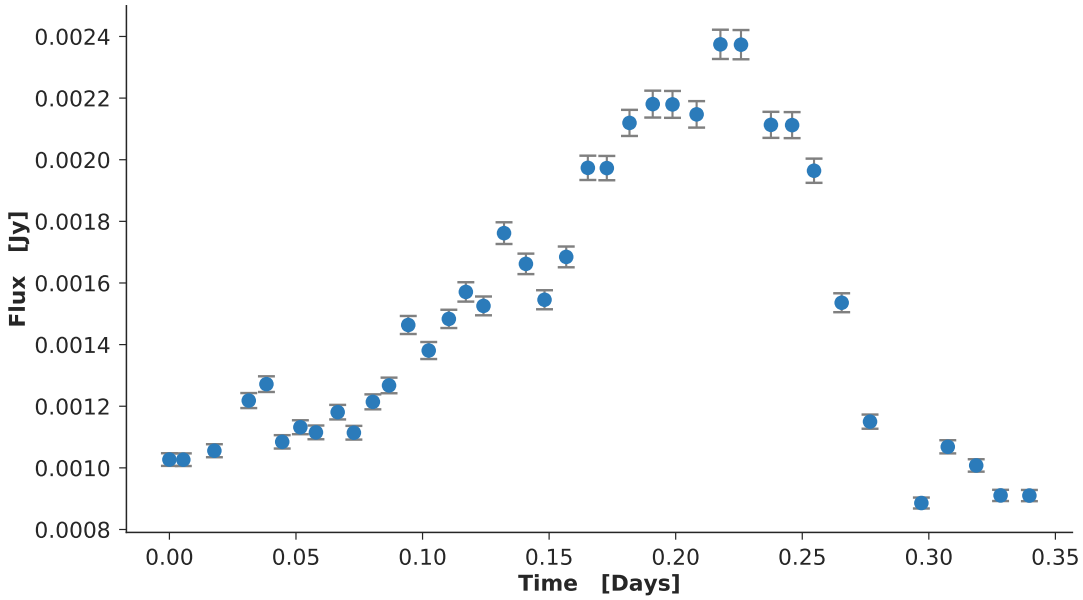


Fig. 1.3. An example Algol radio light curve, plotted as flux (in Jy) as a function of time (in days). It can be seen that the timescale of the variation seen is of the order of hours, with a total observation time of ≈ 8 hours. Data plotted is courtesy of [13].

1.3.2 X-Ray Binary

X-Ray binaries (XRBs; [14], [15]) are binary systems that have high X-ray emissions where one object is a massive star and its companion is a compact object, i.e a black hole or neutron star. These compact objects have large gravitational potential wells. As the name suggests, these binaries release X-rays, which are high energy photons and hence are associated with high temperatures. Mass from the companion is accreted onto the compact object, as it falls into the potential well it is heated to very high temperatures. This accretion can also give rise to relativistic jets similar to that of AGNs (Section 1.3.11) except on stellar scales. We see the reaction between these jets and the surrounding medium in the radio frequencies. X-ray binaries can be split into two main groups, namely, High-mass X-ray binaries (HMXBs) and low-mass X-ray binaries (LMXBs).

HMXBs have O or B stars[7] as the companion to the compact object which accretes matter onto it[17]. Many HMXBs are X-ray pulsars. Pulsars form when matter accretes onto a rotating neutron star. The matter flows down the magnetic field lines of the neutron star until it gets to the poles. The matter is then ejected at the poles creating relativistic jets.

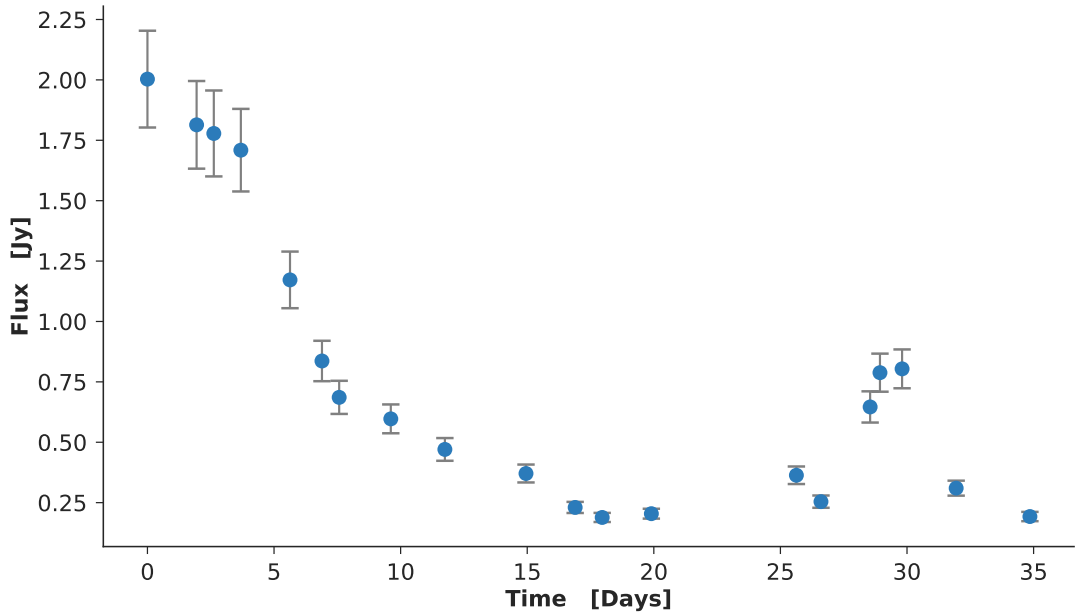


Fig. 1.4. An example XRB radio light curve, plotted as flux (in Jy) as a function of time (in days). It can be seen the timescale of the variations are order of a few days, with a total observation time of ≈ 35 days. Data plotted is courtesy of [13].

LMXBs have G, K or M stars as the companion to the compact object which loses mass through Roche lobe overflows. Mass moves from the Roche lobe to the accretion disk of the compact object. Many of these binaries are found in globular clusters and have very short orbital periods. LMXBs are much fainter than HMXBs. An example XRB radio light curve is shown in Figure 1.4.

1.3.3 Flare Star

Flare stars (FS; [18]), also known as UV Ceti stars, are stars that have large variations in brightness that isn't due to the star being eclipsed. Their variation is due to a similar process that causes solar flares (hence the name). They are dwarf K and M stars. Dwarf stars are the most common stars in the galaxy, hence flare stars are the most common variable stars in our galaxy.

[7] Stars can be classified based on their average temperatures. This classification scheme has seven main types, which in descending temperature order are: O,B,A,F,G,K and M. The temperature ranges for each class are:– O: $> 30\ 000K$, B: $10\ 000 - 30\ 000K$, A: $7\ 500 - 10\ 000K$, F: $6\ 000 - 7\ 500K$, G: $5\ 200 - 6\ 000K$, K: $3\ 700 - 5\ 200K$ and M: $2\ 400 - 3\ 700K$ [16].

These stars have a high rotation rate and have a large magnetic field. This rotation causes the magnetic field lines to get tangled, when these field lines snap and rearrange in to simpler configurations they release large amounts of magnetic energy. The flares are thought to be caused by this energy being released on the surface of the star. When the energy is released it causes a shock wave to move out - this heats and ionises the gas. The flare's emissions are most prominent at short wavelengths. Optical flares are often accompanied by bursts of radio energy, thought to be due to synchrotron radiation. An example flare star radio light curve is shown in Figure 1.5.

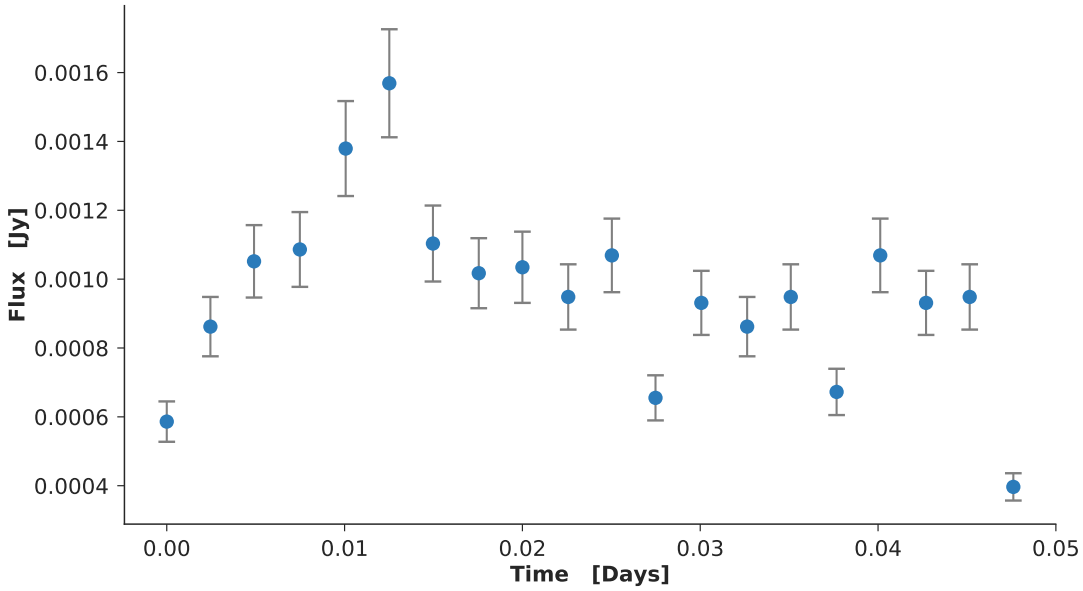


Fig. 1.5. An example FS radio light curve, plotted as flux (in Jy) as a function of time (in days). It can be seen that the timescale of the variation seen is extremely short when compared to most other types of variables and transients, with a timescale on the order of minutes, with a total observation time of ≈ 1 hour. Data plotted is courtesy of [13].

1.3.4 RS Canum Venaticorum

RS Canum Venaticorum (RSCVn; [19]) stars are close detached binaries where one star is a giant with a spectral type of G-K and its companion is a smaller star with a spectral type of F-G. These stars exhibit variability other than that due to eclipsing. This can be seen as a sinusoidal distortion in the stars light curve (Figure 1.6), hence is given the term 'distortion wave'. This is thought to be caused by a large number of sunspots on the larger stars surface.

The small star 'spins up' the rotation period of the large star. This rotation produces a large magnetic field which causes the star to have a very active chromosphere, which in turn produces the large number of starspots. These groups of starspots move around the star due to its rotation, which causes the stars brightness to vary as the dark spots come into and go out of view. It is this varying brightness that we see as the distortion wave. The rotation period and orbital period of the star may not be the same, hence

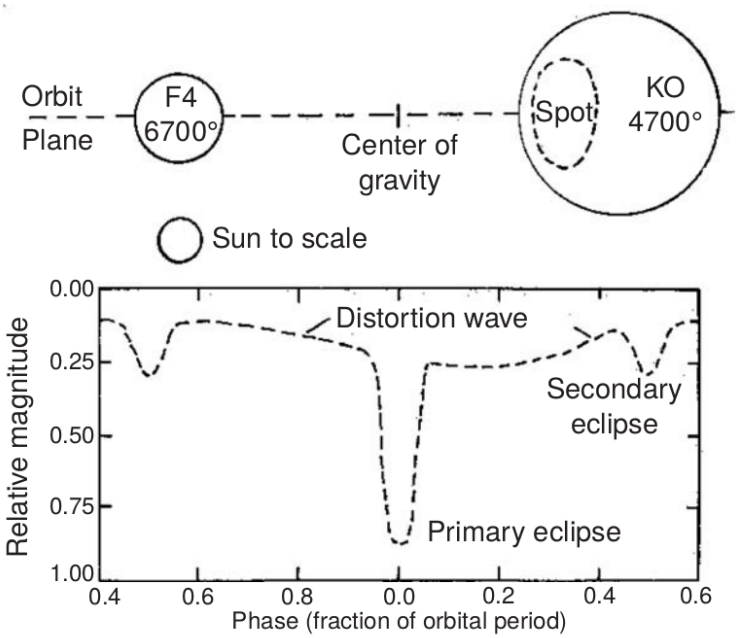


Fig. 1.6. Model of a RSCVn binary with corresponding light curve. The model on the top panel shows the larger star with the sunspots as well as the smaller star. The sun is also shown for scale. On the light curve in the bottom panel we see the primary and secondary eclipses as well as the distortion wave caused by the star spots.

the amplitude and phase of the distortion wave slowly vary with respect to the eclipses in the light curve. An example RSCVn radio light curve is shown in Figure 1.7.

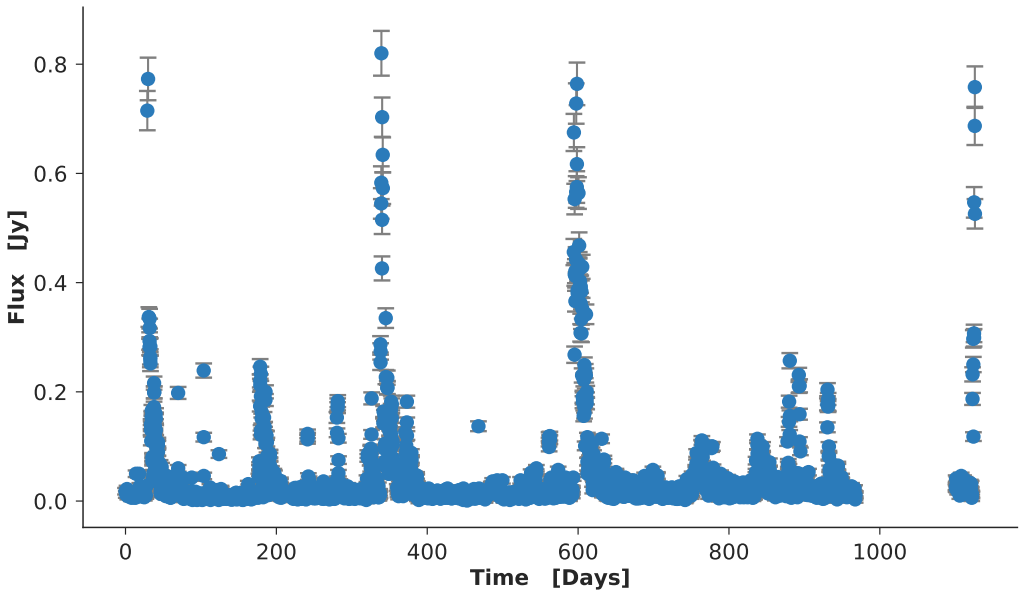


Fig. 1.7. An example RSCVn radio light curve, plotted as flux (in Jy) as a function of time (in days). It can be seen that the variations are frequent but have no obvious periodicity. The timescale of the variations are order of 10s of days, with a total observation time of ≈ 3 years. Data plotted is courtesy of [13].

1.3.5 Nova

Novae are a subclass of cataclysmic variables [20]. They are stars in close binaries where one of the stars is generally a white dwarf that is accreting matter from its companion, a cool main sequence star of type G to M. The main sequence companion fills its Roche lobe and then transfers matter through the inner Lagrangian point to an accretion disk around the white dwarf, similar to Figure 1.2.

As the white dwarf accretes hydrogen from its companion onto its surface, the hydrogen will eventually reach a critical temperature and ignite [21]. The relatively thin layer of hydrogen will quickly burn out after ignition and is pushed off the surface of the white dwarf creating the increase in brightness we see as the Nova. This ejected material interacts with the surrounding gas producing free - free emission that we see in the radio wavelengths.

Novae can also be subdivided into classical nova, recurrent nova and dwarf nova. The process we just described was that of classical novae. Recurrent novae, as the name suggests, have flared up more than once. There are only 8 of these objects known. Dwarf novae are hot dwarf variable stars, that flare up at irregular intervals. An example nova radio light curve is shown in Figure 1.8.

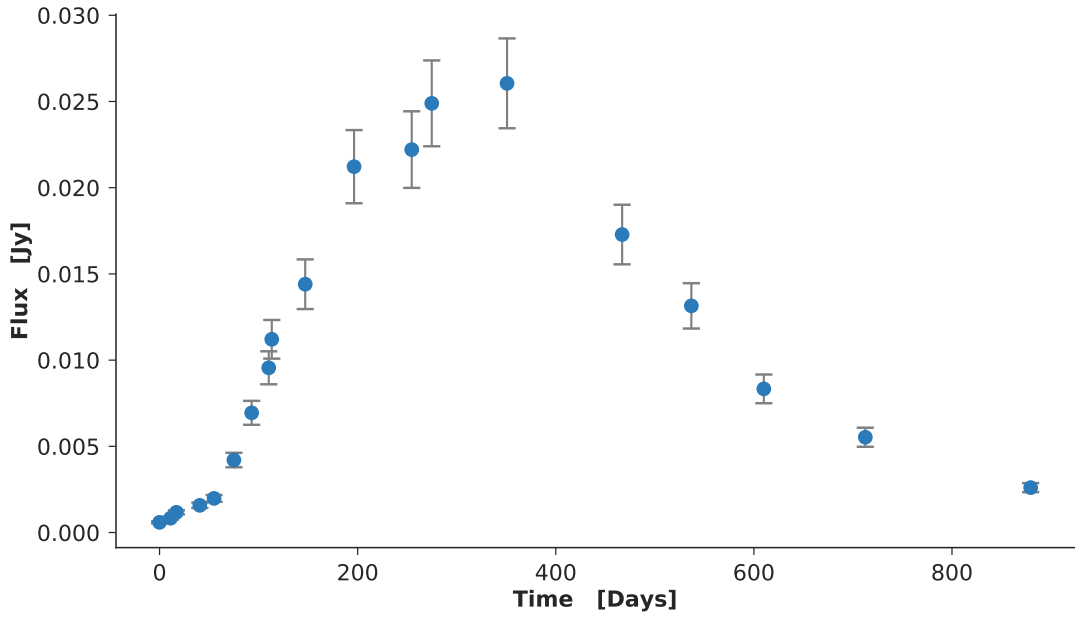


Fig. 1.8. An example nova radio light curve, plotted as flux (in Jy) as a function of time (in days). It can be seen that the timescale of this transient event is on the order of years, with a total observation time of ≈ 2.5 years. Data plotted is courtesy of [13].

1.3.6 Magnetar

A magnetar (mag) is a type of pulsar that has abnormally large magnetic fields [22]. Pulsars are rapidly rotating neutron stars which are remnants left by some supernovae

(Section 1.3.7). Their rapid rotation is due to the conservation of angular momentum. Standard pulsars have magnetic field strengths of between 10^{12} and 10^{14} Gauss where as magnetars were found to have magnetic field strengths of 10^{15} Gauss. Unlike standard pulsars, which are powered my rotation and accretion, magnetars are powered by the decay of their strong magnetic fields.

Pulsars eject matter from their poles, if their axis of rotation is inclined with respect to the earth we would see the brightness of the pulsar increase as the pole came into view then decrease as it went out of view. Thus the variation seen in standard pulsars are due to their rotation and not a physical variation in brightness of the star. Magnetar outbursts however, are thought to be caused by starquakes on the surface of the star that leads to huge amounts of magnetic energy being released. These outburts have been seen in the radio wavelengths as well as the visible and infra-red wavelengths. An example magnetar radio light curve is shown in Figure 1.9.

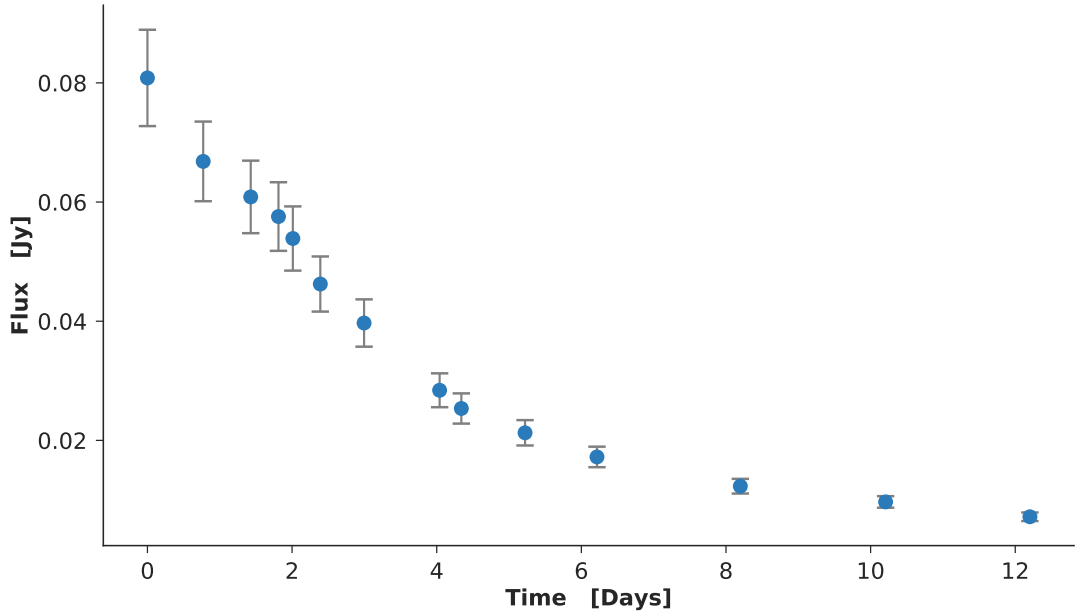


Fig. 1.9. A mag radio light curve, plotted as flux (in Jy) as a function of time (in days). The entire transient event has not been observed. It can be seen that the timescale of the decay of transient event is of a week, with the total observation time of ≈ 12 days. Data plotted is courtesy of [13].

1.3.7 Supernova

Supernovae are a type of exploding star, which peak at absolute magnitude -20 or brighter[11]. Supernovae are broadly grouped into two types based on their spectra. Type I supernovae have no hydrogen absorption lines in their spectrum where as type II do. Type I supernova are further subdivided into three groups:- Type Ia supernovae have no helium lines and strong silicon lines. Type Ib have strong helium lines. Type Ic have no helium lines and no silicon lines.

Type Ib, Ic and II supernovae occur when massive stars run out of fuel to burn in their cores. Once fusion in the core stops there is no longer any radiation pressure to counter act the gravitational force of its large mass, causing the star to collapse in on itself. Depending on the mass of the star before the collapse, it will leave behind either a neutron star or a black hole. Once the star explodes, the blast wave interacts with the interstellar medium as it moves away from the star at high speeds, it is this interaction that gives rise to the radio afterglow that we see as a radio transient. An example supernova afterglow light curve is shown in Figure 1.10.

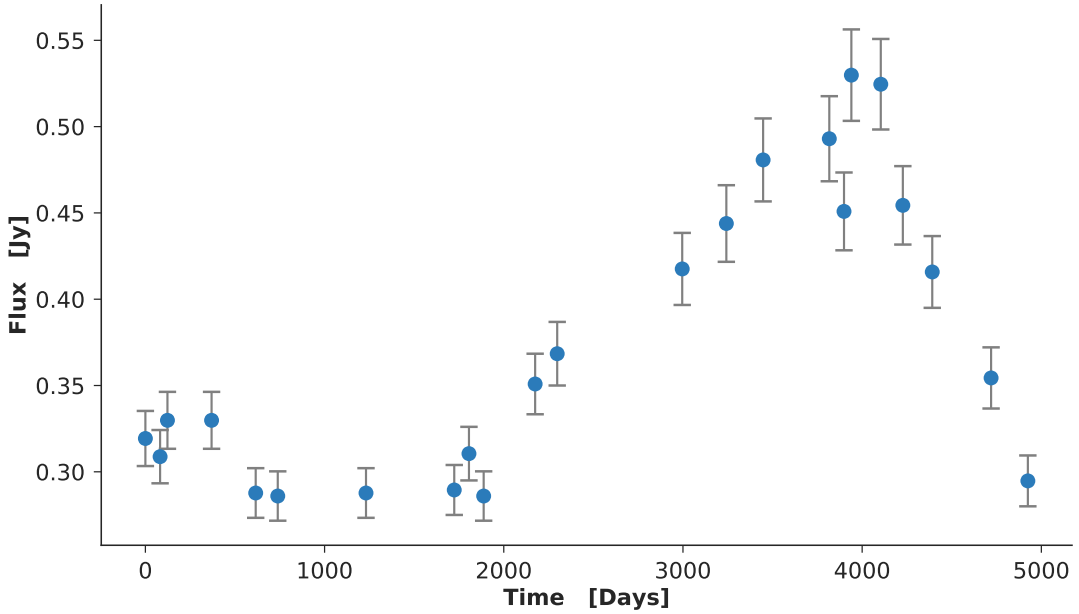


Fig. 1.10. An example SN radio light curve, plotted as flux (in Jy) as a function of time (in days). It can be seen that the timescale of this transient event is on the order of years, with a total observation time of ≈ 14 years. Data plotted is courtesy of [13].

Type Ia supernovae are a special type. The exact process in which these stars 'go' supernova is still uncertain, however it is commonly agreed that the progenitor system contains a white dwarf. The two most common competing theories are the single white dwarf model and the merging white dwarf model [23]. The first model consists of a white dwarf in a binary system where it accretes matter from a companion star until it gets close to the Chandrasekhar limit (1.4 solar masses). It then re-ignites fusion in its core and explodes in a nuclear runaway reaction [24]. In the second model, two white dwarf stars merge creating an object with a mass higher than the Chandrasekhar limit which then re-ignites fusion and explodes [25]. Type Ia supernovae are used as a distance measure to their host galaxies. This is due to the fact that they are thought to have a uniform peak absolute magnitude (once calibrated) and hence their distances can be inferred from their apparent magnitudes [26]. Ia supernovae have never been detected in the radio regime, detecting one is one of the science goals of the ThunderKAT project [3] (see Section 1.1).

1.3.8 Gamma Ray Burst

Gamma ray bursts (GRBs) are very high energy outbursts. They occur over a range of timescales from a few milliseconds to a few hundred seconds. The short time scale GRB's are thought to be the result of neutron star - neutron star mergers or neutron star - black hole mergers [27]–[29]. The long time scale GRB's are thought to be created from the core collapse of massive stars. There is not much known about these objects, however astronomers believe that these long time supernovae are the result of hypernovae – extremely energetic Type II supernovae (Section 1.3.7). A model of the current theory of how these are formed is shown in Figure 1.11.

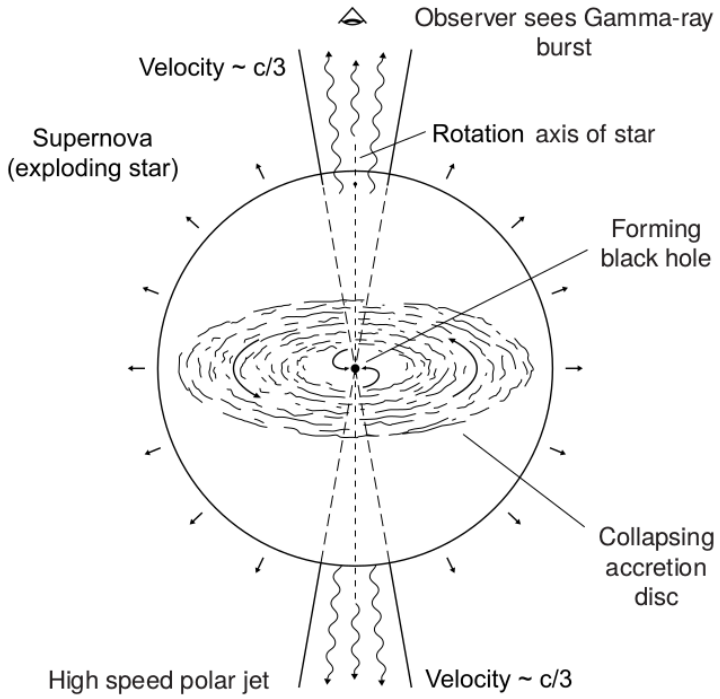


Fig. 1.11. Current model of the formation of GRBs. The shell of the exploding star is shown. At the center, the forming black hole with a collapsing accretion disk can be seen. Relativistic jets are seen ejected perpendicular to the plane of the accretion disk. The current model states that observers line of site coincides with the rotation axis of the black hole then a GRB will be observed. Diagram from [11].

Figure 1.11 shows a massive rotating star just after the core collapse. At the center is a forming black hole with an accretion disk around it. Matter is ejected out along the axis of rotation of the star during the supernova. The shock waves caused by this high energy jet are thought to produce the GRBs. When this high energy matter interacts with the surrounding medium it creates an afterglow that can be seen in both the visible and radio wavelengths. There have only been a few observations of a supernovae accompanying a GRB. This is due to the fact that under this model a GRB will only be observed if the stars rotating axis is along our line of sight. An example GRB radio light curve is shown in Figure 1.12.

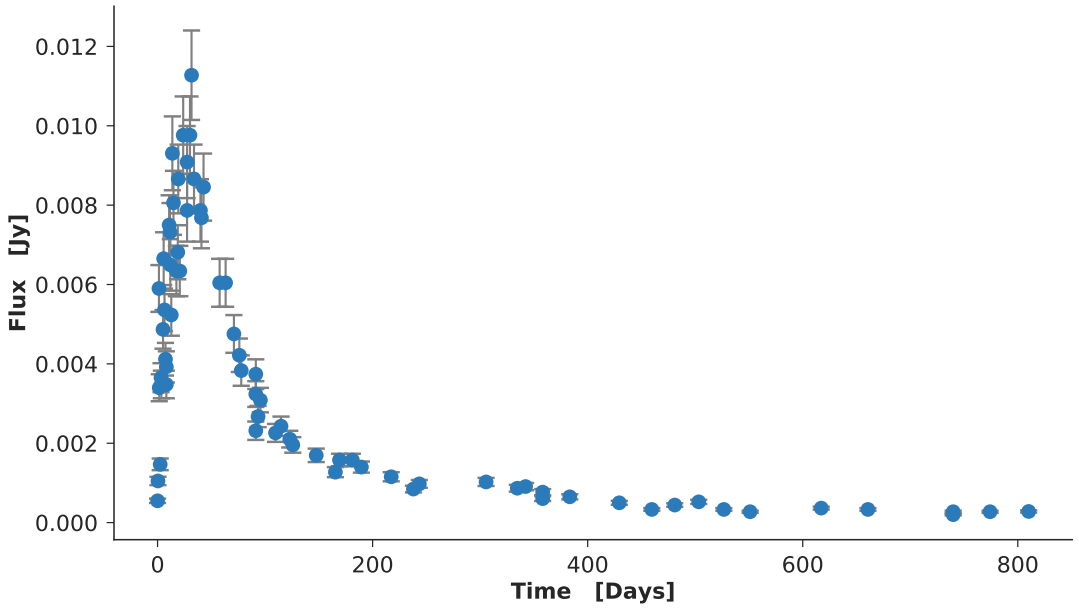


Fig. 1.12. An example GRB radio light curve, plotted as flux (in Jy) as a function of time (in days). It can be seen that the timescale of this transient event is ≈ 6 months, with a total observation time of ≈ 2 years. Data plotted is courtesy of [13].

1.3.9 Kilonova

When the matter ejected from a neutron star - neutron star or neutron star - black hole merger interacts with the interstellar medium it creates a transient 1000s of times brighter than nova, hence it was dubbed 'kilonova' (KN). The material from these mergers are very neutron rich, when this interacts with the interstellar medium it creates heavy radioactive elements via rapid neutron capture. It is the decay of these elements that powers the KN [30].

KN are excellent candidates for multimessenger[8] astronomy as they can be observed not only through electromagnetic radiation such as radio, gamma ray and x-ray emission but also through gravitational waves. The observation of the gravitational wave GW170817 was a great success for multimessenger astronomy as it was observed across the electromagnetic spectrum. The first KN observation was a counterpart to GW170817 [31], [32]. The first KN radio light curve is shown in Figure 1.13.

1.3.10 Tidal Disruption Event

A tidal disruption event (TDE) [33], [34] occurs when a star gets too close (within the tidal radius) to the central black hole of its host galaxy. The tidal radius, R_t, is defined as [35]

[8] Multimessenger astronomy is term that encompasses all four major astrophysical signals, namely: electromagnetic radiation, gravitational waves, neutrinos and cosmic rays.

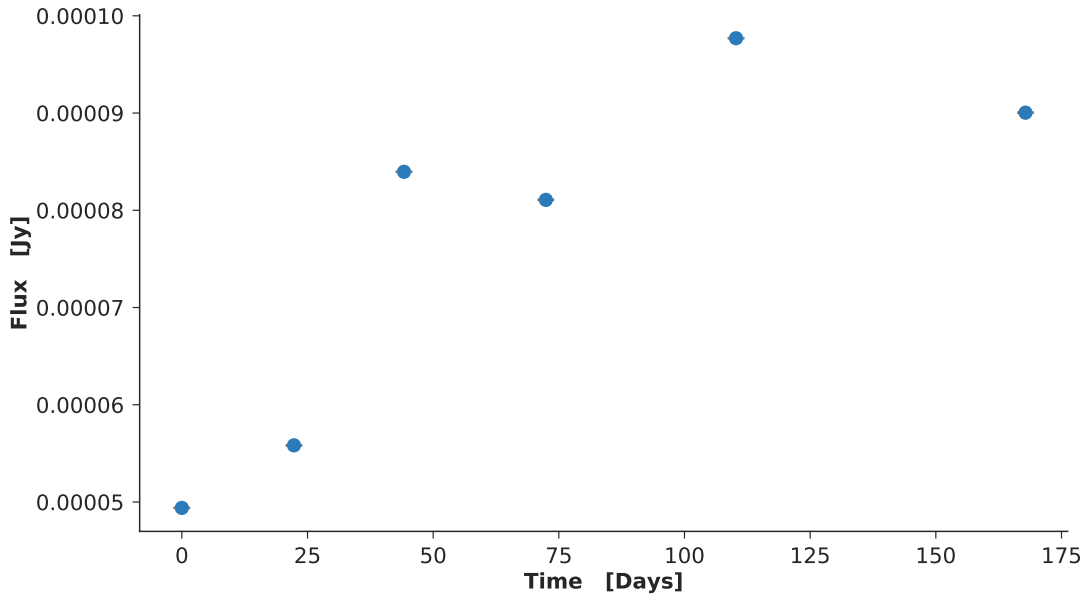


Fig. 1.13. A KN radio light curve, plotted as flux (in Jy) as a function of time (in days). As this is the first observed KN, the entire transient event has yet to be observed. It can be seen that the timescale of the rise of transient event is ≈ 5 months, with a current total observation time of ≈ 6 months. Data plotted is courtesy of [32].

$$R_t = R_* \left(\frac{M_{\rm bh}}{M_*} \right)^{1/3} , \tag{1.1}$$

where R_* and M_* are the stars radius and mass respectively and $M_{\rm bh}$ is the mass of the black hole.

As the star gets close to the black hole the tidal forces rip the star apart. Approximately half of the star matter escapes the system at high speed and the other half gets accreted onto the black hole. This matter is then ejected in relativistic jets. This event differs from normal AGN (Section 1.3.11) activity in that TDEs are much faster and have a much higher amplitude than regular AGN. TDEs are mainly seen in the optical and x-ray bands. These objects can also be sometimes be observed in radio wavelengths depending on the surrounding material with which the jets interact with [36]. An example TDE radio light curve is shown in Figure 1.14.

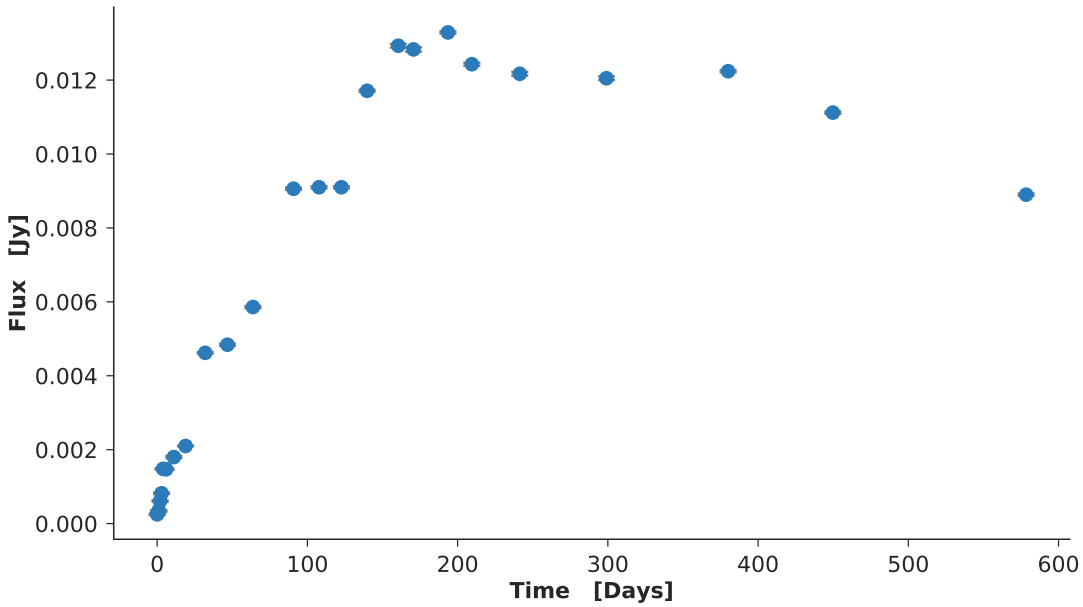


Fig. 1.14. An example TDE radio light curve, plotted as flux (in Jy) as a function of time (in days). The entire transient event has yet to be observed. It can be seen that the timescale of the rise of transient event is ≈ 6 months, with a current total observation time of ≈ 2 years. Data plotted is courtesy of [13].

1.3.11 Active Galactic Nuclei

At the center of most galaxies is a supermassive black hole. The term Active Galactic Nuclei (AGN) is used to describe all galaxies that have active supermassive black holes at their centers [37], [38]. Matter from the galaxy spirals into the center and accretes onto the active supermassive black hole, leading to massive amounts of matter being ejected out of the galaxy in relativistic jets as shown in Figure 1.15.

AGN vary on long time scales, generally on the order of months to years. This is due to the fact that orbital periods for objects around the AGN are generally long. When stars and matter fall into the black hole it causes an increase in brightness, however this is a stochastic process hence the variation we see in AGN is not periodic. Most of the radiation from AGN is obscured by the gas and dust in the accretion disk. This makes it easiest to observe in the radio wavelengths as these frequencies are not affected by dust and gas. An example AGN radio light curve is shown in Figure 1.16.

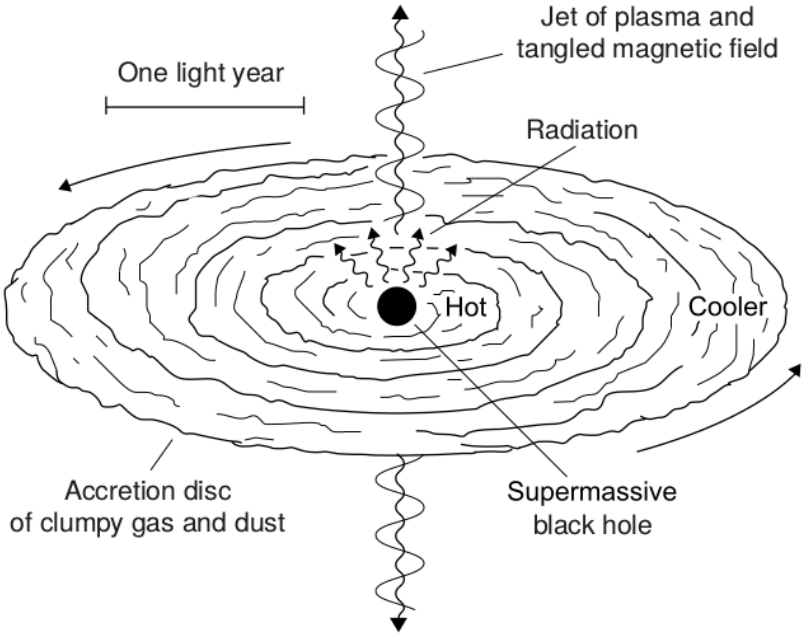


Fig. 1.15. Model of an AGN. The supermassive black hole is shown at the center, with an accretion disk around it. The jets are shown to be ejected perpendicular to the accretion axis. The characteristic scale is also shown. Diagram from [11].

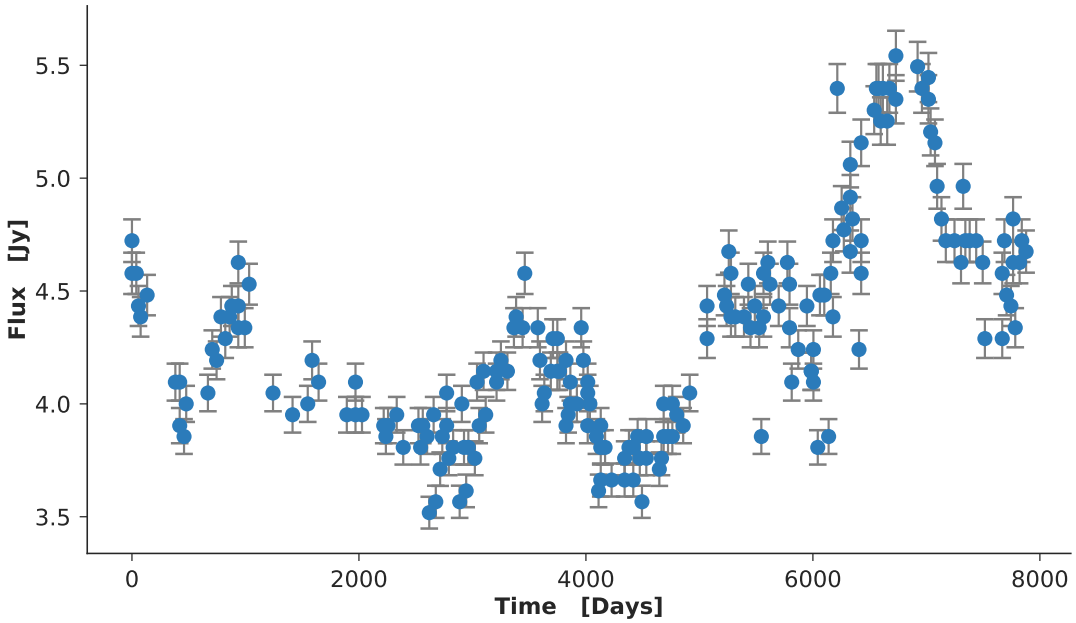


Fig. 1.16. An example AGN radio light curve, plotted as flux (in Jy) as a function of time (in days). It can be seen that the timescale of the variation seen is of the order of years, with a total observation time of $\approx$ 22 years. Data plotted is courtesy of [13].

Machine Learning

2

> *People worry that computers will get too smart and take over the world, but the real problem is that they're too stupid and they've already taken over the world.*
>
> **— Pedro Domingos**
> (Machine Learning Researcher)

Machine learning is the blanket term for how computers learn to apply algorithms to data. The algorithms may be anything from models that describe mappings between datasets to recognising underlying patterns present in datasets. More formally, Tom Mitchell defines as "a computer program learning from *experience* E with respect to some set of *tasks* T and *performance* metric P, if its performance at tasks in T, as measured by P, improves with experience E" [39]. An example of this would be an algorithm that learns to play chess by playing many games against itself.

Machine learning conceptually dates back to the mid 1900s [39]. However, most algorithms require large amounts of computing power. The scale of data available today (which will only grow exponentially in the future) has led to a rise in popularity of these algorithms. There exist problems that would simply take a human being too long to process and solve. Machine learning at its core aims to solve these problems by emulating, in some sense, the way humans learn information. This is far from being truly achieved – however there are many algorithms that perform sufficiently, or even outperform humans, at specific learning tasks [39]. In recent years machine learning has gained mainstream media coverage because of artificial intelligence systems beating humans in tasks such as image recognition [40], and playing the strategy game of Go [41].

There are many sub fields of machine learning, including but not limited to supervised learning, unsupervised learning and reinforcement learning[1]. In this chapter we will be focusing on unsupervised and supervised machine learning. These algorithms can be used for two broad tasks, namely, classification and regression. With classification the algorithm tries to predict discrete values, where as with regression the algorithm predicts continuous real values. In this work we will focus on classification however the topics discussed can be easily generalised to regression as well. In Section 2.1 the general approach to machine learning problems is discussed. In Section 2.2 and Section 2.3 the

[1] A full overview of reinforcement learning is out of the scope of this thesis, however the main goal of reinforcement learning is for a system or agent to automatically learn from actions taken in an environment

reader is given a general overview of common unsupervised and supervised machine learning algorithms respectively. Finally the work done with machine learning in Astronomy is outlined in Section 2.5.

2.1 Applying Machine Learning

Machine learning is a data-driven discipline, and generally approaches begin with extracting useful features (see Section 2.4) from data to support the algorithm's ability to learn from it. These features can be visually interrogated using techniques for visualising high dimensional datasets (see Section 2.1.2). Then, each algorithm usually has two types of parameters (see Section 2.1.3); parameters that the algorithm learns during training and hyperparameters that are chosen by the user. The next step is to train and test the algorithm (see Section 2.1.3). Finally, the performance of the algorithm is evaluated using a metric (see Section 2.1.4) that is generally tailored to the specific problem that the algorithm is being applied to.

2.1.1 Feature extraction

A dataset is the collection of measurements or observations. Examples of which include: images of different objects, time series measurements, data on people etc. Information from individual measurements/observations is stored as vectors. When dealing with classification problems these measurements often have a corresponding label, referred to as a class or type. As an example if the dataset is a collection of images of animals, the labels would be the names of the animal in each image. Each row or vector in the dataset is called an instance. Classical machine learning techniques can seldom use data in its raw format for classification. Feature extraction is a technique used to reduce the dimensionality of the data by summarising the information contained in the original data. The features one uses should also be well-separated between classes. Taking a repeating light curve as an example, features one could extract are the frequency, the amplitude and the phase among others. For more on feature extraction see Section 2.4.

2.1.2 Feature Visualisation

Features extracted from a dataset, while having a dimension smaller than that of the original dataset, can still have quite large dimensions. One way to visualise these features is through geometric projection techniques. These are techniques which find useful ways to project higher dimensional datasets onto 2 or 3 dimensions such that they can be visualised and interrogated. The most common approaches are statistical techniques such as factor analysis (e.g. principal component analysis, see Section 2.2.2), multidimensional scaling [42] or parallel coordinates [43]. For a review on visualisation techniques see de Oliveira et al (2003) [44].

In 2008 van der Maaten et al [45] showed that their technique, t-distributed stochastic neighbourhood embedding (see Section 2.2.3), outperformed the other visualisation techniques for various datasets. Thus this technique was used for feature visualisation for this study.

2.1.3 Training, Testing and Cross-validation

In machine learning, a dataset is generally split into two main subsets: the training set and the test set. Machine learning models have two types of parameters, namely model parameters and hyperparameters.

Model parameters are learned by the algorithm during training using the training set. The test set is reserved in order to check if the algorithm has learned an accurate model, and is only presented to the algorithm after the training step is complete. In some cases an algorithm can perform very well on the training set but perform poorly on the test set. This occurs when the algorithm overfits the model parameters to the training set and hence the model will not generalise to the test set. One method for overcoming this is known as k-fold cross validation[46]. First the dataset is split into the usual training and test sets, the training set is then further split into $k > 2$ subsets. One of these subsets is then used as the validation set while the others are used as the training set. This is then repeated k times until all k subsets has been used as a test set and average results are used for the model parameters. An example of a five fold cross validation is shown in Figure 2.1

Fig. 2.1. 5-fold cross validation is shown. The dataset is split into 5 equal parts. During each run, the algorithm is trained on $4/5$ parts and tested on the remaining part. This is repeated until all parts have been used as a validation set. The results are then averaged over.

Hyperparameters unlike model parameters are not learned during training, but are set by the user. Examples of these include the number of folds k in cross validation, the number of trees in a random forest (see Section 2.3.5) or the number of dimensions in t-SNE (see Section 2.2.3). These parameters can be optimised using cross-validation. First a range of hyperparameter values are defined. This is then iterated through, each

time doing a k-fold cross validation. After looping through all hyperparameters, the hyperparameters that performed the best on the cross validation are chosen.

2.1.4 Evaluating algorithms

Metrics are functions that evaluate the performance of a machine learning algorithm. Thus they vary depending on the problem one is trying to solve. Common metrics include accuracy, precision, recall, specificity and F1 score. Consider a two class binary classification problem with classes 1 and 0 with the positive class being class 1. When given a feature vector to classify a machine learning algorithm outputs the probability of the feature vector belonging to class 1 and 0. A threshold is placed on these probabilities to convert them into class labels, this is commonly set to 50%. For example if the algorithm outputs probabilities of 53% class 1 and 47% class 0 then it will be labelled as class 1.

Common metrics are defined using the following quantities:

- ***True Positives* (TP):** These are instances that the algorithm classified as class 1 that were in fact class 1
- ***True Negatives* (TN):** These are instances that the algorithm classified as class 0 that were in fact class 0
- ***False Positives* (FP):** These are instances that the algorithm classified as class 1 that were in fact class 0
- ***False Negatives* (FN):** These are instances that the algorithm classified as class 0 that were in fact class 1

These quantities can be generalised to multi-class problems.

The accuracy, precision, recall, specificity and F1 score are then defined as:

Accuracy:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.1}$$

Accuracy is one of the most commonly used metrics, it is the ratio of correctly classified instances to total number of instances. This would be a bad metric to use if the dataset was unbalanced, this is because if there were few instances of the negative class, the classifier could get them all wrong and still have a high accuracy.

Precision:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.2}$$

Precision only takes into account the instances for which the classifier labelled as the positive class. Once again this would not be a good metric to use if the dataset was unbalanced, if the there were few instances of the negative class the classifier could label all instances as a part of the positive class and get a relatively high precision.

Recall:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.3}$$

The recall is the ratio of the correctly classified positive instance to all the instances in the positive class. This metric is useful if the positive class is of particular interest as it specifically measures how well the classifier performs on the positive class.

Specificity:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.4}$$

Specificity is like recall except for the negative class. It is the ratio of correctly classified negative instances to all instances in the negative class. This metric is useful if the negative class is of particular interest as it specifically measures how well the classifier performs on the negative class.

F1 score:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

The F1 score is the weighted average of the precision and recall. This metric is particularly useful for unbalanced data sets as it takes into account both the false positives and false negatives making it a much more reliable measure than accuracy.

The performance of a machine learning algorithm can also be visualised in different ways. Two of the most common such methods – confusion matrices and receiver operating characteristic (ROC) curves are described below.

Confusion Matrix

A confusion matrix is a plot showing the true label of the object on one axis and the label predicted by the machine learning algorithm on the other. Once again taking a binary classification problem as an example, the confusion matrix would be a 2×2 matrix with the true positives and true negatives along the diagonal, and the false

negatives and false positives on the off-diagonals as shown in Figure 2.2. Therefore confusion matrices show how well the algorithm classifies each class. Classes classified correctly would appear on the diagonal, and incorrect classifications would appear on the off-diagonals.

Confusion matrices can be generalised to multi-class (more than two classes) classification problems. A multi-class confusion matrix is shown in Figure 2.2.

ROC Curves

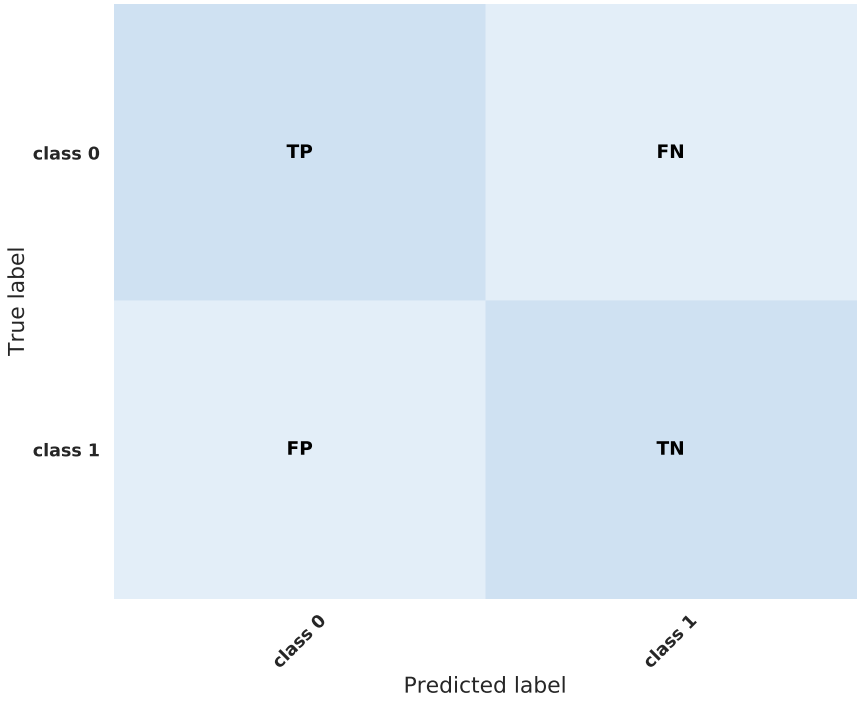
ROC curves show the trade off between the true positive rate (TPR) or recall (Equation 2.3) and the false positive rate (FPR) or 1 - specificity (Equation 2.4). ROC curves were developed during world war II to measure how well radar operators could differentiate between noise and enemy planes (hence the name Receiver Operator Characteristics). It then went on to be used extensively in signal detection theory [47]. ROC curves were first adopted into the medical community before finally being used by the machine learning community [48].

For the other metrics discussed in the section a threshold[2] has to be chosen before the metric is calculated. One of the advantages of the ROC curves is that it plots the performance of the algorithm as the threshold is varied. The TPR and FPR are plotted for different threshold values. A common ROC curve statistic that is used is the area under the ROC curve (AUC). If the machine learning algorithm is performing well, the AUC approaches 1. Example ROC curves are shown in Figure 2.3. Three curves are shown, the first is of an algorithm that chooses classes at random, the second is of an algorithm that is performing poorly and the last curve is of an algorithm that is performing well.

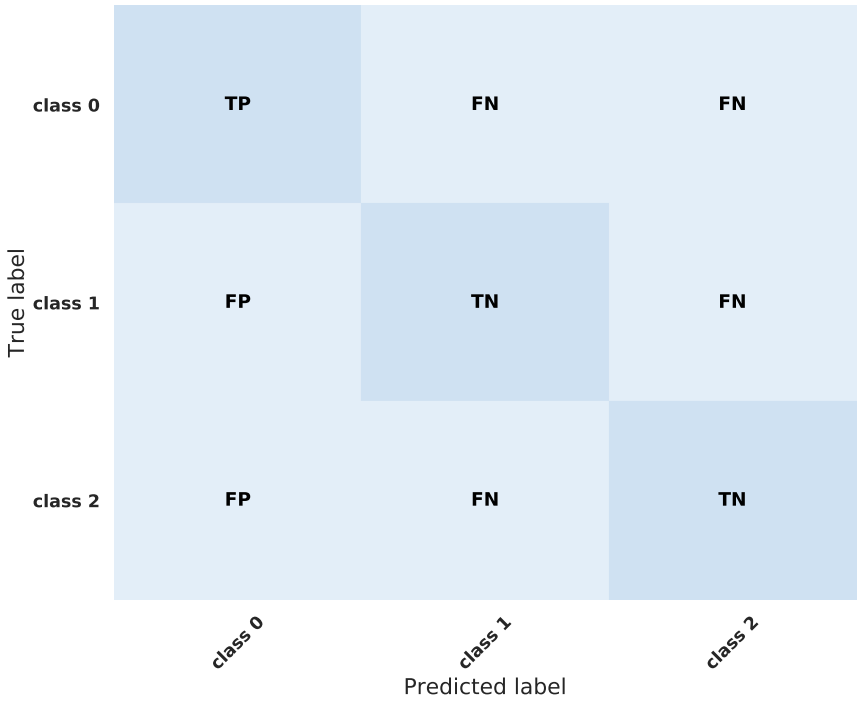
2.2 Unsupervised Learning

Unlike supervised learning, the main goal of unsupervised learning is not to learn a mapping from a set of input vectors to a set of output labels. Instead these algorithms are trained on unlabeled data. The unsupervised learning algorithm's task is then to learn patterns or structure present in the dataset. This is generally done through different clustering and dimensionality reduction algorithms. Example clustering and dimensionality reduction algorithms are outlined below.

[2] In classification, the threshold value is the value above which an instance is considered to be part of the positive class. For example if the threshold is set at 0.6, an output of 0.6 or higher means the instance would be classified as the positive class If the output is less than 0.6, the instance would be classified as the negative class.



(a) A confusion matrix for a binary classification.



(b) A confusion matrix for a multi-class classification.

Fig. 2.2. Confusion matrices with class 0 as the positive class. It can seen that the classifications that the algorithm get right are along the diagonals (TP and TN) and the classifications that the algorithm get wrong are along the off diagonals (FP and FN).

2.2.1 k-means

k-means [49], [50] is one of the most popular clustering algorithms due to its simplicity and ease of use. Given a dataset of n values, the algorithm works by first randomly

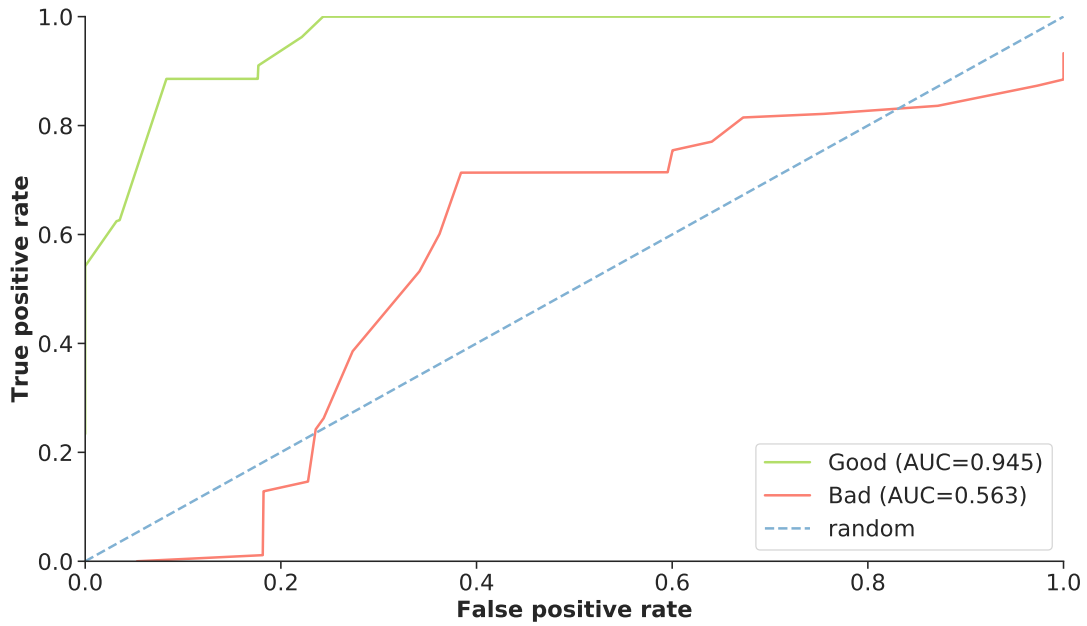


Fig. 2.3. Example ROC curves. Shown is blue is the ROC curve of an algorithm that assigns classes at random. Shown in red and green are the ROC curves of algorithms that are performing poorly and well respectively.

picking k of the data points. These points will be the first guess at the centroid locations for the k clusters. The other data points are then assigned to the clusters of the closest centroid. New centroid positions are calculated by averaging each of the clusters. The processes is then repeated until the centroids don't move within some threshold. The k-means algorithm is best understood through an example. Figure 2.4 shows the k-means algorithm going through four iterations. The centroids are shown as black stars and the 3 clusters are shown in blue, green and red. The first iteration shows the randomly picked centroids with each point coloured according to its closest centroid. The algorithm is then seen iteratively calculating new centroid positions from the mean of the three clusters until the centroid position does not change. The pseudocode for the k-means algorithm is outlined in Algorithm 1.

Algorithm 1: k- means pseudocode

Input: The number of clusters k, Training set $\boldsymbol{X}$

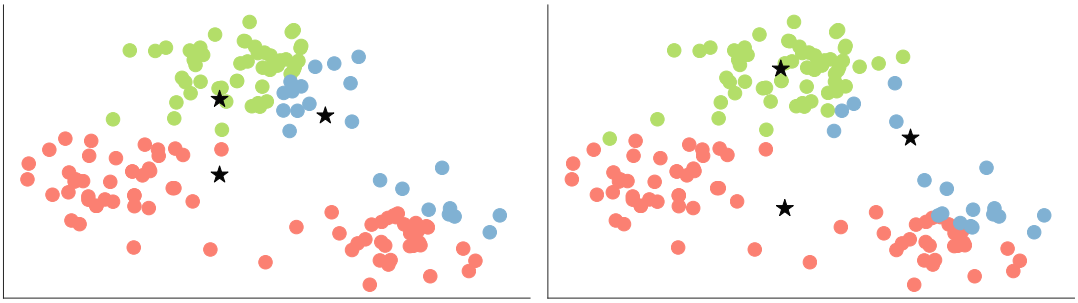
Output: Class labels Corresponding for training set $\boldsymbol{X}$ with k unique values

Initialise clusters by choosing k data points `// these will be initial centroid positions`

while *location of centroids changes* **do**

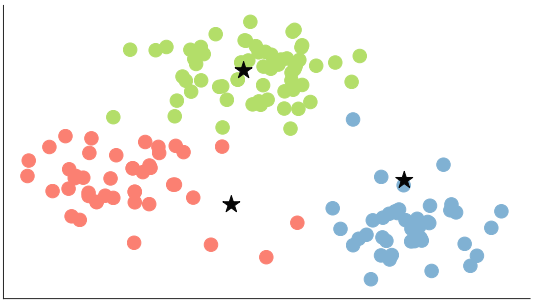
    Assign labels to all other data point to nearest cluster

    Re-calculate centroid position of each of the k clusters

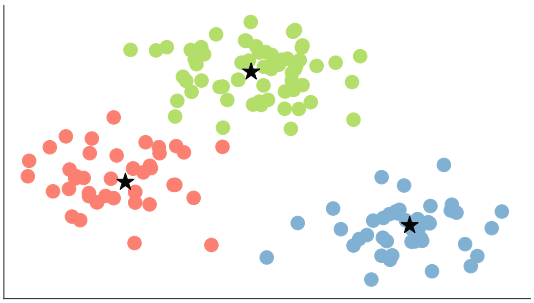


(a) Step 1

(b) Step 2



(c) Step 3



(d) Step 4

Fig. 2.4. An example of the k-means algorithm at work. The centroids are shown as black stars and the three clusters are shown in blue, green and red. Step one shows the randomly picked centroids with each point coloured according to its closest centroid. Step 2 and 3 shows the new centroids calculated from the means of the clusters. Step 4 shows the final clusters found by the k-means algorithm.

2.2.2 Principal Component Analysis

Principal Component Analysis (PCA) [51], [52] is a linear transformation that take a set of correlated variables to a set of uncorrelated variables that maximises variance. It is a widely used as dimensionality reduction technique, that reduces dimensions while minimising information loss.

Given a dataset D with m instances, each instance with n features, this can be written as an $m \times n$ matrix:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} & \mathbf{x_2} & \dots & \mathbf{x_n} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix} \tag{2.6}$$

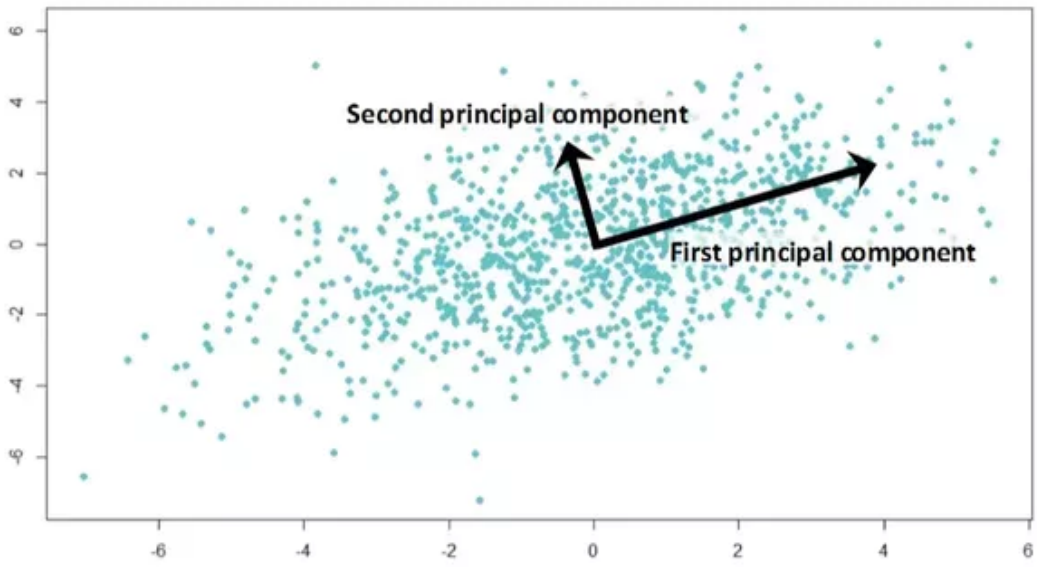


Fig. 2.5. Two dimensional data with first and second principal components shown.

The new uncorrelated variables, $\boldsymbol{Y}$, are given by [53]

$$\boldsymbol{T}\boldsymbol{X} = \boldsymbol{Y} \ . \tag{2.7}$$

The transform, $\mathbf{T}$, is a linear combination of the columns, $\mathbf{x_i}$, in the matrix $\mathbf{X}$,

$$\mathbf{T} = \sum_{i}^{n} \alpha_i \mathbf{x}_i = \boldsymbol{X}\boldsymbol{\alpha} \ , \tag{2.8}$$

where $\boldsymbol{\alpha}$ is a normalised vector of constants such that $\boldsymbol{\alpha}^\intercal\boldsymbol{\alpha} = 1$. The aim is to find the constants that maximises the variance of $\mathbf{T}$. The variance, $V(\boldsymbol{\alpha})$, of $\mathbf{T}$ is given by

$$\begin{aligned} V(\boldsymbol{\alpha}) &= \text{var}(\mathbf{T}) \\ &= \text{var}(\boldsymbol{X}\boldsymbol{\alpha}) \\ &= \boldsymbol{\alpha}^\intercal\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{\alpha} \ , \end{aligned} \tag{2.9}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ is the covariance matrix of $\mathbf{X}$. Under no other constraints the $V(\boldsymbol{\alpha})$ can be maximised by differentiating Equation 2.9 w.r.t $\boldsymbol{\alpha}$ and setting it equal to zero. However, from Equation 2.8 it can be seen that the constraint, $g(\boldsymbol{\alpha})$, is $g(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\intercal\boldsymbol{\alpha} - \mathbf{1} = 0$. The variance can be maximised under this constraint using the technique of Lagrange multipliers. The Lagrangian of this system is given by

$$L(\boldsymbol{\alpha}, \lambda) = V(\boldsymbol{\alpha}) + \lambda g(\boldsymbol{\alpha}) \,, \tag{2.10}$$

where λ is an arbitrary constant. Finding the maximum of Equation 2.10 will maximise $V(\boldsymbol{\alpha})$ under the constraint of $g(\boldsymbol{\alpha})$. The maximum is found by differentiating w.r.t to $\boldsymbol{\alpha}$:

$$\begin{aligned} \partial_{\boldsymbol{\alpha}} L &= 0 \\ \partial_{\boldsymbol{\alpha}} V(\boldsymbol{\alpha}) + \lambda \partial_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}) &= 0 \\ \partial_{\boldsymbol{\alpha}} \boldsymbol{\alpha}^{\intercal} \boldsymbol{\Sigma}_{\boldsymbol{X}} \boldsymbol{\alpha} - \lambda \partial_{\boldsymbol{\alpha}} (\boldsymbol{\alpha}^{\intercal} \boldsymbol{\alpha} - 1) &= 0 \,. \end{aligned} \tag{2.11}$$

Differentiating and rearranging Equation 2.11:

$$\begin{aligned} \boldsymbol{\Sigma}_{\boldsymbol{X}} \boldsymbol{\alpha} - \lambda \boldsymbol{\alpha} &= 0 \\ \implies (\boldsymbol{\Sigma}_{\mathbf{X}} - \lambda \mathbf{I}) \boldsymbol{\alpha} &= 0 \end{aligned} \tag{2.12}$$

Equation 2.12 can be recognised to be an eigenvalue equation, where λ is an eigenvalue and $\boldsymbol{\alpha}$ is an eigenvector of $\boldsymbol{\Sigma}_{\mathbf{X}}$. Using Equation 2.9 and Equation 2.12 we see that:

$$\begin{aligned} \max(\text{var}(\mathbf{T})) &= \max(\boldsymbol{\alpha}^{\intercal} \boldsymbol{\Sigma}_{\boldsymbol{X}} \boldsymbol{\alpha}) \\ &= \max(\boldsymbol{\alpha}^{\intercal} \lambda \boldsymbol{\alpha}) \\ &= \max(\lambda \boldsymbol{\alpha}^{\intercal} \boldsymbol{\alpha}) \\ &= \max(\lambda) \end{aligned} \tag{2.13}$$

Thus the eigenvector that captures most of the variation in the dataset $\boldsymbol{X}$ corresponds to the highest eigenvalue. This eigenvector is called the *first principal component* ($\boldsymbol{PC_1}$), the eigenvector corresponding to the next highest eigenvalue is called the *second principal component* ($\boldsymbol{PC_2}$) and so on. The transform, $\boldsymbol{T}$, can then be constructed to be

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{PC_1} & \boldsymbol{PC_2} & \dots & \boldsymbol{PC_m} \end{bmatrix} \,, \tag{2.14}$$

where the principal components $\boldsymbol{PC_i}$ are the eigenvectors ordered in descending order according to their corresponding eigenvalues. Thus the new uncorrelated variables, $\boldsymbol{Y}$ are found by projecting the original dataset $\boldsymbol{X}$ onto the principal components:

$$\mathbf{Y} = \begin{bmatrix} \boldsymbol{PC_1} & \boldsymbol{PC_2} & \dots & \boldsymbol{PC_m} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix} \tag{2.15}$$

The matrix $\mathbf{Y}$ still has the same dimensions as the original dataset $\mathbf{X}$. The dimensionality can be reduced by projecting onto only the eigenvectors ($\boldsymbol{PC_i}$) with the largest eigenvalues, which describe the majority of the variability in the dataset. The total variance of the data can be quantified by calculating the sum total of the eigenvalues. The number of eigenvectors kept are decided by the fraction of variability one would want to retain in the dataset; variability is defined as the sum of all eigenvalues. For example, if one wanted to keep 90% variability in the dataset, then one would keep the corresponding eigenvectors of the largest eigenvalues (in descending value) until their sum equals 90% of the sum of all eigenvalues.

Algorithm 2: PCA pseudocode

Input: Dataset $\boldsymbol{X}$

Output: Transformed dataset

for *each instance in X* **do**
    Calculate mean
    Normalise by subtracting mean

```
// Assign new matrix to variable called Xnorm
```

Xnorm ← Create new matrix with normalised instances
Cov ← Calculate the covariance matrix of Xnorm

Calculate eigenvectors and eigenvals of Cov
Project X onto eigenvectors and order according to eigenvals

2.2.3 t-distributed Stochastic Neighbour Embedding

t-distributed Stochastic Neighbour Embedding (t-SNE; [45]) is primarily a data visualisation technique (see Section 2.1.2) that aims to find lower dimensional representations of data. It works by computing the probability that two points are similar in the higher dimensional space based on its Euclidean distance. It does this for every pair of points in the feature set, then attempts to find a lower dimensional representation of these points that preserves the probability distribution. Given two higher dimensional data points x_i and x_j, the conditional probability, $p(x_j|x_i)$, is defined as

$$p(x_j|x_i) = \frac{\exp(-|x_i - x_j|^2/2\boldsymbol{\sigma}_i^2)}{\sum_{i \neq k} \exp(-|x_i - x_k|^2/2\boldsymbol{\sigma}_i^2)} \tag{2.16}$$

where $\boldsymbol{\sigma}_i$ is the variance of a Gaussian distribution centered at x_i. The $p(x_i|x_i)$ terms are set to zero. The conditional probability, $q(y_j|y_i)$, of the lower dimensional representations of data points x_i and x_j, y_i and y_j, is defined as

$$q(y_j|y_i) = \frac{(1 + |y_i - y_j|^2)^{-1}}{\sum_{l \neq k}(1 + |y_l - y_k|^2)^{-1}} \tag{2.17}$$

It can be seen that this lower dimensional representation is modelled as a Student t-distribution [54] with 1 degree of freedom unlike the higher dimensional probabilities which are modelled as Gaussian distributions.

The lower dimensional representations y_i and y_j correctly match the similarities between the higher dimensional data points x_i and x_j if the conditional probabilities p and q are the same. Defining P_i and Q_i as the symmetric joint probability distribution over the higher dimensional data points and lower dimensional representations respectively, such that $p(x_i|x_j) = p(x_j|x_i)$ and $q(y_i|y_j) = q(y_j|y_i)$ for all i, j. The difference between the two distributions can be minimised by minimising the Kullback-Leibler (KL; [55]) divergence between them.

The cost function, C, to minimise is then given by

$$\begin{aligned} C &= KL(P_i|Q_i) \\ &= \sum_i \sum_j p(x_i|x_j) \log \left[\frac{p(x_i|x_j)}{q(y_i|y_j)} \right] , \end{aligned} \tag{2.18}$$

The minimisation is commonly performed using the gradient descent method [45].

The main hyperparameter in the t-SNE algorithm is the variance, $\boldsymbol{\sigma}_i$ of the Gaussian distributions centered over each of the higher dimensional data points x_i. The variance is found using a fixed perplexity that is specified by the user. The perplexity, `Perp`, is defined as

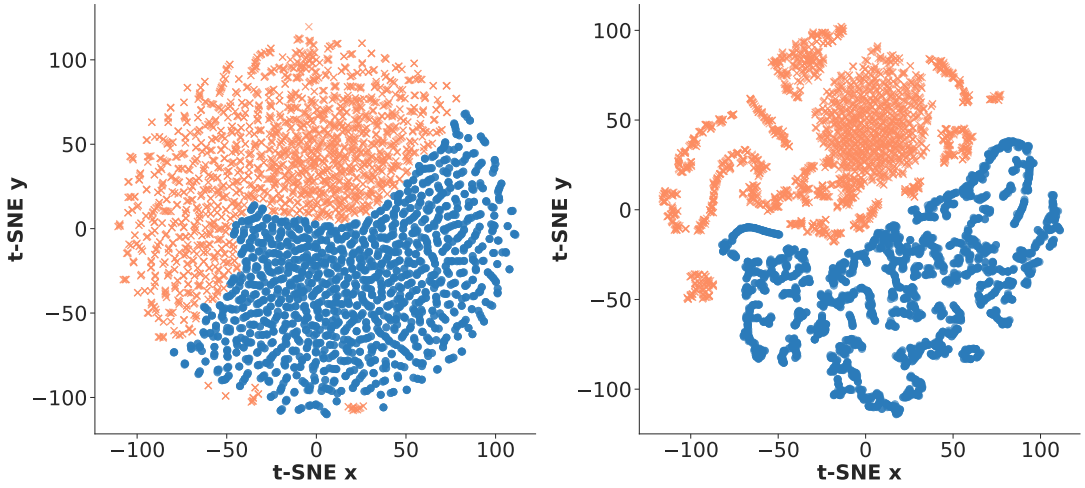
$$\texttt{Perp}(P_i) = 2^{H(P_i)} \tag{2.19}$$

where $H(P_i)$ is the Shannon Entropy defined as

$$H(P_i) = -\sum_j p(x_j|x_i) \log_2 p(x_j|x_i) \tag{2.20}$$

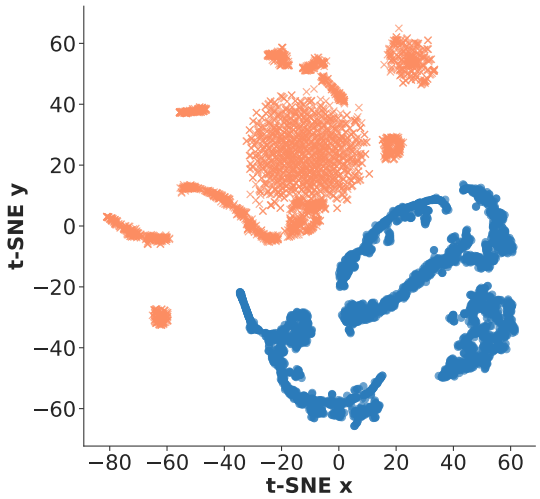
where H is measured in bits. The perplexity is a smooth measure of the number of effective neighbours of a data point x_i. Once a perplexity is chosen, t-SNE uses a binary search to find a value for $\boldsymbol{\sigma}_i$, with that fixed perplexity.

Because t-SNE finds a lower dimensional representation of these points while preserving the probability distribution it is very useful for visualising higher dimensional datasets by finding two or three dimensional representations of the original dataset which can then be visualised. An example of a dataset visualisation in two dimensions with different perplexities is shown in Figure A.9

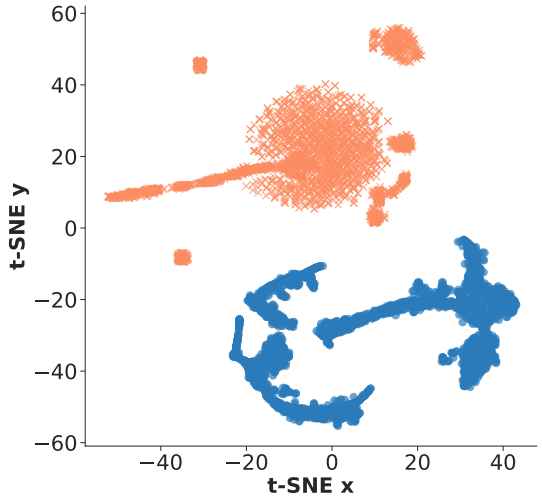


(a) Perplexity = 2

(b) Perplexity = 10



(c) Perplexity = 50



(d) Perplexity = 100

Fig. 2.6. t-SNE plots showing different perplexity values. It can be seen that as perplexity is increased the class clusters become tighter. It can also be seen that a perplexity of 100 gives no better clustering than perplexity 50, thus a higher perplexity is not always better.

Algorithm 3: t-SNE pseudocode

Input: Perplexity, no. of dim in lower dim representation, Dataset $\boldsymbol{X}$

Output: Lower dimensional representation of dataset

Calculate σ_i using a binary search and perplexity

$\boldsymbol{Y}$ ←*initialise lower dimensional representation by randomising all elements in* $\boldsymbol{Y}$

while *KL divergence not minimum* **do**

  for $i \leftarrow 1$ **to** n **do**

    `// n is the number of datapoints in X`

    for $j \leftarrow 1$ **to** n **do**

      Calculate $p(x_j|x_i)$

      Calculate $q(y_j|y_i)$

      Calculate KL divergence

  Update $\boldsymbol{Y}$ *using gradient descent*

2.3 Supervised Learning

Supervised machine learning makes use of a training set where each instance has a known label in order to predict quantities for an unlabelled test set. Thus supervised machine learning algorithms all perform one basic task; automatically creating a mapping from a set of input vectors or instances to a set of corresponding output labels. Once this mapping is learnt, new instances can be given to the algorithm and it will predict the best output label for this input. The labels can either have discrete single values as with *classification* problems or have continuous values as with *regression* problems. Here we briefly review the most common supervised machine learning algorithms. These include artificial neural networks, naive bayes classifiers, support vector machines, Gaussian process regression and random forests together with decision trees. In the analysis we will present in this thesis we will use exclusively the random forest algorithm, primarily because of their robustness to overfitting and overall good performance [56]–[58].

Many of these algorithms can be used for both classification and regression, however we will be looking at them in the context of classification problems unless stated otherwise. This chapter is primarily adapted from Mitchell et al [39].

2.3.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) get their name from the loose connection they share with a simple model of the brain. The most elementary model of the brain is one that

consists of many interconnected neurons. Each neuron takes inputs from many other neurons and has one output, which may in turn be connected to another neuron (and so on). ANNs, similarly, are built from multiple nodes that take in many inputs and produces one output which could then be connected to other nodes [39].

Nodes are considered to be single layer neural networks, that are able to model linear functions. By combining many nodes together more complex models can be modelled. Nodes can be combined by passing the output from one node into another. For example if you have two nodes, f_1 and f_2 then the combination would be, $F = f_2(f_1(\mathbf{x}, b))$. Adding a node like this is called adding another layer to the neural network. Many layers can be added in this fashion. The first layer of the ANN is called the input layer, the middle layers are called the hidden layers, and the final layer is called the output layer. The number of layers, the number of neurons in each layer and in the most general form for deep learning (see Equation 2.3.1) the type of each layer is defined as the architecture of the network.

The simplest nodes that can be used to build ANNs are called perceptrons [59]. A perceptron maps an input vector to a single real valued output between 0 and 1. First the dot product of the input vector and a weight vector, ω, is calculated, after which a bias constant is added. This is then passed through a function commonly called the activation function that produces a real valued output. Graphical models of a perceptron and an ANN are shown in Figure 2.7. There are many different activation functions that could be used, the most common of which include: ReLU, sigmoid and tanh functions. The ReLU, $R(x)$, and sigmoid, $S(x)$, activation functions are given by,

$$R(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \qquad S(x) = \frac{1}{1 + \exp^{-x}} \tag{2.21}$$

Once an architecture is chosen, the network is trained. At the start of training, all weight vectors in the network are initialised to some random value between 0 and 1. The weights are then optimised using an optimisation function, the most common include the mean squared error (Equation 2.22) and cross entropy log loss (Equation 2.23).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 \,, \tag{2.22}$$

where n is the number of data points, y_i is the true value and y_i' is the value predicted by the network. This optimisation function is used for regression problems.

$$\text{CE} = -\sum_{i}^{N} y_i \log p(x_i) \,, \tag{2.23}$$

where N is the number of training instances, y_i is the true label and $p(x_i)$ is the output of the network. This optimisation function is commonly used for classification problems. Based on the choice of activation function on the last layer of the network, the output of the network can be interpreted as a probability of input x_i being apart of the positive class.

These functions are generally optimised using stochastic gradient descent (SGD) [60] and backpropagation [61]. Methods such as SGD make use of gradient of the cost function to optimise the weights. ANNs have nested cost function (due to connecting layers together). Calculating the gradient of a nested function will require a lot of repeated calculations due to the chain rule, this makes it very computationally expensive. Backpropagation provides a more efficient method of calculating these derivatives

Deep learning

Recently deep neural networks (DNN) [62], [63] have become increasingly popular. The artificial intelligence systems that out performed human experts image classification [40] and the strategy game of Go [41] were built using deep neural networks.

Deep neural networks are ANNs with many hidden layers. There are different types of layers that could be used to construct a DNN, two common layers are the fully connected layer and convolutional layer [64], [65]. A fully connected layer is one in which every neuron in the layer is connected to every neuron in the previous layer. An example of this type of layer is shown in Figure 2.7. A convolutional layer is commonly used in image classification. It uses the convolution operation to automatically extract features from an image. The convolution operation can be thought of as sliding a filter over the image. The sliding starts at the top left of the image and moves to the bottom right one pixel at a time. Each time the filter is moved the dot product between the filter and the part of the image it is covering is computed.

DNNs can also be used for unsupervised learning tasks, for example a specific type of DNN called an autoencoder can be used to learn nonlinear compressions of data [66]. These networks are constructed to be symmetric about a central hidden layer with less nodes than the input layer. The network maps the inputs back to themselves, in doing so the network has to learn lower dimensional representations of the inputs as the central layer has less nodes than the input layer. It then has to learn how to convert these lower dimensional representations back to the input vectors (since it maps inputs back to themselves). The first half of this network is referred to as the *encoder* and the second half is referred to as the *decoder*

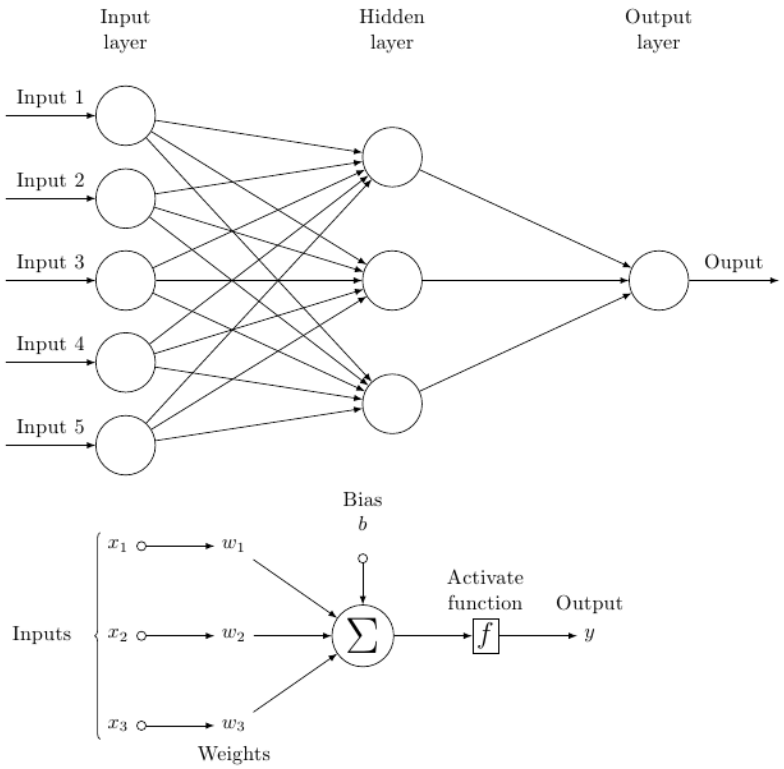


Fig. 2.7. A graphical model of a perceptron and ANN. The ANN consists of an input layer with five neurons, a hidden layer with three neurons and an output layer with a single neuron. The perceptron model shows three inputs with weights and a bias which is feed into the activation function to produce the output.

Algorithm 4: ANN pseudocode

Input: Network architecture, Training set $\boldsymbol{X}$,num of epochs

Randomly initialise weights ω

count $\leftarrow 0$

while *weights are not optimal or count != num of epochs* **do**

    Calculate output at first layer

    Apply activation function

    for *l in hidden layers* **do**

        for *neurons in l* **do**

            Calculate output from neuron using previous layers output as input

            Apply activation function

            count++

    Calculate error between output of last layer and true labels

    update ω *using backpropagation*

2.3.2 Naive Bayes classifiers

Bayes' Theorem states that the probability, $P(B|A)$, of event B given event A is given by:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{2.24}$$

Where $P(B|A)$ is the probability of event A given B and is called the posterior. $P(A|B)$ is the probability of event B given A and is called the likelihood, $P(B)$ is the probability of event B and is called the prior and $P(A)$ is the the probability of event A and is called the evidence.

The Naive Bayes' (NB) classifier algorithm adapts Bayes' Theorem to calculate the probability, $P(\mathbf{C_i}|\mathbf{x})$, that an instance with a given feature vector, $\mathbf{x}$, belongs to class $\mathbf{C_i}$[67]:

$$\begin{aligned} P(\mathbf{C_i}|\mathbf{x}) &= \frac{P(\mathbf{x}|\mathbf{C_i})P(\mathbf{C_i})}{P(\mathbf{x})} \\ P(\mathbf{C_i}|x_0, x_1, x_2, ...x_n) &= \frac{P(x_0, x_1, x_2, ...x_n|\mathbf{C_i})P(\mathbf{C_i})}{P(x_0, x_1, x_2, ...x_n)} \end{aligned} \tag{2.25}$$

The NB algorithm assumes that each feature in the feature vector is independent, hence Equation 2.25 can be simplified to

$$\begin{aligned} P(\mathbf{C_i}|x_0, x_1, x_2, ...x_n) &= \frac{P(x_0, x_1, x_2, ...x_n|\mathbf{C_i})}{P(x_0, x_1, x_2, ...x_n)} P(\mathbf{C_i}) \\ &= \frac{P(x_0|\mathbf{C_i}) \times P(x_1|\mathbf{C_i}) \times .... \times P(x_n|\mathbf{C_i})}{P(x_0, x_1, x_2, ...x_n)} P(\mathbf{C_i}) \\ &= \frac{\Pi_{j=1}^{n} P(x_j|\mathbf{C_i})}{P(x_0, x_1, x_2, ...x_n)} P(\mathbf{C_i}) \end{aligned} \tag{2.26}$$

The denominator of Equation 2.26 does not depend on class $\mathbf{C_i}$. Hence the final probability that a feature vector, $\mathbf{x}$, belongs to class $\mathbf{C_i}$ can be written as [68]:

$$P(\mathbf{C_i}|x_0, x_1, x_2, ...x_n) \propto \Pi_{j=1}^{n} P(x_j|\mathbf{C_i})P(\mathbf{C_i}) \tag{2.27}$$

The NB algorithm calculates the probability, $P(\mathbf{C_i}|\mathbf{x})$, for all classes $\mathbf{C_i}$. The form of the likelihood is often assumed to be Gaussian with the mean and standard deviation being learned from the training data. The feature vector $\mathbf{x}$ is then assigned the class $\mathbf{C_i}$ that has the highest probability.

Thus the class outputted by a NB classifier is given by:

$$\texttt{class} = \underset{\mathbf{C_i}}{argmax}\ \Pi_{j=1}^{n} P(x_j|\mathbf{C_i})P(\mathbf{C_i}) \tag{2.28}$$

Algorithm 5: Naive Bayes pseudocode

Input: Dataset $\boldsymbol{X}$

```
// Initialise to 1
```
$mulP \leftarrow 1$
```
// Initialise to empty array
```
$P \leftarrow [\,]$
```
// For class in all classes
```
for C_i *in* $\boldsymbol{C}$ **do**
    Calculate $P(C_i)$
    // For instance in X
    for x_j *in* $\boldsymbol{X}$ **do**
        $mulP \leftarrow mulP \times P(x_j|C_i)$
    $P \leftarrow append(mulP \times P(C_i))$

2.3.3 Support Vector Machines

Support Vector Machines (SVMs; [69]) are binary classifiers. This algorithm works by computing a hyper-plane in n-dimensional feature space that best separates the classes.

Consider a two class classification problem, with instances $\boldsymbol{X}$ and corresponding labels y_i. For the sake of easy visualisation the feature vectors will be two-dimensional, however this generalises to n-dimensions. We define a plane separating the two classes to be of the form:

$$\boldsymbol{\omega} \cdot \mathbf{X} + b = 0 \tag{2.29}$$

Where $\boldsymbol{\omega}$ is a weight vector and b is a bias constant.

As shown in Figure 2.8 there are many possible planes that can be constructed that accurately separates the two classes.

To find the “best" hyper-plane we first define two planes, called support planes, as follows:

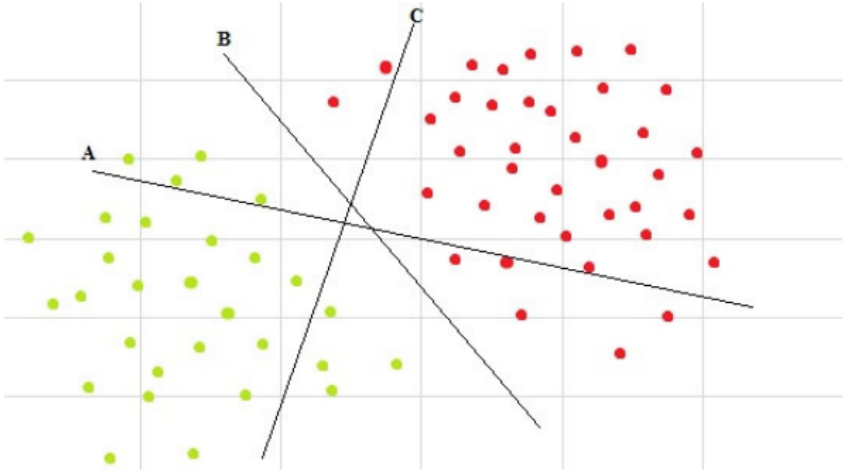


Fig. 2.8. The two classes are shown in red and yellow. Three different supporting planes are shown. It can be seen from this that there are an infinite number of supporting planes that could be drawn to separate the two classes. Figure courtesy of Ayush Ranjan[2].

$$\begin{aligned} \boldsymbol{\omega} \cdot \mathbf{X} + b \geq +1 \quad &\text{for the positive class} \quad (y_i = +1) \\ \boldsymbol{\omega} \cdot \mathbf{X} + b \leq -1 \quad &\text{for the negative class} \quad (y_i = -1) \end{aligned} \tag{2.30}$$

The distance between these planes are maximised by moving them apart until they hit points from their respective classes as shown in Figure 2.9. The points that are touching the planes are called support vectors and the distance between the planes is called the margin.

The distance between the two supporting planes is given by $d = \frac{2}{|\boldsymbol{\omega}|}$. Thus the maximum margin would be found by minimising the weight vector $\boldsymbol{\omega}$, with the constraints defined above. The hyperplane that best separates the two classes would then be the plane in the middle of the maximal margin.

Thus far we have assumed that the two classes are linearly separable. SVMs can also deal with non linearly separable classes. This is done by making use of the "kernel trick". In this context, a kernel is a function that maps an n-dimensional feature space to a higher dimensional space in which the classes may be linearly separable, as shown in Figure 2.10. It is important to note that the kernel trick is not appropriate for every data set.

Some of the most commonly used kernel functions are:

[2]Figures taken from SVM tutorial:
www.kaggle.com/ayushranjan15/eda-and-svm-basic-beginner/notebook

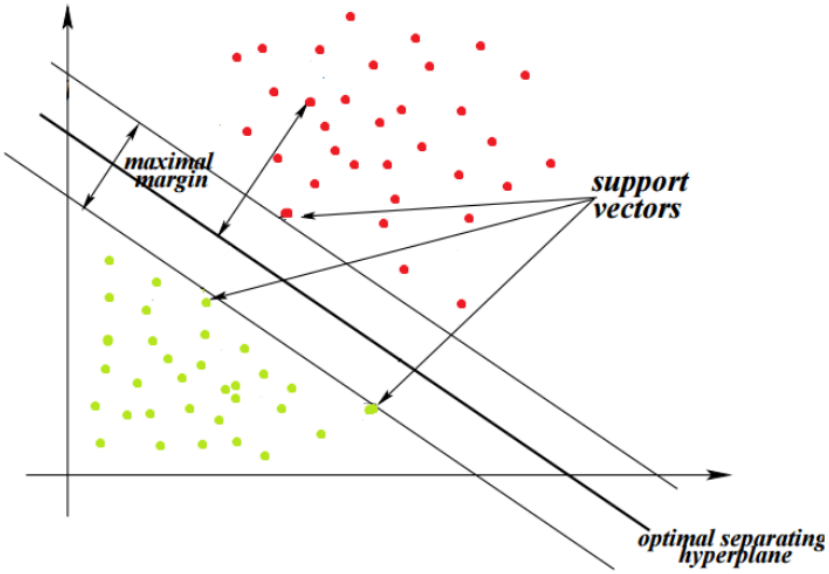


Fig. 2.9. Two classes shown in red and yellow. Two supporting planes as well as the support vectors are shown. The distance between the support planes is defined as the margin. The optimal separating hyper-plane is directly in the middle of the margin. Figure courtesy of Ayush Ranjan[2].

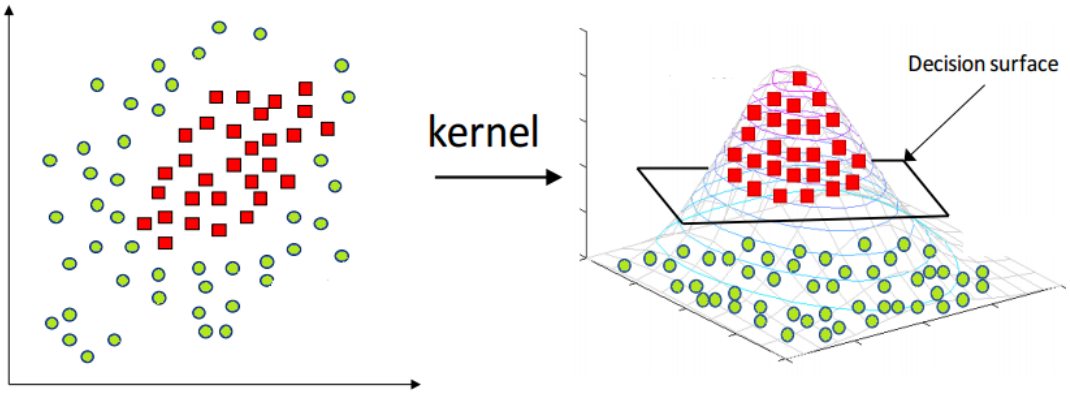


Fig. 2.10. The left panel shows the two dimensional features. It can be seen that the classes can not be separated with a hyper-plane in this space. Using a kernel function these features can be transformed into a three dimensional space (right panel). It can be seen that in this space the classes can be separated by a hyper-plane. Figure courtesy of Ayush Ranjan[2].

$$K(x_i, x_j) = \begin{cases} (x_i \cdot x_j)^p & \text{polynomial} \\ \exp\left[\frac{|x_i - x_j|^2}{2\sigma^2}\right] & \text{radial base} \\ \tanh(\kappa x_i \cdot x_j - \delta) & \text{neural net activation function} \end{cases} \tag{2.31}$$

There is no formal procedure for finding the 'best' kernel to use, however it is common practice to start with a linear SVM then move to more complex kernels using the linear

SVM results as a baseline. The hyperparameters of the kernel are optimised using cross validation (see Section 2.1.3) with a grid search over hyperparameter space.

Algorithm 6: SVM pseudocode

Input: Dataset $\boldsymbol{X}$

Output: weights for optimal separating hyperplane

Initialise b

Initialise $\boldsymbol{\omega}$

$d \leftarrow \frac{2}{|\boldsymbol{\omega}|}$

while d *not maximum* **do**

  update $\boldsymbol{\omega}$ *under constraints defined in Equation 2.30*

  $d \leftarrow \frac{2}{|\boldsymbol{\omega}|}$

2.3.4 Gaussian Process Regression

Gaussian processes (GPs; [70]) are a set of indexed random variables, defined such that every finite subset of a GP follows a multivariate normal distribution. Thus every subset of the GP can be characterised fully by its mean and covariance functions.

We define the mean to be

$$m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})] \,, \tag{2.32}$$

and covariance function (kernel) of the GP to be

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x'}) - m(\boldsymbol{x'}))] \,, \tag{2.33}$$

where $\boldsymbol{x}$ is a location vector and $f(\boldsymbol{x})$ is the random variable at that location.

Three of the most widely used kernels are defined below:

$$k(x_i, x_j) = \begin{cases} \sigma^2 \exp(-\frac{|x_i - x_j|^2}{2l^2}) & \text{squared exponential} \\ \exp(-|x_i - x_j|) & \text{exponential} \\ \exp\left(-\Gamma \sin^2\left[\frac{\pi}{P}|x_i - x_j|\right]\right) & \text{sine squared} \end{cases} \tag{2.34}$$

with hyperparameters, $\boldsymbol{\theta} = \{l, \sigma, \Gamma, P\}$.

A covariance function specifies a prior distribution over functions f. These functions are then marginalised over given the observed data $(\boldsymbol{x}, f(\boldsymbol{x}))$. This marginalised distribution can then be sampled at different x positions, $\boldsymbol{x}_*$ to get new function values $f(\boldsymbol{x}_*)$

$$f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{x}, f(\boldsymbol{x}) \sim \mathcal{N}\left(m(f(\boldsymbol{x}_*)) \ , \ cov(f(\boldsymbol{x}_*))\right) \ , \tag{2.35}$$

where

$$m(f(\boldsymbol{x}_*)) = k(\boldsymbol{x}, \boldsymbol{x}_*)k(\boldsymbol{x}, \boldsymbol{x})^{-1} f(\boldsymbol{x}) \tag{2.36}$$

$$cov(f(\boldsymbol{x}_*)) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{x}_*, \boldsymbol{x})k(\boldsymbol{x}, \boldsymbol{x})^{-1} k(\boldsymbol{x}, \boldsymbol{x}_*) \ . \tag{2.37}$$

If the observed data is noisy then the covariance functions can be written as

$$\begin{aligned} cov(x_i, y_i) &= k(x_i, x_j) + \sigma_n^2 \delta_{ij} \\ cov(\boldsymbol{x}, \boldsymbol{x}) &= k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \boldsymbol{I} \ , \end{aligned} \tag{2.38}$$

where σ_n is the observed variance, δ_{ij} is the Kronecker delta and $\boldsymbol{I}$ is the identity matrix. With this additional term, Equation 2.36 and Equation 2.37 become

$$m(f(\boldsymbol{x}_*)) = k(\boldsymbol{x}, \boldsymbol{x}_*)[k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \boldsymbol{I}]^{-1} f(\boldsymbol{x}) \tag{2.39}$$

$$cov(f(\boldsymbol{x}_*)) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{x}_*, \boldsymbol{x})[k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \boldsymbol{I}]^{-1} k(\boldsymbol{x}, \boldsymbol{x}_*) \ . \tag{2.40}$$

It can be seen from Equation 2.34 and Equation 2.38 that covariance functions often have hyperparameters, $\boldsymbol{\theta} = \{l, \sigma, \Gamma, P\}$. These hyperparameters can be optimised during the GP regression by maximising the log marginal likelihood. The log marginal likelihood, $\log p(f(\boldsymbol{x})|\boldsymbol{x}, \boldsymbol{\theta})$ is given by

$$\begin{aligned} \log p(f(\boldsymbol{x})|\boldsymbol{x}, \boldsymbol{\theta}) = &-\frac{1}{2} f(\boldsymbol{x})^\top [k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \boldsymbol{I}]^{-1} f(\boldsymbol{x}) \\ &-\frac{1}{2} \log |k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \boldsymbol{I}| - \frac{n}{2} \log 2\pi \ . \end{aligned} \tag{2.41}$$

Using Equation 2.38, this can be written as

$$\log p(f(\boldsymbol{x})|\boldsymbol{x}, \boldsymbol{\theta}) = -\frac{1}{2} f(\boldsymbol{x})^\top cov(\boldsymbol{x}, \boldsymbol{x})^{-1} f(\boldsymbol{x}) - \frac{1}{2} \log |cov(\boldsymbol{x}, \boldsymbol{x})| - \frac{n}{2} \log 2\pi \ . \tag{2.42}$$

To find the best set of hyperparameters, Equation 2.42 is maximised. The most common way of doing this is through a gradient based optimiser.

Algorithm 7: GP regression pseudocode

Input: kernel, $\boldsymbol{x}$, $\boldsymbol{y}$

Output: best fit kernel parameters

Initialise parameters, $\boldsymbol{\theta}$*, of kernel*
Calculate observed variance
Calculate covariance matrix
Calculate inverse covariance matrix
$logL \leftarrow \log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$

while $logL$ *not maximum* **do**

Calculate gradient of logL w.r.t $\boldsymbol{\theta}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \nabla \text{logL}$
$logL \leftarrow \log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$

2.3.5 Random Forests

Ensemble methods such as random forests [19], [71] build robust classifiers out of a multitude of weak learners such as decision trees. One of the main drawbacks of decision trees is the high variance on the outputs, which can be overcome by the averaging processes done by ensemble methods. To understand random forests we first need to take a detour into decision trees.

Decision Trees

A decision tree is a popular machine learning algorithm that approximates discrete valued target functions [39]. It creates a mapping, by making a series of "yes/no" decisions, from an input vector to an output label [72]. The tree is built in a top down approach, through a series of nodes. At each node a criterion is put on one of the components of the input vector. This node then splits into two branches one for each of the criteria options. This process is repeated at the nodes created at the ends of these branches. An instance is then classified by starting at the top of the tree, and working down through the nodes until a final answer is reached. The first node is referred to as the *root* node and the terminal nodes are referred to as *leaf* node.

Figure 2.11 shows an example of a decision tree. In this example the input vector has three components (X, Y and Z) and the output label has four options (A, B, C and D). The criteria imposed on the components of the input vector are shown in black. The leaf nodes are shown in blue. Consider classifying an input vector, $\vec{V} = (5, -2, -10)$. Starting at the top of the tree, we see that the $V_x < 10$ is our condition - thus, you move down the tree to the left. The next condition is $V_y < 0$, so to the left again. Finally,

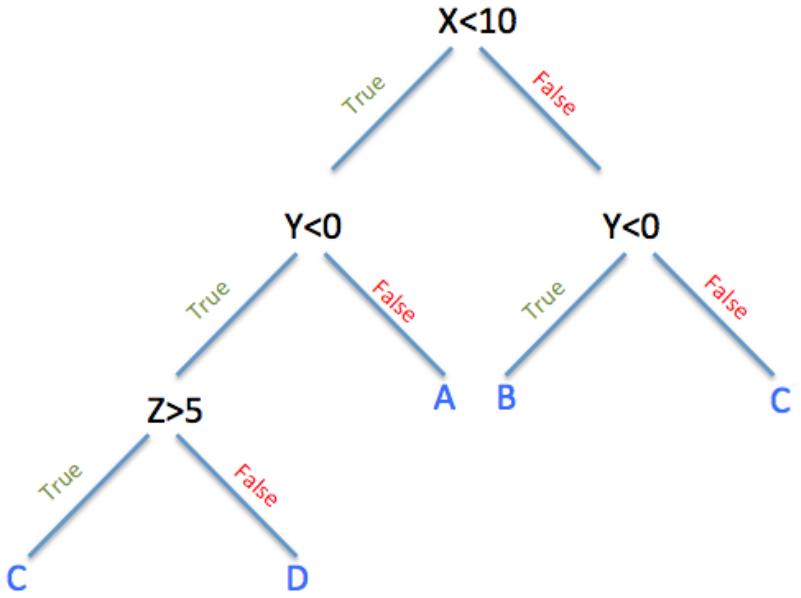


Fig. 2.11. Decision tree example with input vector with three components (X, Y and Z) and output label with four options (A, B, C and D). Following this tree it can be seen that an input vector, $\vec{V} = (5, -2, -10)$ would be classified as class D where as an input vector $\vec{V} = (11, 5, -10)$ would be classified as class C.

$V_y > 5$ moves you down the tree to the right, ending at the leaf node D. Hence the input vector falls into class D.

Constructing a forest

The first step in creating a random forest classifier is to create multiple decision trees. Each of these trees are trained on its own training set D_i. The training sets D_i are created from the original training data, D, through a process called *bagging* or *bootstrapping*. This is a sampling method that samples with replacement. If the original dataset D has m instances then the new datasets D_i are created by randomly sampling m instances from D each time replacing the instance that has been sampled. This means that instances can be picked more than once and others may never be picked at all. The probability of picking the data point x_i from a dataset of m instances is given by

$$P(x_i) = \frac{1}{m} \,. \tag{2.43}$$

When sampling with replacement, the probability of picking a data point is independent of all other draws, thus the probability of picking the same data point m times is given by

$$\underbrace{P(x_i) \times P(x_i) \times \cdots \times P(x_i)}_{m \text{ times}} = \left(\frac{1}{m}\right)^m . \tag{2.44}$$

Thus the probability of never picking a data point x_1 after m draws is:

$$\underbrace{P(\neg x_i) \times P(\neg x_i) \times \cdots \times P(\neg x_i)}_{m \text{ times}} = \left(1 - \frac{1}{m}\right)^m . \tag{2.45}$$

For large m this can be approximated as

$$\lim_{m\to\infty} \left(1 - \frac{1}{m}\right)^m \approx \frac{1}{e} \approx 0.37 , \tag{2.46}$$

thus the percentage of the original dataset that will not be in any given D_i is $\approx 37\%$.

Random forests also has a second stage of randomisation to further improve the robustness of the algorithm. Each node in the trees created are given a random subset of the full feature vector, i.e if each instance has p features, then each node is given $q < p$ features. The best feature to split on is then chosen from this random subset of features. A split is considered 'good' if it separates the classes in that subset. The two most common methods to choose the best splitting feature are the Gini index and information gain.

Gini index

The Gini index is a function that measures the impurity of the class labels if a split is applied. It minimises the function

$$G(x_i) = \sum_{j=1}^{N_i} P(x_{ij}) G(C|x_{ij}) , \tag{2.47}$$

where x_i is the feature that was used for the split, N_i is the total number of possible values x_i can take on and $p(x_{ij})$ is the probability of the feature x_i taking on its j^{th} value. $G(C|x_{ij})$ is defined as

$$G(C|x_{ij}) = 1 - \sum_{q=1}^{K} P^2(C_q|x_{ij}) , \tag{2.48}$$

where $P(C_q|x_{ij})$ is the probability of an instance belonging to class C_q given that the feature x_i has taken on its j^{th} value. K is the total number of values that x_i can take.

Information gain:

The information gain is given by

$$I(x_i) = \sum_{j=1}^{N_i} P(x_{ij}) H(C|x_{ij}) \ , \tag{2.49}$$

where x_i is the feature that was used for the split, N_i is the total number of possible values of x_i and $P(x_{ij})$ is the probability of feature x_i taking on its j^{th} value. The classification entropy $H(C|x_{ij})$ is defined as

$$H(C|x_{ij}) = \sum_{q=1}^{K} P(C_q|x_{ij}) \log_2 P(C_q|x_{ij}) \ , \tag{2.50}$$

where $p(C_q|x_{ij})$ is the probability of an instance belonging to class C_q given that the feature x_i has taken on its j^{th} value. This is similar to the Gini index as this too is a definition of an impurity measure.

Once a forest has been constructed, it assigns a class label to an input instance by a majority vote. The instance is given to each tree in the forest, each tree then outputs a label, the label that occurred the most would then be the final class label outputted by the random forest.

Algorithm 8: Random forest pseudocode

Input: Dataset $\boldsymbol{X}$, num of trees
Output: Trained forest

$n \leftarrow$ num of trees
$forest \leftarrow$ []
for $i = 0$ **to** n **do**
    $train \leftarrow$ bootstrap sample of $\boldsymbol{X}$
    $f \leftarrow$ random subset of feature vecs in $train$
    initialise tree
    while *not leaf node* **do**
        At each node split on best feature in f
    $forest \leftarrow$ append(tree)

2.4 Feature extraction

As mentioned previously feature extraction is a technique used to reduce the dimensionality of the data by summarising the information contained in the original data. For this study the feature extraction technique that was used was that of the wavelet transform.

2.4.1 Wavelet Transform

Similar to Fourier decomposition where time series data is decomposed into a sum of sine and cosine basis functions, wavelet decomposition breaks up the time series data into a sum of wavelet basis functions. The work reviewed in this section follows that of Chun-Lin [73] and Mallat et al [74]. The continuous wavelet transform (CWT) is given by

$$f(s,\tau) = \int f(t)\psi^*_{s,\tau}(t)dt \ , \tag{2.51}$$

where $\psi^*_{s,\tau}$ is the complex conjugate of the wavelet bases. Wavelet basis functions have the form

$$\psi_{s,\tau}(t) = \frac{1}{s}\psi\left(\frac{t-\tau}{s}\right) \ , \tag{2.52}$$

where s is a scaling variable and τ is a translation variable.

One condition imposed on wavelet basis functions is referred to as the admissibility condition which requires that the Fourier transform of the wavelets vanish at zero, that is

$$|\Psi(\omega)|^2\Big|_{\omega=0} = 0 \ , \tag{2.53}$$

where Ψ is the Fourier transform of ψ. This condition implies that wavelets must have a bandpass like spectrum and hence can be interpreted as high pass filters. Low pass filters, ϕ, can be defined as decomposition of the wavelet functions, hence

$$\phi(t) = \sum_{j,k} a(j,k)\psi_{j,k}(t) \ , \tag{2.54}$$

where $a(j,k)$ are coefficients and $\psi_{j,k}(t)$ are the wavelets. ϕ is may also be refered to as the scaling function.

Wavelet bases are problem specific and can be constructed to fit the problem. Some of the most commonly used wavelets are highlighted below:

- Haar Wavelets [75]:

$$\psi(t) = \begin{cases} -1 & \text{if } 0 \leq t < 1/2 \\ 1 & \text{if } 1/2 \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.55}$$

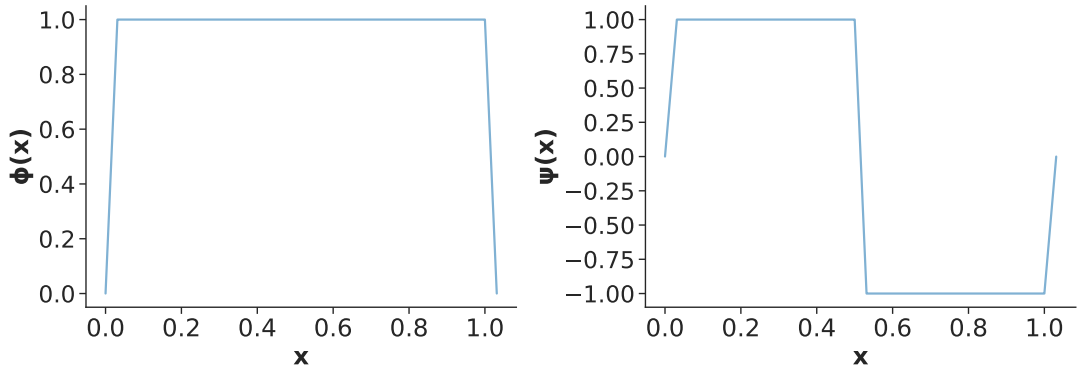


Fig. 2.12. Example plots for the Haar scaling, ϕ, and wavelet, ψ, functions.

- Meyer wavelets [76]:

$$\psi(\omega) = \begin{cases} 1 & \text{if } |\omega| \leq 2\pi/3 \\ \cos\left[\frac{\pi}{2}\nu(\frac{3}{2\pi}|\omega| - 1)\right] & \text{if } 2\pi/3 \leq |\omega| \leq 4\pi/3 \\ 0 & \text{otherwise} \end{cases} \tag{2.56}$$

where ν is any infinitely differentiable function such that:

$$\nu(t) = \begin{cases} 0 & \text{if } t \leq 0 \\ 1 & \text{if } t \geq 1 \end{cases} \tag{2.57}$$

and the $\nu(t)$ smoothly increases from 0 to 1.

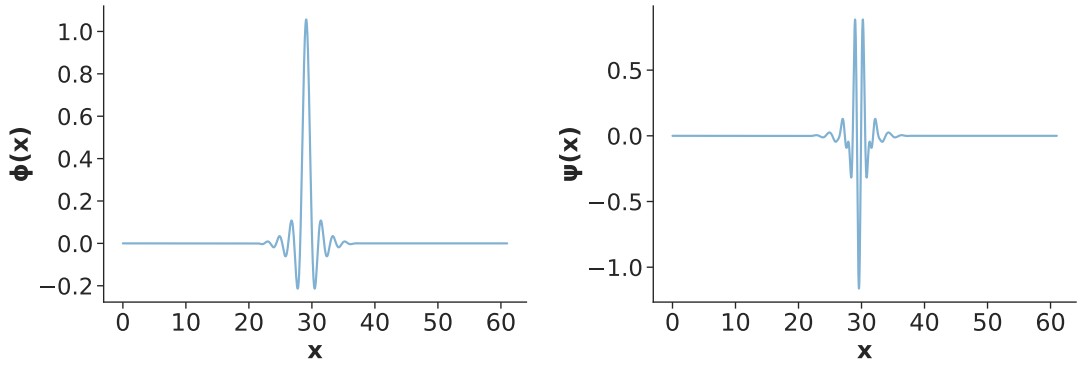


Fig. 2.13. Example plots for the Meyer scaling, ϕ, and wavelet, ψ, functions.

- Daubechies symmlet wavelets [77]:

  Equation 2.54 can also be written in the form

$$\frac{1}{\sqrt{2}}\phi\left(\frac{t}{\sqrt{2}}\right) = \sum_{n=-\infty}^{n=\infty} h[n]\phi(t-n)\ , \tag{2.58}$$

  where

$$h[n] = \left\langle \frac{1}{\sqrt{2}}\phi\left(\frac{t}{\sqrt{2}}\right), \phi(t-n)\ . \right\rangle \tag{2.59}$$

  The term $h[n]$ is referred to as the discrete filter. Daubechies wavelets are defined as

$$\hat{h}(\omega) = \sqrt{2}\left(\frac{1+e^{-i\omega}}{2}\right)^p R(e^{-i\omega})\ , \tag{2.60}$$

  where $\hat{h}$ is the Fourier transform of $h[n]$ and $R(e^{-i\omega})$ is a designed to be a polynomial of degree m such that $\hat{h}$ satisfies

$$|\hat{h}(\omega)|^2 + |\hat{h}(\omega)|^2 = 2\ . \tag{2.61}$$

  Rewriting Equation 2.60 as a polynomial in $\cos\omega$, we get

$$|\hat{h}(\omega)|^2 = 2\cos(\omega/2)^{2p} P(\sin^2(\omega/2))\ , \tag{2.62}$$

  where $|R(e^{-i\omega})|^2 = P(\sin^2(\omega/2))$. Daubechies found the polynomial P to be [74]

$$P(y) = \sum_{k=0}^{p-1} \binom{p-1+k}{k} y^k\ , \tag{2.63}$$

  and

$$R(e^{-i\omega}) = r_0 \prod_{k=0}^{m} (1 - a_k e^{-i\omega})\ . \tag{2.64}$$

  Thus the relationship between R and P was shown to be

$$|R(e^{-i\omega})|^2 = P\left(\frac{2 - e^{i\omega} - e^{-i\omega}}{4}\right)\ . \tag{2.65}$$

Finally, Daubechies symlet wavelets are created by selecting roots of $P\left(\frac{2-e^{i\omega}-e^{-i\omega}}{4}\right)$ that have a linear complex phase then substituting those back into Equation 2.60. This leads to Daubechies wavelets that are more symmetric.

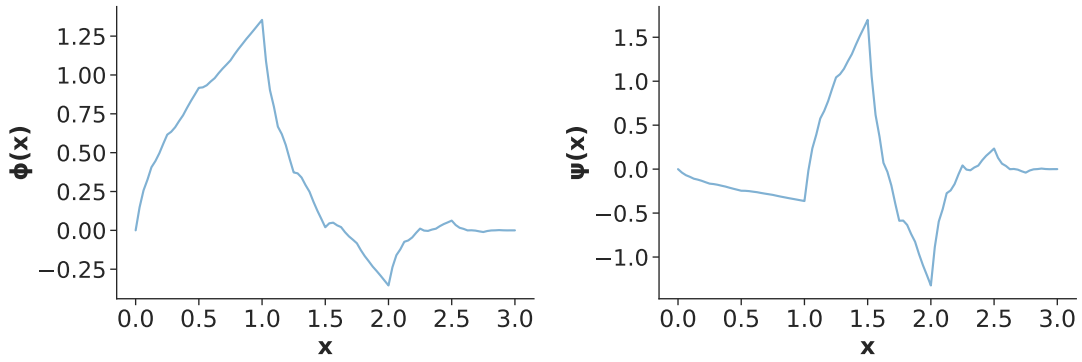


Fig. 2.14. Example plots for the Symlet scaling, ϕ, and wavelet, ψ, functions.

Discrete Wavelet Transform

Given some wavelet basis functions(or low and high pass filters), $\phi(x)$ and $\psi(x)$, a discrete time series, $f(x)$, can be decomposed using the discrete wavelet transform (DWT) as

$$f(x) = \frac{1}{\sqrt{M}} \sum_k W_\phi(j_0, k)\phi_{j_0,k}(x) + \frac{1}{\sqrt{M}} \sum_{j=j_0}^{\infty} \sum_k W_\psi(j, k)\psi_{j,k}(x) \ , \tag{2.66}$$

where M is the total number of data points in the discrete time series and

$$\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k) \ , \tag{2.67}$$
$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k) \ , \tag{2.68}$$

where $j \in \mathbb{Z}$, 2 is a scaling constant and k is the position parameter.

The wavelet coefficients $W_\phi(j_0, k)$ and $W_\psi(j, k)$ are then given by

$$W_\phi(j_0, k) = \frac{1}{\sqrt{M}} \sum_n f(x)\phi_{j_0,k}(x) \ , \tag{2.69}$$
$$W_\psi(j, k) = \frac{1}{\sqrt{M}} \sum_n f(x)\psi_{j,k}(x) \quad j \geq j_0 \ . \tag{2.70}$$

Equation 2.69 and Equation 2.70 are referred to as the approximate and detailed coefficients respectively.

Figure 2.15 shows the iterative DWT: Equation 2.66 is applied to the time series data which outputs the detailed coefficient, D_1, and approximate coefficients, A_1. Equation 2.66 can then be applied to the approximate coefficients to get detailed coefficient, D_2, and approximate coefficients, A_2. This process can be done iteratively an arbitrary number of times. When a DWT is performed once it is referred to as a one level decomposition, if a DWT is performed twice iteratively it is referred to as a two level decomposition and so on.

Stationary Discrete Wavelet Transform

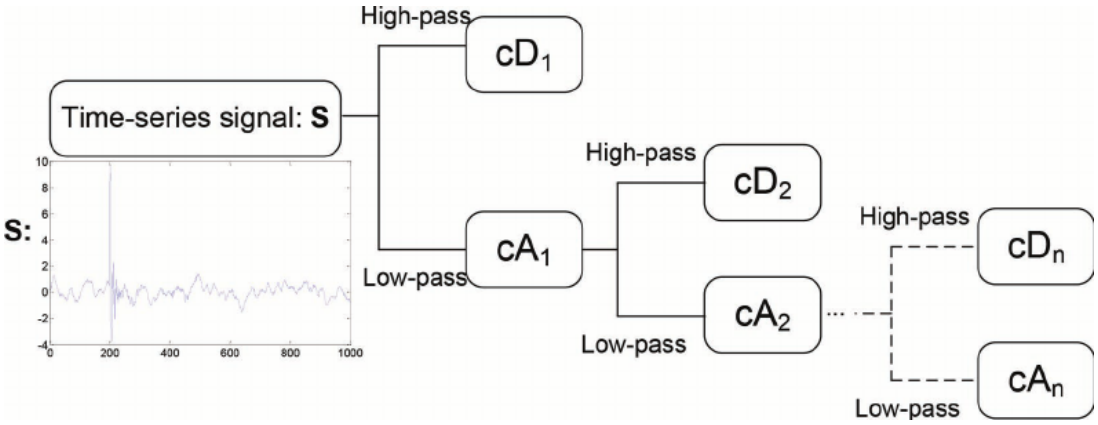


Fig. 2.15. Wavelet decomposition illustration.

Each iteration of the DWT downsamples the original time series data by a factor of two i.e only every second data point is used. This is done so that the output coefficients is of the same dimension as the input time series. For example if a DWT is performed on an input time series with 100 data points, then the output will be 50 D_1 coefficients and A_1 coefficients. This downsampling limits the memory usage of the algorithm however, due to using only every second data point the DWT loses information that is retained by CWT. This information loss leads to the DWT losing the property of translation invariance that is possessed by the CWT. Approximate translation invariance can be recovered by removing the downsampling at the cost of using more computer memory. A DWT algorithm that does not downsample the input signal is called a stationary discrete wavelet transform (SWT). Due to not being downsampled SWTs have a larger number of redunant features.

2.5 Machine Learning in Astronomy

The work reviewed in this section closely follows that of [72]. There has been a considerable amount of research done in astronomy using machine learning techniques. This work can be split up into their respective astronomy sub-fields.

Stellar and galaxy classification

Most of the early work done in astronomy using machine learning was on separating galaxies and stars from digitised photographic plats. Odewahn et al [78] presented work on using ANNs as a stellar - galaxy discriminator. Lahav et al [79] showed in the mid 90's how ANNs could be used to classify galaxies. They could classify galaxies from images and spectra equally as well as human experts. In the early 2000's Bailer-Jones et al [80] showed how ANNs could be used for stellar classification. Zhang et al [81] split early and late type galaxies using an unsupervised machine learning algorithm called k nearest neighbours. White et al [82] showed how decision tree classifier could be used for a range of astronomy classification problems including discriminating between stellar objects and galaxies. More recently, Khalifa et al [83] , showed that deep convolutional neural networks could separate galaxies into types: spiral, elliptical and irregular with an overall accuracy of 97%. Wathela et al [84] used deep convolutional neural networks to classify galaxies into classes of compact, bent, Fanaroff-Riley (FR) I and II, achieving and overall accuracy of 97%.

Photometric redshifts

Estimating redshifts using photometry has greatly increased in popularity in recent years. This is because even though it is less accurate than spectroscopic methods, the sheer volume of photometric data available reduces the statistical noise in ensemble calculations. A review on the early work done in using photometry to measure redshifts was done by Koo et al [85] in 1999. The two most common approaches are using templates or training machine learning algorithms on the empirical data. There has been extensive work done on using ANNs to find the redshift for low redshift galaxies, some include Banerji et al [86] who used ANNs to find the photometric redshifts for the Dark Energy Survey, Oyaizu et al [87] and Zang et al [81] who used ANNs to find the photometric redshifts for the Sloan Digital Sky Survey. There has been work done using other machine learning algorithms, Carliles et al [88] used random forests to measure the redshifts of galaxies in the Sloan Digital Sky Survey. Ball et al [89] used k nearest neighbours to generate full photometric redshift probability density functions for galaxies in the Sloan Digital Sky Survey. For low redshift galaxies the results from the two methods are similar with a root-mean-square deviation of ≈ 0.02 in redshift. For higher redshifts the accuracy declines. While some supervised machine learning techniques have been successful, they are hampered by the small datasets available. It is likely that hybrid template - supervised learning as well as semi-supervised learning will improve these results [72]. Recently, D'Isanto et al [90] used a deep convolutional network combined with a mixture density network to predict redshift probabililty density functions.

Variables and transient classification

There has been a substantial amount of work done with machine learning in transient astronomy over the last decade. This includes research done by Bailer-Jones el al [91] in stellar classification, image-based classification of supernovae done by Romano et al [92] and Bailey et al [93], classifying variable stars done by Richards et al [94], and photometric supernovae classification done by Newling et al [95]. In recent years, these algorithms have been used successfully in classifying optical transients, such as classification of transients in SDSS images done by Buisson et al [96], supernovae done by Lochner et al [58], variable sources done by Farrell et al [97] or general optical transients done by Mahabal et al [98]. Some machine learning methods have been investigated for the upcoming ASKAP survey for Variables and Slow Transients (VAST) [99], but these were only applied to optical data.

Part II

MALT - MAchine Learning for Transients

Applying MALT

3

> *Without ambition, one starts nothing. Without work, one finishes nothing. The prize will not be sent to you. You have to win it.*
>
> — **Ralph Waldo Emerson**
> (American essayist and philosopher)

In the first half of this chapter we outline a general approach to classifying transients with multiwavelength data (which we call MALT - MAchine Learning for Transients) which is heavily based on the work done by Lochner et al [58]. The approach is split into two main sections; the first builds a machine learning classifier that uses light curve data of any wavelength and the second deals with combining data from different sources. This creates a general approach useful for combining all sources of information that may be useful to classifying transients, in addition to classification using the light curves themselves. In the second half of this chapter we apply MALT to radio and optical transient data. The data are described in Section 3.2, the data augmentation method is described in Section 3.3, the feature extraction methods are described in Section 3.4, lastly the additional information that was added is outlined in Section 3.5.

3.1 General approach to multiwavelength transient classification

Section 3.1.1 describes techniques required for data preprocessing, including data augmentation which is required to increase the number of training samples for machine learning. Section 3.1.2 describes the feature extraction technique which is a technique used to summarise the information contained in a dataset. Section 3.1.3 outlines the dimensionality reduction technique, finally the method for adding multiple sources of information is described in Section 3.1.4.

3.1.1 Data Augmentation

Machine learning algorithms require very large datasets to train on. In many cases large datasets do not exist, for example, there are only ≈ 100 labelled radio transients. In cases such as these, where the dataset is too small to use for machine learning, a technique called data augmentation is used to create new data from existing data. This artificially creates a large dataset that can then be used for machine learning. Data augmentation can be broken down into two main approaches: augmenting the real

data in data-space or augmenting the features in feature-space [100]. Using image classification as an example, the first type of augmentation would be to change the original images by rotating, cropping, mirroring etc (actions that preserve the image labels), then extract features from these new images. The second type of augmentation would be to extract features from the original images, then transform these features in a way that preserves the image labels to create new features.

One widely used method of augmenting data is to use a model to simulate new data, however models are not available for all classes of radio transient hence a different technique had to be used. The shapes of the radio transient light curves are determined by the underlying physical processes that govern these objects. We expect these physical processes to be fairly similar for objects in the same class hence we expect the light curves of objects in the same class to be statistically similar. We use a data augmentation technique called Gaussian processes (see Section 2.3.4) that creates new data that is statistically consistent with the original data uses the supervised machine learning technique . We implement this technique to augment our data, similar to [101].

3.1.2 Feature extraction

As mentioned in Section 2.4, feature extraction is a technique that is used to reduce the dimensionality of the data by summarising the information contained in the original data. A useful set of features for time series data, $f(x)$, is a decomposition into a linear combination of basis functions

$$f(x) = \sum_k a_k \phi_k(x) \ , \tag{3.1}$$

where $\phi_k(x)$ are orthogonal basis functions and a_k are the respective coefficients.

This is a common approach in the field of signal processing and can be a powerful tool for feature extraction, the set of coefficients used as the features with a machine learning algorithm. One widely used form of this is a Fourier decomposition, where a signal can be decomposed into the component frequencies. However Fourier decomposition loses all localisation information and is thus mostly applicable to regular, repeating signals.

By contrast, in transient classification, the object may be observed at any point in its light curve and the algorithm must determine its class in this setting. Thus, we require a decomposition method that is translation-invariant but still sensitive to the intrinsic shape of the curve. A form of decomposition that is approximately scale and translation-invariant is known as the stationary wavelet transform [74], [102]. See Section 2.4.1 for more details. Following its successful use in [58] and [103], we make use of the stationary wavelet transform with the *symlet* family, as implemented in the package `PyWavelets`[1]

[1] https://github.com/PyWavelets/pywt

3.1.3 Dimensionality Reduction

The feature extraction technique outlined in Section 2.4.1 outputs a large number of features, thus the dataset has a large number of dimensions. The performance of most machine learning algorithms degrade with higher dimensionality. Working with higher-dimensional datasets can also be very computationally expensive. Thus the dimensions of the dataset are generally reduced before using it for any machine learning. The are many techniques available for dimensionality reduction, common techniques including: multidimensional scaling [104], [105], linear discriminate analysis [106] and singular value decomposition [107]. For this work we used principal component analysis (see Section 2.2.2).

3.1.4 Combining multiple data sources

The last piece of the puzzle required is a framework for incorporating additional data that can't be decomposed with wavelets. This data can be prior or external information about the sources, such as fluxes of the source in different wavelengths or, contextual information, such as position of the object in the sky. Multimessenger information from alert streams from other observatories can also be added as external information (e.g. the presence of gamma ray emission from the Fermi[2] telescope or a gravitational wave detected by LIGO[3] in the region of a new radio transient source can be highly discriminating).

There are two methods for incorporating information from other sources: a probabilistic approach or adding the information in as an extra feature. The specific setup of the problem will dictate which approach is more appropriate, but formalising this process is a step towards automated multimessenger machine learning pipelines, that can then be fed into downstream analysis including spectroscopic follow-up prioritisation.

Probabilistic Approach:

Most machine learning classification algorithms are capable of producing a score that can be interpreted as a probability of an object belonging to a particular class. To combine this with external information, such as the presence of a coincident alert from another observatory, we can calculate the prior probability, $P(\mathcal{C})$, of the object being in a certain class $\mathcal{C}$, given all prior information. This probability, $P(\mathcal{C})$, would then be multiplied by the probability given by the classifier to give a final probability of some object being in class $\mathcal{C}$.

[2] https://fermi.gsfc.nasa.gov
[3] www.ligo.org

Extra Features:

The second method is to use the information as an extra feature in the machine learning process. For example, if one has a flux measurement at any other wavelength, one could add that flux as a feature. The advantage of this approach is that correlations between the different features are learned automatically by the machine learning algorithm, potentially resulting in improved classification accuracy.

The disadvantage of this latter approach is that machine learning algorithms do not intrinsically handle missing data. This could happen if, for instance, MeerKAT detects a transient during a daytime observation when MeerLICHT[4] [1] cannot observe. While feature imputation techniques exist [108], a more interpretable approach may be to combine the probabilities where, if data are missing, a default probability based on prior observations (for example, known transient rates) can be used.

3.2 Data

3.2.1 Radio Data

The radio transient data used were collected by [13] except for the kilonova light curve, which was collected by [32]. Most of the data is obtained from the literature and the rest is from the Green Bank Interferometer[5] (GBI). Data collected from the GBI have a much higher cadence than the data obtained from the literature. The data consists of time series light curve data (radio flux as a function of time). These radio transient light curves can be separated into eleven different classes or types. The total number of light curves in each type and their sources are shown in Table 3.1. It can be seen from this table that dataset consists of only 89 light curves. This was all the publicly available data at the time of this study. An example light curve for each class is shown in Fig. 3.2.

It is important to note that all the light curves have different lengths. The length of the light curve is correlated with the type of object, due to observational biases. Some objects are observed over years (e.g. AGN) and others are observed over only a few hours (e.g. FS). Figure 3.1 shows the number of light curves for each class as a function of the total length of observation for that object. Because of this bias, we restrict our study to a timescale of eight hours, which is the longest observation time for which we have measurements for all classes (see Figure 3.1). Classification on this timescale will also allow relatively prompt follow-up triggers. The technique is applicable on even shorter timescales, even if there are very few flux measurements, although classification accuracy will likely decrease.

[4] www.meerlicht.uct.ac.za
[5] https://public.nrao.edu/telescopes/green-bank-interferometer

Tab. 3.1. Breakdown of radio transient data into the relevant types and data sources. GBI refers to data collected from the Green Bank Interferometer. From Lit. refers to data collected from the literature.

Type	From Lit.	GBI	Total
AGN	17	13	30
Algol	1	2	3
FS	5	0	5
GRB	4	0	4
KN	1	0	1
Mag	1	0	1
Nova	8	0	8
RSCVn	0	2	2
SN	13	0	13
TDE	2	0	2
XRB	11	9	20
Totals	**63**	**26**	**89**

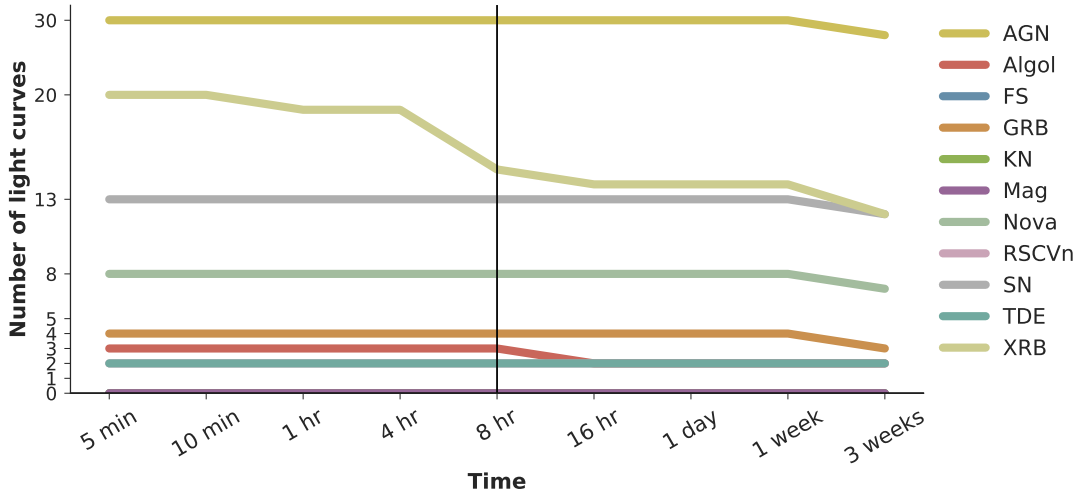


Fig. 3.1. The number of light curves for each class as a function of the length of observation for that source (in other words, there are y objects with light curves that were observed for at least time x). It can be seen that 8 hrs is the shortest observation period for which all the classes have a non-zero number of light curves.

3.2.2 Optical data

Simultaneous optical observations do not exist for all the classes in our dataset. To show how the addition of multiwavelength data could improve classification, an optical flux measurement had to be simulated. Optical data for four classes, namely: AGN, XRB, SNe and GRB, were collected from [109]. The data consisted of single flux measurements in both optical and radio wavelengths for each source. The dataset had total of $11,882$ measurements of which $11,782$ were AGN[6]. Figure 3.3 shows the

[6]While some objects appear in both the Stewart et al. sample and Pietka et al. sample that we use in this work, it should be noted that the vast majority of objects in the Stewart et al. sample consist of only a single flux measurement and thus are not appropriate for this work.

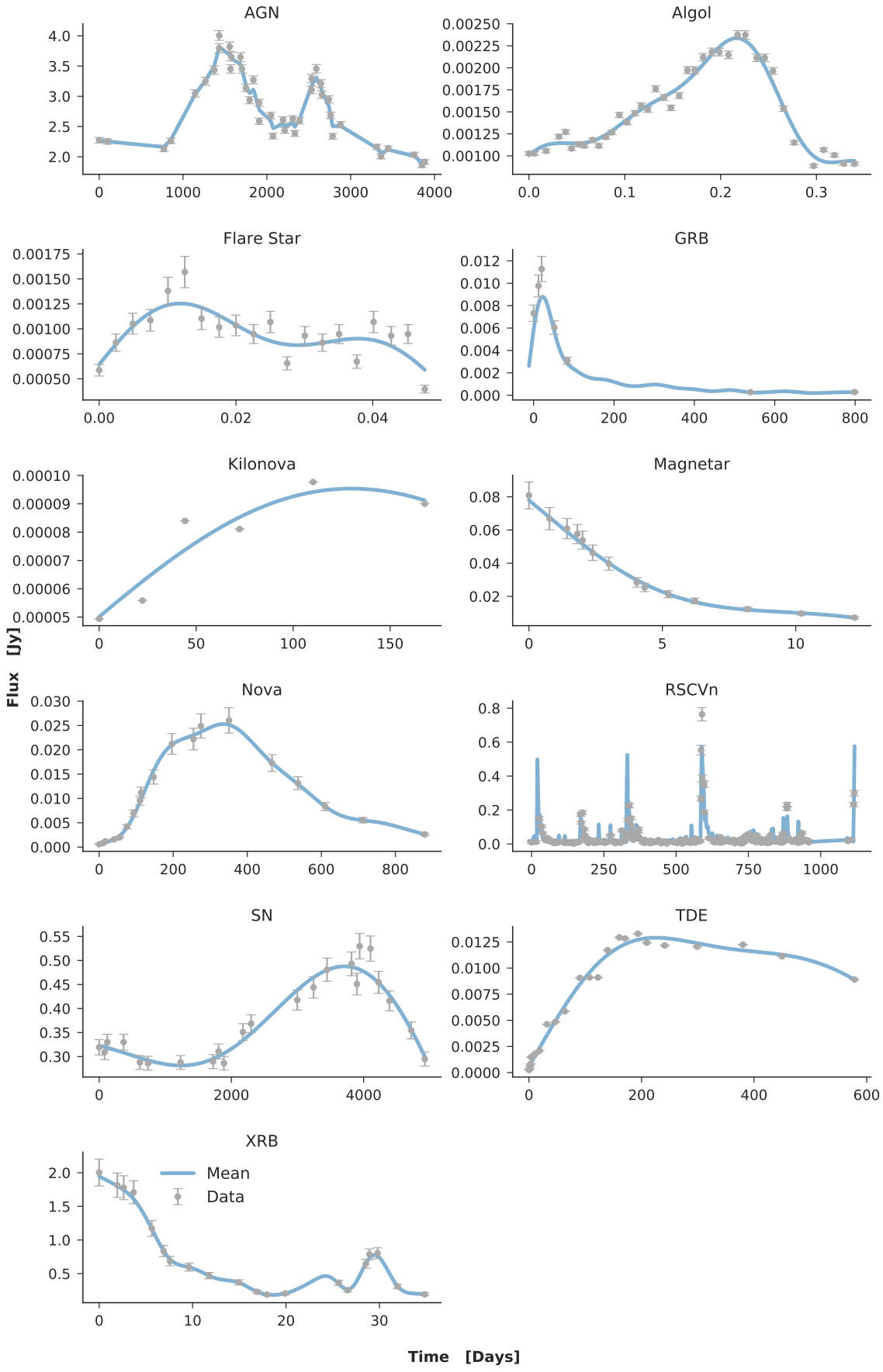


Fig. 3.2. An example light curve for each of the the radio transient types plotted as flux (in Jy) as a function of time (in days). It is clear that the data are taken on extremely different timescales. The original data points are shown in grey and the mean of the Gaussian Process is shown in blue.

optical-radio flux distributions for each of the four classes. A two dimensional Gaussian was used to model the distribution of all the classes except for AGN which can be clearly seen to be highly non-Gaussian. The Gaussian fits are shown as contours. These Gaussian distributions were used to sample new optical fluxes for the three classes, new optical fluxes were sampled directly from the distribution for AGNs as there are enough data points to do so.

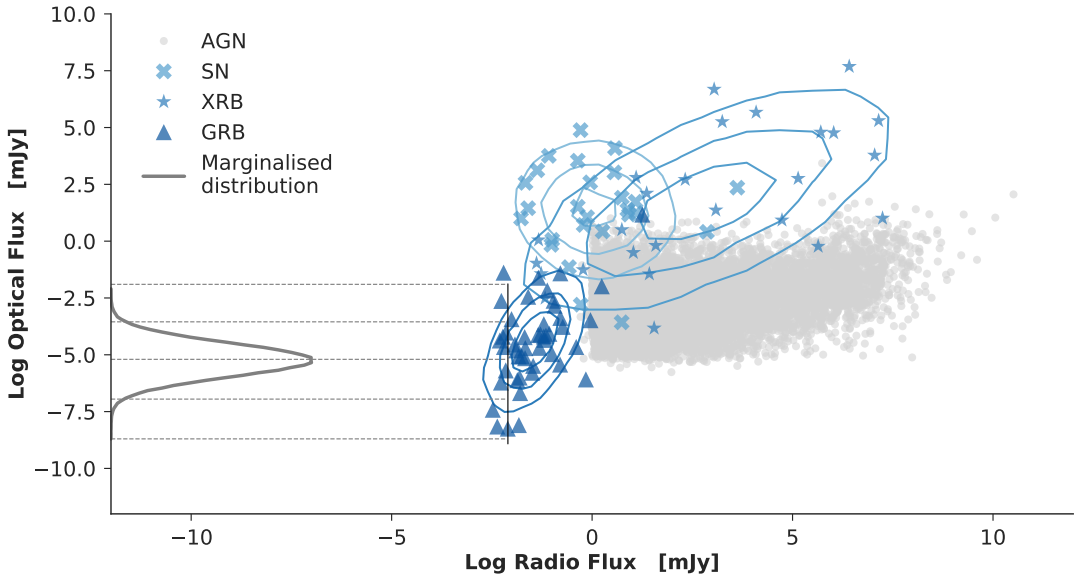


Fig. 3.3. Optical - radio flux distributions for the classes of AGN,XRB,GRB and SNe. Two dimensional Gaussian fits to the distributions of three of the classes are shown with contours. These fit the data for the three classes reasonably well, however it can be seen that the distribution for AGNs is highly non-Gaussian. An example peak GRB radio flux measurement is shown as a black vertical line. The GRB optical-radio flux distribution marginalised over this peak flux is shown in dark grey. Optical fluxes were sampled from these marginals.

The process for simulating simultaneous optical and radio observations are as follows, taking the class of GRB as an example. First, a GRB radio light curve was simulated as described earlier, the peak flux of this light curve was found. The radio-optical flux distributions, given the peak radio flux, were marginalised over as shown in black in Fig. 3.3. An optical flux was then drawn from this marginal distribution. Finally, this optical flux was added as an extra feature in the machine learning process. For the class of AGN, a peak radio flux was found as before, points in the radio-optical flux distribution were then binned, centered on the peak radio flux with a bin width of 0.1 mJy. These binned points, now marginalised on the peak radio flux follow a Gaussian distribution similar to that of the other classes. An optical flux was then drawn from this marginal distribution and added an extra feature in the machine learning process.

3.3 Augmentation

As mentioned in Section 3.1.1 when a dataset is too small for machine learning, a technique called data augmentation is often used to create new data from the existing data. For example, in deep convolutional networks used for image classification it is critical to augment the training data, usually by flipping, rotating and zooming examples in the training data. In the case of time series data such simple techniques are not available and we must do something more sophisticated. Fortunately, because we have error bars on our training data we can resample the light curves and interpolate between points. One could use splines but we choose to use Gaussian processes (introduced in section 2.3.4) for this data augmentation since it gives significantly more freedom to learn the characteristic temporal correlations for each class.

Each class of light curve has different general characteristics. This necessitated a combination of a few different kernel (or covariance) functions to be used for regression, which are defined in Section 2.3.4. After testing many different combinations and given the general characteristics of the different classes, it was found that one of three combinations of these kernels fit the data best. These combinations were defined as follows

$$
\begin{aligned}
K_1 &= \omega_1 k_{\text{rad}}(r) \\
K_2 &= \omega_1 k_{\text{rad}}(r) + \omega_2 k_{\text{rad}}(r) k_{\text{sine}}(r) \\
K_3 &= \omega_1 k_{\text{rad}}(r) + \omega_2 k_{\text{exp}}(r)
\end{aligned}
$$

where ω_i are weights on each kernel.

GP regression was performed using each of these combinations. During regression, the negative log likelihood, given a set of hyperparameters and the data, was calculated. The negative log likelihood (defined in Section 2.3.4) was minimised using the Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BRGS) optimisation algorithm from the `SciPy`[7] package, to get the best set of hyperparameters for that kernel combination. The combination of kernels with the lowest negative log likelihood was used to construct the GP from which to sample. An example GP for a SN light curve is shown in Figure 3.4.

3.4 Feature extraction

As mentioned in Section 2.4, feature extraction is used to reduce the dimensionality of the data by summarising the information contained in the original data. Two different feature extraction techniques were used. The first was a simple feature

[7] www.scipy.org

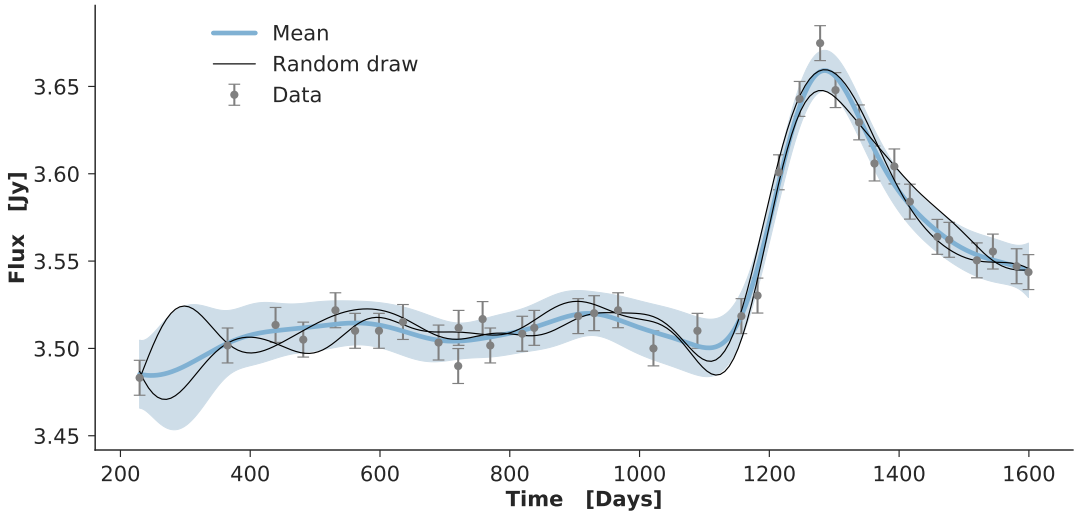


Fig. 3.4. An example light curve for SN plotted as flux (in Jy) as a function of time (in days) with the GP overlaid. The original data points are shown in grey and the mean of the Gaussian Process is shown in blue. For the SN light curve one standard deviation away from the mean is shown as a blue envelope. The black lines shown in the SN light curve are two random draws from the GP. It can be seen that these lines are different from, but still consistent with the original data.

set based on the differences in flux of the objects on different timescales, this is described in Section 3.4.1. The second method was to use the coefficients from wavelet decomposition (see Section 2.4.1), this is described in Section 3.4.2.

3.4.1 Flux Features

For the first attempt at classifying the radio transients a very simple feature set was used. The feature vector, ΔF, was defined to be

$$\Delta F = F_t - F_{t_0} \ , \tag{3.2}$$

where ΔF is the difference in flux between a reference flux, F_{t_0}, chosen at random from the light curve and the flux, F_t, at time $t > t_0$.

In anticipation of a fast imager on MeerKAT we chose the minimum difference between successive flux measurements to be two seconds (Woudt, private communication). The maximum time difference was chosen to be three months to account for the objects that vary on very long time scales such as AGN. The complete set of t were chosen to be:

$$\begin{aligned} t = [&2 \text{ sec}, 1 \text{ min}, 5 \text{ min}, 10 \text{ min}, 30 \text{ min}, 1 \text{ hr}, 2 \text{ hr}, 4 \text{ hr}, 6 \text{ hr}, \\ &8 \text{ hr}, 12 \text{ hr}, 18 \text{ hr}, 1 \text{ day}, 2 \text{ day}, 4 \text{ day}, 1 \text{ day}, 2 \text{ week}, \\ &3 \text{ week}, 1 \text{ month}, 1.5 \text{ month}, 2 \text{ month}, 3 \text{ month}] \end{aligned} \tag{3.3}$$

The feature extraction method was then as follows: First GP regression was performed on the original data set. A reference time, t_0, was then drawn at random to be somewhere within the curve. The GP was then sampled at this t_0 to obtain an F_{t_0}. The GP was then sampled at points $(t_0 + t)$ where t is defined above. F_{t_0} was then subtracted from these fluxes to obtain the feature vector ΔF. This process was repeated multiple times for each light curve, each time generating a random t_0. This was done to simulate the fact that the transient may be detected at any point on the light curve.

3.4.2 Wavelet Features

It will be seen in Section 4.1 that we found that the flux features were inadequate to capture the variation between classes and so we instead follow the feature extraction procedure used by Lochner et al [58] as outlined in Section 2.4.1.

The complete data augmentation and feature extraction method was as follows: First GP regression was performed on the original data set. From this, many example light curves can be generated, each statistically consistent with the original data, which allows us to generate a realistic synthetic dataset of any size. To simulate the fact that the transient may be detected at any point on the light curve, a reference time, t_0, was then drawn at random to be somewhere within the curve. We then sampled 100 flux values between t_0 and $t_0 + 8$ hrs from the GP. This approximates an 8 hour radio observation where an image is produced every five minutes. We then used `PyWavelets` to perform a two-level wavelet decomposition on these 100 points, which returns 400 coefficients. PCA (see Section 2.2.2) was performed on these, keeping 20 coefficients which corresponds to retaining 99% of the variability in the dataset.

3.5 Incorporating other data sources

Some classes can have similar light curves but are generally found in different parts of the sky. For example we are more likely to find novae in the Galactic plane, while SNe are more likely to be extragalactic.

In order to break the degeneracies between these classes we added a feature that characterises where the object is in the sky. Telescopes will always have access to the position in the sky in which it is pointing, hence the drawback of adding features that may sometimes be missing, as outlined in Section 3.1.4, will not be an issue.

The RA and Dec coordinates were obtained for all the objects in our dataset using a combination of Simbad[8] and NED[9]. These coordinates were then converted to Galactic coordinates. We defined a feature that specified whether or not the object was in the

[8] www.simbad.u-strasbg.fr
[9] https://ned.ipac.caltech.edu

Galactic plane. Any object with a Galactic declination of above 10° or below -10° was considered to be out of the Galactic plane as shown in Figure 3.5.

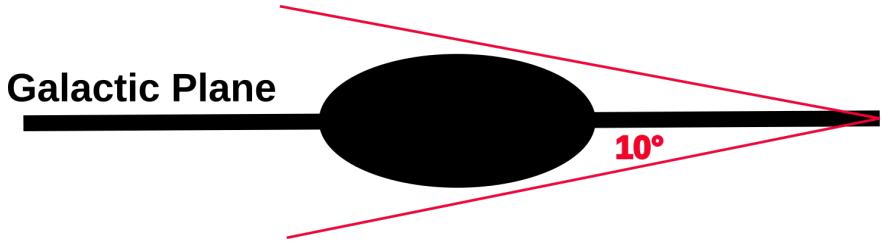


Fig. 3.5. An illustration of the Galactic coordinate system, showing a galactic latitude of -10° and $+10^\circ$.

Results

4

> *If we knew what it was we were doing, it would not be called research, would it?*
>
> — **Albert Einstein**
> (Theoretical physicist)

In this chapter we present the results of applying our framework to transient data. The results of the flux feature extraction are presented in Section 4.1. The results shown in the rest of the chapter are using the more sophisticated wavelet extraction technique. The effects of training set are discussed in Section 4.2. The features are visualised in Section 4.4. In Section 4.3, the effect of splitting the dataset into variables and transients are explored. Finally in Section 4.5 the results of adding multiwavelength data are presented.

4.1 Flux Features

Once the flux features were extracted as described in Section 3.4.1, different subsets of the total feature set was used to train different classifiers. The subsets used were created by truncating the features at different timescales, i.e a classifier was trained on all features up to a maximum of 5min, then another was trained on all features up to a maximum of 10min and so on for all t in Equation 3.3. The accuracy of each of these classifiers is shown in Figure 4.1.

It can be seen from Figure 4.1 that the classifier performs well on long timescales. This however, is not useful as interesting objects can be classified on long timescales by other methods such as spectroscopic follow up. What we would like to investigate is how well the classifier performs on short time scales for each of this classes. From Figure 3.1 it can be seen that the number of FS light curves in the dataset goes to zero, hence 8 hrs is the longest timescale at which we have a complete set of classes. Thus the time vector from Section 3.4.1 is changed to:

$$t = [2 \text{ sec}, 1 \text{ min}, 5 \text{ min}, 10 \text{ min}, 30 \text{ min}, 1 \text{ hr}, 2 \text{ hr}, 4 \text{ hr}, 6 \text{ hr}, 8 \text{ hr}]$$

A classifier was trained on these 8 hr features, the results of which are shown in Figure 4.2. It can be seen that on this short timescale the classifier performs poorly, achieving an overall accuracy of just 54.7%. As described in Section 3.5, one method to break degeneracies between classes is to add extra information we have on each of the

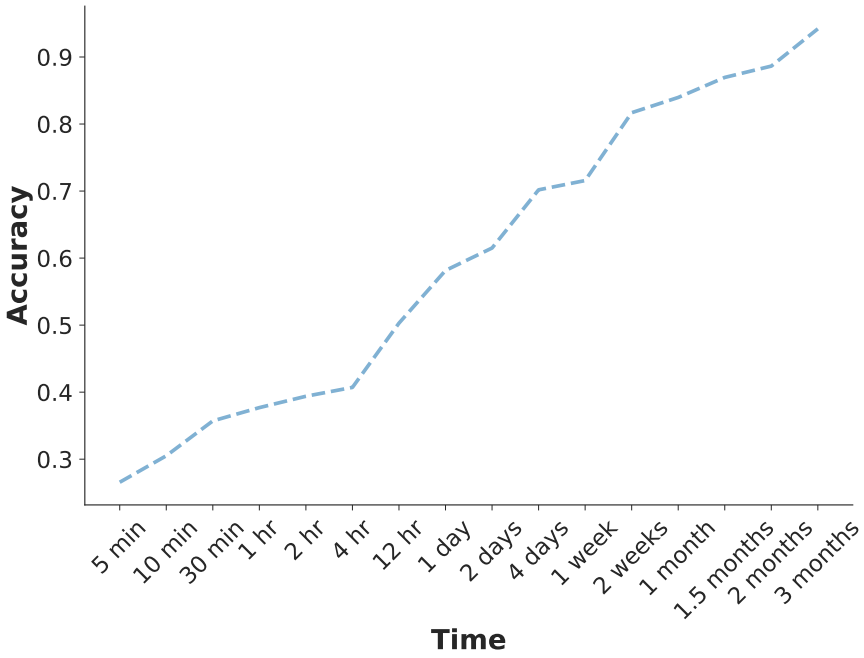
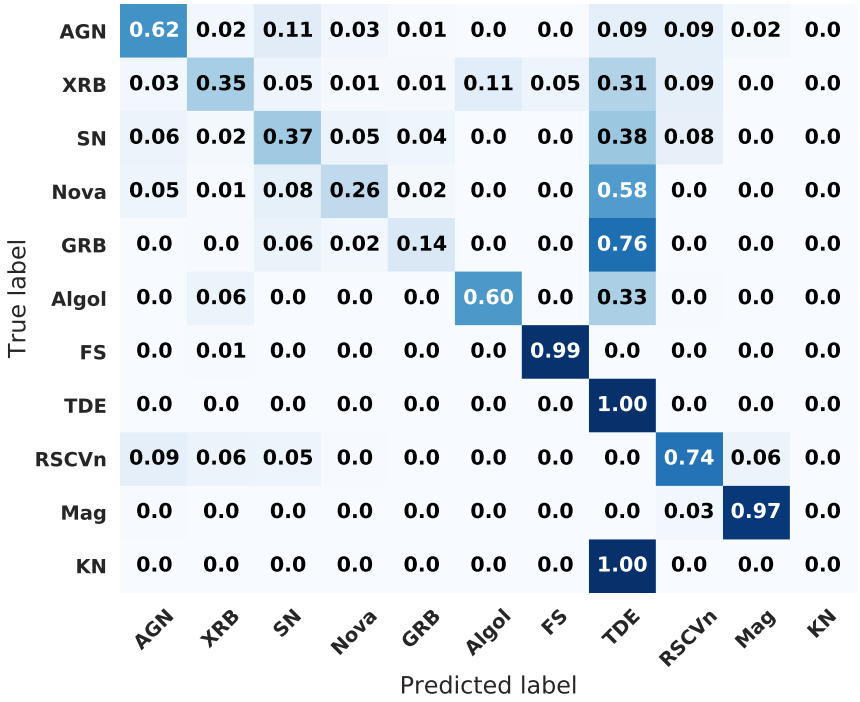


Fig. 4.1. The accuracy of different random forest classifiers each trained on increasing timescales of the total feature set. The y-axis shows the accuracy and the x-axis shows the time observed. It can be seen that as we increase the time observed the accuracy of the classifier also increases, however the main objective is early time classification hence a more sophisticated feature set was considered.

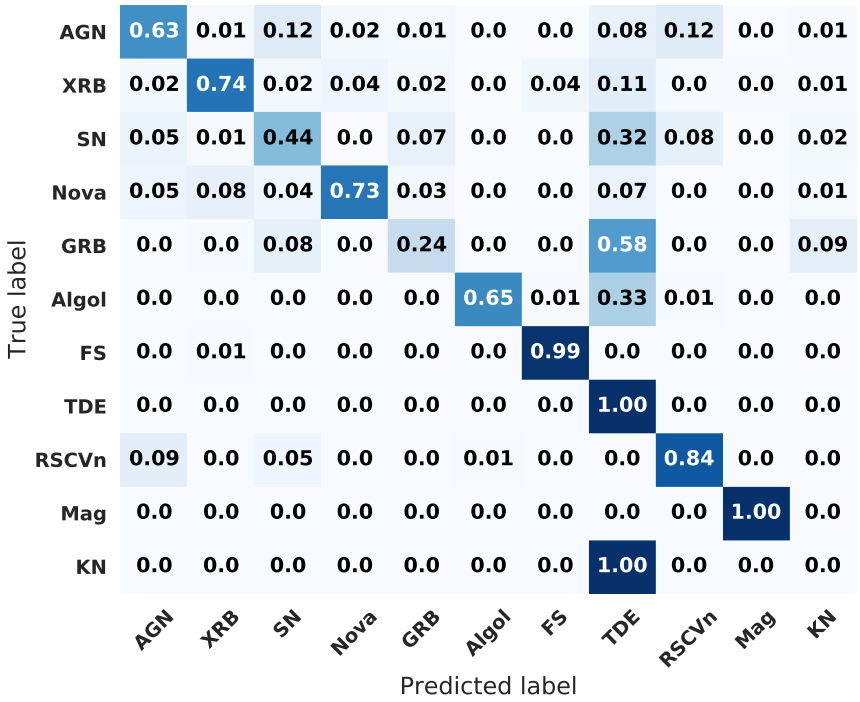
objects. Figure 4.2 shows the results of training a classifier on the 8 hr features together with the additional feature describing the object's position in 4he sky as outlined in Section 3.5.

We can see that the addition of the contextual feature does improve the performance of the classifier with the overall accuracy increasing by $\approx 11\%$. The improvement can be clearly seen in the difference confusion matrix shown in Figure 4.3.

Recall from Section 2.1.4 that a classifier is performing well if classifications appear along the diagonals, thus Figure 4.3 has two colour schemes. The first corresponds to the diagonals. If the values along the diagonal increase they will show in green; if they decrease they will show in red. The second corresponds to the off-diagonals. If the values along the off-diagonals increase they will show in red; if they decrease they will show in green. From this we can see that the class of novae receives the biggest improvement in performance with the addition of the contextual feature with a 54% increase in accuracy. This is to be expected as almost all novae observed would be in the galaxy.



(a) Confusion matrix without contextual feature



(b) Confusion matrix with contextual feature

Fig. 4.2. The normalised confusion matrix for flux features extracted from 8hrs of data. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class). The top panel shows the confusion matrix without contextual information with an overall accuracy achieved of 54.7%. The bottom panel shows the confusion matrix with contextual information with an overall accuracy achieved of 65.5%

Fig. 4.3. The normalised difference confusion matrix for flux features extracted from 8hrs of data with the added contextual feature. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class). Two colour schemes are shown. The first corresponds to the diagonals. If the values along the diagonal increase they will show in green; if they decrease they will show in red. The second corresponds to the off-diagonals. If the values along the off-diagonals increase they will show in red; if they decrease they will show in green. From this it can be seen that the new feature greatly benefits the class of nova which increases in accuracy by 54%.

With the extra feature the performance of the classifier is still poor, achieving an overall accuracy of 65%. This shows that these flux features are unable to describe the variation present between the classes in the dataset. This led us to consider the feature extraction method outlined in Section 3.4.2.

4.2 Wavelet Features

The wavelet features were extracted as described in Section 3.4.2. The radio transient data currently available are unfortunately too small hence it will not encompass the inter-class variability we would see with a larger dataset. Thus we performed three studies to investigate the effect of the training set on the performance of the classifier. In the first case the classifier was trained on a training set that is representative of the test set. The results of this is presented in Section 4.2.1. In the second case the classifier was trained on a training set that is non-representative of the test set. The results of this is presented in Section 4.2.2. In the final case the importance of each of the original light curves is investigated. The results of this is presented in Section 4.2.3.

4.2.1 Fully representative training data

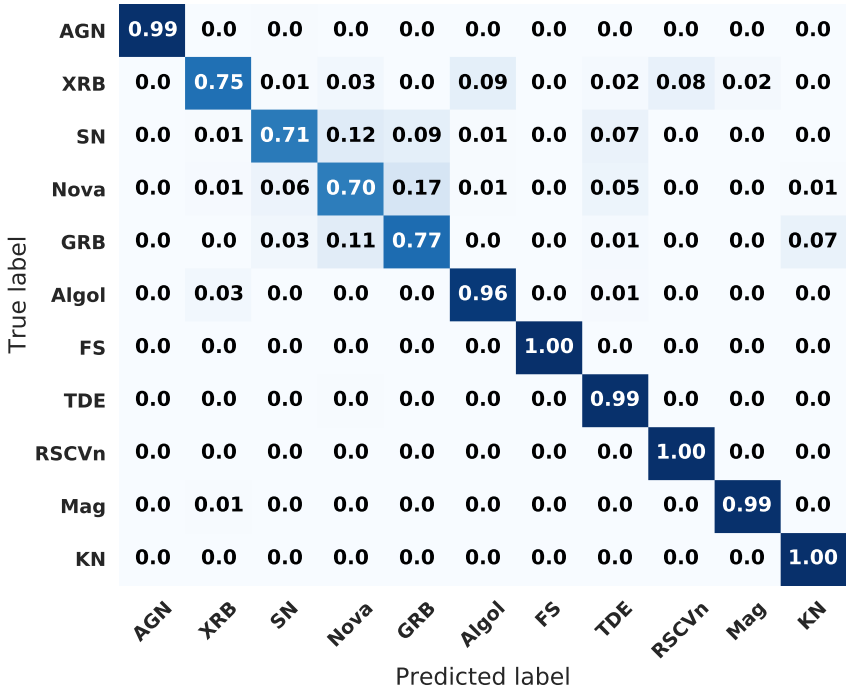
A representative dataset was simulated by training the classifier on samples from all the light curves in our dataset. The process was as follows. GP regression was performed on the original 87 light curves. From these, $10,000$ "simulated" light curves were generated for each of the eleven classes, to create a balanced training set. Wavelet feature extraction was then performed on each of the simulated light curves resulting in 400 wavelet coefficients. After performing PCA on these coefficients, the 20 most important components were selected and used as features.

In order to show the effect of adding contextual information, two separate classifiers were trained: the first was trained without any contextual feature and the other was trained with contextual information as described in Section 3.5. The results are shown in Figure 4.4. It can be seen that without any contextual information the classifier confuses the classes of XRB, SNe, Novae and GRBs. However after the contextual feature is added, the accuracies of these classes improve greatly with the overall accuracy increasing by 6%. The quantitative improvements can be easily seen from the difference confusion matrix shown in Figure 4.5. We can see that XRB accuracy increases by 20%, SN increases by 10%, Novae increases by 22% and GRBs by 13%. We expect the confusion matrix with contextual information to characterise the performance of the classifier in practice; thus a contextual feature was used in the rest of this work. These confusion matrices show that once adequate training data is collected the classifier will perform very well.

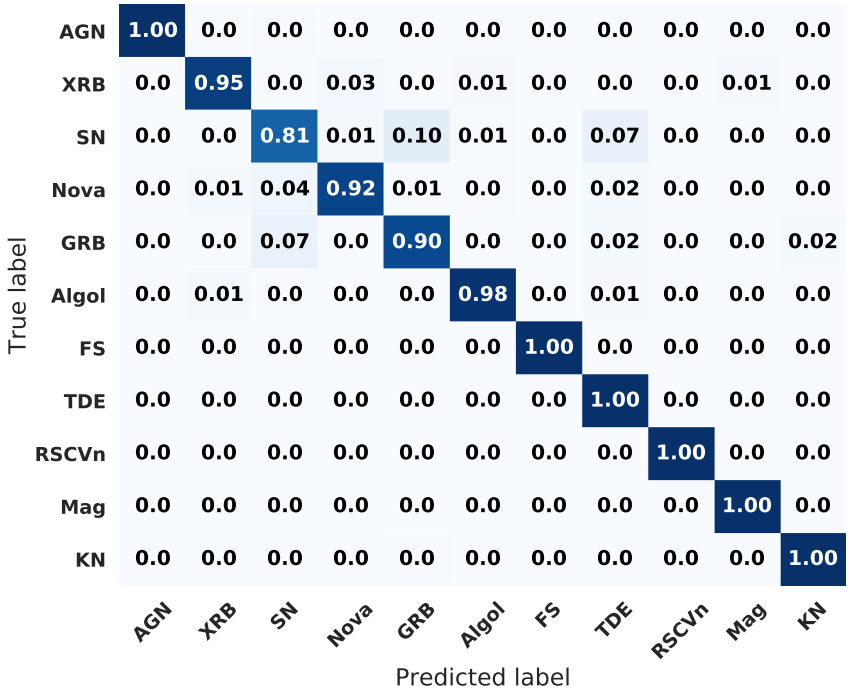
4.2.2 Non-representative training set

Because our original data set is limited and does not cover the full anticipated variablity of all the classes, we expect the results outlined in Section 4.2.1 to represent an idealised case. In practice, we anticipate that any test set would consist of some objects not present in our training set. To construct a more realistic test of the classifier performance the data currently available, instead of training the classifier on samples from all the light curves, we trained it on a subset of light curves. As can be seen from Figure 3.1, only five classes have greater than four light curves. In order to ensure the training subset contained all classes, we only removed light curves from these five classes. We still included the classes with only one object in both the training and test sets, because these objects can cause confusion between classes which would be artificially removed if they were excluded.

We removed 25% of the original light curves in these five classes at random, GP regression was performed on the remaining 75% of original light curves. From these, 7500 simulated light curves were generated for each of the eleven classes. Wavelet feature extraction was then performed on each of the simulated light curves followed by PCA as before. This was used as the training set. 2500 simulated light curves were



(a) The normalised confusion matrix without contextual feature



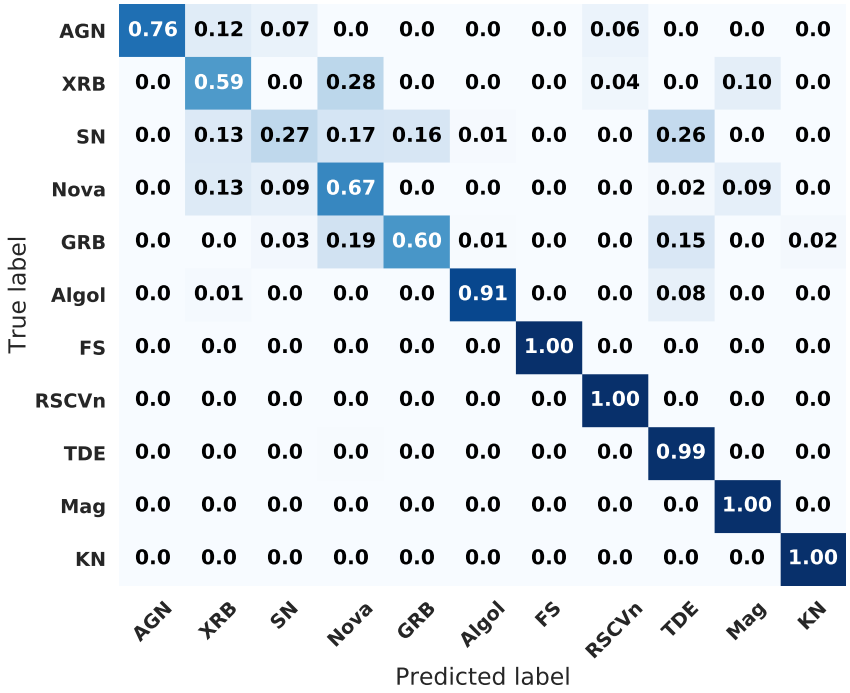
(b) The normalised confusion matrix with contextual feature

Fig. 4.4. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class). The top panel shows the confusion matrix without contextual information with an overall accuracy achieved of 89.8%. The bottom panel shows the confusion matrix with contextual information with an overall accuracy achieved of 95.8%. It can be seen that without contextual information the classes of Nova, GRB and Algols are confused.

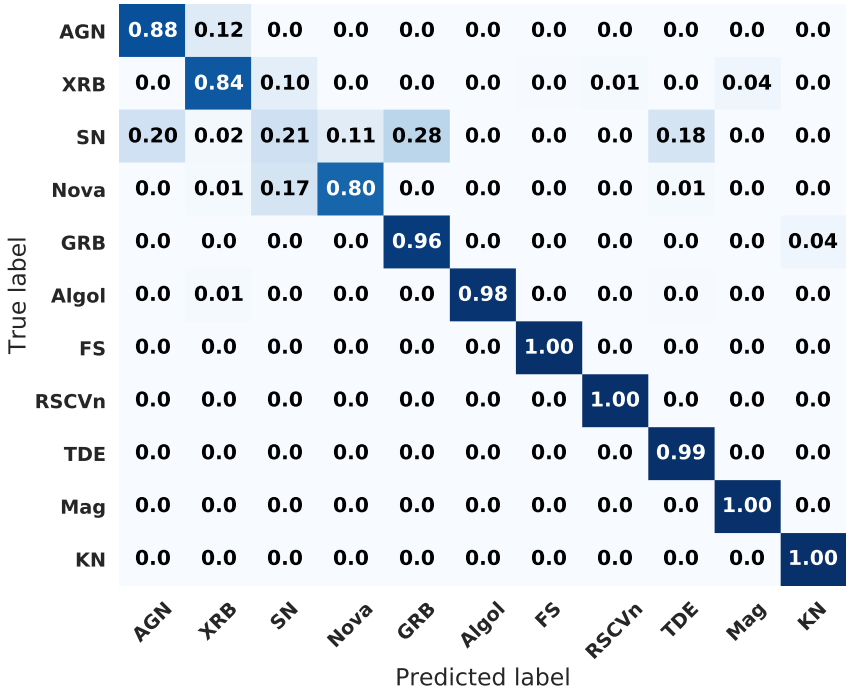
Fig. 4.5. The normalised difference confusion matrix for wavelet features extracted from 8hrs of data with the added contextual feature. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class). Two colour schemes are shown. The first corresponds to the diagonals. If the values along the diagonal increase they will show in green; if they decrease they will show in red. The second corresponds to the off-diagonals. If the values along the off-diagonals increase they will show in red; if they decrease they will show in green. XRB increases by 20%, SN increases by 10%, Novae increases by 22% and GRBs by 13%.

then drawn from GPs of the 25% of the light curves that were removed, then wavelet extraction and PCA was performed as before. This was used as the testing set. We repeated this 20 times, each time removing 25% of the light curves at random.

The confusion matrices with the minimum, maximum and average overall accuracies over the 20 runs is shown in Figure 4.6 and Figure 4.7 respectively. It can be seen that the performance of our classifier has decreased by $\approx$ 22%, which is to be expected given the non-representativeness of the training set.



(a) Confusion matrix showing the results of the classifier with the minimum overall accuracy of the 20 runs.



(b) Confusion matrix showing the results of the classifier with the maximum overall accuracy of the 20 runs.

Fig. 4.6. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class). The confusion matrix shown in the top panel has an overall accuracy of 69.7%. The confusion matrix shown in the bottom panel has an overall accuracy of 83.7%.

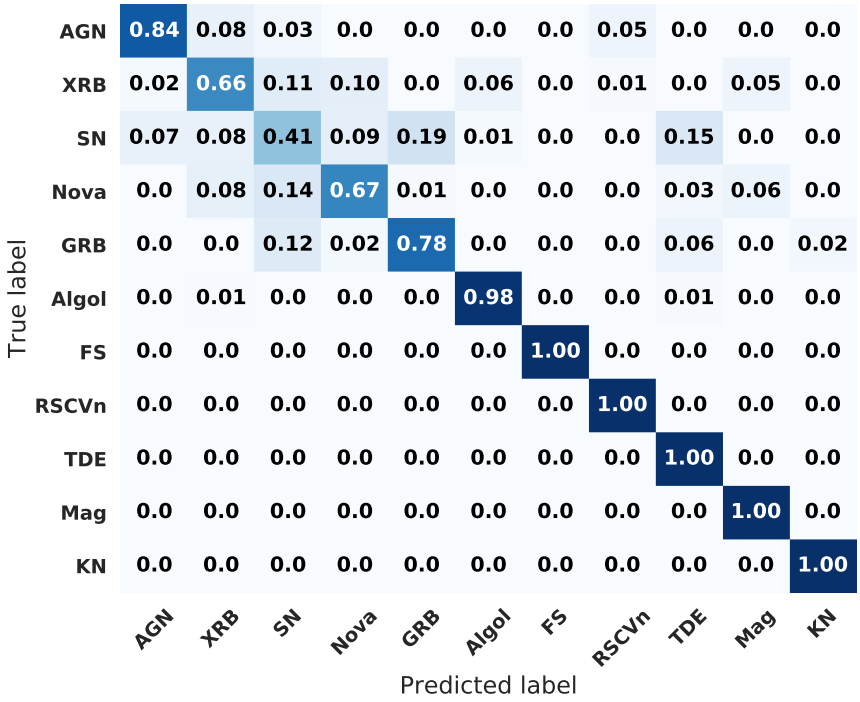


Fig. 4.7. Confusion matrix showing the results of the classifier averaged over 20 runs. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class). It can be seen that the classes that the algorithm performs poorly on are SNe and Novae. The overall accuracy achieved is 77.8%.

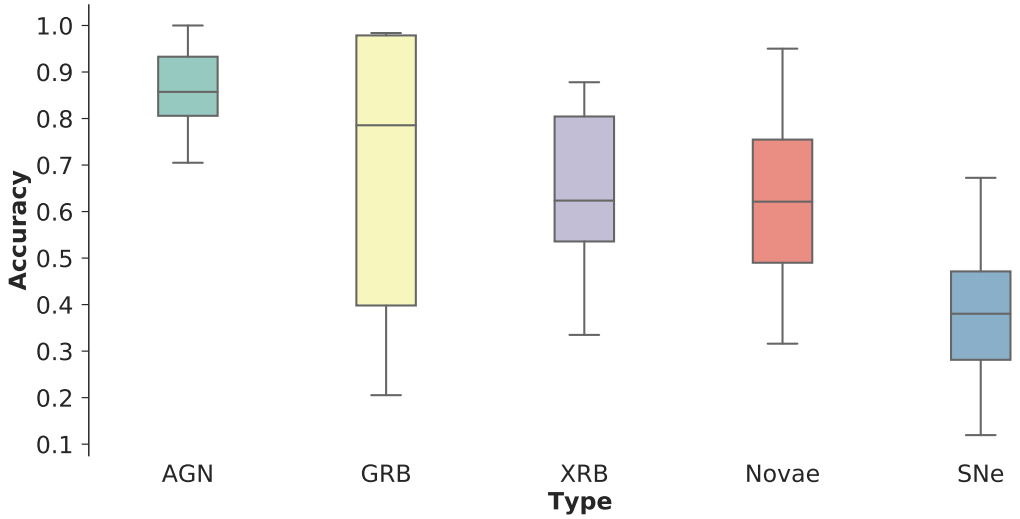


Fig. 4.8. Summary of results from non-representative training set are shown in these box and whisker plots. The bar in each box shows the accuracy for each class averaged over the 20 runs (on each run a classifier was trained on 75% of the total dataset and tested on the remaining 25%, this process was repeated 20 times, each time randomising the training and testing sets). The coloured boxes show the interquartile range. The grey bars (whiskers) show the minimum and maximum accuracies in the 20 runs. It can be seen that the classifier performs poorly for SNe as it has the lowest average accuracy of $\approx 40\%$.

The results for the five main classes, AGN, XRBs, SNe, Novae and GRB are summarised in the box and whisker diagram Figure 4.8. From this it can seen that well-represented classes like AGN still perform well as their accuracies have a high average and low variability. Classes such as SN however, with a highly diverse training set with many dissimilar objects, are poorly classified.

4.2.3 Single curve testing

To further demonstrate the effect of a non-representative training set, we tested the classifier's performance on light curves not represented in a training set. We focused on the five classes (AGN, SNe, XRBs, Novae and GRBs) that contain four or more light curves.

We trained the classifier on samples from all of the original light curves, except for one. GP regression was thus performed on 86 of the original light curves and from these simulated light curves were drawn. Wavelet feature extraction was then performed on each of the simulated light curves followed by PCA as before. We then test the classifier on simulated light curves from the one that was left out of the training set in the training process. This process was then repeated, each time omitting a different light curve in the training set, until every light curve in the dataset had been removed during the training of a classifier at least once. The results for these five main classes are summarised in the violin plots shown in Figure 4.9.

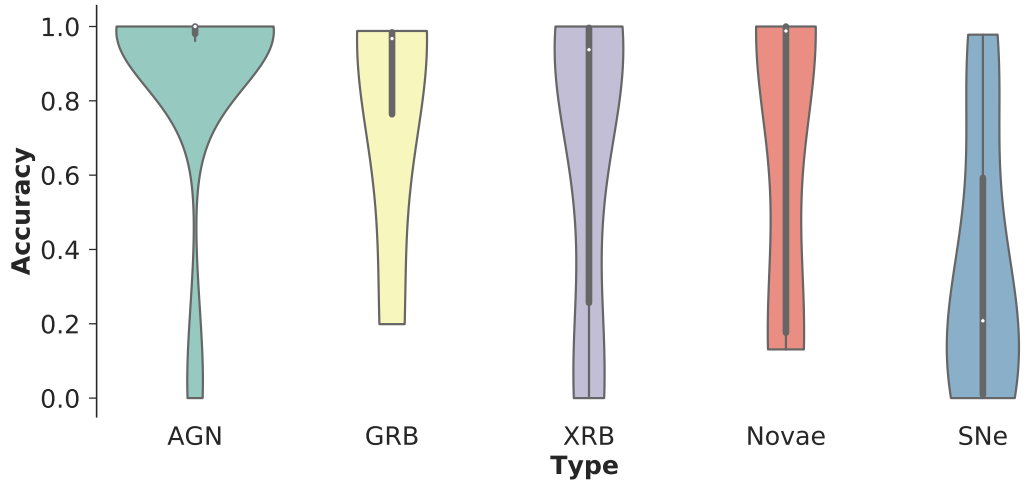


Fig. 4.9. The single curve testing results for the five main classes are summarised in these violin plots. Each violin plot represents a different class as shown. The coloured outline shows the smoothed kernel count distribution of the accuracies. The thick central black line represents the interquartile range. The thin central black line shows the 95% interval. From this is can be seen that the classifier performs well with AGN as most of the accuracies are above 80%. It can also be seen that the classifier performs poorly with SNe as most of the accuracies are below 50%.

Tables showing the accuracy for each of the dropped out curves is shown in Table 4.1. The coloured outline shows the smoothed kernel count distribution of the accuracies.

The thick central black line represents the interquartile range. The thin central black line shows the 95% interval. From this is can be seen that the classifier does very well with AGN as most of the accuracies are above 80%. It can also be seen that the classifier struggles with SNe as most of the accuracies are below 50%.

Tab. 4.1. The results of the individual dropout test. Samples from one light curve was removed during training. These samples were then used to test the classifier. The accuracy for the samples of each of the light curves for the five main classes are shown. From left to right the classes are: AGN, XRB, SN, Nova and GRB. From these tables it can be seen that the classifier does really well for AGNs, XRBs and GRBs but the performance decreases for SNe and Novae. This shows that the intrinsic variabilities of these two classes are not captured by our dataset.

AGN		XRB		SN		Nova		GRB	
Name	**Acc**	**Name**	**Acc**	**Name**	**Acc**	**Name**	**Acc**	**Name**	**Acc**
NGC7213	0.0	B1259-63	0.0	SN1993J	0.0	V1974Cyg	0.131	GRB030329	0.199
0850-121	0.0	MAXIJ1836	0.007	SN2008iz	0.0	RSOph	0.152	GRB970508	0.952
0954+658	0.206	ScoX-1	0.014	SN1980K	0.0	V1500Cyg	0.184	GRB060418	0.983
0224+671	0.784	CygX-2	0.148	SN1988z	0.009	V407Cyg	0.976	GRB110709B	0.988
2005+403	0.824	XTEJ1550	0.581	SN2003L	0.147	Sco2012	1.0		
NRAO530	0.931	aqlX1	0.707	SN1998bw	0.195	TPyx	1.0		
2223-052	0.951	GROJ1655	0.927	SN2003bg	0.208	SSCyg	1.0		
1413+135	0.961	1909+048	0.948	SN2011dh	0.307	V1723Aql	1.0		
0528+134p	0.98	SS433	0.954	SN2004dk	0.593				
1622-297	1.0	0236+610	0.981	SN1994I	0.593				
CTA102	1.0	1915+105	0.986	SN2008ax	0.861				
0336-019	1.0	GX17+2	1.0	SN2004cc.	0.931				
3C345	1.0	CygX-1	1.0	SN2004gq	0.978				
B0605-085	1.0	CICam	1.0						
3C454.3	1.0	CirX-1	1.0						
1328+254	1.0								
3C120	1.0								
3C273	1.0								
0458-020	1.0								
3C279	1.0								
1237+049	1.0								
0851+202	1.0								
2200+420	1.0								
0528+134	1.0								
PKS2004-447	1.0								
1803+784	1.0								
0954+65	1.0								
1749+096	1.0								
NGC4278.	1.0								
AO0235+164	1.0								

Table 4.1 shows the results of the individual dropout test. Samples from one light curve was removed during training. These samples were then used to test the classifier. The accuracy for the samples of each of the light curves for the four classes are shown. From left to right the classes are: AGN, XRB, SN, Nova and GRB. From these tables it can be seen that the classifier is very accurate for AGNs, XRBs and GRBs but misclassifies SNe and Novae. This shows that the intrinsic variabilities of these two classes are not captured by our dataset.

4.3 Variables and Transients

One method which we explored to separate the confused classes was to split the dataset. From Chapter 1 we know that these radio objects can be split into two types: Variables – objects that have multiple outbursts, i.e. constantly vary (e.g. FS) and Transients – objects that vary once, i.e have one outburst that then decays (e.g. SNe). The dataset was split into these two groups then a classifier was trained on the binary classification of transient vs variable, results are shown in Figure 4.10.

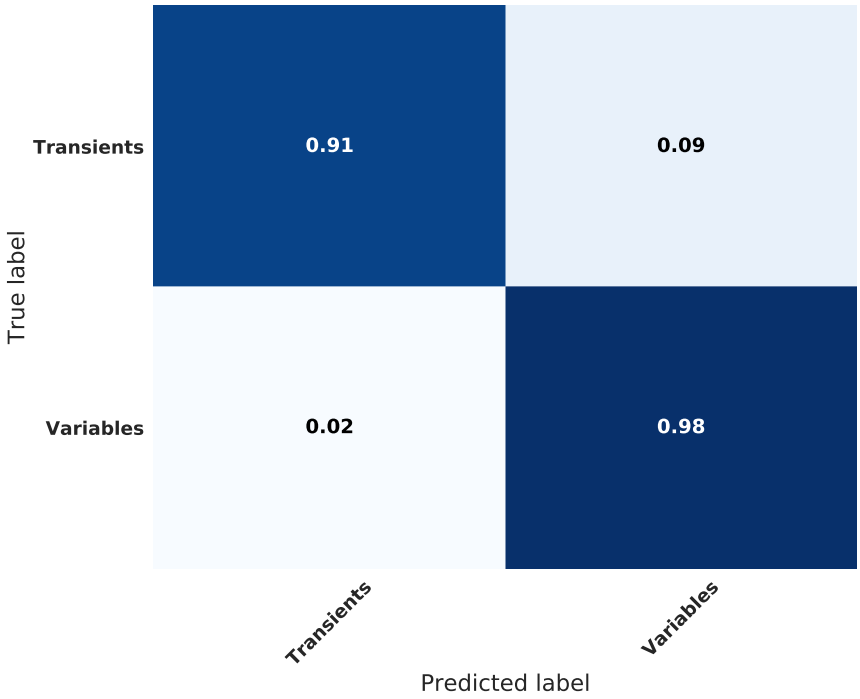
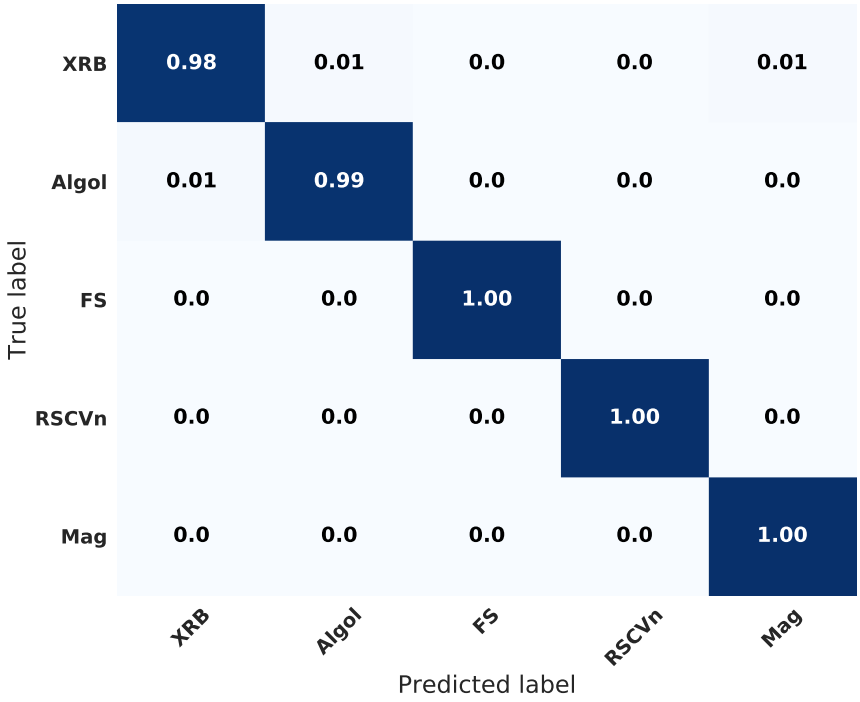


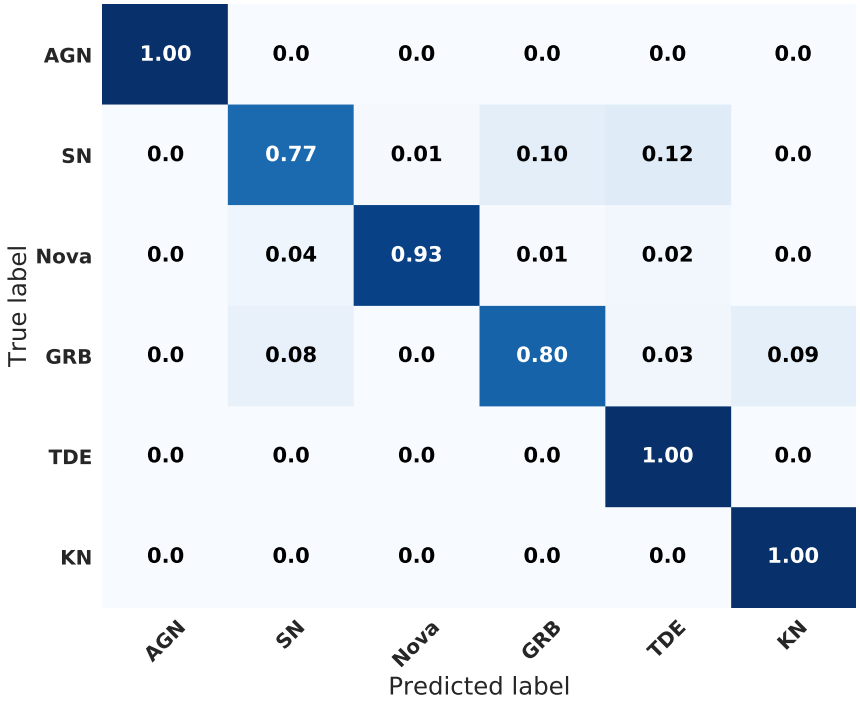
Fig. 4.10. The normalised confusion matrix for wavelet features extracted from 8hrs of data for a binary Transient/Variable classification. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class).

Classifiers were then trained on the individual groups of variables and transients, results are shown in Figure 4.11.

It can be seen from Figure 4.10 that the classifier can distinguish between variables and transients extremely well, with an overall accuracy of 98.5%. By making this split we increase the accuracy of the XRB and SN classifications slightly when compared to Figure 4.4. The class of GRB, however, has little improvement. As with Figure 4.4, SNe are still being confused with TDEs. It can be seen that this split is unnecessary as it achieves little to no improvement in the performance of the classifier.



(a) Confusion matrix for variable classes

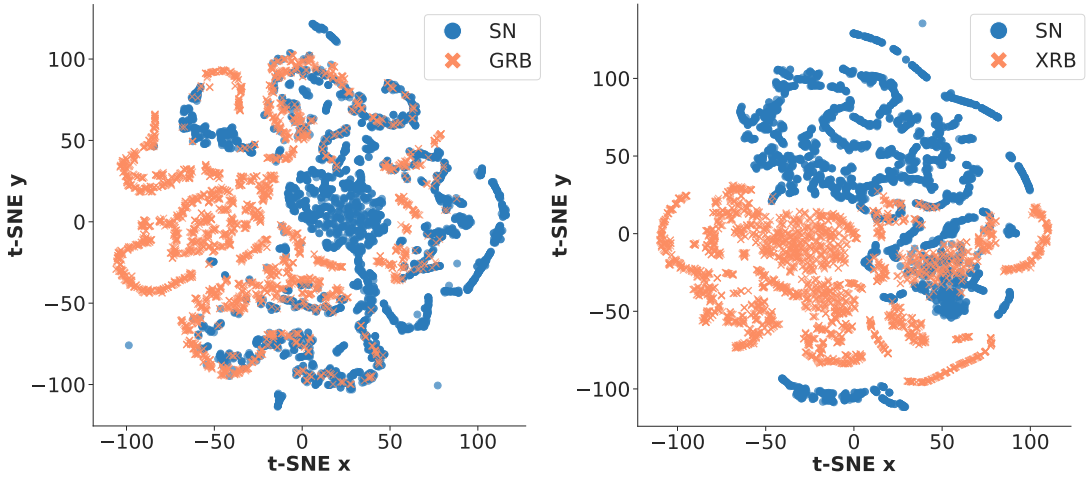


(b) Confusion matrix for transient classes

Fig. 4.11. The normalised confusion matrix for wavelet features extracted from 8hrs of data. The y-axis shows the true label of the object(true class). The x-axis shows the label which the algorithm predicts for the object(predicted class). The top panel shows the confusion matrix for the variable classes, which achieved an overall accuracy of 99%. The bottom panel shows the confusion matrix for the transient classes, which achieved an overall accuracy of 91.6%.

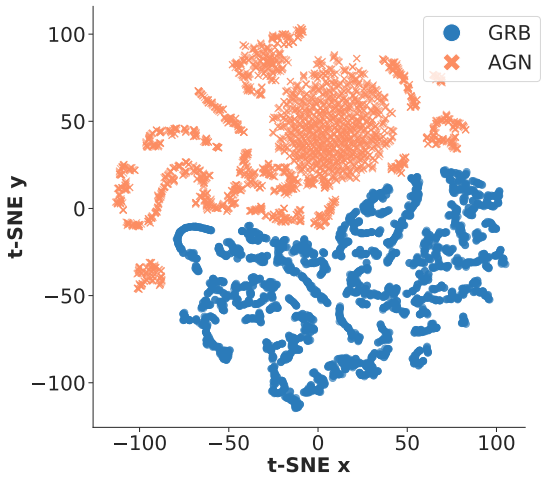
4.4 Visualising features

From Section 4.2 and Section 4.3 it is evident that the classifier performs well for some classes and poorly for others. One method that can be used to probe why this may be the case is to visualise the features that are being used. Recall from Section 2.2.3 that t-SNE plots capture the distribution of features, mapped down into two dimensions, hence can be used for visualising how well-separated classes are in feature space.

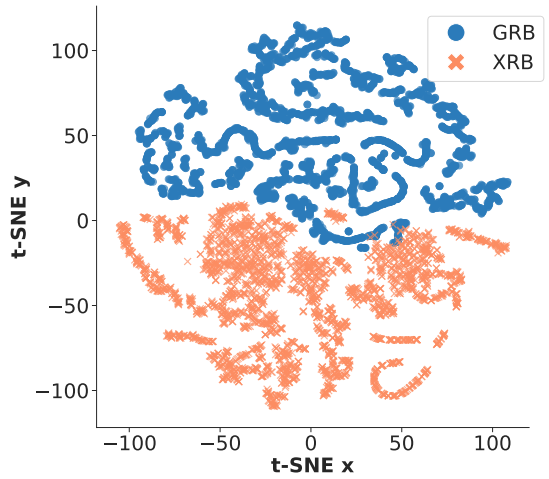


(a) SN - GRB

(b) SN - XRB



(c) GRB - AGN



(d) GRB - XRB

Fig. 4.12. The t-SNE plots for four separate cases. These plots show the high dimensional feature space embedding in two dimensions, hence the x and y axes are arbitrary. The top panel shows two cases for which the classifier performs poorly. The bottom panel shows two cases for which the classifier performs well. From this it can be seen that classes for which the classifier confuses (such as GRBs and XRBs) have features that overlap in feature space. Classes for which the classifier performs well on have features that are well separated in feature space, thus making it easier for the classifier to discriminate between them.

The first attempt at visualising the features was to plot all classes on one t-SNE plot. The number of classes made this very hard to interpret. Instead we chose subsets of classes for which the classier performed well on and a subset for which it performs poorly, for illustration purposes. Plots of these are shown in Figure 4.12. We expect that classes for which the classifier performs well to be easily discriminated between. This can be seen in the bottom panel of Figure 4.12 as you can easily separate the two classes in each of the plots. The opposite is true for top panel as the features of the two classes overlap in the each of the plots. This shows that it is much harder to discriminate between these classes hence the classifier performs poorly. The top panel of Figure 4.12 show that the features we are currently using is insufficient to separate these classes in feature space.

4.5 Adding optical data

The method explored in Section 4.3 achieved little to no improvement in the overall performance of the classifier. In Section 4.4 it was shown that the current feature set can not adequately separate some of the classes. Thus we investigated if adding multiwavelength data could break the degeneracies of some of the classes. Optical data was simulated as described in Section 3.2.2. These optical features were added to the the training set. A classifier was trained 20 different times using a non-representative training set of the four classes, AGN, XRB, SN and GRB following the method outlined in Section 4.2.2.

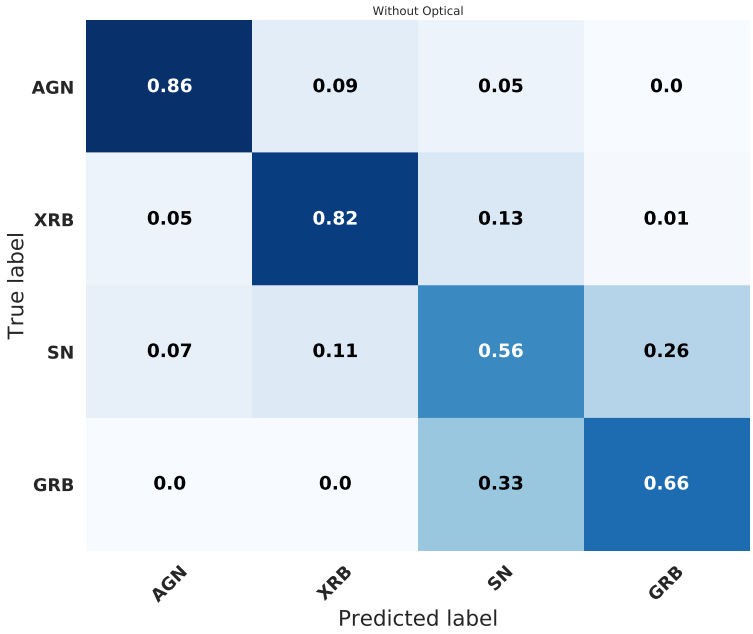


Fig. 4.13. Confusion matrix showing the results of the classifier, trained on non-representative data using wavelet features together with a contextual feature, averaged over 20 runs. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class). It can be seen that the classes that the performs poorly on are SNe and GRB.

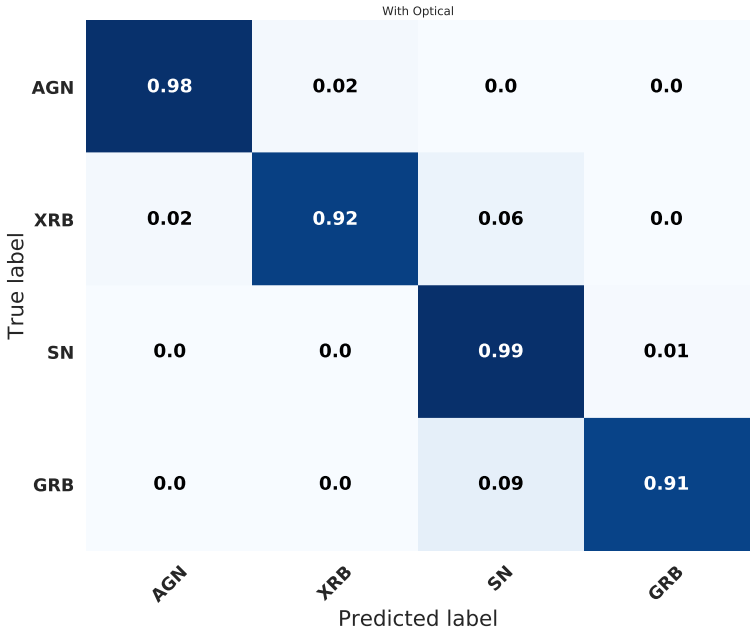


Fig. 4.14. Confusion matrix showing the results of the classifier averaged over 20 runs, with the optical feature. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class). With the added optical feature the performance of the classifier improves as the SNe and GRBs are much less confused.

First the classifier was trained 20 times using the wavelet features with a contextual feature. The confusion matrix averaged over the 20 runs is shown in Figure 4.13. This

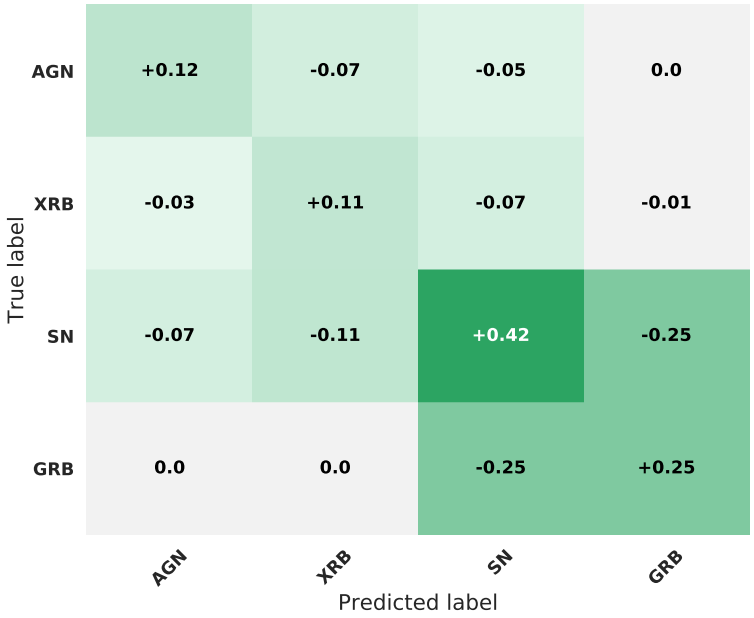


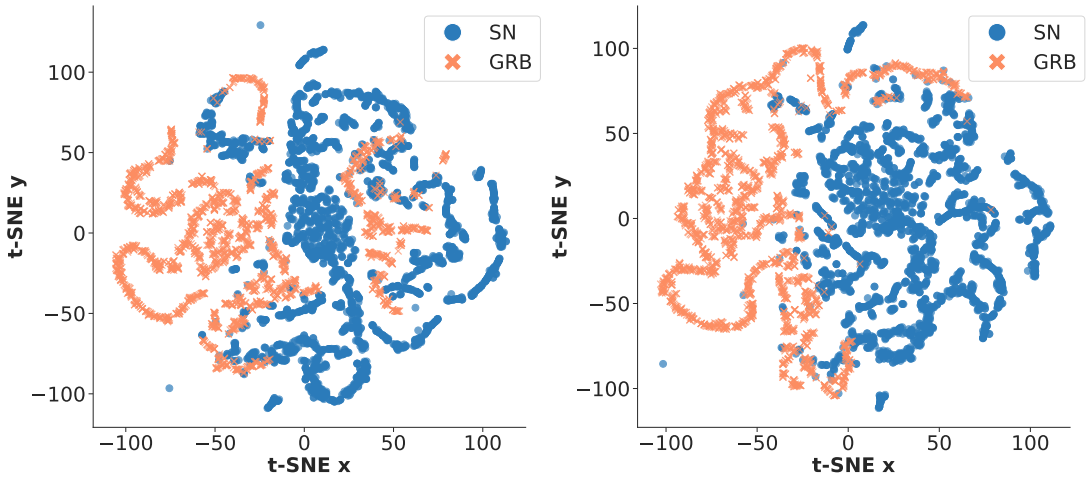
Fig. 4.15. The normalised difference confusion matrix for wavelet features extracted from 8hrs of data with the added optical feature. The y-axis shows the true label of the object (true class). The x-axis shows the label which the algorithm predicts for the object (predicted class). Two colour schemes are shown. The first corresponds to the diagonals. If the values along the diagonal increase they will show in green; if they decrease they will show in red. The second corresponds to the off-diagonals. If the values along the off-diagonals increase they will show in red; if they decrease they will show in green. We can see that SNe and GRBs have a substantial increase in accuracy, with increases of 42% and 25% respectively.

was then repeated using the additional optical feature, the result of which is shown in Figure 4.14. It can be seen that the addition of this optical feature significantly improves the performance of the classifier for classes such as SNe and GRBs. This improvement can be clearly seen from the difference confusion matrix shown in Figure 4.15.

Figure 4.15 once again uses the two different colour schemes for the diagonals and off-diagonals. We can see that the addition of the optical feature significantly improves the performance of the classifier. All the off-diagonals are reduced and all diagonals have increased, with GRBs and SNe receiving the biggest increase in accuracy of 25% and 42% respectively.

As a final sanity check we can once again visualise the features using a t-SNE plot. From Figure 4.13 we can see that the classes of SN and GRB are confused with each other. The features for these classes with and without an optical feature are shown in Figure 4.16.

Figure 4.16 matches what we see from Figure 4.15. It shows that with the addition of the optical feature, it is easy to discriminate between the classes of SN and GRB. This improvement in performance demonstrates the value of simultaneous optical observations, such as from MeerLICHT in classifying radio transients.



(a) Features of SN and GRB with no optical feature

(b) Features of SN and GRB with an optical feature

Fig. 4.16. The t-SNE plots for two classes of radio transients for which the classifier performs poorly. These plots show the high dimensional feature space embedding in two dimensions, hence the x and y axes are arbitrary. The blue points represent feature vectors of SN objects, orange crosses represent the feature vectors of GRB objects. The left panel shows the feature set with no optical feature. The right panel shows the feature set with an optical feature. From this it can be clearly seen that the addition of the optical feature helps to separate the two previously confused classes.

Conclusions

5

> *The significance of our lives and our fragile planet is then determined only by our own wisdom and courage. We are the custodians of life's meaning. We long for a Parent to care for us, to forgive us our errors, to save us from our childish mistakes. But knowledge is preferable to ignorance. Better by far to embrace the hard truth than a reassuring fable. If we crave some cosmic purpose, then let us find ourselves a worthy goal.*
>
> — **Carl Sagan**
> (Astronomer)

We have presented a general formalism for multiwavelength transient classification with machine learning that will be useful in the era of MeerKAT/MeerLICHT as well as eventually for SKA, LSST and LIGO. We outlined the different approaches that can be taken to include data from multiple telescopes, as well as additional information such as source location.

Extending Lochner et al [58], we developed a machine learning pipeline for classifying transients, and illustrated how to include contextual information (such as whether or not the source is in the Galactic plane) and simultaneous observations from an optical telescope.

We trained and tested our pipeline using existing radio transient light curves gathered from literature. Because these light curves were few in number, the first step in the pipeline was to artificially augment the dataset using a machine learning technique called Gaussian processes that learns the characteristic correlations and shapes of the different classes. This technique allow for the creation of new light curves that are statistically consistent with the existing radio transient light curves.

The second step in the pipeline is feature extraction. This summarises the information contained in the dataset such that machine learning algorithms can learn from it. Two feature extraction techniques were considered. First, features were extracted from the augmented dataset using a simple method based on the change in flux of the light curve over different time periods. The second technique used was a wavelet decomposition. Lochner et al [58] showed that these features perform well for supernovae and we expect them to generalise to all types of transients. The dimensionality of these feature

sets were reduced using principal component analysis. It was found that the wavelet features have much higher performance than the simpler flux difference features.

The final stage of the pipeline is training a machine learning algorithm on the features. We chose a random forest classifier as it has been shown to a robust algorithm that outperforms other algorithms in many cases [56]–[58]. As the available radio data is limited, the performance of the classifier was tested on different (augmented) training sets to see the effects this has on performance. The first case is where the training data are representative of the test data, which can be seen as the best case scenario. The classifier was trained on features without a contextual feature and achieved an average accuracy across all 11 classes of 89.8%. After the contextual feature was added the average accuracy increased to 95.8%. The improvement was particularly noticeable for novae, objects typically found within the galactic plane, which are otherwise confused with GRBs. This result illustrates the importance of adding additional pieces of information about the objects in question. As a result of this improvement, the contextual feature was used for the rest of this study.

In this case the performance of the algorithm was found to be excellent. However, it is important to note that the training set currently available is very small, thus it is highly likely that sources observed by new telescopes such as MeerKAT would not be similar to anything in the training set. In practise this problem will be dealt with by online learning: continuously re-training the classifier as new data comes in from MeerKAT and is labelled by follow-up from other telescopes.

When 25% of the objects from the training set were removed and the classifier was tested on those removed light curves to simulate a non-representative training set, an average accuracy of only 77.8% was achieved. From the investigation of the effect of dropping out single individual light curves from the pre-augmented training set, it was found that, while most of the light curves are still classified well, several light curves in the dataset are poorly classified. This is because they are dissimilar to anything in the training set. This effect is most pronounced for supernovae, which are generally easily confused with other classes such as GRBs or novae. This will improve as more training data is added.

Our dataset contained only 13 supernovae and the single light curve dropout tests illustrate the diversity of this population. With ThunderKAT anticipated to detect 20-50 supernovae every year [3], we can expect this training set to improve dramatically within a year or two of MeerKAT observations.

Finally, the effect of including optical data was illustrated. Unfortunately simultaneous optical and radio transient data are scarce, so optical fluxes were simulated for the radio light curves, drawing on existing distributions. Optical measurements for only four of our original 11 classes were found, namely AGN, XRB, SN and GRB. A classifier was trained using the more realistic “dropped out” training set using the feature set

with the contextual feature, achieving an average accuracy of 72.9%. Once the optical feature was added this average increased to 94.7% . This highlights the importance of incorporating multimessenger data in order to make full use of the data available. The accuracy increased by $\approx 21\%$ with only the addition of one other electromagnetic source (optical flux), implying data from other telescopes is useful for distinguishing between different classes of transients. Our framework is general enough to include information from other sources such as LIGO.

These results indicate that by including multimessenger information and making use of a sophisticated machine learning approach, we can expect accurate classification of radio transients with even a relatively small training set created with early MeerKAT and MeerLICHT data. This should be invaluable as the next generation SKA and LSST telescopes come online, yielding far too many transients each night to follow up. Instead machine learning classification will allow us to cherry pick the most interesting objects from the expected deluge.

Light curves

A

In this appendix we present the light curves of all the data used in this study. For light curves with a large number of data points, we randomly selected only 300 points to be plotted. As a result the GP fits (in blue) may appear to deviate significantly from the data. This is because there is data that we have not plotted for clarity

A.1 AGN

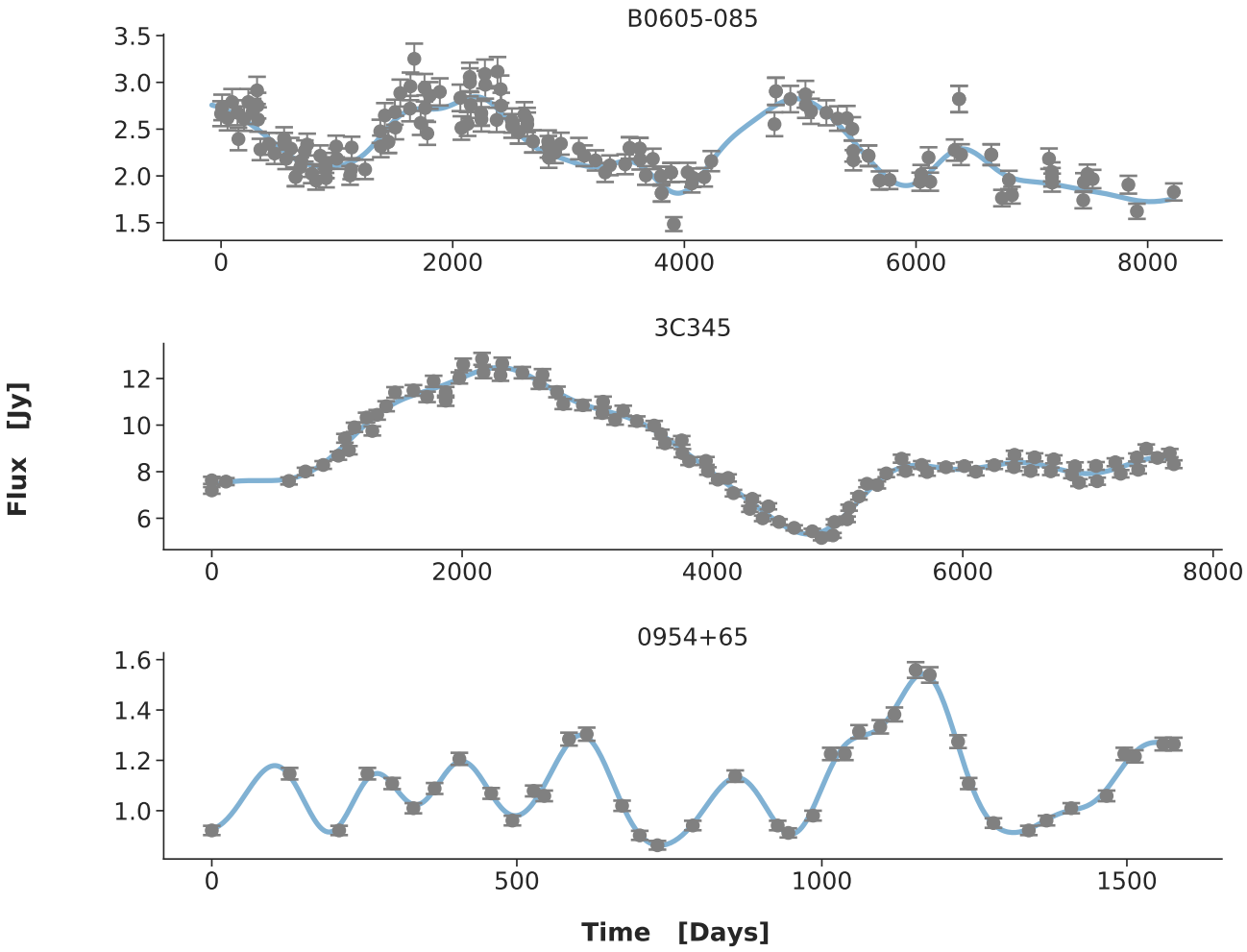


Fig. A.1. The light curves of AGN 0528+134p, 3C273 and 3C345. Data plotted is courtesy of [13].

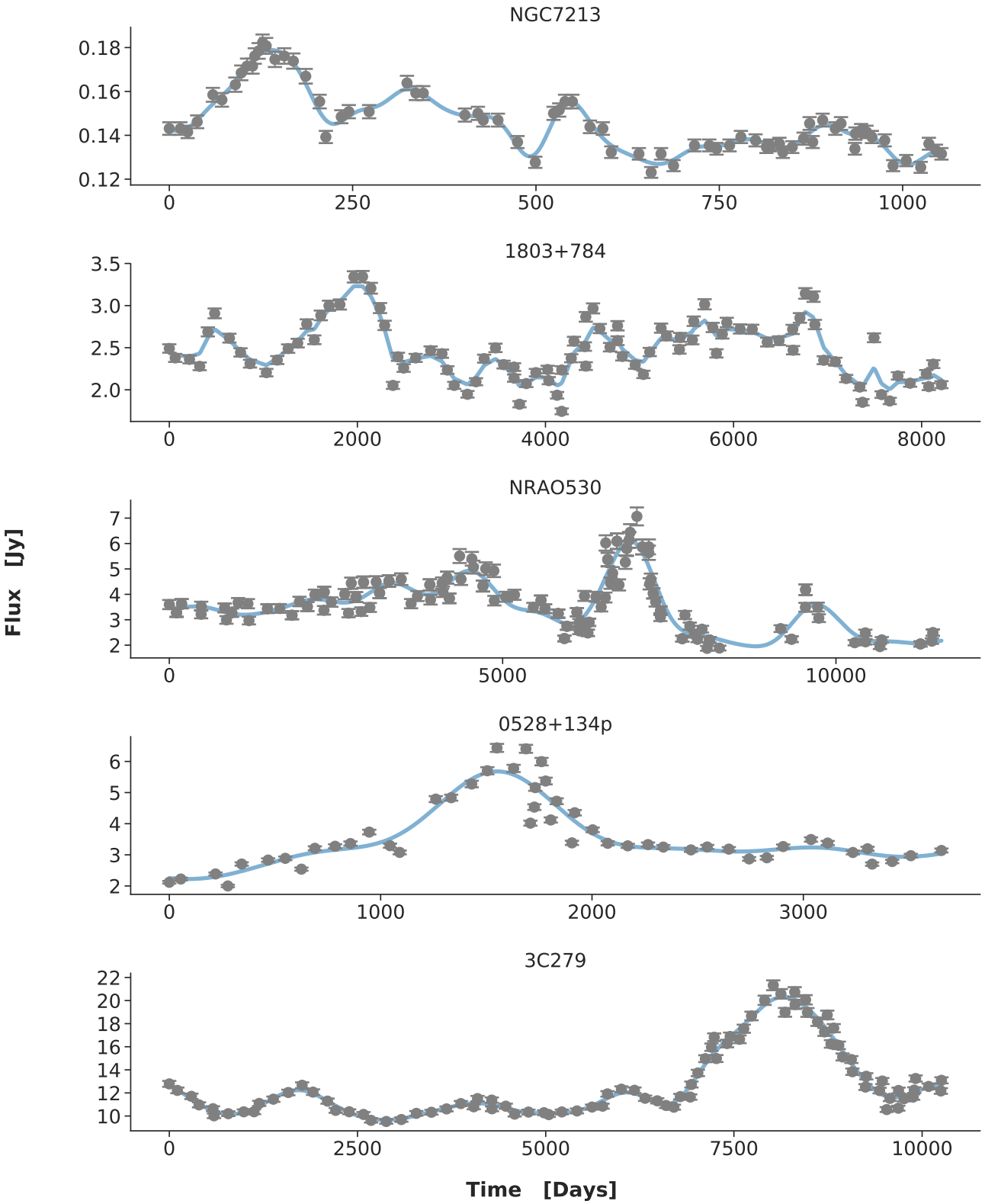


Fig. A.2. The light curves of AGN 0458-020, CTA102, 3C279, NGC4278 and 0954+65. Data plotted is courtesy of [13].

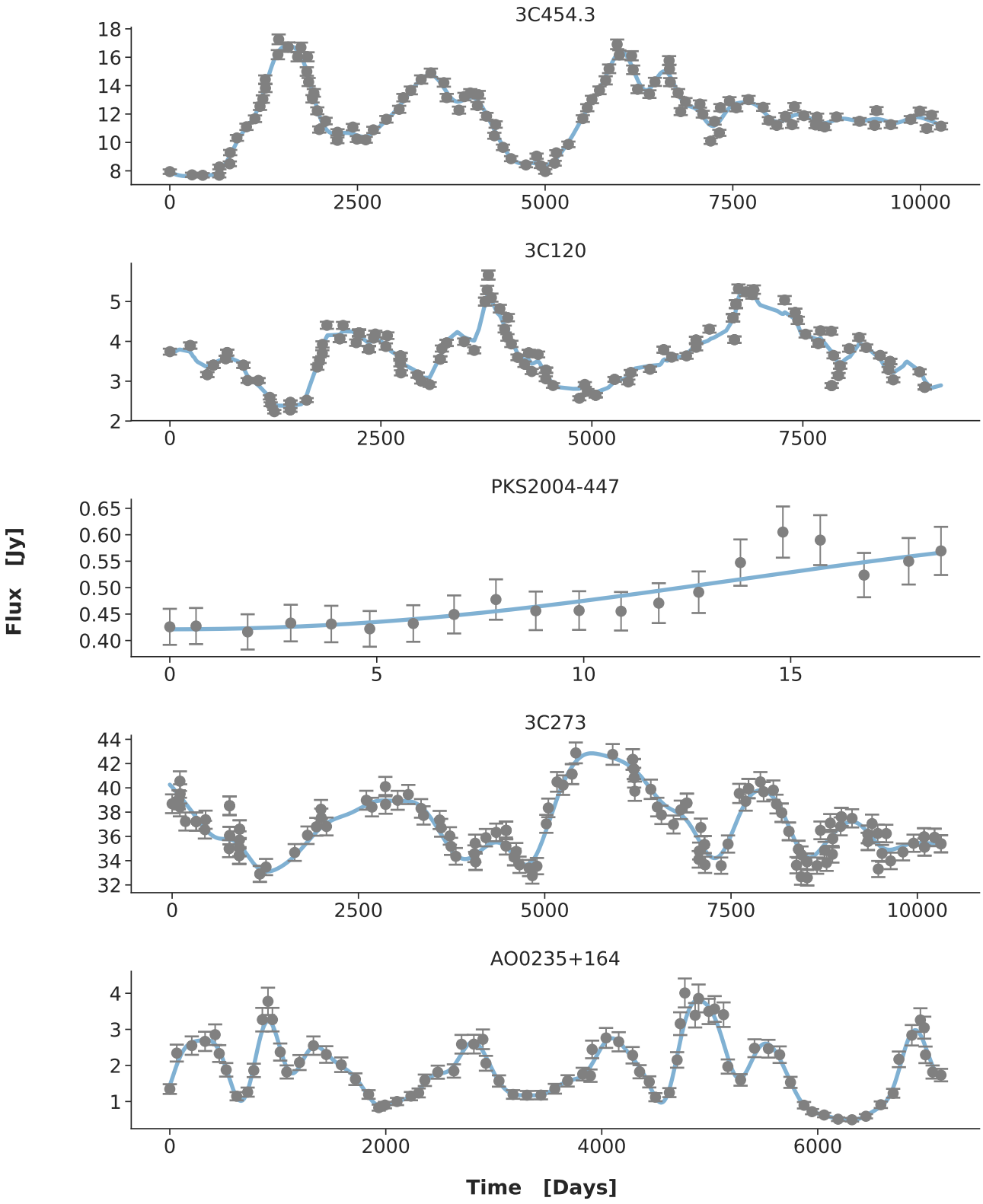


Fig. A.3. The light curves of AGN AO0235+164, 3C120, PKS2004-447, NRAO530 and 1803+784. Data plotted is courtesy of [13].

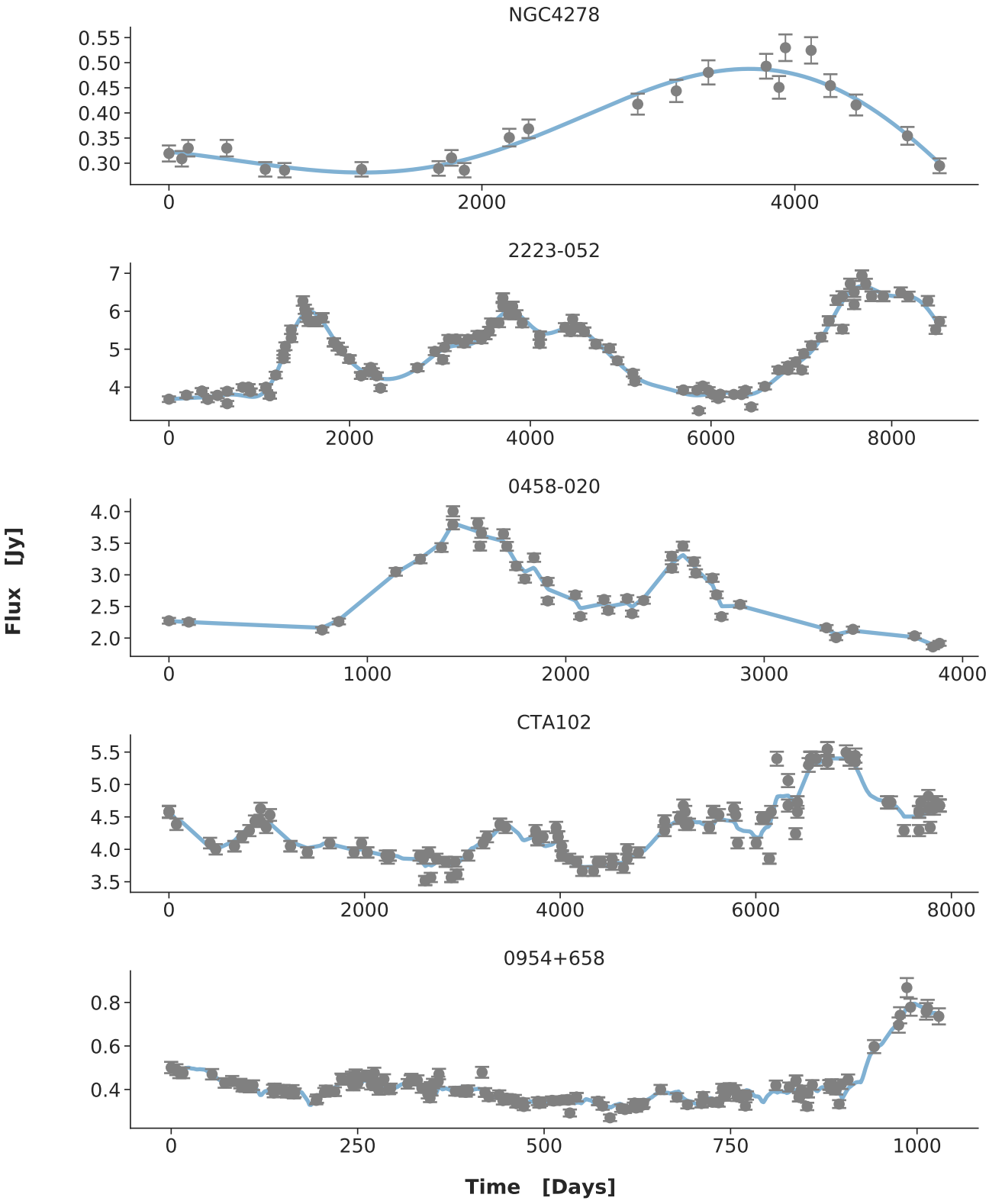


Fig. A.4. The light curves of AGN B0605-085, 2223-052, NGC7213, 3C454.3 and 2005+403. Data plotted is courtesy of [13].

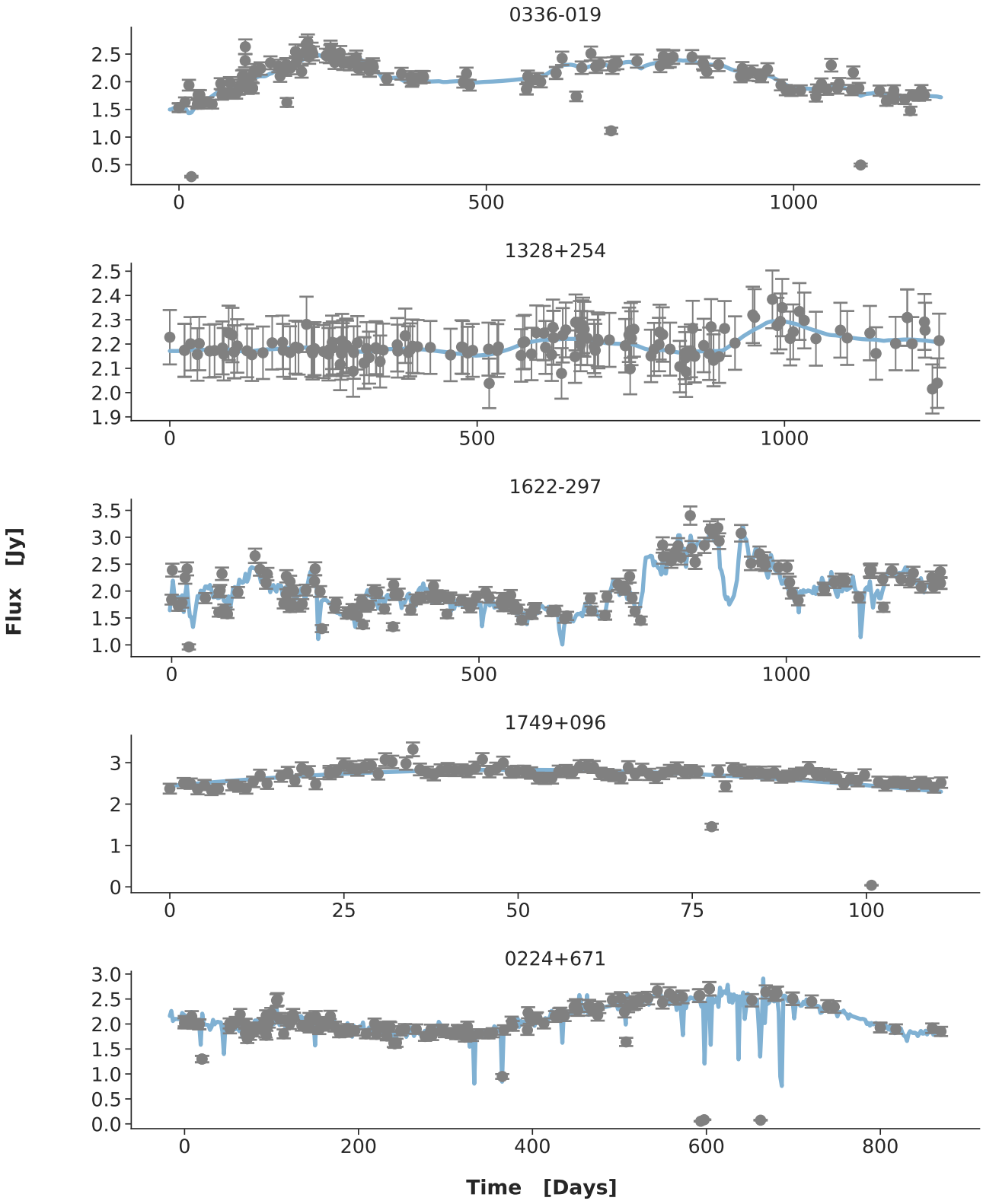


Fig. A.5. The light curves of AGN 0850-121, 0336-019, 0528+134, 2200+420 and 1237+049. Data plotted is courtesy of [13].

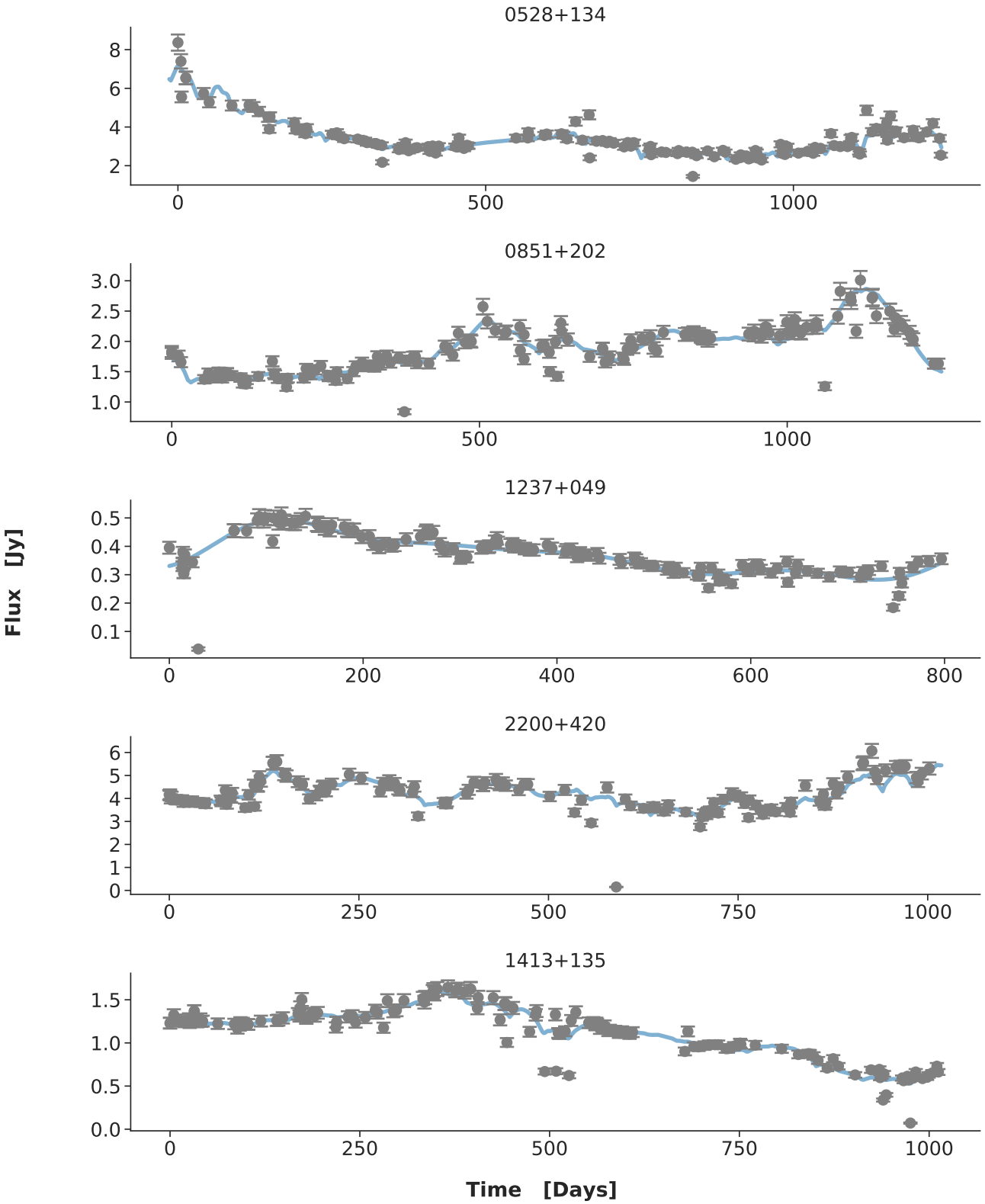


Fig. A.6. The light curves of AGN 0954+658, 0851+202, 1749+096, 1413+135 and 0224+671. Data plotted is courtesy of [13].

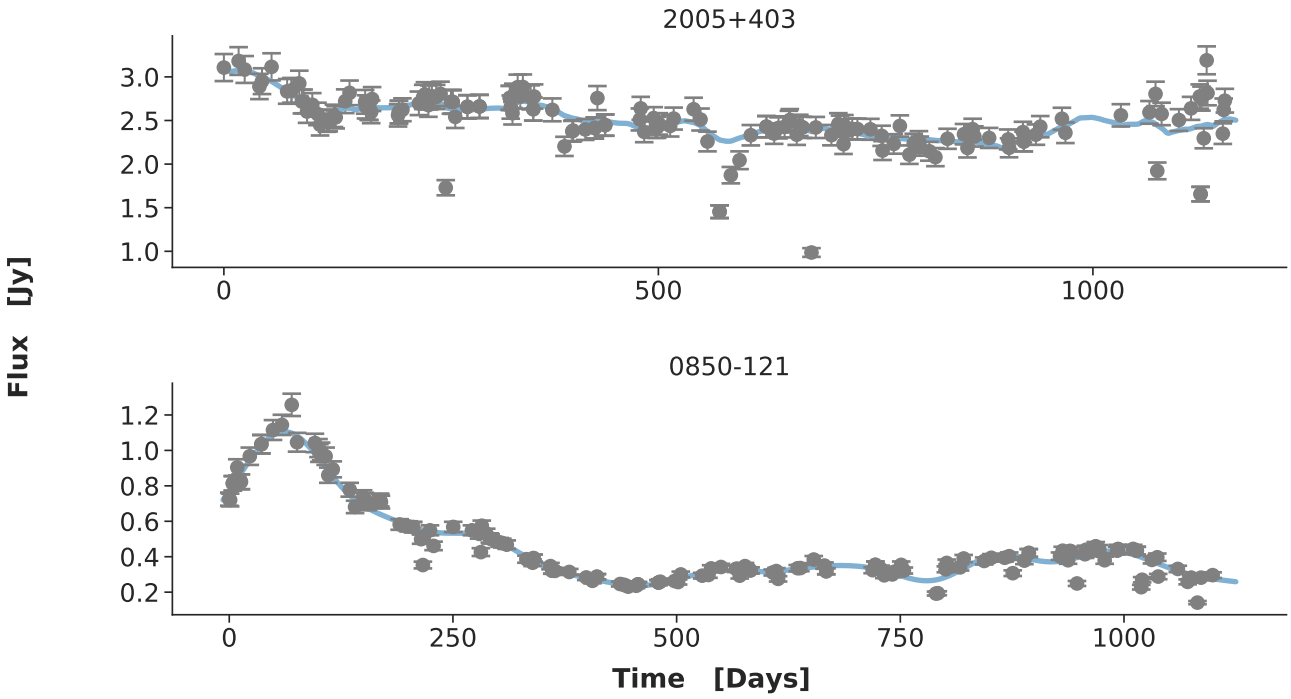


Fig. A.7. The light curves of AGN 1622-297 and 1328+254. Data plotted is courtesy of [13].

A.2 Algol

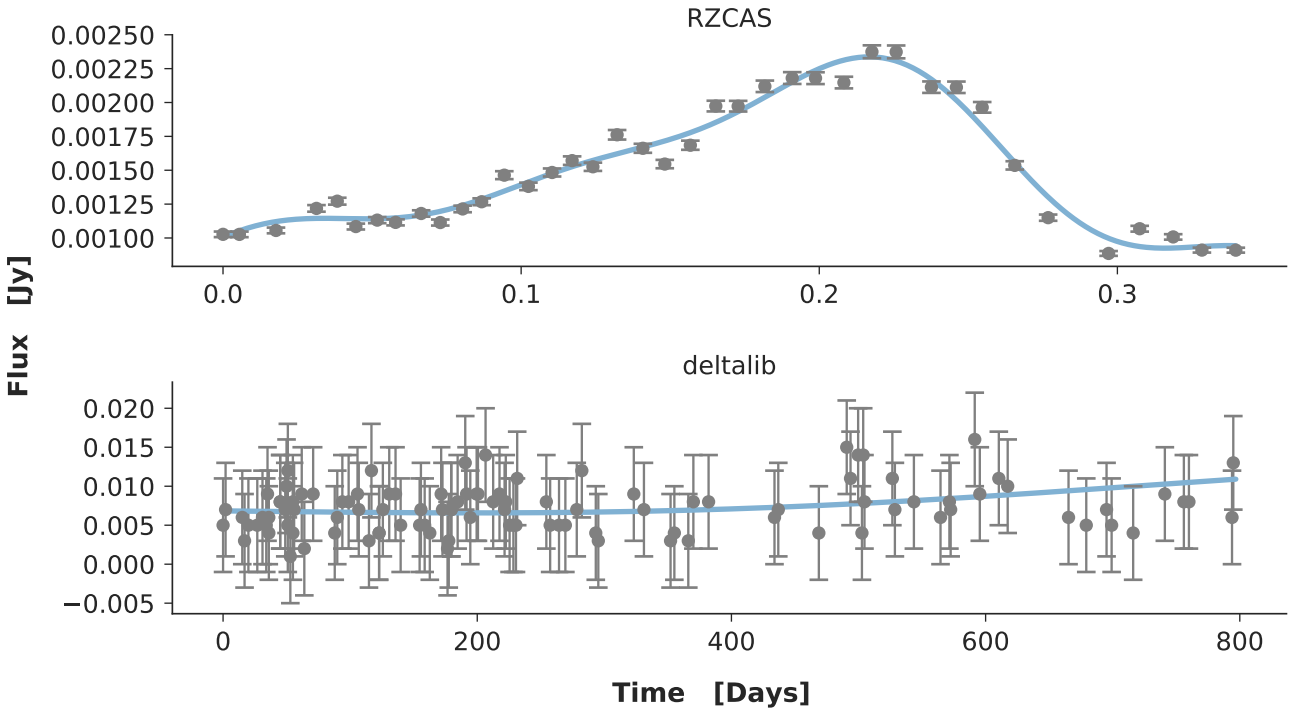


Fig. A.8. The light curves of Algol RZCAS and deltalib. Data plotted is courtesy of [13].

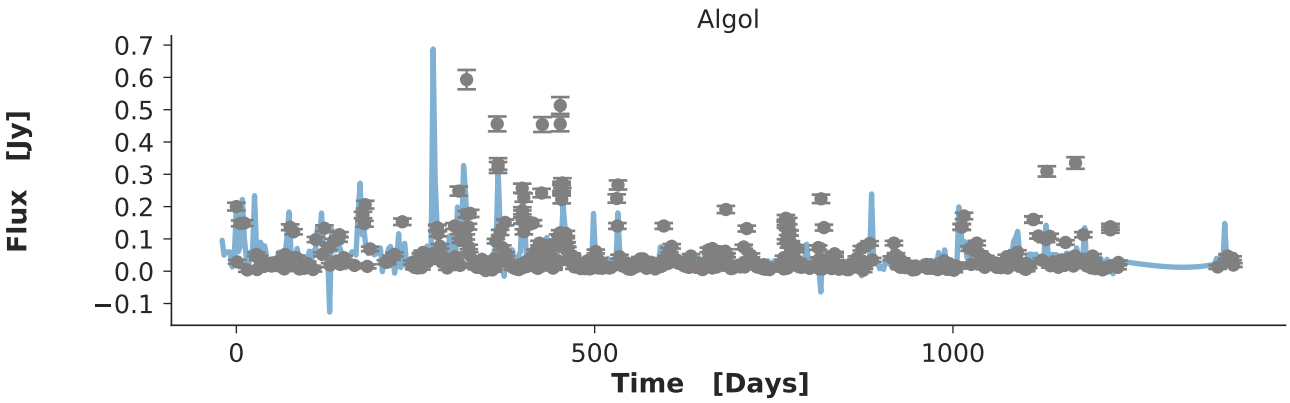


Fig. A.9. The light curve of the original Algol system. Data plotted is courtesy of [13].

A.3 Flare Star

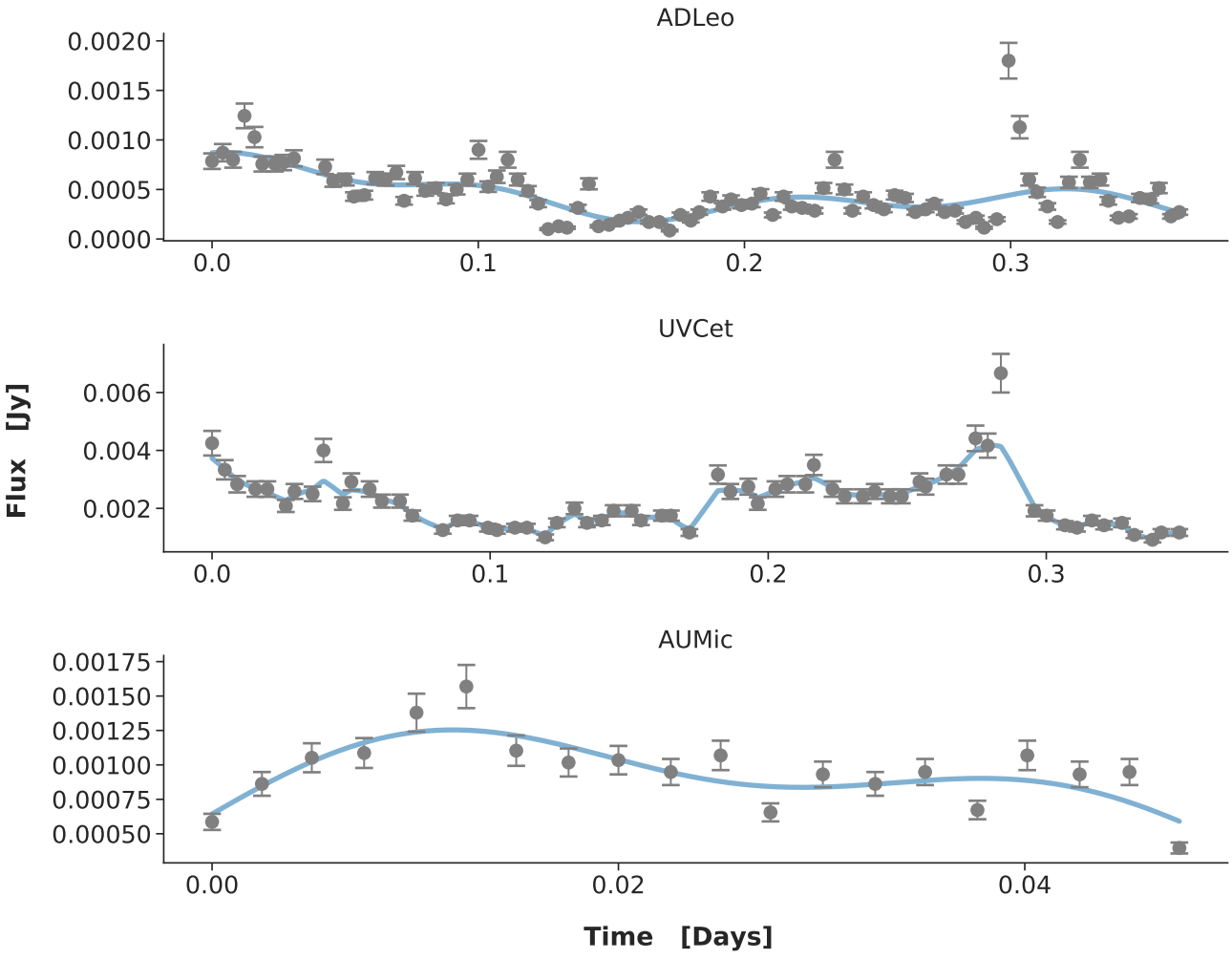


Fig. A.10. The light curves of flare stars ADLeo, UVCet and AUMic. Data plotted is courtesy of [13].

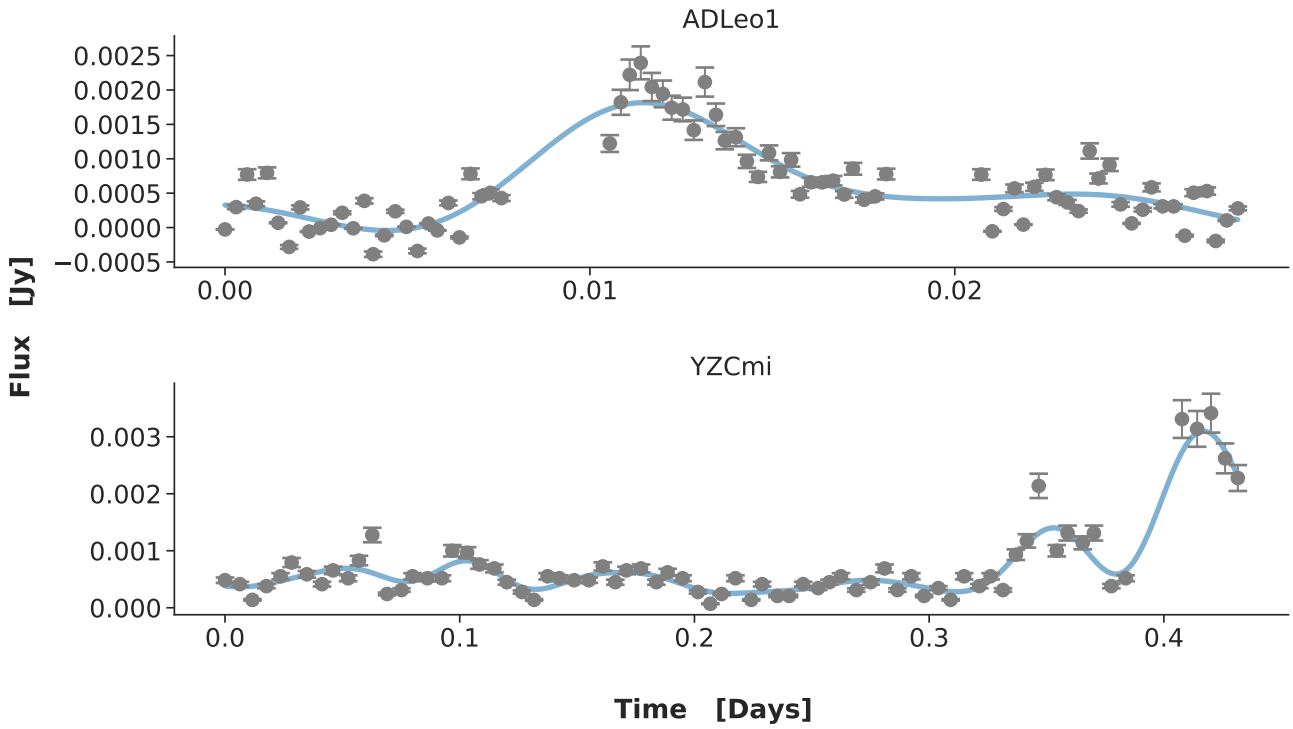


Fig. A.11. The light curves of flare stars ADLeo1 and YZCmi. Data plotted is courtesy of [13].

A.4 GRB

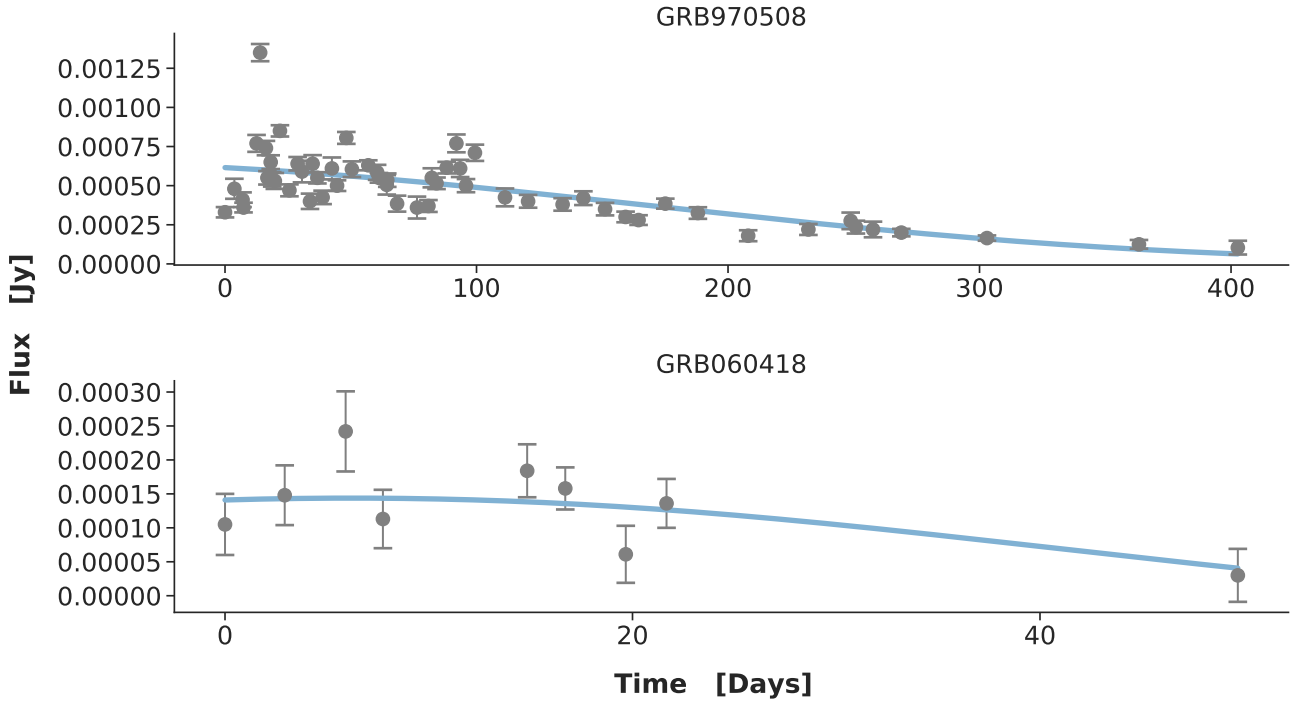


Fig. A.12. The light curves of GRBs GRB970508 and GRB060418. Data plotted is courtesy of [13].

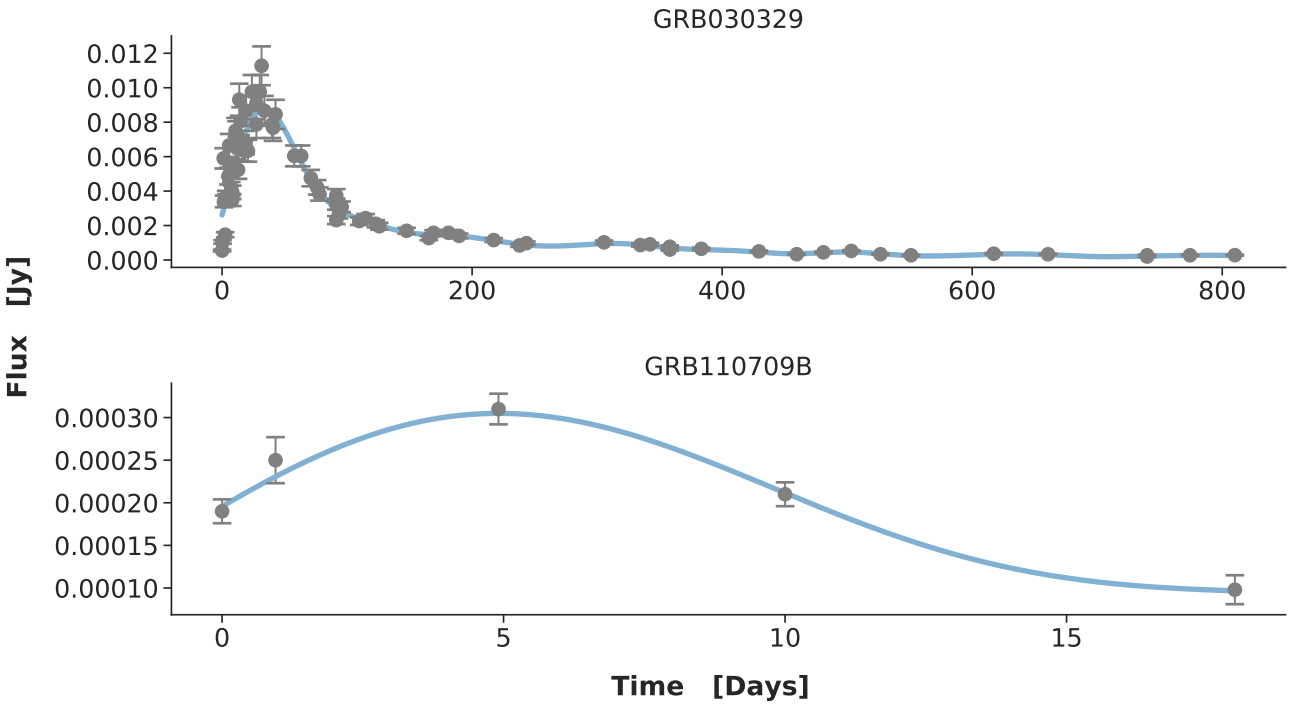


Fig. A.13. The light curves of GRBs GRB030329 and GRB1110709B. Data plotted is courtesy of [13].

A.5 Kilonova

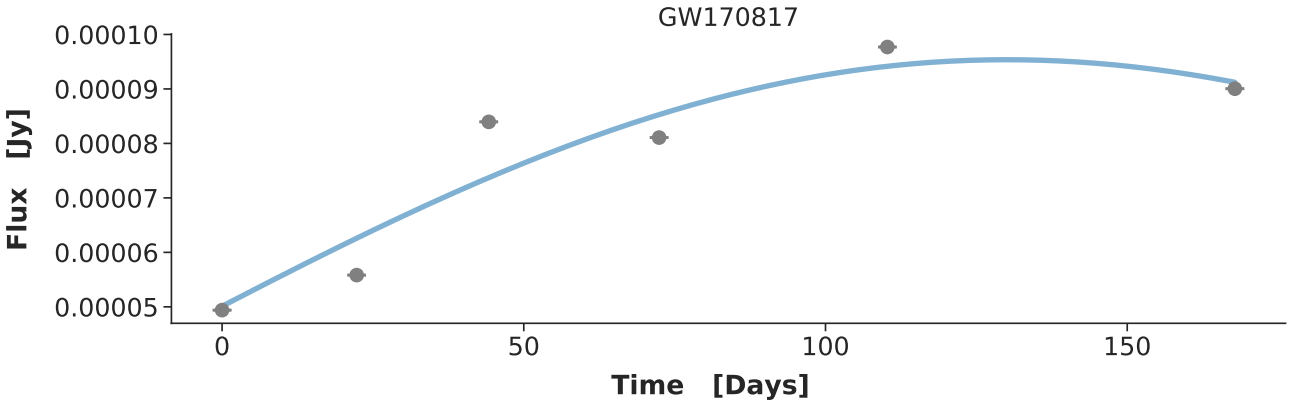


Fig. A.14. The light curve of Kilonova GW170817. Data plotted is courtesy of [32].

A.6 Magnetar

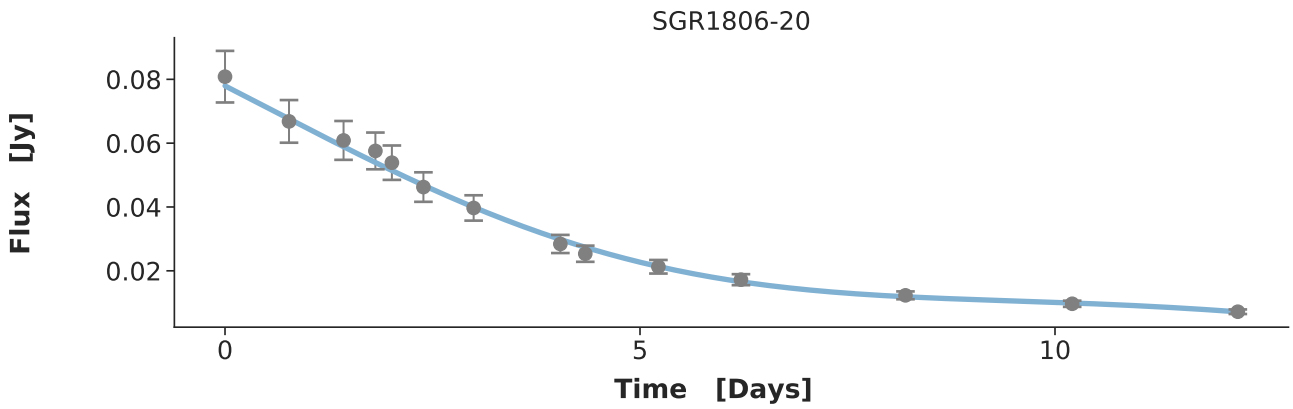


Fig. A.15. The light curve of Magnetar SGR1806-20. Data plotted is courtesy of [13].

A.7 Nova

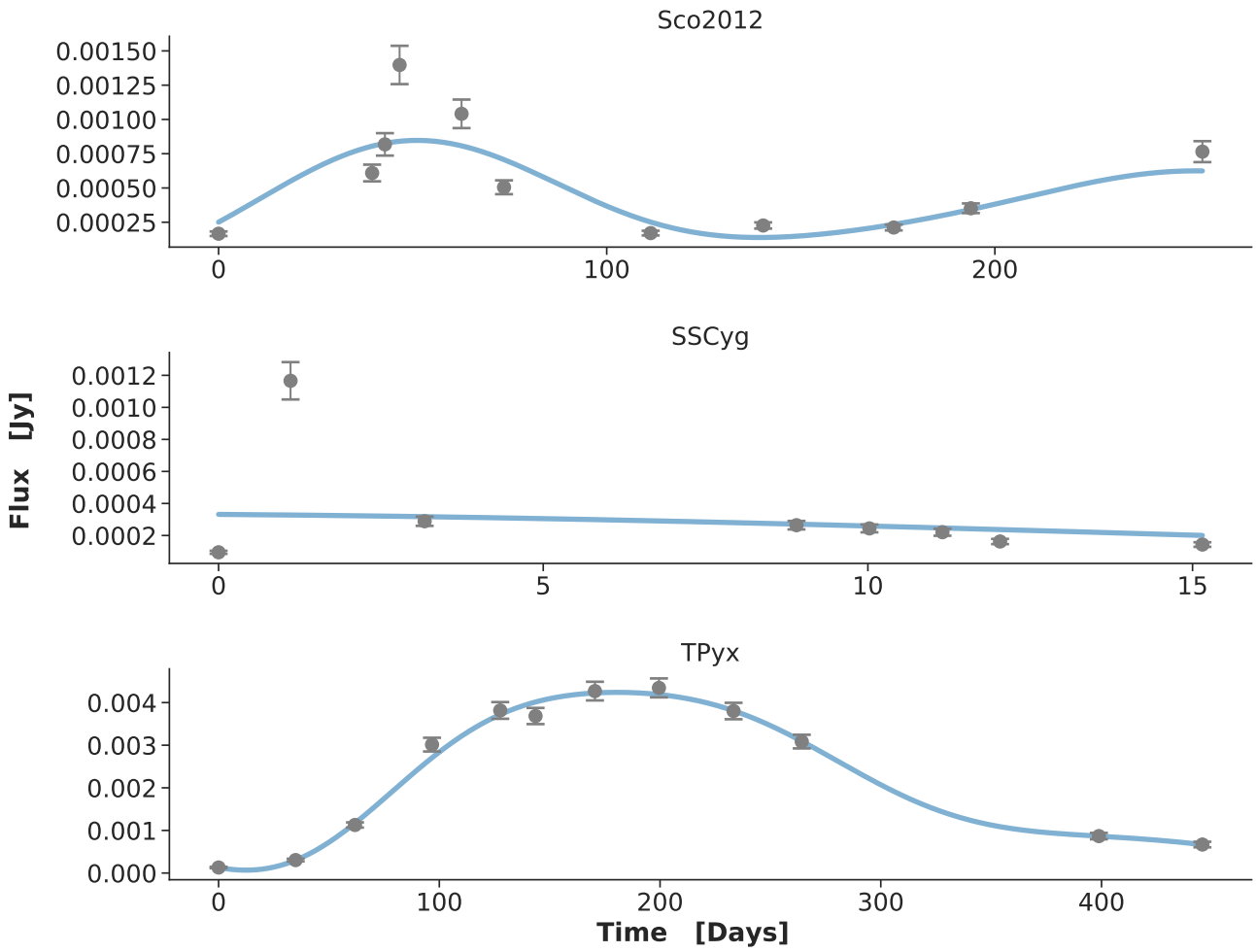


Fig. A.16. The light curves of Nova Sco2012, SSCyg and TPyx. Data plotted is courtesy of [13].

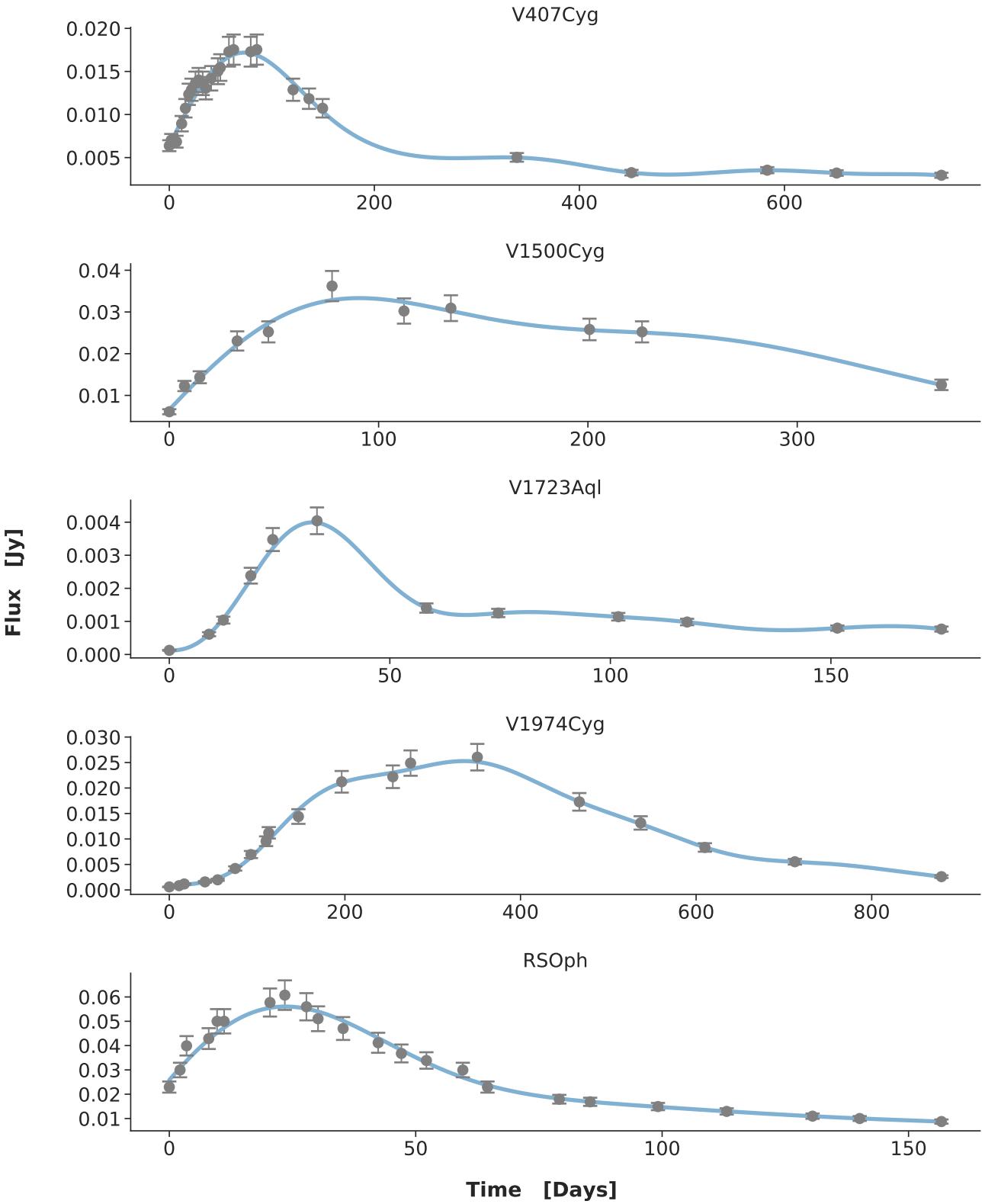


Fig. A.17. The light curves of Nova V407Cyg, V1500Cyg, V1723Aql, V1974Cyg and RSOph. Data plotted is courtesy of [13].

A.8 RSCVn

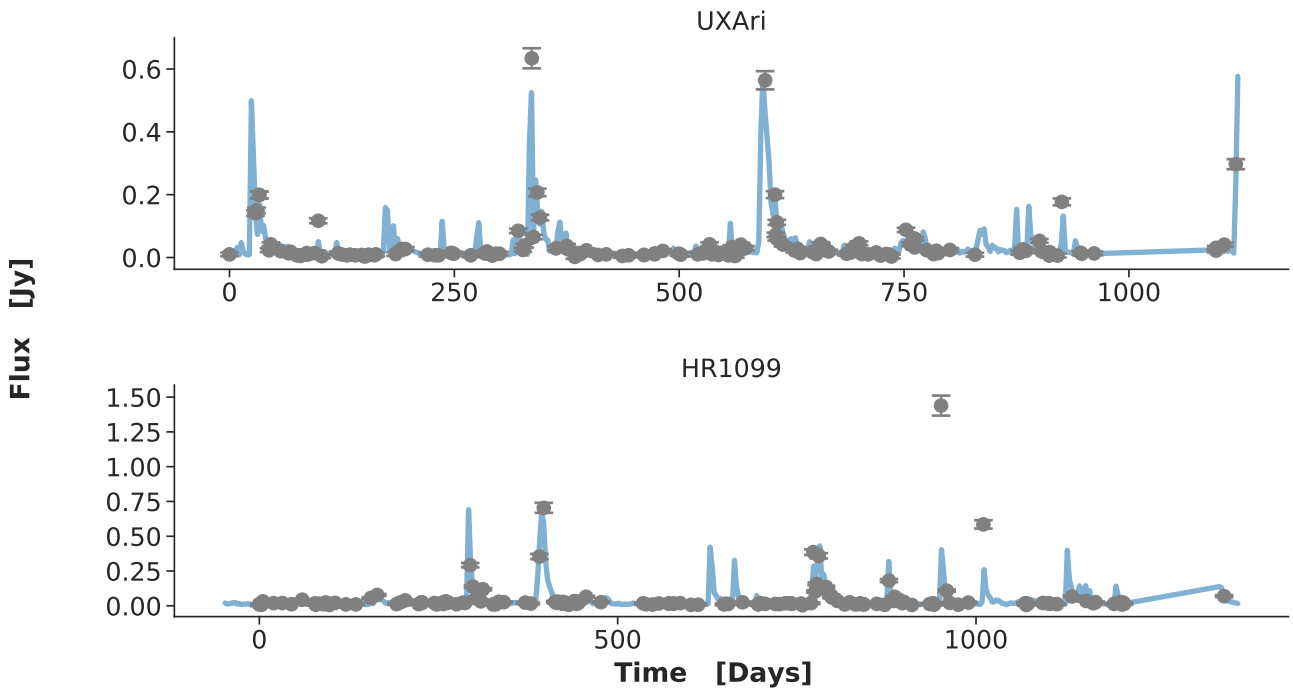


Fig. A.18. The light curves of RSCVn UXAri and HR1099. Data plotted is courtesy of [13].

A.9 Supernova

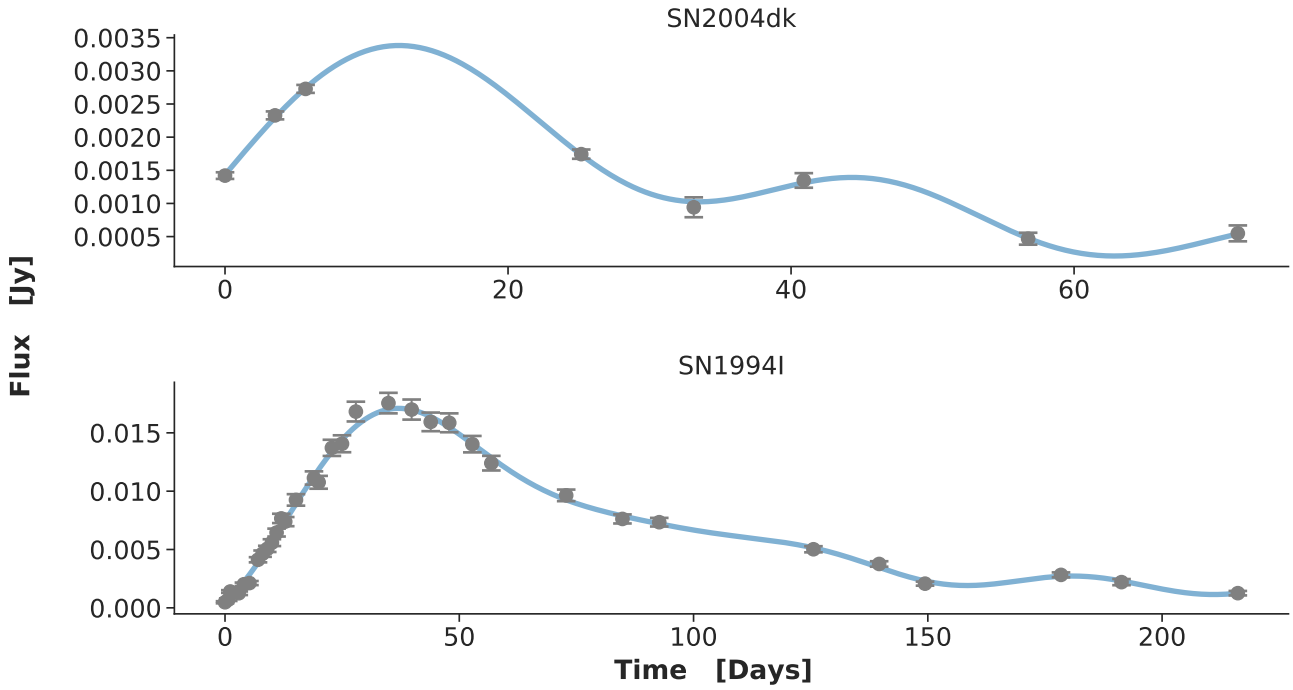


Fig. A.19. THe light curves of supernova SN2004dk and SN1994l. Data plotted is courtesy of [13].

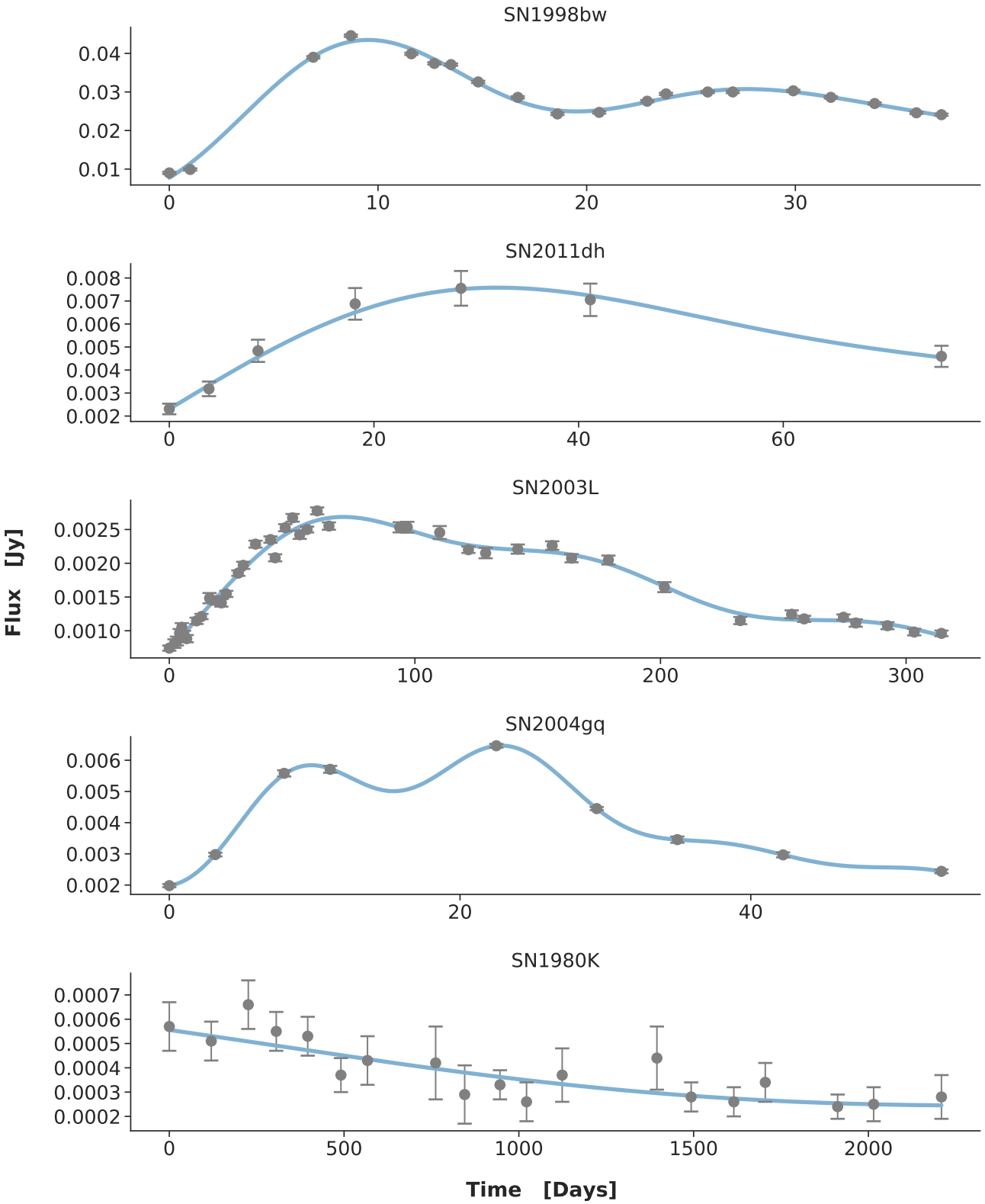


Fig. A.20. The light curves of supernova SN1998bw, SN2011dh, SN2003L, SN2004gq and SN1980K. Data plotted is courtesy of [13].

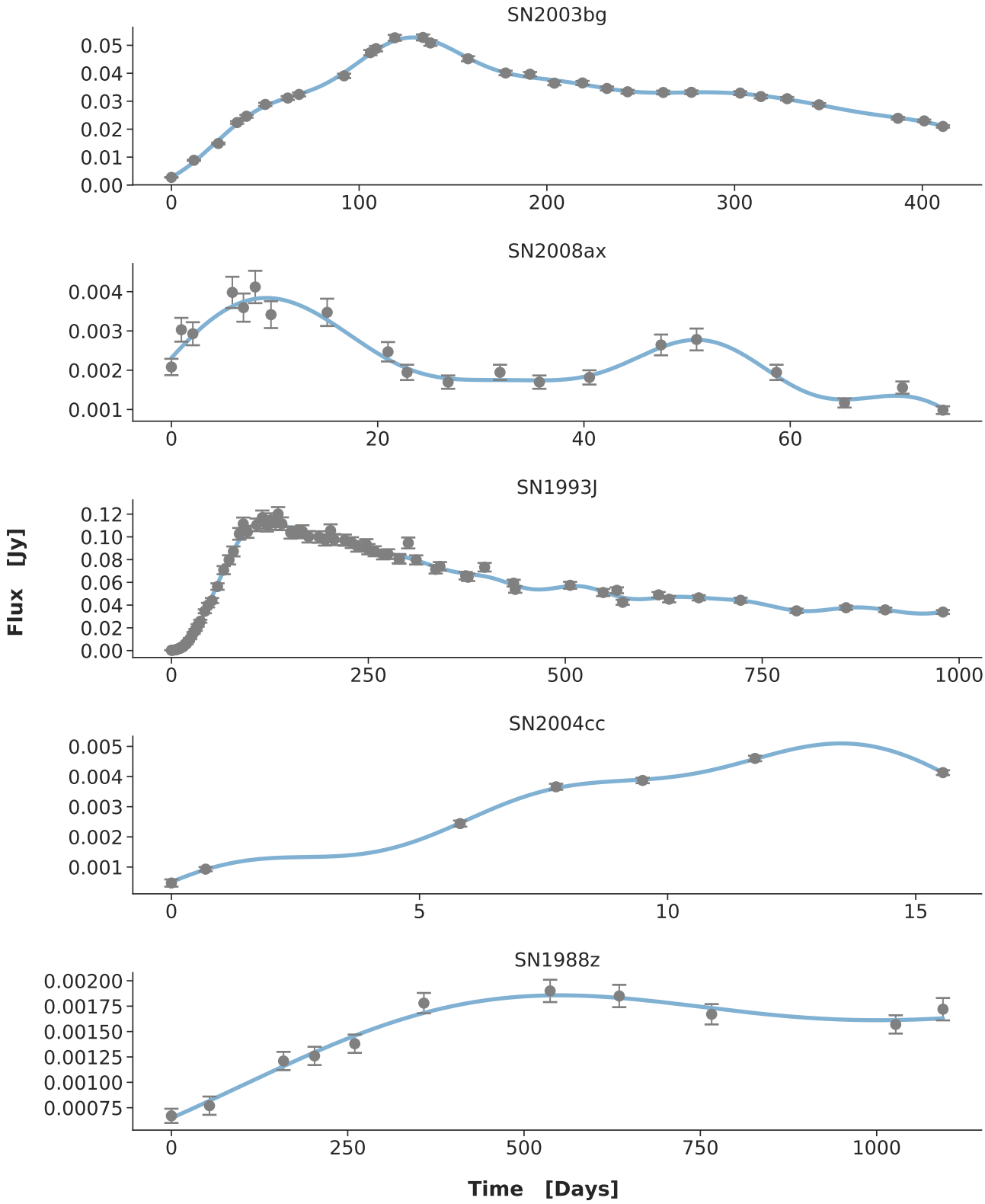


Fig. A.21. The light curves of supernova SN2003bg, SN2008ax, SN1993j, SN2004cc and SN1988z. Data plotted is courtesy of [13].

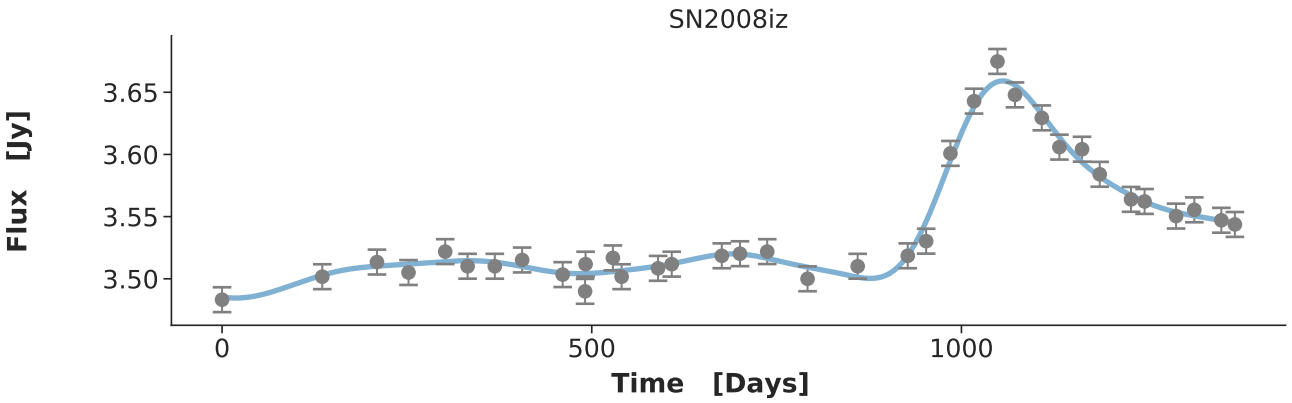


Fig. A.22. The light curve of supernova SN2008iz. Data plotted is courtesy of [13].

A.10 TDE

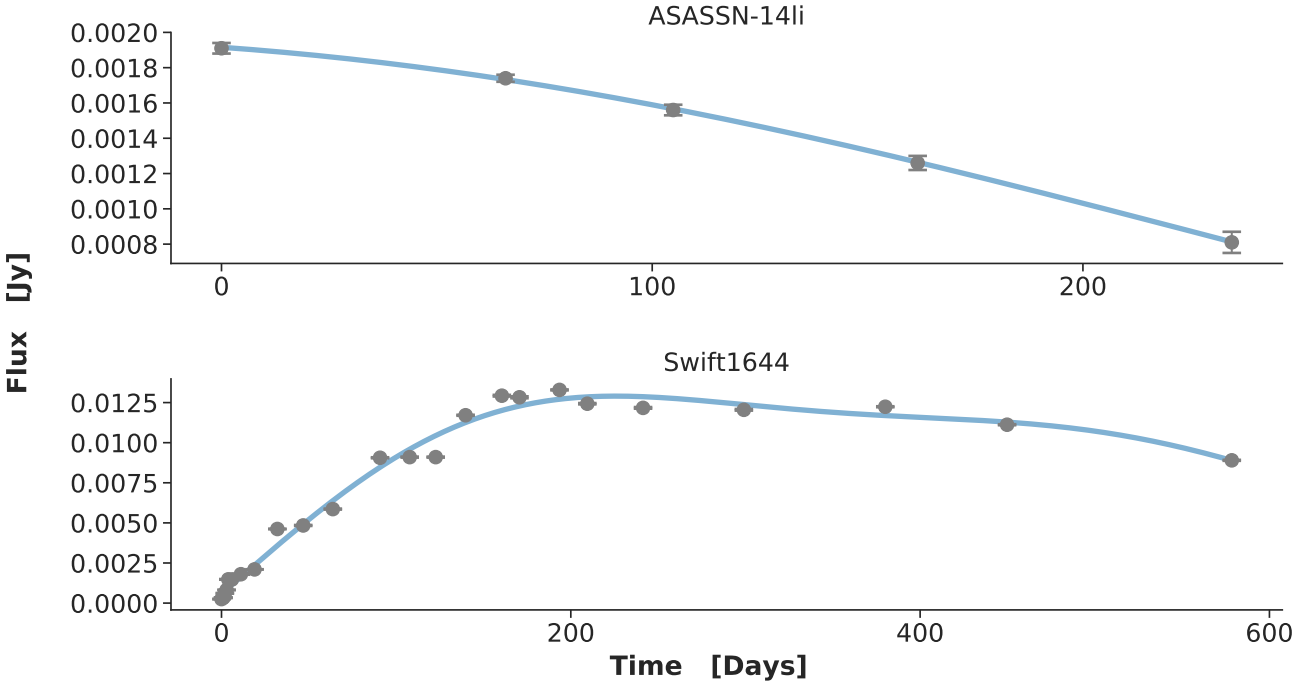


Fig. A.23. The light curves of TDE ASASSN-14li and Swift1644. Data plotted is courtesy of [13].

A.11 XRB

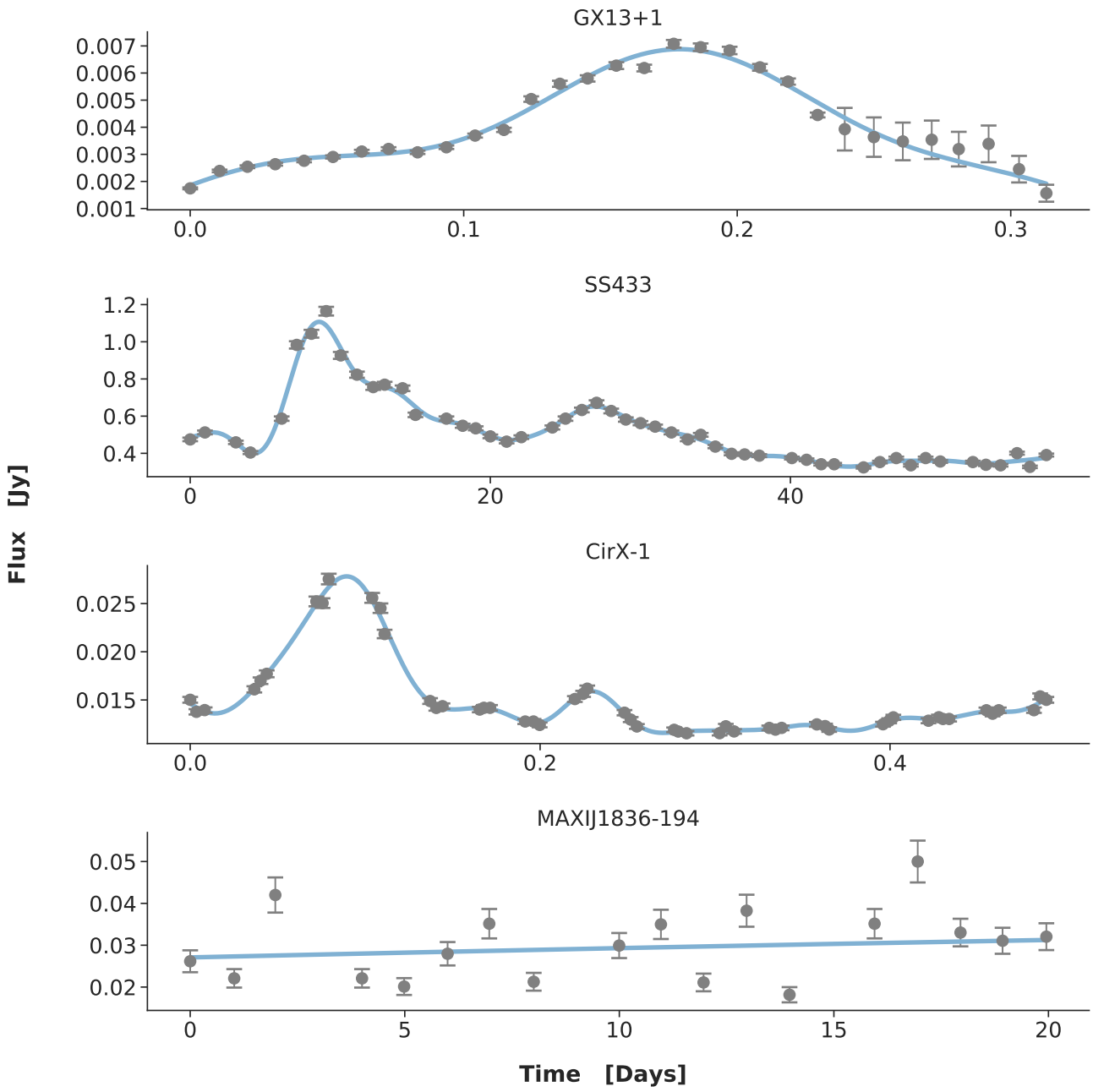


Fig. A.24. The light curves of XRB GX13+1, SS433, CirX-1 and MAXlj1836-194.

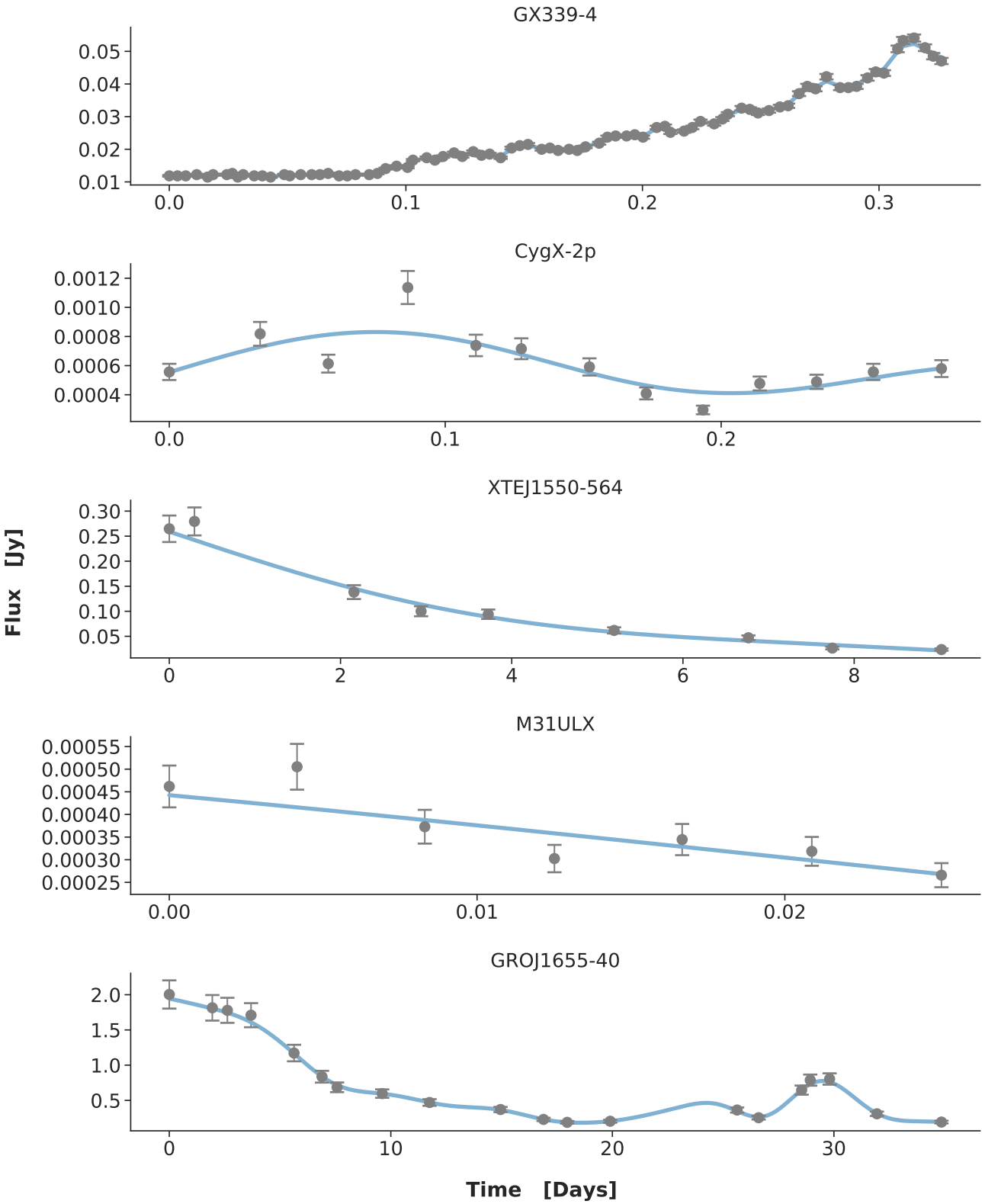


Fig. A.25. The light curves of XRB GX229-4, CygX-2p, XTEj1550-564, M31ULX and GROj1655-40.

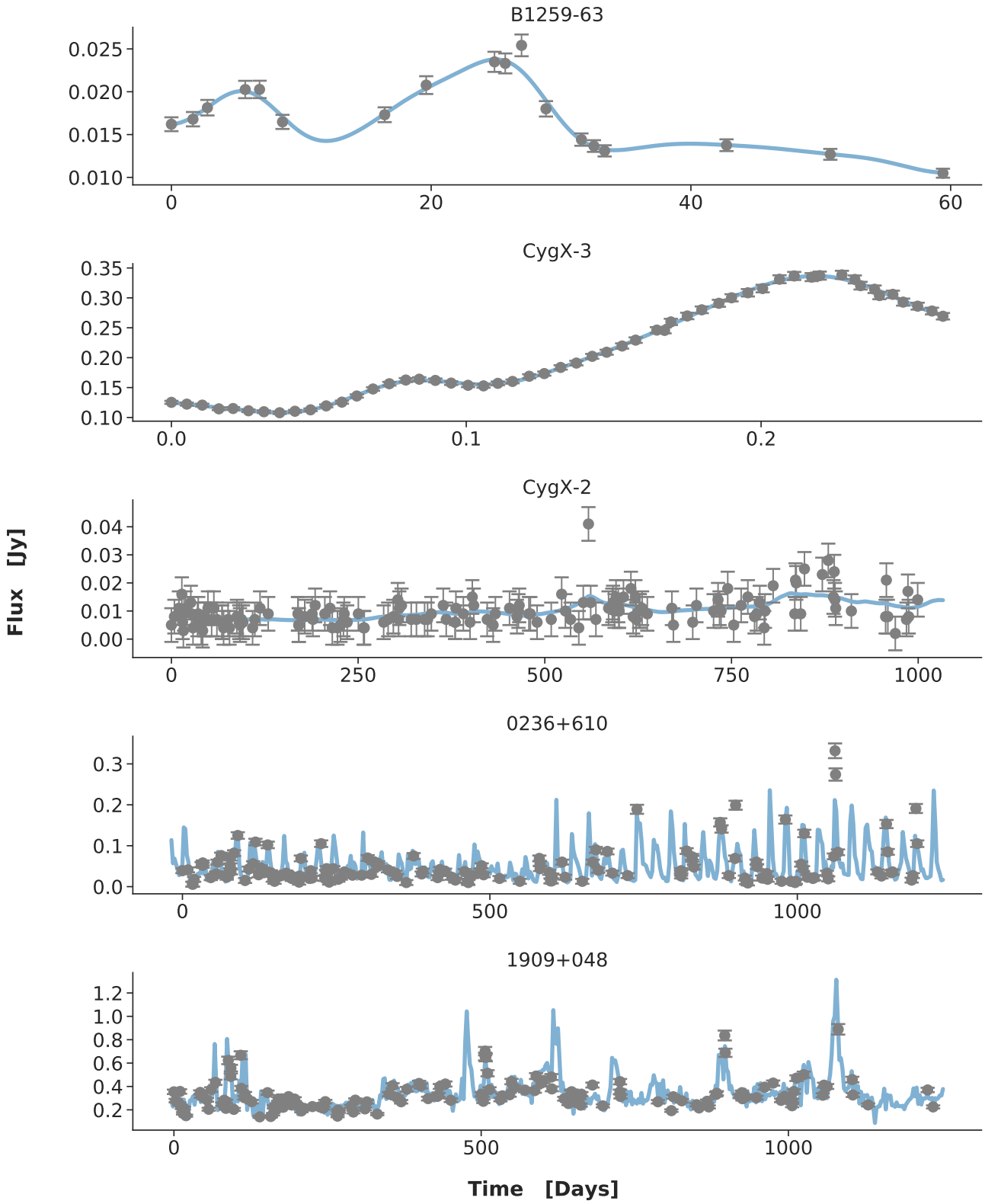


Fig. A.26. The light curves of XRB B1259-63, CygX-3, CygX-2, 0236+610 and 1909+048. Data plotted is courtesy of [13].

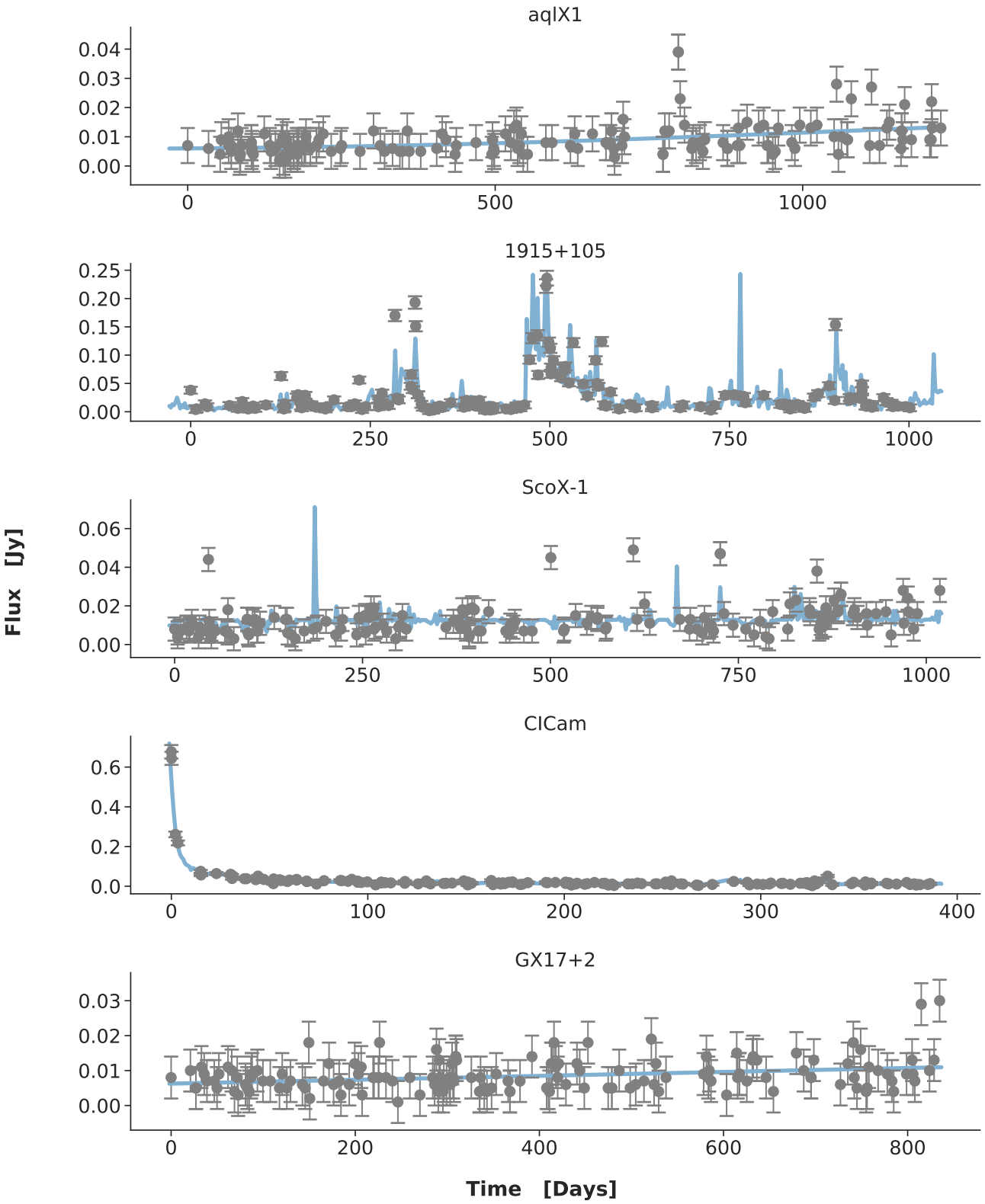


Fig. A.27. The light curves of XRB aqlX1, 1915+105, ScoX-1, ClCam and GX17+2. Data plotted is courtesy of [13].

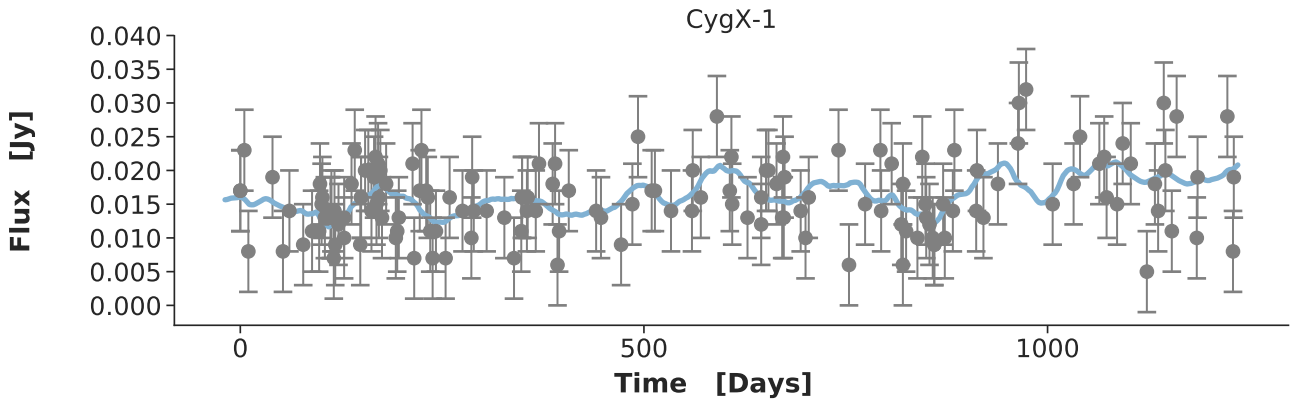


Fig. A.28. The light curve of XRB CygX-1. Data plotted is courtesy of [13].

Bibliography

[1] S. Bloemen *et al.*, "MeerLICHT and BlackGEM: custom-built telescopes to detect faint optical transients", *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 9906, p. 990 664, 2016 (cit. on pp. xxiii, 3, 60).

[2] LSST Science Collaboration, P. A. Abell, J. Allison, *et al.*, "LSST Science Book, Version 2.0", *ArXiv e-prints*, Dec. 2009. arXiv: 0912.0201 [astro-ph.IM] (cit. on p. xxiii).

[3] R. Fender *et al.*, "ThunderKAT: The MeerKAT Large Survey Project for Image-Plane Radio Transients", *PoS*, vol. MeerKAT2016, p. 013, 2018. arXiv: 1711.04132 [astro-ph.HE] (cit. on pp. 3, 13, 88).

[4] M. I. Large, "The mechanisms of radio emission", *Radiotekhnika*, pp. 126–137, 1963 (cit. on p. 5).

[5] J. J. Condon and S. M. Ransom, *Essential Radio Astronomy*. 2016 (cit. on p. 5).

[6] R. Fender, A. Stewart, J.-P. Macquart, *et al.*, "Transient Astrophysics with the Square Kilometre Array", 2015. arXiv: 1507.00729 [astro-ph.HE] (cit. on p. 5).

[7] A. Hewish, S. J. Bell, J. D. H. Pilkington, P. F. Scott, and R. A. Collins, "Observation of a Rapidly Pulsating Radio Source", *nat*, vol. 217, pp. 709–713, Feb. 1968 (cit. on p. 5).

[8] V. S. Beskin, "Radio pulsars: already fifty years!", *Physics Uspekhi*, vol. 61, pp. 353–380, Apr. 2018. arXiv: 1807.08528 [astro-ph.HE] (cit. on p. 5).

[9] S. B. Popov, K. A. Postnov, and M. S. Pshirkov, "Fast Radio Bursts", *Phys. Usp.*, vol. 61, no. 10, pp. 965–979, 2018. arXiv: 1806.03628 [astro-ph.HE] (cit. on p. 5).

[10] E. Platts, A. Weltman, A. Walters, *et al.*, "A Living Theory Catalogue for Fast Radio Bursts", 2018. arXiv: 1810.05836 [astro-ph.HE] (cit. on p. 5).

[11] J. R. Percy, *Understanding Variable Stars*. May 2007 (cit. on pp. 6, 12, 14, 18).

[12] W.-C. Chen, X.-D. Li, and S.-B. Qian, "Orbital Evolution of Algol Binaries with a Circumbinary Disk", *Astrophys. J.*, vol. 649, pp. 973–978, 2006. arXiv: astro-ph/0606081 [astro-ph] (cit. on p. 6).

[13] M. Pietka, T. D. Staley, M. L. Pretorius, and R. P. Fender, "On the use of variability time-scales as an early classifier of radio transients and variables", *Mon. Not. Roy. Astron. Soc.*, vol. 471, no. 4, pp. 3788–3805, 2017. arXiv: 1707.04265 [astro-ph.HE] (cit. on pp. 7–13, 15, 17, 18, 60, 91–106, 109–111).

[14] F. Verbunt, "Origin and evolution of x-ray binaries and binary radio pulsars", *Annual Review of Astronomy and Astrophysics*, vol. 31, no. 1, pp. 93–127, 1993 (cit. on p. 7).
[15] T. M. Tauris and E. P. J. van den Heuvel, "Formation and evolution of compact stellar x-ray sources", *Submitted to: To appear in the book 'Compact Stellar X-Ray Sources'*, 2003. arXiv: astro-ph/0303456 [astro-ph] (cit. on p. 7).
[16] G. M. H. J. Habets and J. R. W. Heintze, "Empirical bolometric corrections for the main-sequence", *aaps*, vol. 46, pp. 193–237, Nov. 1981 (cit. on p. 8).
[17] S. Chaty, "Nature, Formation, and Evolution of High Mass X-Ray Binaries", in *Evolution of Compact Binaries*, L. Schmidtobreick, M. R. Schreiber, and C. Tappert, Eds., ser. Astronomical Society of the Pacific Conference Series, vol. 447, Sep. 2011, p. 29. arXiv: 1107.0231 [astro-ph.HE] (cit. on p. 8).
[18] W. E. Kunkel, "Solar neighborhood flare stars - A review", in *Variable Stars and Stellar Evolution*, V. E. Sherwood and L. Plaut, Eds., ser. IAU Symposium, vol. 67, 1975, pp. 15–46 (cit. on p. 8).
[19] L. Breiman, "Random forests", *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001 (cit. on pp. 9, 43).
[20] M. F. Bode, "Classical and Recurrent Nova Outbursts", *arXiv e-prints*, arXiv:1111.4941, arXiv:1111.4941, Nov. 2011. arXiv: 1111.4941 [astro-ph.SR] (cit. on p. 11).
[21] S. Starrfield, C. Iliadis, and W. R. Hix, "The Thermonuclear Runaway and the Classical Nova Outburst", *Publications of the Astronomical Society of the Pacific*, vol. 128, p. 051 001, May 2016. arXiv: 1605.04294 [astro-ph.SR] (cit. on p. 11).
[22] V. M. Kaspi and A. Beloborodov, "Magnetars", *Ann. Rev. Astron. Astrophys.*, vol. 55, pp. 261–301, 2017. arXiv: 1703.00068 [astro-ph.HE] (cit. on p. 11).
[23] D. Maoz and F. Mannucci, "Type-Ia Supernova Rates and the Progenitor Problem: A Review", *Publications of the Astronomical Society of Australia*, vol. 29, pp. 447–465, Jan. 2012. arXiv: 1111.4492 [astro-ph.CO] (cit. on p. 13).
[24] J. Whelan and I. Iben Jr., "Binaries and Supernovae of Type I", *apj*, vol. 186, pp. 1007–1014, Dec. 1973 (cit. on p. 13).
[25] R. F. Webbink, "Double white dwarfs as progenitors of R Coronae Borealis stars and Type I supernovae", *apj*, vol. 277, pp. 355–360, Feb. 1984 (cit. on p. 13).
[26] D. Branch and D. L. Miller, "Type IA supernovae as standard candles", *apjl*, vol. 405, pp. L5–L8, Mar. 1993 (cit. on p. 13).
[27] B. Abbott *et al.*, "GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral", *Phys. Rev. Lett.*, vol. 119, no. 16, p. 161 101, 2017. arXiv: 1710.05832 [gr-qc] (cit. on p. 14).
[28] C. Ashall *et al.*, "GRB 161219B-SN 2016jca: a powerful stellar collapse", 2017. arXiv: 1702.04339 [astro-ph.HE] (cit. on p. 14).
[29] K. C. Sahu, M. Livio, L. Petro, *et al.*, "Observations of grb 970228 and grb 970508, and the neutron-star merger model", *Astrophys. J.*, vol. 489, p. L127, 1997. arXiv: astro-ph/9706225 [astro-ph] (cit. on p. 14).
[30] S. Rosswog, "The multi-messenger picture of compact binary mergers", *Int. J. Mod. Phys.*, vol. D24, no. 05, p. 1 530 012, 2015. arXiv: 1501.02081 [astro-ph.HE] (cit. on p. 15).

[31] S. J. Smartt *et al.*, "A kilonova as the electromagnetic counterpart to a gravitational-wave source", *Nature*, vol. 551, no. 7678, pp. 75–79, 2017. arXiv: 1710.05841 [astro-ph.HE] (cit. on p. 15).

[32] D. Dobie, D. L. Kaplan, T. Murphy, *et al.*, "A turnover in the radio light curve of GW170817", *Astrophys. J.*, vol. 858, no. 2, p. L15, 2018. arXiv: 1803.06853 [astro-ph.HE] (cit. on pp. 15, 16, 60, 100).

[33] J. G. Hills, "Possible power source of Seyfert galaxies and QSOs", *nat*, vol. 254, pp. 295–298, Mar. 1975 (cit. on p. 15).

[34] M. J. Rees, "Tidal disruption of stars by black holes of 10 to the 6th-10 to the 8th solar masses in nearby galaxies", *nature*, vol. 333, pp. 523–528, Jun. 1988 (cit. on p. 15).

[35] A. Ulmer, "Flares from the tidal disruption of stars by massive black holes", *The Astrophysical Journal*, vol. 514, no. 1, pp. 180–187, 1999 (cit. on p. 15).

[36] I. Donnarumma, E. M. Rossi, R. Fender, *et al.*, "SKA as a powerful hunter of jetted Tidal Disruption Events", *PoS*, vol. AASKA14, p. 054, 2015. arXiv: 1501.04640 [astro-ph.HE] (cit. on p. 17).

[37] P. Padovani, D. M. Alexander, R. J. Assef, *et al.*, "Active galactic nuclei: what's in a name?", *Astronomy and Astrophysics Review*, vol. 25, 2, p. 2, Aug. 2017. arXiv: 1707.07134 [astro-ph.GA] (cit. on p. 17).

[38] C. Tadhunter, "Radio AGN in the local universe: unification, triggering and evolution", *Astronomy and Astrophysics Review*, vol. 24, 10, p. 10, Jun. 2016. arXiv: 1605.08773 [astro-ph.GA] (cit. on p. 17).

[39] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997 (cit. on pp. 19, 33, 34, 43).

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12, Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105 (cit. on pp. 19, 35).

[41] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of Go with deep neural networks and tree search", *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016 (cit. on pp. 19, 35).

[42] J. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis", eng, *Psychometrika*, vol. 29, no. 1, pp. 1,27, 1964 (cit. on p. 20).

[43] A. Inselberg, "The plane with parallel coordinates", eng, *The Visual Computer*, vol. 1, no. 2, pp. 69,91, 1985 (cit. on p. 20).

[44] M. Ferreira de Oliveira and H. Levkowitz, "From visual data exploration to visual data mining: A survey", eng, *Visualization and Computer Graphics, IEEE Transactions on*, vol. 9, no. 3, pp. 378,394, 2003 (cit. on p. 20).

[45] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE", *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008 (cit. on pp. 21, 30, 31).

[46] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation", in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 532–538 (cit. on p. 21).

[47] J. P. Egan, *Signal detection theory and ROC analysis*, ser. Series in Cognition and Perception. New York, NY: Academic Press, 1975 (cit. on p. 24).

[48] T. Fawcett, "An introduction to roc analysis", *Pattern Recogn. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006 (cit. on p. 24).

[49] J. Macqueen, "Some methods for classification and analysis of multivariate observations", in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297 (cit. on p. 25).

[50] S. Shukla, "A review on k-means data clustering approach", 2014 (cit. on p. 25).

[51] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space", *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901, PCA beginnings. eprint: https://doi.org/10.1080/14786440109462720 (cit. on p. 27).

[52] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments", *Philos Trans A Math Phys Eng Sci*, vol. 374, 2016 (cit. on p. 27).

[53] J. Shlens, "A tutorial on principal component analysis", in *Systems Neurobiology Laboratory, Salk Institute for Biological Studies*, 2005 (cit. on p. 28).

[54] J. F. Box, "Gosset, fisher, and the t distribution", *The American Statistician*, vol. 35, no. 2, pp. 61–66, 1981 (cit. on p. 31).

[55] S. Kullback, *Information Theory and Statistics*. Wiley, 1959 (cit. on p. 31).

[56] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms", eng, in *Proceedings of the 23rd international conference on machine learning*, ser. ICML '06, vol. 148, ACM, 2006, pp. 161,168 (cit. on pp. 33, 88).

[57] M. Liu, M. Wang, J. Wang, and D. Li, "Comparison of random forest, support vector machine and back propagation neural network for electronic tongue data classification: Application to the recognition of orange beverage and chinese vinegar", *Sensors and Actuators B: Chemical*, vol. 177, pp. 970 –980, 2013 (cit. on pp. 33, 88).

[58] M. Lochner, J. D. McEwen, H. V. Peiris, O. Lahav, and M. K. Winter, "Photometric Supernova Classification With Machine Learning", *Astrophys. J. Suppl.*, vol. 225, no. 2, p. 31, 2016. arXiv: 1603.00882 [astro-ph.IM] (cit. on pp. 33, 53, 57, 58, 66, 87, 88).

[59] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain", *Psychological Review*, pp. 65–386, 1958 (cit. on p. 34).

[60] H. Robbins and S. Monro, "A stochastic approximation method", *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, Sep. 1951 (cit. on p. 35).

[61] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", *nat*, vol. 323, pp. 533–536, Oct. 1986 (cit. on p. 35).

[62] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview", *arXiv e-prints*, arXiv:1404.7828, arXiv:1404.7828, Apr. 2014. arXiv: 1404.7828 [cs.NE] (cit. on p. 35).

[63] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016 (cit. on p. 35).

[64] Y. L. Cun, B. Boser, J. S. Denker, *et al.*, "Advances in neural information processing systems 2", in, D. S. Touretzky, Ed., San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. Handwritten Digit Recognition with a Back-propagation Network, pp. 396–404 (cit. on p. 35).

[65] Y. LeCun and Y. Bengio, "The handbook of brain theory and neural networks", in, M. A. Arbib, Ed., Cambridge, MA, USA: MIT Press, 1998, ch. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258 (cit. on p. 35).

[66] P. Graff, F. Feroz, M. P. Hobson, and A. N. Lasenby, "SKYNET: an efficient and robust neural network training tool for machine learning in astronomy", *Mon. Not. Roy. Astron. Soc.*, vol. 441, no. 2, pp. 1741–1759, 2014. arXiv: 1309.0790 [astro-ph.IM] (cit. on p. 35).

[67] I. Rish, "An empirical study of the naive bayes classifier", Tech. Rep., 2001 (cit. on p. 37).

[68] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss", eng, *Machine Learning*, vol. 29, no. 2, pp. 103,130, 1997-11 (cit. on p. 37).

[69] C. Cortes and V. Vapnik, "Support-vector networks", *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995 (cit. on p. 38).

[70] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005 (cit. on p. 41).

[71] T. K. Ho, "Random decision forests", in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, 1995, 278–282 vol.1 (cit. on p. 43).

[72] N. M. Ball and R. J. Brunner, "Data Mining and Machine Learning in Astronomy", *International Journal of Modern Physics D*, vol. 19, pp. 1049–1106, 2010. arXiv: 0906.2173 [astro-ph.IM] (cit. on pp. 43, 51, 52).

[73] C.-L. Liu, "A tutorial of the wavelet transform", Jan. 2010 (cit. on p. 47).

[74] S. G.S. G. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Third. 2009, pp. xx + 805 (cit. on pp. 47, 49, 58).

[75] A. Haar, "Zur theorie der orthogonalen funktionensysteme", *Mathematische Annalen*, vol. 69, no. 3, pp. 331–371, 1910 (cit. on p. 48).

[76] Y. Meyer, *Wavelets and operators*, ser. Cambridge studies in advanced mathematics ; 37. Cambridge: Cambridge University Press, 1992 (cit. on p. 48).

[77] I. Daubechies, "Orthonormal bases of compactly supported wavelets", *Communications on Pure and Applied Mathematics*, vol. 41, no. 7, pp. 909–996, 1988. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.3160410705 (cit. on p. 49).

[78] S. Odewahn, E. Stockwell, R. Pennington, R. Humphreys, and W. Zumach, "Automated star/galaxy discrimination with neural networks", English (US), *Astronomical Journal*, vol. 103, no. 1, pp. 318–331, Jan. 1992 (cit. on p. 52).

[79] O. Lahav, A. Naim, L. Sodré Jr., and M. C. Storrie-Lombardi, "Neural computation as a tool for galaxy classification: methods and examples", *mnras*, vol. 283, p. 207, Nov. 1996. eprint: astro-ph/9508012 (cit. on p. 52).

[80] C. A. L. Bailer-Jones, R. Gupta, and H. P. Singh, "An introduction to artificial neural networks", 2001. arXiv: astro-ph/0102224 [astro-ph] (cit. on p. 52).

[81] Y. Zhang, L. Li, and Y. Zhao, "Morphology Classification and Photometric Redshift Measurement of Galaxies", *Mon. Not. Roy. Astron. Soc.*, vol. 392, p. 233, 2009. arXiv: 0810.0867 [astro-ph] (cit. on p. 52).

[82] R. L. White, "Astronomical Applications of Oblique Decision Trees", in *American Institute of Physics Conference Series*, C. A. L. Bailer-Jones, Ed., ser. American Institute of Physics Conference Series, vol. 1082, Dec. 2008, pp. 37–43 (cit. on p. 52).

[83] N. E. M. Khalifa, M. H. N. Taha, A. E. Hassanien, and I. M. Selim, "Deep Galaxy: Classification of Galaxies based on Deep Convolutional Neural Networks", *ArXiv e-prints*, arXiv:1709.02245, arXiv:1709.02245, Sep. 2017. arXiv: 1709.02245 [cs.CV] (cit. on p. 52).

[84] W. Alhassan, A. R. Taylor, and M. Vaccari, "The FIRST Classifier: compact and extended radio galaxy classification using deep Convolutional Neural Networks", *mnras*, vol. 480, pp. 2085–2093, Oct. 2018 (cit. on p. 52).

[85] D. C. Koo, "Overview - Photometric Redshifts: A Perspective from an Old-Timer[!] on their Past, Present, and Potential", in *Photometric Redshifts and the Detection of High Redshift Galaxies*, R. Weymann, L. Storrie-Lombardi, M. Sawicki, and R. Brunner, Eds., ser. Astronomical Society of the Pacific Conference Series, vol. 191, 1999, p. 3. eprint: astro-ph/9907273 (cit. on p. 52).

[86] M. Banerji, F. B. Abdalla, O. Lahav, and H. Lin, "Photometric Redshifts for the Dark Energy Survey and VISTA and Implications for Large Scale Structure", *Mon. Not. Roy. Astron. Soc.*, vol. 386, pp. 1219–1233, 2008. arXiv: 0711.1059 [astro-ph] (cit. on p. 52).

[87] H. Oyaizu, M. Lima, C. E. Cunha, *et al.*, "A Galaxy Photometric Redshift Catalog for the Sloan Digital Sky Survey Data Release 6", *apj*, vol. 674, pp. 768–783, Feb. 2008. arXiv: 0708.0030 (cit. on p. 52).

[88] S. Carliles, T. Budavari, S. Heinis, C. Priebe, and A. Szalay, "Photometric Redshift Estimation on SDSS Data Using Random Forests", *ASP Conf. Ser.*, vol. 394, p. 521, 2008. arXiv: 0711.2477 [astro-ph] (cit. on p. 52).

[89] N. M. Ball, R. J. Brunner, A. D. Myers, *et al.*, "Robust Machine Learning Applied to Astronomical Data Sets. III. Probabilistic Photometric Redshifts for Galaxies and Quasars in the SDSS and GALEX", *apj*, vol. 683, pp. 12–21, Aug. 2008. arXiv: 0804.3413 (cit. on p. 52).

[90] A. D'Isanto and K. L. Polsterer, "Photometric redshift estimation via deep learning. Generalized and pre-classification-less, image based, fully probabilistic redshifts", *aap*, vol. 609, A111, A111, Jan. 2018. arXiv: 1706.02467 [astro-ph.IM] (cit. on p. 52).

[91] C. A. L. Bailer-Jones, "Automated stellar classification for large surveys: a review of methods and results", 2001. arXiv: astro-ph/0102223 [astro-ph] (cit. on p. 53).

[92] R. Romano, C. Aragon, and C. Ding, "Supernova recognition using support vector machines", eng, 2006 (cit. on p. 53).

[93] S. Bailey, C. Aragon, R. Romano, *et al.*, "How to Find More Supernovae with Less Work: Object Classification Techniques for Difference Imaging", *apj*, vol. 665, pp. 1246–1253, Aug. 2007. arXiv: 0705.0493 (cit. on p. 53).

[94] J. W. Richards, D. L. Starr, N. R. Butler, *et al.*, "On machine-learned classification of variable stars with sparse and noisy time-series data", *The Astrophysical Journal*, vol. 733, no. 1, p. 10, 2011 (cit. on p. 53).

[95] J. Newling, M. Varughese, B. Bassett, *et al.*, "Statistical classification techniques for photometric supernova typing", *mnras*, vol. 414, pp. 1987–2004, Jul. 2011. arXiv: 1010.1005 (cit. on p. 53).

[96] L. d. Buisson, N. Sivanandam, B. A. Bassett, and M. Smith, "Machine Learning Classification of SDSS Transient Survey Images", *Mon. Not. Roy. Astron. Soc.*, vol. 454, no. 2, pp. 2026–2038, 2015. arXiv: 1407.4118 [astro-ph.IM] (cit. on p. 53).

[97] S. A. Farrell, T. Murphy, and K. K. Lo, "VizieR Online Data Catalog: Autoclassification of the variable 3XMM sources (Farrell+, 2015)", *VizieR Online Data Catalog*, vol. 181, Feb. 2016 (cit. on p. 53).

[98] A. Mahabal, K. Sheth, F. Gieseke, *et al.*, "Deep-Learnt Classification of Light Curves", *ArXiv e-prints*, Sep. 2017. arXiv: 1709.06257 [astro-ph.IM] (cit. on p. 53).

[99] T. Murphy, S. Chatterjee, D. L. Kaplan, *et al.*, "VAST: An ASKAP Survey for Variables and Slow Transients", *pasa*, vol. 30, e006, e006, Feb. 2013. arXiv: 1207.1528 [astro-ph.IM] (cit. on p. 53).

[100] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?", *ArXiv e-prints*, Sep. 2016. arXiv: 1609.08764 [cs.CV] (cit. on p. 58).

[101] E. A. Revsbech, R. Trotta, and D. A. van Dyk, "STACCATO: a novel solution to supernova photometric classification with biased training sets", *mnras*, vol. 473, pp. 3969–3986, Jan. 2018. arXiv: 1706.03811 [astro-ph.IM] (cit. on p. 58).

[102] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform", in *Wavelets. Time-Frequency Methods and Phase Space*, J.-M. Combes, A. Grossmann, and P. Tchamitchian, Eds., 1989, p. 286 (cit. on p. 58).

[103] G. Narayan *et al.*, "Machine Learning-based Brokers for Real-time Classification of the LSST Alert Stream", *Astrophys. J. Suppl.*, vol. 236, no. 1, p. 9, 2018. arXiv: 1801.07323 [astro-ph.IM] (cit. on p. 58).

[104] *Multidimensional scaling: History, theory, and applications*. Hillsdale, NJ, US: Lawrence Erlbaum Associates, Inc, 1987, pp. xv, 307–xv, 307 (cit. on p. 59).

[105] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005 (cit. on p. 59).

[106] G. J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*. Newark, NJ: Wiley, 2005 (cit. on p. 59).

[107] Z. Zhang, “The Singular Value Decomposition, Applications and Beyond”, *arXiv e-prints*, arXiv:1510.08532, arXiv:1510.08532, Oct. 2015. arXiv: 1510.08532 [cs.LG] (cit. on p. 59).

[108] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993 (cit. on p. 60).

[109] A. J. Stewart, T. Muñoz-Darias, R. P. Fender, and M. Pietka, “On the optical counterparts of radio transients and variables”, 2018. arXiv: 1806.09815 [astro-ph.HE] (cit. on p. 61).

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at http://cleanthesis.der-ric.de/.