Noname manuscript No.
(will be inserted by the editor)

Bayesian Anomaly Detection and Classification

Ethan Roberts · Bruce A. Bassett · Michelle Lochner

Received: date / Accepted: date

Abstract Statistical uncertainties are rarely incorporated in machine learning algorithms, especially for anomaly detection. Here we present the Bayesian Anomaly Detection And Classification (BADAC) formalism, which provides a unified statistical approach to classification and anomaly detection within a hierarchical Bayesian framework. BADAC deals with uncertainties by marginalising over the unknown, true, value of the data. Using simulated data with Gaussian noise, BADAC is shown to be superior to standard algorithms in both classification and anomaly detection performance in the presence of uncertainties, though with significantly increased computational cost. Additionally, BADAC provides well-calibrated classification probabilities, valuable for use in scientific pipelines. We show that BADAC can work in online mode and is fairly robust to model errors, which can be diagnosed through model-selection methods. In addition it can perform unsupervised new class detection and can naturally be extended to search for anomalous subsets of data. BADAC is therefore ideal where computational cost is not a limiting factor and statistical rigour is important. We discuss approximations to speed up BADAC, such as the use of Gaussian processes, and finally introduce a new metric, the Rank-Weighted Score (RWS), that is particularly suited to evaluating the ability of algorithms to detect anomalies.

Keywords machine learning · anomalies · classification · novelty · Bayesian · unsupervised class detection

1 Introduction

In any fully rigorous or scientific analysis, uncertainties must be quantified and propagated through the full analysis pipeline. This is difficult to do with traditional machine learning algorithms that do not

E. Roberts[1,2]
E-mail: rbreth001@myuct.ac.za

B. A. Bassett[1,2,3,4]
E-mail: bruce.a.bassett@gmail.com

M. Lochner[2,3]
E-mail: michelle@aims.ac.za

1. University of Cape Town, Rondebosch, Cape Town, 7700
2. African Institute of Mathematical Sciences, Muizenburg, Cape Town, 7950
3. South African Radio Astronomical Observatory, Observatory, Cape Town, 7295
4. South African Astronimcal Observatory, Observatory, Cape Town, 7295

explicitly take into account uncertainties on the data or features. As machine learning is increasingly given authority for making more important and high-risk decisions, (e.g. in self-driving cars), and with the potential for adversarial attacks (Akhtar and Mian 2018), there is an increasing need for interpretable models and rigorous statistical uncertainties on machine learning predictions.

In classification problems class labels are typically inferred through the use of a separation boundary that is learned from training data (Fawzi et al 2017) and is based on a score combined with a threshold. This threshold is often arbitrary or learned as a hyperparameter to minimise some chosen loss function (Niculescu-Mizil and Caruana 2005a). Any resulting class "probabilities" are systematically distorted in ways unique to the classification algorithm used and are not true probabilities, though in some cases these can be calibrated in a frequentist sense with more training data using e.g. isotonic regression or Platt scaling (Niculescu-Mizil and Caruana 2005b).

However, particularly in the physical sciences, we desire an algorithm that automatically outputs unbiased, accurate probabilities, since knowing the probabilities of an object belonging to various classes is typically more useful than the class label alone. The classification process is often just one step in a multi-stage pipeline, and it is important to propagate class uncertainties through the additional steps in the analysis pipeline. This need is especially true in cases where the true class labels of the training data are noisy or subjective, or the training data are not representative of the test set. An example in astronomy is provided by the photometric classification of type Ia supernovae which are subsequently used for studies of dark energy. Hard label classification leads to contamination from non-Ia supernovae that leads to biases in dark energy properties while fully propagating class probabilities instead allows for unbiased results at the end of the pipeline (Kunz et al 2007; Hlozek et al 2012).

In this context Bayesian methods are ideal (Denison et al 2002), as they have been proven optimal for classification for certain loss metrics, e.g. Domingos and Pazzani (1997), and allow the option of both supervised or unsupervised classification (Cheeseman and Stutz 1996). In the context of astronomy, Bayesian techniques have been applied to classification of transient objects such as supernovae (Connolly and Connolly 2009). A common limitation in the classification of noisy data however, is that the classes in the training data are typically represented by a single template with zero variability (e.g. Sako et al (2011)). This allows straightforward Bayesian methods to be applied but does not apply if there is significant intraclass variability. Ignoring this intraclass variability also makes principled anomaly detection challenging: how unlikely is an example if one doesn't know the underlying distribution within a class?

Here we address these limitations, constructing what we will argue is a natural, statistically robust supervised Bayesian method that can simultaneously be used for both anomaly detection and classification in the presence of measurement uncertainties on all data. Our method works directly with raw data, requiring no feature extraction, and requires minimal assumptions about the nature of the anomalies or classes.

We begin by describing the formalism in section 2. We introduce a new metric optimised for anomaly detection, namely the Rank-Weighted Score (RWS) in section 3 (the other metrics we also use to assess algorithm performance are discussed in appendix C). Finally, we compare algorithm performance against various benchmark algorithms on simulated data in section 4. The full derivation of the formalism is given in appendix A.

2 The BADAC Formalism

Bayesian Anomaly Detection and Classification (BADAC) is a Bayesian hierarchical formalism that provides the probabilities that a new measurement belongs to each of a set of known classes, and also ranks objects by the probability that they are anomalous. We make use of language common to machine learning by referring to *features* for the data for a specific instance of a class, *training data* (i.e. data for which we know the class label τ) and referring to *test data* for data that we wish to classify as either anomalous or belonging to a known class τ.

We start by assuming we have a set of multiple classes, τ, with each class having a set of (noisy) training feature instances $\{y^i_{o,j}\}_\tau$, along with associated uncertainties on the features, typically due to some form of measurement error. Here i indexes the instances/examples in a class, while j indexes the specific features within an instance.

For example, if we have a set of one-dimensional time-series data instances, then j would index time. If our data were MNIST images[1], then j would index pixels, τ would index the integers $\{0, ..., 9\}$, and i would index the examples of each digit. As implied by the above examples, the BADAC formalism we develop will work whether j indexes a continuous underlying variable (e.g. time or space), or is a nominal index with no preferred ordering. The o subscript denotes *observed* data, to distinguish it from the *true* (and unknown) underlying value which may differ because of noise or uncertainties. Allowing for the true and observed values separately will be useful, as we show later. For notational convenience we now suppress the class subscript on the $y^i_{o,j}$ but it should always be assumed. The test data to classify and rank for anomalousness, which we call d, should also be assumed to have measurement uncertainties which need to be included in any analysis.

Since the final formulae are notationally complex, we gain intuition of the general case by considering the simplest possible example. Assume each instance consists of just one feature, and that each class τ has only one instance. This would correspond to taking one point from one instance from each class in the time-series example. This simplification allows us to suppress both the i and j indices (as well as the τ label) and write y_o for the moment. We assume y_o has a known measurement error distribution. Then our final goal is to compute the posterior probability $P(\tau|d, y_o)$, of a single test data point d, with known error distribution, of belonging to a class τ. This is discussed in more detail, along with the general derivation, in appendix A and shown schematically in figure 13. Bayes' theorem gives this posterior probability as:

$$P(\tau|d, y_o) \propto P(d, y_o|\tau)P(\tau) \tag{1}$$

where $P(\tau)$ is the prior probability of belonging to class τ.

If we have measurement errors associated with both d and y_o, we cannot compute $P(\tau|d, y_o)$ with equation 1 as is. Instead we introduce a latent variable y_t which represents the underlying true value, i.e. what would be measured if there were no measurement error. This can then be marginalised over to rigorously handle our uncertainty of the true value y_t. By application of the product rule for probability densities the posterior becomes:

$$\begin{aligned} P(\tau|d, y_o) &\propto P(\tau) \int dy_t P(d, y_o, y_t|\tau) && (2)\\ &\propto P(\tau) \int dy_t P(d, y_o|y_t, \tau)P(y_t|\tau) && (3)\\ &\propto P(\tau) \int dy_t P(d|y_o, y_t, \tau)P(y_o|y_t, \tau)P(y_t|\tau) && (4) \end{aligned}$$

If we assume d and y_o are statistically independent of one another[2], then $P(d|y_o, y_t, \tau) = P(d|y_t, \tau)$. We then arrive at:

$$P(\tau|d, y_o) \propto P(\tau) \int dy_t P(d|y_t, \tau)P(y_o|y_t, \tau)P(y_t|\tau) \tag{5}$$

For Gaussian error distributions this can be solved analytically, as we show below.

[1] http://yann.lecun.com/exdb/mnist/

[2] We consider correlations in appendix B.

Next let us generalise equation 5 to allow each class τ to have multiple training instances, $\{y_o^i\}$, each still consisting of a single scalar feature value. In this case we need to introduce n latent variables, y_t^i, one for each instance in the class. Then the posterior probability that a new test instance d belongs to class τ - assuming the instances in the training data are uncorrelated - is given by (see Appendix A for derivation):

$$P(\tau|d,\{y_o^i\}) \propto P(\tau)\int dy_t^1...dy_t^n \left[\frac{1}{n}\sum_{i=1}^{n} P(d|y_t^i,\tau)\right]\prod_{i=1}^{n} P(y_o^i|y_t^i,\tau)\times\prod_{i=1}^{n} P(y_t^i|\tau) \tag{6}$$

Here $P(d|y_t^i,\tau)$ is the likelihood of observing the data d, conditioned on both the class type and an estimate of the true values of the training data. $P(y_t^i|\tau)$ is a prior on the true value y_t^i given the class τ. Two things are worth noting: first, because of the uncertainties in the training data, the classification of just a single scalar data point requires an n-dimensional integral over the n instances in each class τ[3]. Second, equation 6 is not just a product of terms like equation 4, even if the instances are independent.

We next generalise to consider the case of a multi-dimensional feature vector for each instance, i.e. $\{y_{o,j}^i\}$ for the training data and $\{d_j\}$ for the test data instance, where as usual i, j index the instance and data dimension within an instance respectively (e.g. for time-series data j would index the time). In the special case of uncorrelated Gaussian distributed test and training data, and for (improper) flat priors, we can analytically compute the posterior probability, $P(\tau|\{d_j\},\{y_{o,j}^i\})$ (see Appendix A), yielding our main result:

$$P(\tau|\{d_j\},\{y_{o,j}^i\}) \propto \frac{1}{n}\sum_{i=1}^{n} P(\tau_i)\prod_{j=1}^{m}(2\pi\sigma_{d_j}\sigma_{y_{o,j}^i})^{-1}\left[\frac{\pi}{\frac{1}{2}(\mathit{\Gamma}_d+\mathit{\Gamma}_i)}\right]^{1/2}$$
$$\times\ \exp\left(-\frac{1}{2}\left(\mathit{\Gamma}_d d_j^2+\mathit{\Gamma}_i {y_{o,j}^i}^2-\frac{(\mathit{\Gamma}_d d+\mathit{\Gamma}_i y_{o,j}^i)}{\mathit{\Gamma}_d+\mathit{\Gamma}_i}\right)\right) \tag{7}$$

where $\mathit{\Gamma}_d \equiv \sigma_{d,j}^{-2}$ and $\mathit{\Gamma}_i \equiv \sigma_{y_{o,j}^i}^{-2}$ are the precision of the datapoints, n is the total number of training instances and m is the number of datapoints per instance. We use equation 7 in all our experiments in section 4 to evaluate BADAC, though it is of course not the most general form.

To normalise equation 7 we need to compute the Bayesian evidence for all the observed data, $P(\{d_j\},\{y_{o,j}^i\})$. For m different classes in total, the evidence is then:

$$P(\{d_j\},\{y_{o,j}^i\}) = \sum_{k}^{m} P(\{d_j\},\{y_{o,j}^i\}|\tau_k)P(\tau_k) \tag{8}$$

This is a summation of the posterior probability terms returned for each class type which requires the prior probability, $P(\tau_k)$, for each class, to be supplied by the user as usual. Classification then occurs by choosing the class with the highest normalised posterior probability.

We illustrate BADAC for Bayesian classification in figure 1. Here we show a toy example with training data synthesised into two smooth Gaussian templates (whose 1σ regions are shown by the filled blue and orange areas) corresponding to two classes, $\tau = 0, 1$. We have a single test example which we wish to classify, consisting of two data points (the black triangles with error bars). The middle and right panels show the unnormalised posteriors for the true values of the points, i.e. $P(y_t^i|d,\tau = 0, 1)$, marginalised over the true value of the other point. For the first data point the posterior conditioned on the blue class is pulled upwards and that conditioned on the orange class downwards. However, because the blue

[3] Strictly speaking we should write n_τ since the number of samples in each class will be different but we suppress this to keep the notation relatively simple.

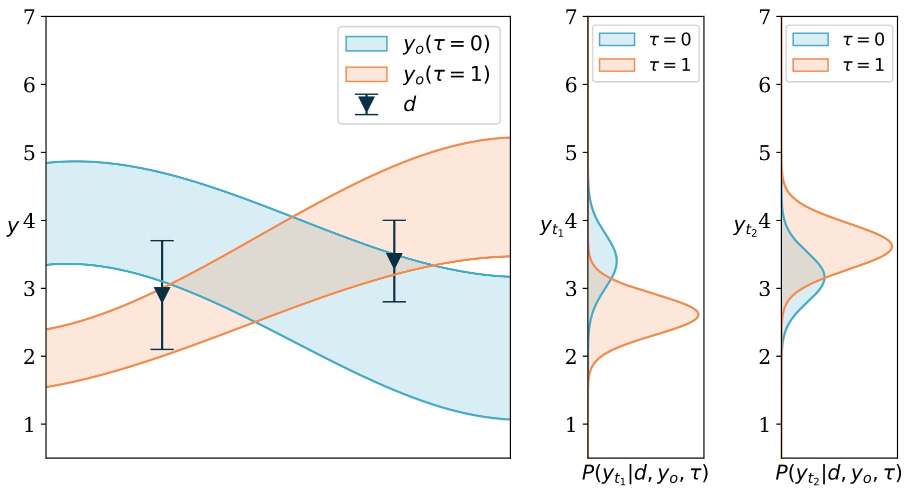

Fig. 1: Schematic representation of BADAC as a classifier. *Left*: a single test example consisting of just two data points (black triangles with error bars). The training data comes from two classes shown schematically as the blue ($\tau = 0$) and orange ($\tau = 1$) 1-σ error envelopes. Which of these two classes does the test data come from? *Middle and Right*: panels showing the unnormalised posterior probability for the true value, y_t^i, for the first (middle panel) and second (right panel) data point, marginalised over the true value of the other point and conditioned on belonging to either class (class 0 - blue or class 1 - orange). The relative area of the corresponding Gaussians in the middle and right panels gives the probability for the data to belong to either class. As can be seen, the data is more likely to come from class 1 (the orange class), in this case with a probability of 73%.

error envelope at the first data point is broader than the orange one, the first point is less likely to come from the blue class, a fact reflected in the smaller area under the blue Gaussian in the first of the right narrow panels. The same happens for the 2nd point. The Bayesian evidence therefore predicts that the test example is more likely to come from the $\tau = 1$ (orange) class, with probability 65%.

2.1 Anomaly Detection

Since we want a simultaneous anomaly detection and classification algorithm we assign the m-th class to the unknown anomalies with the known classes assigned to the remaining $m-1$ τ labels. A key issue is the choice of the likelihood for the anomaly class: $P(\{d_j\}, \{y_{o,j}^i\}|\tau = m)$. If we have seen examples of the anomaly before, this can be estimated from the data. If the anomaly has never been seen before however, this reduces essentially to a prior about our expectations regarding the anomaly. Broad, normalised Gaussian or top-hat functions are simple choices allowing for a wide range of behaviours for the anomalies. The shadow of the No Free Lunch theorem appears here: if our prior about the anomaly's behaviour is completely wrong, we may miss the anomaly entirely.

Instead of specifying a generic prior for the anomaly likelihood, an alternative approach is to assume only the known $m-1$ classes, but then to rank instances in the test data from smallest to largest values of

$$\sum_{k}^{m-1} P(\{d_j\}, \{y^i_{o,j}\}|\tau_k)P(\tau_k)\,, \tag{9}$$

i.e. to rank the instances from least to most likely to belong to any of the $m-1$ known classes. This is useful if one has some insight into the expected fraction of anomalous events, since one can then use it to determine a cutoff in this ranked list.

2.2 Comparison with kNN and KDE

It is interesting to compare BADAC to the k-nearest neighbours (kNN) algorithm (Altman 1992), with the best value of k selected on a validation set. Since each object can be considered as a single point in a high dimensional feature space, in a sense BADAC generalises kNN to allow for error bars on the test and training points, and uses all the training data from each class. BADAC is also similar to the Kernel Density Estimation (KDE) approach where classification is performed by putting a kernel on each training data point and summing the resulting smooth functions. In our case the bandwidth of the kernel is given by the error bars on the data while it also handles errors on the test data. BADAC can then be thought of as a natural extension of both kNN and KDE to a more rigorous statistical setting.

2.3 Non-Gaussian data

Often, the standard deviation is used as a proxy for the error distribution on an observation, even when the distribution is non-Gaussian. In section 4 we test how the algorithm developed in section 2 performs while assuming a Gaussian error distribution, even when the error distribution is non-Gaussian. However, if the error distribution is known, the forms of $P(d|y^i_t,\tau)$ and $P(y^i_o|y^i_t,\tau)$ in equation 6, can be replaced with the known non-Gaussian distribution. These could be the binomial distribution in the case of count data, or the Poisson distribution in the case of certain time series. Any appropriate distribution that can be modelled can be used in this formalism. In the case of the binomial distribution, one would do a summation rather than an integration over the latent variables. For any distribution that doesn't yield an analytically integrable form for $P(\tau|d, y^1_o, ..., y^n_o)$, one can do the marginalisation numerically, though at increased computational cost.

2.4 Online Learning of new classes

Once we have confirmed that an anomaly represents a new class (i.e., if the Bayesian evidence for the anomaly class is higher than that of any of the existing classes) it can automatically be added to the training data as a new class (with a single example) to be compared with. This provides an online learning version of the BADAC algorithm. Any future data belonging to the new anomaly class will be automatically assigned the new anomaly class label.

This process in no way limits us to a single new anomaly class. The BADAC formalism allows for the automatic addition of new classes as demanded by the data. If a new kind of anomaly is different from any previously identified anomalies it will automatically be assigned to a new class.

2.5 BADAC with Parametric Models

It is often the case that the training data can be efficiently modeled by some general parametrisation (with class-dependent parameters θ_τ), either because we have a physical model of the classes or have performed a general expansion on each class, e.g. through a suitable basis set such as wavelets (Mallat 2009) or principal component analysis (Hotelling 1933). As a result we assume here that we can parametrise our data d as $y(\theta_\tau, x)$. Then we can express the classification probability as:

$$P(\tau|d) \propto \int_{\theta_\tau} d\theta_\tau P(d, \theta_\tau|\tau)P(\tau) \tag{10}$$

$$P(\tau|d) \propto \int_{\theta_\tau} d\theta_\tau P(d|\theta_\tau, \tau)P(\theta_\tau|\tau)P(\tau) \tag{11}$$

Here $P(\theta_\tau|\tau)$ is the prior on the parametrised variables θ_τ, for a given class τ. $P(d|\theta_\tau, \tau)$ is the likelihood for given parameters θ_τ, given data d. For models derived from physical phenomena, we can usually evaluate this.

If one is using some general parametrisation, such as a wavelet transform, then the parameters θ_τ can be thought of as similar to (and hence grouped with) y_o. In this case, if there were measurement uncertainties associated with each θ_τ, we could then proceed in the same way as in equation 6, where instead of marginalising over a large range of possible y_t values, we now marginalise over the more constrained θ_t. This could be extended to the hierarchical context where the parameters θ_t are drawn from some parent distribution.

This shift from the raw training data, to a parametrised model of the data, can have significant computational advantages when there is a lot of training data, as we discuss further in section 6.3.

3 Rank-Weighted Score

In this section we introduce a new anomaly-sensitive metric that we call the Rank-Weighted Score (RWS). In addition to being insensitive to class imbalance, this metric is sensitive to the relative *ranking* of anomalous objects. In many cases there is a clear hierarchy of how interesting anomalous objects are (a new class is generally more interesting than a new subclass for example), and hence we prefer algorithms that correctly rank more anomalous objects more highly[4]. This is natural because following up and investigating potential anomalies typically consumes resources (whether human or instrumental) and false positives, at any anomaly threshold, must be minimised.

The RWS is defined by ranking the N objects according to their degree of anomalousness (from high to low) as identified by an algorithm. Here N is a user-supplied integer (the expected number of anomalies in the dataset). The RWS score is then computed as the weighted sum:

$$S_{\mathrm{RWS}} = \frac{1}{S_0}\sum_{i=1}^{N} w_i I_i \tag{12}$$

where:

$$w_i = (N + 1 - i) \tag{13}$$

Note that this gives (linearly) more weight to correctly identifying anomalies at the top of the ranks (with low values of i) compared to lower down the list. In equation 12, I_i is an indicator variable: $I_i = 1$

[4] Compare for example the Spearman's rank correlation (Spearman 1904) which does not give higher weight to the top-ranking objects.

if the i-th object is an outlier, and $I_i = 0$ otherwise. S_0 is a normalisation factor: $S_0 = \frac{N}{2}(N+1)$. This means the RWS score has a possible range of $[0,1]$, where 0 implies that no true outliers were found in the N most anomalous objects ranked by the algorithm, while an RWS score of 1 would mean that all N most anomalous objects identified by the algorithm were in fact outliers. The value of N must be chosen on a per problem basis, and be kept consistent across the various algorithms being considered to allow fair comparison. In section 4 we use this metric along with several other commonly used metrics to gauge algorithm performance. We discuss these metrics in Appendix C.

4 Evaluation of BADAC: Gaussian Case

To illustrate and test the performance of BADAC, we simulate a number of one-dimensional datasets and compare results with multiple metrics including the Rank-Weighted Score (RWS) introduced in section 3, that is optimised for anomaly detection.

4.1 Simulations

We simulate data from arbitrary mathematical functions. We use two mathematical functions to build two "normal" classes and use three other functions as anomalies. Each function has parameters which, when generating the data, are randomly drawn from a Gaussian distribution. The class functions and their corresponding parameter distributions are given in table 1.

Table 1: Description of the functions used to create the simulated data. 99% of the test objects in the dataset are of the type "inlier" and 1% are "outliers". Each class has the corresonding functional form with parameters drawn randomly for each instance from Gaussian distributions with hyperparameters specified in the table.

Class label	Type	Functional Form	Parameter distributions
0	Inlier	$y = \sin(\omega \text{x})$	$\omega \sim \mathcal{N}(5,2)$
1	Inlier	$y = \alpha x^2 + \beta x + \gamma$	$\alpha \sim \mathcal{N}(0.5,0.2)$ $\beta \sim \mathcal{N}(0.5,0.2)$ $\gamma \sim \mathcal{N}(0,0.2)$
2	Outlier	$y = h$ if $x \leq x_0$, else $y = 0$	$h \sim \mathcal{N}(1,0.3)$ $x_0 \sim \mathcal{N}(0.5,0.2)$
3	Outlier	$y = A \exp\left(-\left(\frac{x-\mu}{w}\right)^2\right)$	$A \sim \mathcal{N}(0.5,0.2)$ $\mu \sim \mathcal{N}(0.1,0.05)$ $w \sim \mathcal{N}(1,0.5)$
4	Outlier	$y = 0.2\Big(\sin(\omega_1 x) + \sin(\omega_2 x) +$ $\sin(\omega_3 x) + \sin(\omega_4 x) + \sin(\omega_5 x)\Big)$	$\omega_i \sim \mathcal{N}(30,20)$

For each experiment, we generate 15000 curves of roughly equal number of objects from class 0 and 1 as training data. In the test data, we add 1% outliers from classes 2, 3 and 4. Figure 2 illustrates some randomly drawn objects from the training and test sets (equal numbers from each class).

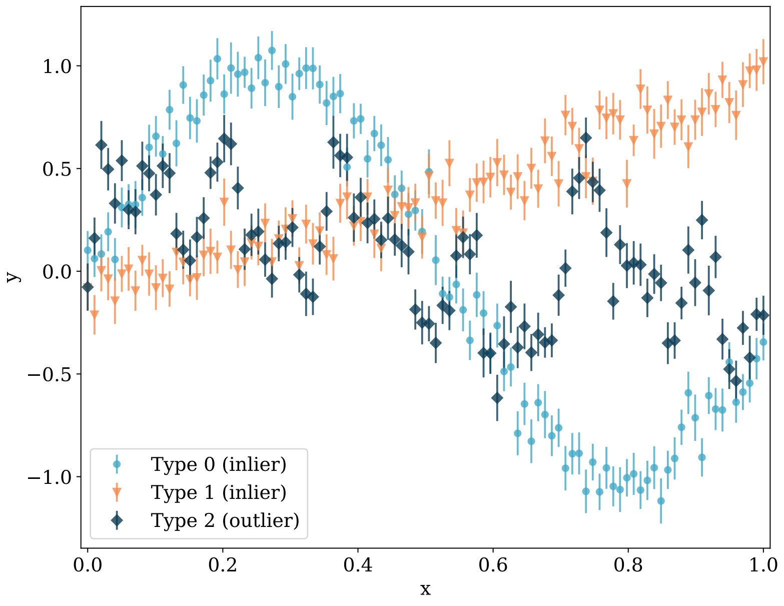

Fig. 2: Illustrations of example objects from the simulated data. The plotted error bars correspond to 1σ error of Gaussian noise. The functional form and distribution of hyperparameters used to generate these examples is shown in in table 1. The points are coloured by true type, where light blue circles correspond to a type 0 object, orange triangles to type 1 and dark indigo diamonds is an outlier. Only type 0 and type 1 curves are used during the training phase.

4.1.1 Experiment 1: Gaussian errors

We use the framework of section 4.1 to create a variety of experiments to test our anomaly detection and classification algorithm. Here we simulate the data as described in section 4.1 with uncorrelated Gaussian errors on all data points. The standard deviation of the underlying noise distribution depends on the class, and is given by: $\sigma_0 = \sigma_2 = \sigma_3 = \sigma_4 = 0.3$ and $\sigma_1 = 0.5$. This experiment is the ideal case in which the noise distribution used for generating the simulated data is the same as that in the mathematical formulation of equation 7.

4.1.2 Experiment 2: Compact Anomalies

In this experiment, we test BADAC's ability to detect curves with a compact anomaly embedded somewhere in them. We use one of the base inlier classes described in experiment 1, the sine curve (class 0), and place on top of it a narrow Gaussian. We randomly draw the parameters of the sine curve from the distribution described in table 1 and draw the parameters of the compact anomalies as described in table 2. An example of a compact anomaly is shown in figure 3.

Table 2: Description of the functions used to create the compact anomaly simulated data. 99% of the test objects in the dataset are of the type "inlier" which are the same as class 0 and 1 in table 1. The remaining 1% are drawn from one of two compact anomaly classes. These are narrow Gaussians added to a randomly generated function of class 0. The parameters of the Gaussian are drawn randomly for each object from a distribution with hyperparameters as specified in the table.

Class label	Type	Functional Form	Parameter distributions
0	Inlier	$y = \sin(\omega \mathrm{x})$	$\omega \sim \mathcal{N}(5, 2)$
1	Inlier	$y = \alpha x^2 + \beta x + \gamma$	$\alpha \sim \mathcal{N}(0.5, 0.2)$ $\beta \sim \mathcal{N}(0.5, 0.2)$ $\gamma \sim \mathcal{N}(0, 0.2)$
2	Outlier	$y = \sin(\omega \mathrm{x}) + \mathrm{A}\ \exp\left(-\left(\frac{\mathrm{x}-\mu}{\mathrm{w}}\right)^2\right)$	$\omega \sim \mathcal{N}(5, 2)$ $A \sim \mathcal{N}(1.5, 0.5)$ $\mu \sim \mathcal{U}(0, 1)$ $w \sim \mathcal{N}(0.03, 0.01)$
3	Outlier	$y = \sin(\omega \mathrm{x}) + \mathrm{A}\ \exp\left(-\left(\frac{\mathrm{x}-\mu}{\mathrm{w}}\right)^2\right)$	$\omega \sim \mathcal{N}(5, 2)$ $A \sim \mathcal{N}(-1.5, 0.5)$ $\mu \sim \mathcal{U}(0, 1)$ $w \sim \mathcal{N}(0.03, 0.01)$

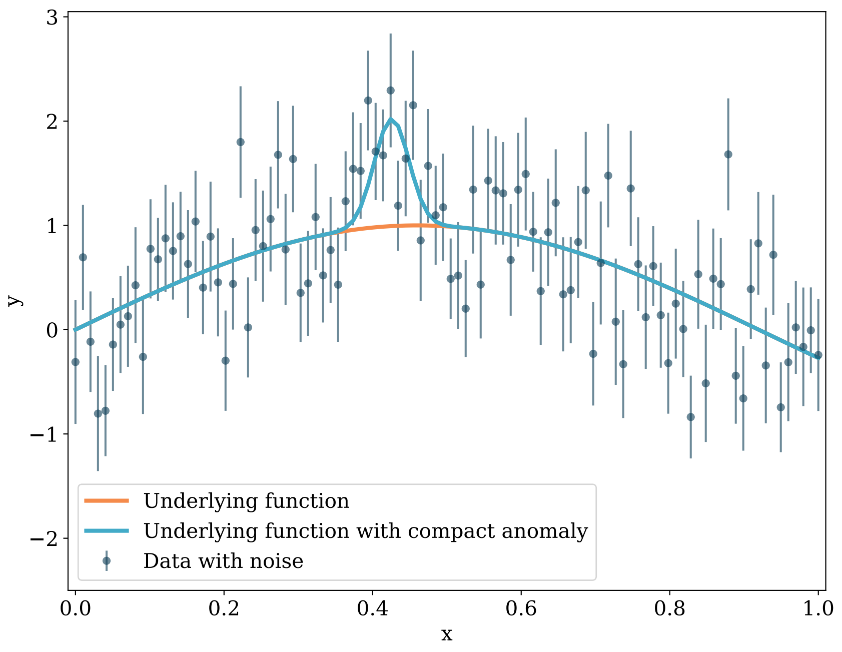

Fig. 3: Example of the compact anomaly simulations. The underlying function from which the data were generated is shown as an orange solid line. The underlying function with the compact anomaly superposed is shown as the light blue solid line. The final data with noise are shown by the dark indigo scatter where the errorbars represent the 1σ Gaussian measurement error.

4.2 Comparison of algorithm performance

We assess the performance of our algorithm on the simulated data discussed in section 4.1. We then compare our algorithm to a series of benchmark algorithms, namely IsolationForest (Liu et al 2008) and Local Outlier Factor (LOF) (Breunig et al 2000) for anomaly detection, and random forests (Breiman and Schapire 2001) for classification.

We use `sklearn` (Pedregosa et al 2011) implementations for all of the benchmark algorithms we compare against BADAC. For anomaly detection, all algorithms receive only the input training data, and the percentage of outliers of 1%. For classification with random forests, we set the input parameter `n_estimators=1000`. There are unsupervised implementations of IsolationForest and LOF, but here we consider the supervised methods only.

Table 3: Anomaly detection challenge results for the Gaussian noise simulations and Gaussian noise compact anomalies (Comp. Anom.) for the three metrics (MCC, AUC and RWS) discussed in section C. The best performer is shown in bold. Note the particularly poor performance of IsolationForest in the MCC and RWS metrics. BADAC significantly outperforms the other algorithms in the Gaussian case.

	BADAC			IsolationForest			LOF		
	MCC	AUC	RWS	MCC	AUC	RWS	MCC	AUC	RWS
Gaussian	**0.95**	**0.99**	**0.99**	0.00	0.89	0.02	0.83	0.97	0.96
Comp. Anom.	0.41	**0.91**	0.59	0.11	0.80	0.14	**0.44**	0.90	**0.63**

Table 4: Comparison of BADAC's *classification* performance to that of random forests using average accuracy across both inlier classes.

	BADAC	Random Forests
Gaussian Noise	**99.02**	98.66
Compact Anomalies	**95.51**	95.18

4.2.1 Gaussian noise

Here we illustrate the performance of BADAC, as well as the benchmark algorithms, on the data discussed in section 4.1.1 with Gaussian measurement error. We use the formalism shown in section 2 to provide two probabilities, P_0 and P_1, which are the un-normalised probabilities of belonging to class 0 and class

1 respectively. These probabilities are plotted in figure 4.

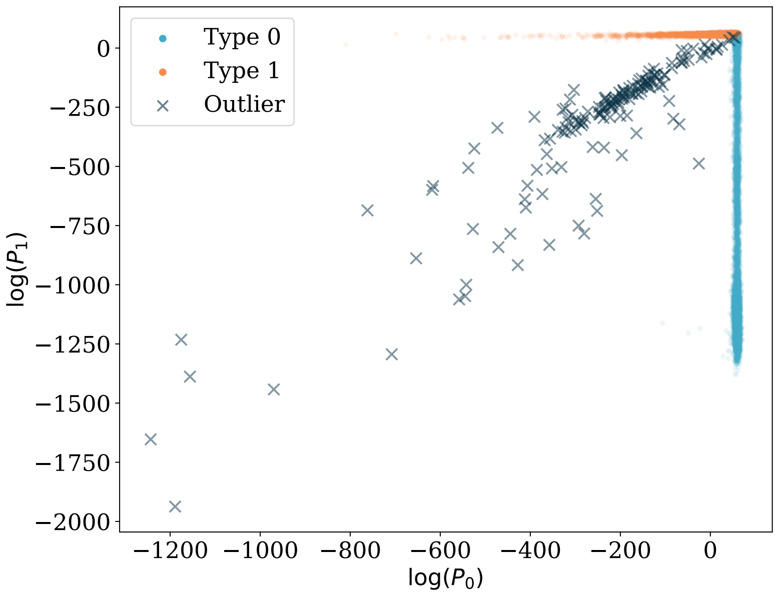

Fig. 4: Scatter plot showing the computed log-probabilities for the test data discussed in section 4.1.1. Each point corresponds to a test object, which is shown in the log(P_0)-log(P_1) space. Points that appear high on the y-axis have a high likelihood of being type 1. Points that appear higher (to the right) on the x-axis have a high likelihood of being type 0. The points are coloured by true type, where light blue corresponds to type 0, orange is type 1 and the dark crosses are all outliers.

Plotting the unnormalised probabilities is useful for visualising the decision boundary that separates both the known classes and anomalies. It also does not require us to make any assumptions about the nature of the anomalies we expect to see. However, to make use of these probabilities in an analysis pipeline, they must be normalised. In order to normalise the probabilities, we compute the Bayesian evidence. The evidence in the case where one is interested in classification only would be $P_0 + P_1$. In the case where anomaly detection is of interest as well, the evidence is $P_0 + P_1 + P_{anomaly}$, where in this case, we choose to evaluate $P_{anomaly}$ using a top-hat likelihood equal to $1/(b - a)$ over the range of $[a, b]$, and equal to 0 otherwise. Here we choose a and b to cover twice the observed range of the input data.

If we bin the normalised probabilities for a single class, we can measure whether or not they are calibrated. It is a well known problem that many machine learning algorithms give uncalibrated probabilities that do not correspond to the true probability of an object belonging to a certain class. The reliability of probabilities can be investigated by plotting a probability calibration curve: the output probabilities from the algorithm for a selected class only are binned and compared with the actual fraction of objects in that bin belonging to the class. We show this result for classification only for type 1 objects in figure 5, and compare the results of BADAC with those of random forests.

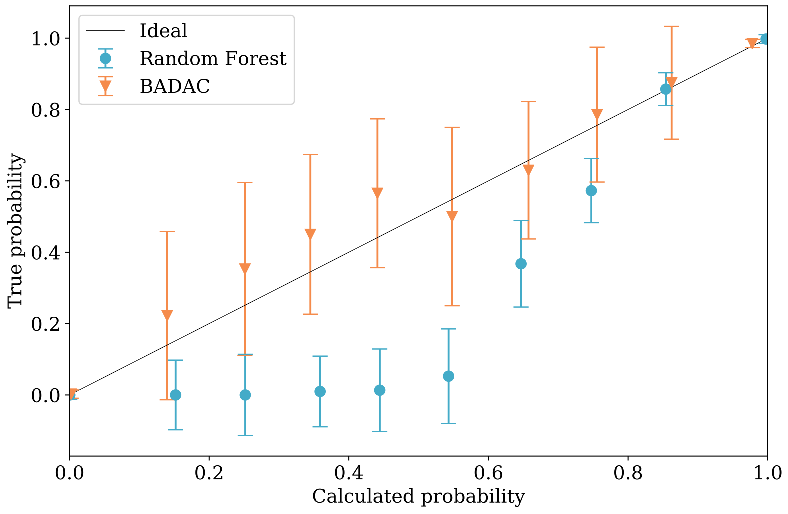

Fig. 5: Probability calibration curve for the Gaussian case for BADAC and random forests (for classification only). Perfectly calibrated probabilities would lie on the line $y = x$. Here we consider the probability of an algorithm classifying an object as type 1. All objects within a particular probability range are binned, and the fraction of correct positive predictions plotted. The errorbars show the Poisson uncertainties given by the number of objects in each bin, and the x-coordinate for each bin is given by the mean calculated probability for that bin. Random forest gives poorly calibrated probabilities while BADAC automatically returns well-calibrated probabilities. This is to be expected since the model we use accounts for the Gaussian noise in the data and follows a principled Bayesian approach.

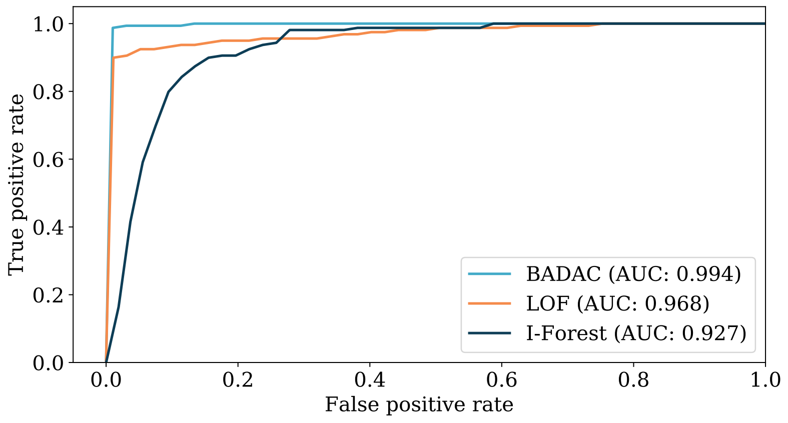

Fig. 6: ROC curves for BADAC, LOF and IsolationForest on the dataset with uncorrelated Gaussian error, for anomaly detection. BADAC performs best under the AUC metric shown in each legend. The best classification algorithms have a ROC curve that reaches close to the top left hand corner, with perfect performance corresponding to an AUC of one.

We show the ROC curves (see Appendix C for a description of ROC curves) for BADAC as well as LOF and IsolationForest in figure 6 in order to compare algorithm performance in anomaly detection. A summary of algorithm performance from all algorithms on all the datasets we consider in both anomaly detection and classification is shown in tables 3 and 4.

4.2.2 Compact anomaly Performance

In this section we illustrate the performance of BADAC as well as the benchmark algorithms we consider on the compact anomaly data discussed in section 4.1.2. It should be noted that the compact anomaly data is generated with Gaussian noise, which is the type of noise we assume in this implementation of our formalism, and is also the same kind of noise as the data described in section 4.1.1. This means we would expect the algorithms to have similar performance in *classification* only in this section as in section 4.2.1. For this reason, we don't discuss classification performance of any of the algorithms on the compact anomaly dataset. We proceed in the exact same manner as we did in section 4.2.1, except here we test how robust the algorithms are to different types of anomalies (compact ones).

The importance of an algorithm being able to detect compact anomalies is twofold. Firstly, compact anomalies are often interesting in science when one wishes to measure or detect aberrant behaviour of known sources. Secondly, an algorithm's ability to detect compact anomalies demonstrates its overall sensitivity in measuring small variations within data.

The probabilities, P_0 and P_1, generated by the formalism we discuss in section 2, are shown in figure 7.

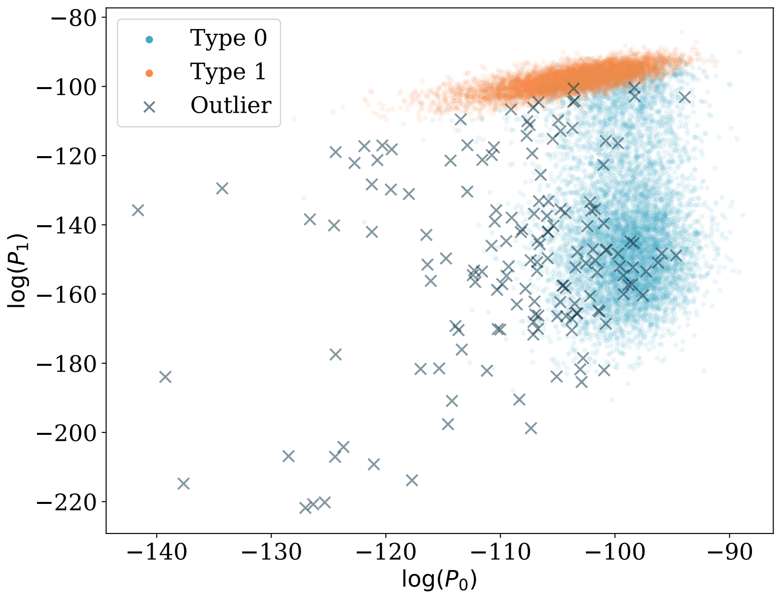

Fig. 7: Scatter plot showing the computed log-probabilities for the test data discussed in section 4.1.2. Each point corresponds to a test object, which is shown in the log(P_0)-log(P_1) space. Points that appear high on the y-axis have a high likelihood of being type 1. Points that appear higher (to the right) on the x-axis have a high likelihood of being type 0. The points are coloured by true type, where light blue corresponds to type 0, orange is type 1 and the dark crosses are outliers.

As we can see from figure 7, the outlier data has significant overlap with type 0 data. This is because we create compact anomalies on top of type 0 data only. The varying scale/amplitude to the anomaly is responsible for where the outlier data is positioned on the $\log(P_0)$-axis (further left indicates a more anomalous object). Outlier points with high $\log(P_0)$ values are likely associated with compact anomalies with very low amplitudes.

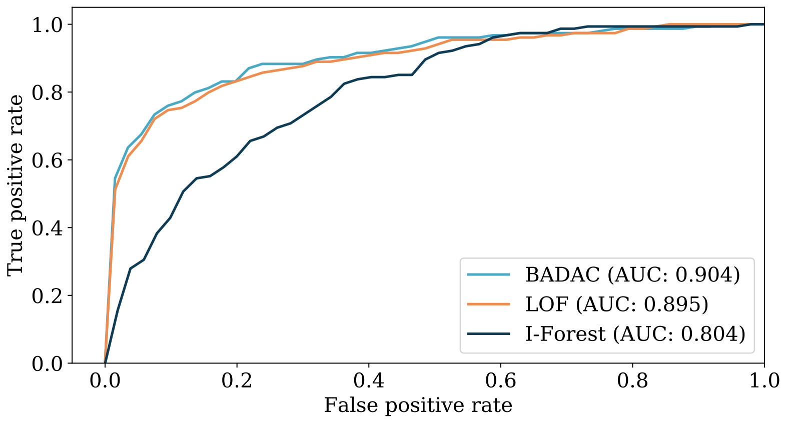

Fig. 8: ROC curves for anomaly detection with BADAC, LOF and IsolationForest on the dataset with compact anomalies. BADAC performs best under the AUC metric, whose values in each case are shown in the legend.

We show the ROC curves for BADAC as well as LOF and IsolationForest in figure 8 in order to compare algorithm performance in anomaly detection. Under the AUC metric, BADAC performs the best in this case. LOF is almost as good, and actually performs better under the MCC and RWS metrics. A summary of algorithm performance from all algorithms on all the datasets we consider in both anomaly detection and classification is shown in tables 3 and 4.

4.3 Computational performance

It is difficult to give a “fair” comparison of computational performance between BADAC, random forests, LOF and IsolationForest, since unlike the benchmark algorithms we compare it with, our algorithm has no distinct training and testing phases. This means that these algorithms scale very differently (depending on amount of training and test data available). For example, for a dataset with m training and n test examples, the computational time required for random forests, IsolationForest and LOF would increase as $f(m) + f(n)$. For our algorithm, the computational time required increases as $f(n \times m)$. In fact the computational time increases linearly as a function of $n \times m$. For an even comparison of computational performance, we have compared the same number of training and testing samples as were used in section 4.2 (15000 training samples and 15000 test samples). We quote the total time (training time + testing time) in table 5. We note, however, that there is ample room for optimisation and parallelisation in our BADAC code and the timings could be considerably improved.

As is evident in table 5, BADAC has a computational cost of around an order of magnitude more than any of the competing algorithms we considered. We discuss ways of mitigating this in section 6.3.

Table 5: Comparison of the computational performance between the three algorithms we compare in section 4. All measurements were made on the dataset used in experiment 1 (Gaussian noise) with 15000 training and 15000 test curves. There are no values shown for testing and training times for BADAC, since there are no distinct training and testing phases. Measurements were made on a 2.9GHz processor, where each algorithm was limited to use a single core.

Algorithm	Training time (s)	Testing time (s)	Total time (s)
Random Forests	96.30	2.94	99.24
IsolationForest	1.62	1.21	2.83
Local Outlier Factor	13.21	27.25	40.46
BADAC	-	-	1281.82

5 Breaking BADAC

So far, i.e. in the case of Gaussian errors, BADAC has outperformed random forests, LOF and Isolation-Forest under most metrics we consider. This is perhaps not surprising since BADAC was designed to use the extra information available, namely that there are uncorrelated errors on the data that are Gaussian distributed. Here we try a series of more challenging tests where we use the uncorrelated Gaussian BADAC formalism, but test it on data that do not obey this model.

5.1 Experiment 3: Non-Gaussian errors

Here we simulate the data exactly as in Experiment 1, however we use non-Gaussian errors instead of the Gaussian errors of Experiment 1. For 80% of the y values (randomly selected) of any given simulated object, the noise is drawn from a Gaussian distribution with standard deviation as described in section 4.1, meaning the scatter matches the error bar. However, for the remaining 20% of the values, the noise is drawn from a Gaussian distribution of five times the width, resulting in scatter dramatically underestimated by the reported error bar.

5.2 Experiment 4: Correlated Gaussian Noise

To test the sensitivity of our algorithm to the uncorrelated noise assumption, we generate correlated Gaussian data. We choose to only correlate Class 0, according to a “wedding cake” covariance matrix (based on Kim and Linder (2011) and Knights et al (2013)):

$$C_{ij} = \sigma_i \sigma_j \delta_{ij} + V_{ij}, \tag{14}$$

where

$$V_{ij} = \sum_{k=1}^{n_{i,j}} s_k \tag{15}$$

where i and j are indices of the data (in order of x value) and $n_{i,j}$ is the bin to which the object belongs. To produce the step-like structure, $n_{i,j} = \lfloor \frac{min(i,j)}{N/5} \rfloor + 1$ (where “$\lfloor\rfloor$” indicates the floor function, rounding down to the nearest integer). We use $s_k = 0.1$ for each k in this work. The result is that the data are correlated in such a way that the points at higher x-values are more correlated than the lower ones.

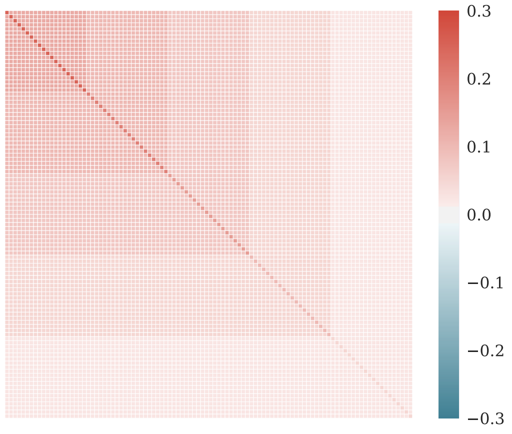

Fig. 9: The covariance matrix used for correlating the class 0 data for experiment 3. This is a “wedding cake” covariance matrix, the form of which is shown in equation 14. The data are ordered by x-value starting at the top left corner (so values near the beginning of a given curve would be more highly correlated than those near the end). Class 1 and anomaly data remain uncorrelated.

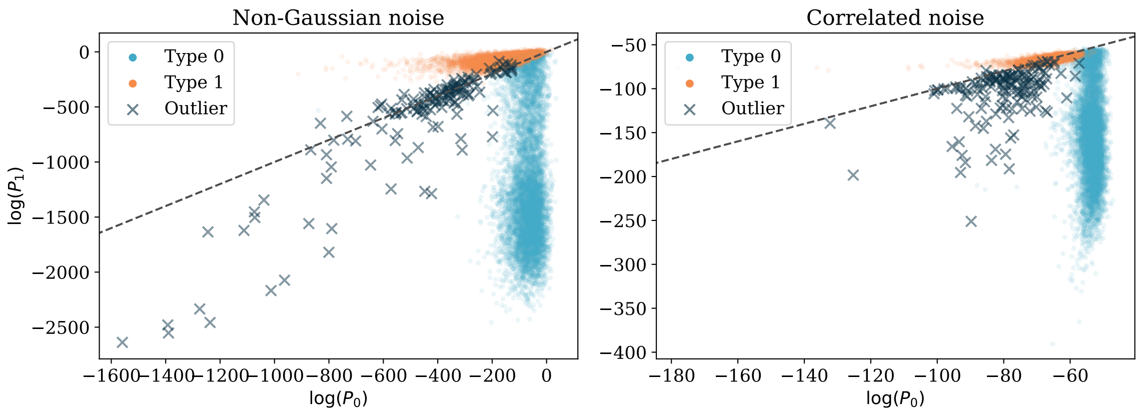

Fig. 10: Probability scatter plot for the dataset with non-Gaussian noise (left panel), and the dataset with correlated Gaussian noise (right panel). Each point corresponds to a test curve, which is shown in the log(P0) − log(P1) space. The line $y = x$ has been added to each plot to highlight the bias introduced by using the wrong model for the noise with BADAC. Here the bias is only visible in the correlated noise case since only class 0 was correlated.

5.3 Results for non-Gaussian noise and correlated Gaussian noise

Here we present the results for both classification and anomaly detection for the data discussed in sections 5.1 and 5.2 with both non-Gaussian and correlated Gaussian noise. It should be noted that we still use equation 7 to determine classification/anomaly detection probabilities, despite the fact that the data does not have Gaussian uncorrelated noise as equation 7 assumes. Thus we must expect BADAC performance to decrease; the question is how much?

In order to normalise the probabilities shown in figure 10, we compute the evidence, $P(\{d_j\}, \{y^i_{o,j}\}) = P_0 + P_1 + P_{anomaly}$ once again. As before, we choose to evaluate $P_{anomaly}$ using a top-hat likelihood equal to $1/(b-a)$ over the range of $[a, b]$, and equal to 0 otherwise. In this case, naively choosing the width of the top-hat does not work, as the model used to determine P_0 and P_1 is incorrect, and hence returns low probabilities. As a result, $P_{anomaly}$ is a much higher probability than P_0 and P_1, even for inlier data, when the incorrect model for the noise is used. To get around this we determine the height of the top-hat likelihood, $1/(b-a)$, by equating it to $P_0 + P_1$ computed for the object corresponding the 99th percentile. What this means is, we enforce that the algorithm labels the most anomalous 1% of objects (as determined by the algorithm) as outliers. This is still a fair comparison with the benchmark algorithms, as both IsolationForest and LOF receive the percentage contamination of 1% as an input parameter. We discuss how to extend this method to be suitable for modelling data with different types of noise in section 2.3.

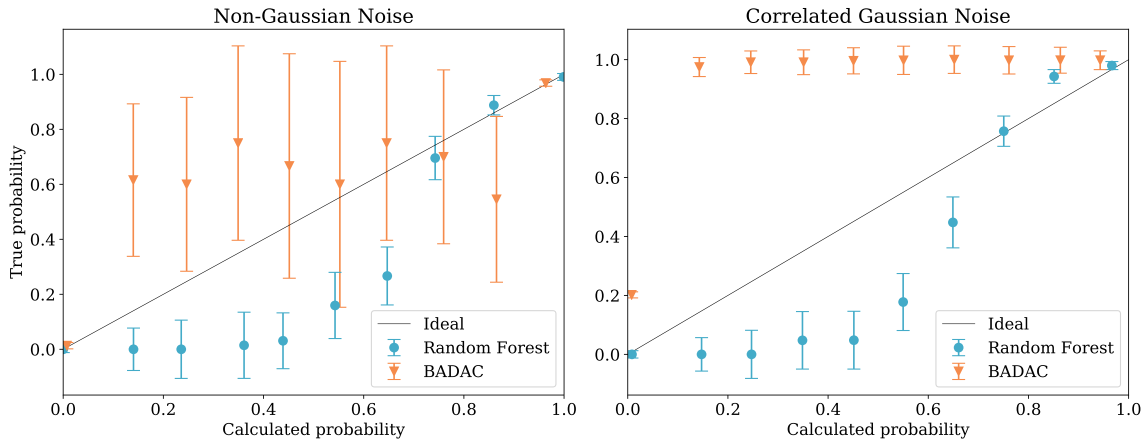

Fig. 11: Probability calibration curves showing the degree to which the probabilities returned by each algorithm (in classification only) are calibrated for the non-Gaussian case we consider (left panel) and the correlated Gaussian case (right panel) respectively. Perfectly calibrated probabilities would lie on the line $y = x$. Here we consider the probability of an algorithm classifying an object as type 1. All objects within a particular probability range are binned, and the fraction of correct predictions plotted. The errorbars show the Poisson uncertainties given by the number of object in each bin. While non-Gaussian noise does not distort the probabilities dramatically, correlated noise has a strong effect due to a fundamentally incorrect noise model assumption.

As we can see from the scatter of classification probabilities shown in figure 10, there is more overlap of different object types present than in the uncorrelated Gaussian noise case we considered. Additionally, in the correlated case, there is a significant bias introduced due to the noise from only one of the classes being correlated. Since the model does not favour fitting this class, the classification probabilities are not reliable. This is illustrated both by figure 10, where the diagonal dashed line shows where type 0 and type 1 clusters should be separated, and figure 11, where we can see that the classification probabilities are far from calibrated. This is due to the fact that we use a uncorrelated Gaussian model for the noise, despite the fact this model is wrong.

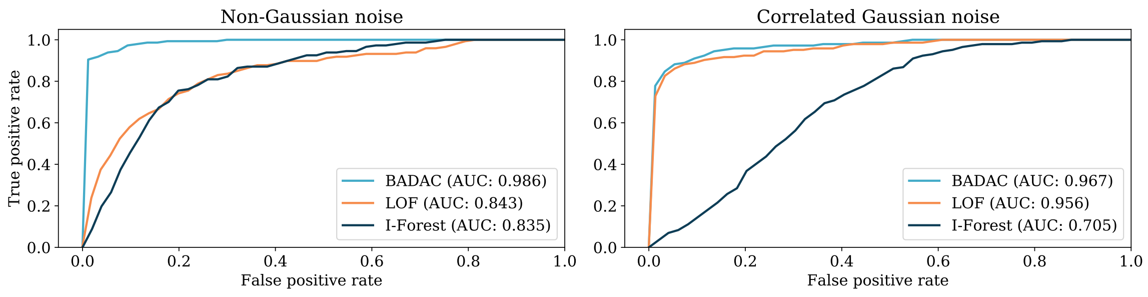

Fig. 12: ROC curves for anomaly detection with BADAC, LOF and IsolationForest on the dataset with non-Gaussian error (left pane), and the dataset with correlated Gaussian error (right pane). BADAC performs best in both cases under the AUC metric (values shown in the legend).

We show the ROC curves for BADAC as well as LOF and IsolationForest in figure 12 in order to gauge performance in anomaly detection. In these two cases, it is surprising BADAC performs best, since we don't correctly model the noise. Random forests however achieves a higher accuracy in classification in these two cases. A summary of algorithm performance from all algorithms on all the datasets we consider in both anomaly detection and classification is shown in tables 3 and 4.

Table 6: Results for anomaly detection only: Non-Gaussian and correlated Gaussian noise. BADAC shows the best performance in both experiments, showing some robustness to incorrectly choosing the model of the noise. In the non-Gaussian case both IsolationForest and LOF perform poorly in terms of MCC and RWS due to the wide tails allowing for large noise fluctuations.

	BADAC			IsolationForest			LOF		
	MCC	AUC	RWS	MCC	AUC	RWS	MCC	AUC	RWS
Non-Gauss.	**0.84**	**0.99**	**0.96**	0.06	0.84	0.10	0.16	0.84	0.18
Corr. noise	**0.68**	**0.97**	**0.84**	0.01	0.70	0.03	0.61	0.96	0.76

Table 7: We compare BADAC average accuracy for *classification* of all classes to that of Random forests. BADAC performs reasonably in the case of non-Gaussian noise but fairly poorly on the correlated noise case, due to the incorrect model assumption in BADAC, while random forests is more robust as it can learn a model from the training data, while BADAC insists on interpreting the fluctuations as coming from an uncorrelated Gaussian distribution. This relatively poor performance of BADAC can be rectified by using, or learning, the right noise model.

	BADAC	Random forests
Non-Gaussian noise	97.71	**98.14**
Correlated Gaussian noise	68.88	**96.72**

6 Extensions

6.1 Learning Subclasses

The BADAC algorithm we have presented can classify and identify anomalies. It can also add new classes as needed when run in online learning mode, as discussed in section 2.4. The following example illustrates how BADAC can potentially identify subclasses of existing classes. But first, what do we mean by a subclass? Here a subclass corresponds to a large intrinsic variation in a class inconsistent with measurement errors.

In equation 7 we explicitly allowed each instance of a class τ to have its own true value, $y^i_{t,j}$, for each feature j. However a very homogeneous class will not require this flexibility, and will only require one latent variable for each feature j. Hence, we can define the number of subclasses to be the number of different latent variables per feature j, required by the data in class τ.

Let us define a hierarchy of models, $\mathcal{M}_\alpha$ where α is the number of subclasses of class τ and also the number of true latent variables per feature. Using all the data from class τ we infer the latent variables $\theta_{ij} \equiv y^i_{t,j}$. We can then select the preferred number of subclasses α by maximising the Bayesian evidence after marginalising over θ_{ij}. The subtlety is that in the models with more than one subclass, we do not know, *a priori*, which subclass an instance may belong to. We solve this by introducing new latent parameters for the subclass which allows each instance to belong to any of the subclasses and then writing the likelihood as a mixture model as was done in (Kunz et al 2007; Hlozek et al 2012; Newling, J., Bassett, B., Hlozek, R. et al. 2012; Knights et al 2013). Finally we marginalise over these subclass labels to compute the model evidence and find the preferred number of subclasses.

6.2 Dealing with Missing Data

In our discussion so far we have assumed the idealised case that we have data at the same points for all training and test data. This is clearly unrealistic and an important limitation. How do we deal with missing data?

There are two approaches. The first, more conservative, approach is to sample from the prior distribution with the error given by the prior distribution for that class. If the data is missing from test data, the missing data can be sampled as above, but in each case we use the prior for the class that it is being compared against.

The second approach is to use some form of interpolation. A natural approach is to use Gaussian processes, since these give both an expected value and Gaussian error at the missing data. Gaussian processes need a covariance function which encodes how rapidly the underlying class varies. As a result each class will have their own Gaussian process and covariance function which should be learned from the training data. Test data should then be compared to training classes using the appropriate Gaussian process for each of the training classes.

6.3 Template construction and speeding up BADAC

As shown in table 5 the full BADAC calculation is much slower than other classification or anomaly detection algorithms. This stems from the pairwise comparison of all data in the test dataset with all data in the training set, something which becomes computationally infeasible for very large amounts of training data.

Fortunately in the limit of large training data we can expect to sample the class distribution well and therefore we can instead create a single template for each class (or as an intermediate step, each sub-class), which will dramatically speed up BADAC, though at the cost of having a non-Gaussian spread in general.

How should we compute class or sub-class templates? An elegant solution is to fit a single Gaussian process to the data of each class (Williams and Barber 1998). This has the advantage of automatically dealing with any missing data, but will not deal with non-Gaussian or multi-model intra-class variability. To get around this limitation one could use a Kernel Density Estimate summed over the training data in each class at each value of the independent variable (or in bins). However, since this is a still a sum over all the training data examples it will be slow. To speed it up we need some approximation to the KDE sum.

Probably the simplest approximation - which also preserves the Gaussian distribution - is to use the inverse-variance estimator $\hat{y}$ with standard deviation $\hat{\sigma}$:

$$\hat{y} = \hat{\sigma}^2 \sum_i y_i/\sigma_i^2 \tag{16}$$

$$\hat{\sigma}^2 = \left(\sum_i 1/\sigma_i^2 \right)^{-1} \tag{17}$$

If the intraclass variability is highly non-Gaussian then it would be better to fit a more appropriate low-dimensional distribution to describe this to create the template. We then effectively reduce to the formalism described in section 2.5.

6.4 Intraclass Variability

In the formulation presented earlier we assumed that the variability in the observed data for a given class was given by the measurement errors on the observed data, i.e. that the intraclass variability was small. If this is not the case one can build more complex models for the intra-class variability. The simplest is to fit for a global standard deviation, σ_*, at training for each class (for example by using a validation subset of the training data). The intra-class variability model can be made arbitrarily complex and the Bayesian evidence could be used to select the best model.

6.5 Calibration and Zero-point Issues

In applying BADAC to real examples there may be systematic differences in the data between test and training. This could, for example, be because the data comes from different instruments or is taken under different conditions. As an example, consider applying BADAC to images where there may be large-scale calibration differences across the images. How can one deal with such effects which will invalidate the use of the simple versions of the BADAC formalism presented earlier, along with most anomaly and classification algorithms?

In the spirit of the Bayesian approach, one way to deal with such large-scale artefacts is to model their effects and introduce nuisance parameters φ, with their own prior distributions $P(\varphi)$, which are then marginalised over before classification or anomaly ranking. Intuitively this means that the algorithm will exploit the freedom implicit in the calibration model to try to fit each test curve to the training data and will only highlight as outliers those data which are poor fits no matter the calibration freedom.

A related problem is the issue of zero-points, which occurs if the examples in training and test data are not all aligned on the x-axis. This is common when working with time series data. In principle this can be dealt with in a similar way, by allowing each data example to have an extra translation parameter which allows one to shift all points in the example left or right. One must then marginalise over this nuisance parameter when doing the fits.

Depending on the exact nature of the data these translation parameters may be well-constrained. For example, one may be able to align all examples approximately, in which case one can put priors on the translation parameters. However, the zero-point issue does raise significant complications. For each pair in the training and test sets one should in principle allow a translation parameter. This leads to $n \times M$ new nuisance parameters where n, M are the number of training and test set examples respectively. Unless the marginalisation can be performed analytically this will typically be prohibitively expensive.

A cheaper alternative is to pre-align all the training data by class. Now there is only one translation nuisance parameter per class and per instance in the test set. However, the alignment of the training data will not be perfect in general. This can be handled by adding an x-error bar to each data point in the training data, corresponding to small errors in the alignment of the data. These x-errors are perfectly correlated however (since the translation affects all data in the same way) and the BADAC formalism would need to be extended to account for such correlations, as done in e.g. Heavens et al (2014); Roberts et al (2017).

7 Conclusions

We have presented a novel statistically robust joint anomaly detection and classification method, Bayesian Anomaly Detection And Classification (BADAC), that is designed to take advantage of any knowledge of the underlying noise distribution in the training and test data. Although we perform tests for the case of Gaussian distributed data, our formalism is general.

Using simulated one-dimensional data, we test the classification and anomaly detection capabilities of BADAC. We make use of several metrics, including our novel Rank-Weighted-Score that rewards algorithms for ranking more anomalous objects above those that have been commonly seen. We find that in the case where the correct noise model is known BADAC outperforms random forests at classification and both IsolationForest and local outlier factor (LOF) at anomaly detection, due to its ability to correctly exploit uncertainty information. In the case of compact anomalies, which could emulate noisy spikes in data, we find that BADAC's performance is comparable to LOF and superior to IsolationForest. We demonstrate how BADAC produces calibrated classification probabilities, which is crucial if a machine learning algorithm is to be incorporated into a precise, scientific analysis pipeline.

We performed tests to investigate the degradation of performance if the assumptions of BADAC are violated. Interestingly, we find BADAC still outperforms the other anomaly detection algorithms in the presence of non-Gaussian and correlated noise. However we find its classification performance degrades, especially in the correlated case. We also note that with an incorrect noise model, the probabilities of BADAC are no longer guaranteed to be calibrated. However, if the structure of the noise is known, the correct noise model can be incorporated into the BADAC likelihood.

While BADAC provides excellent performance by exploiting the extra information about the underlying noise distributions, the computational limitations discussed in section 4.3 mean that it does not scale well to large training datasets. In this case one must either use prototype templates to represent the classes (e.g. through Gaussian processes) or parameterise the data, to speed up classification and anomaly detection with BADAC.

We find ourselves in an era of exponentially increasing data volume, driving the need for machine learning algorithms. However in the physical sciences there is equal need for accurate propagation of uncertainties from all parts of an analysis pipeline, including any machine learning algorithms. With its statistically principled approach to both classification and anomaly detection, BADAC is able to provide believable and interpretable probabilities in the presence of measurement uncertainties, as required by high precision scientific analysis.

Acknowledgements We thank Alireza Vafaei Sadr, Martin Kunz and Boris Leistedt for discussions and comments. We acknowledge the financial assistance of the National Research Foundation (NRF). Opinions expressed and conclusions arrived at, are those of the authors and are not necessarily to be attributed to the NRF. This work is partially supported by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 306478-CosmicDawn.

References

Akhtar N, Mian A (2018) Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access 6:14,410–14,430

Altman NS (1992) An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46(3):175–185

Breiman L, Schapire E (2001) Random forests. In: Machine Learning, pp 5–32, DOI 10.1.1.23.3999

Breunig MM, Kriegel HP, Ng RT, Sander J (2000) Lof: Identifying density-based local outliers. SIGMOD Rec 29(2):93–104, DOI 10.1145/335191.335388, URL http://doi.acm.org/10.1145/335191.335388

Cheeseman P, Stutz J (1996) Advances in knowledge discovery and data mining. American Association for Artificial Intelligence, Menlo Park, CA, USA, chap Bayesian Classification (AutoClass): Theory and Results, pp 153–180, URL http://dl.acm.org/citation.cfm?id=257938.257954

Connolly N, Connolly B (2009) A Bayesian Approach to Classifying Supernovae With Color. ArXiv e-prints 0909.3652

Davis J, Goadrich M (2006) The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning, ACM, New York, NY, USA, ICML '06, pp 233–240, DOI 10.1145/1143844.1143874, URL http://doi.acm.org/10.1145/1143844.1143874

Denison DG, Holmes CC, Mallick BK, Smith AF (2002) Bayesian methods for nonlinear classification and regression, vol 386. John Wiley & Sons

Domingos P, Pazzani M (1997) On the optimality of the simple bayesian classifier under zero-one loss. Machine learning 29(2-3):103–130

Fawzi A, Moosavi-Dezfooli SM, Frossard P, Soatto S (2017) Classification regions of deep neural networks. ArXiv e-prints 1705.09552

Gelman A (2006) Prior distributions for variance parameters in hierarchical models (comment on article by browne and draper). Bayesian Anal 1(3):515–534, DOI 10.1214/06-BA117A, URL https://doi.org/10.1214/06-BA117A

Gull SF (1989) Bayesian Data Analysis: Straight-line fitting, Springer Netherlands, Dordrecht, pp 511–518. DOI 10.1007/978-94-015-7860-8_55

Heavens AF, Seikel M, Nord BD, Aich M, Bouffanais Y, Bassett BA, Hobson MP (2014) Generalized Fisher matrices. Mon Not Roy Astron Soc 445(2):1687–1693, DOI 10.1093/mnras/stu1866, 1404.2854

Hlozek R, Kunz M, Bassett B, Smith M, Newling J, Varughese M, Kessler R, Bernstein JP, Campbell H, Dilday B, et al (2012) Photometric supernova cosmology with beams and sdss-ii. The Astrophysical Journal 752(2):79

Hotelling H (1933) Analysis of a complex of statistical variables into principal components. Journal of educational psychology 24(6):417

Kim A, Linder E (2011) Correlated Supernova Systematics and Ground Based Surveys. JCAP 06(020), 1102.1992v1

Knights M, Bassett BA, Varughese M, Hlozek R, Kunz M, Smith M, Newling J (2013) Extending BEAMS to incorporate correlated systematic uncertainties. Journal of Cosmology and Astroparticle Physics 1:039, DOI 10.1088/1475-7516/2013/01/039, 1205.3493

Kunz M, Bassett BA, Hlozek R (2007) Bayesian Estimation Applied to Multiple Species: Towards cosmology with a million supernovae. Phys Rev D75:103,508, DOI 10.1103/PhysRevD.75.103508, astro-ph/0611004

Liu FT, Ting KM, Zhou ZH (2008) Isolation forest. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA, ICDM '08, pp 413–422, DOI 10.1109/ICDM.2008.17, URL http://dx.doi.org/10.1109/ICDM.2008.17

Mallat SG (2009) A wavelet tour of signal processing, 3rd edn. Academic Press, Burlington

Matthews BW (1975) Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA) - Protein Structure 405(2):442–451, DOI 10.1016/0005-2795(75)90109-9, URL https://doi.org/10.1016/0005-2795(75)90109-9

Newling, J, Bassett, B, Hlozek, R et al (2012) Parameter estimation with Bayesian estimation applied to multiple species in the presence of biases and correlations. Monthly Notices of the Royal Astronomical Society 421(2):913–925, DOI 10.1111/j.1365-2966.2011.20147.x, URL http://dx.doi.org/10.1111/j.1365-2966.2011.20147.x, 1110.6178

Niculescu-Mizil A, Caruana R (2005a) Predicting good probabilities with supervised learning. In: Proceedings of the 22Nd International Conference on Machine Learning, ACM, New York, NY, USA, ICML '05, pp 625–632, DOI 10.1145/1102351.1102430, URL http://doi.acm.org/10.1145/1102351.1102430

Niculescu-Mizil A, Caruana R (2005b) Predicting good probabilities with supervised learning. In: Proceedings of the 22nd international conference on Machine learning, ACM, pp 625–632

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830

Roberts E, Lochner M, Fonseca J, Bassett BA, Lablanche PY, Agarwal S (2017) zBEAMS: A unified solution for supernova cosmology with redshift uncertainties. JCAP 1710(10):036, DOI 10.1088/1475-7516/2017/10/036, 1704.07830

Sako M, Bassett B, Connolly B, Dilday B, Cambell H, Frieman JA, Gladney L, Kessler R, Lampeitl H, Marriner J, Miquel R, Nichol RC, Schneider DP, Smith M, Sollerman J (2011) Photometric Type Ia Supernova Candidates from the Three-year SDSS-II SN Survey Data. The Astrophysical Journal 738:162, DOI 10.1088/0004-637X/738/2/162, 1107.5106

Spearman C (1904) The proof and measurement of association between two things. The American Journal of Psychology 15(1):72–101, URL http://www.jstor.org/stable/1412159

Williams CK, Barber D (1998) Bayesian classification with gaussian processes. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(12):1342–1351

A Detailed derivation of BADAC

In this section we go through the details of the formalism we discussed in section 2. We begin with the simple case of one training and test datapoint, and then add n training points. We finish by generalising the formalism to objects with m datapoints each.

A.1 Modelling one training point

Our goal is to determine the probability of an object d belonging to class τ. This is calculated assuming that we have an observation of another object y_o, which we know is from class τ. The probability of d belonging to class τ, given y_o can hence be written as $P(\tau|d, y_o)$. By implementing Bayes' theorem, we can arrive at equation 18:

$$P(\tau|d, y_o) \propto P(d, y_o|\tau)P(\tau) \tag{18}$$

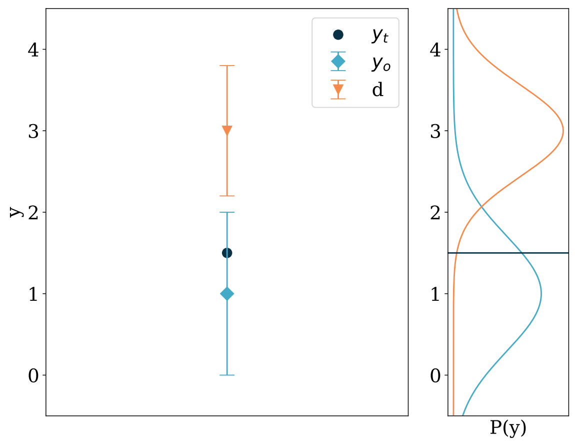

Fig. 13: Plot of the schematic representation of this Bayesian hierarchical model with a single datapoint. This representation is shown in the left pane, and the probability distributions associated with the uncertainty on the data are shown in the right pane. The orange triangular marker for d shows the new test point we would like to consider. The light blue diamond marker for y_o shows the training datum of known class type. The black dot represents y_t, a latent parameter that we introduce that can be thought of as the "true value" for y_o, if we were able to know y_o exactly. This is the parameter we wish to marginalise over.

If we now assume that we have a measurement error associated with both d and y_o, we cannot expand equation 18 to calculate the probability as is. To solve this we introduce a latent parameter y_t, which we wish to marginalise over. Here y_t is the "true" value of y_o, as though we were able to know the correct value of y_o exactly. By application of the product rule, we obtain the result shown in equation 21.

$$P(\tau|d, y_o) \propto P(\tau) \int dy_t P(d, y_o, y_t|\tau) \tag{19}$$

$$P(\tau|d, y_o) \propto P(\tau) \int dy_t P(d, y_o|y_t, \tau)P(y_t|\tau) \tag{20}$$

$$P(\tau|d, y_o) \propto P(\tau) \int dy_t P(d|y_o, y_t, \tau)P(y_o|y_t, \tau)P(y_t|\tau) \tag{21}$$

We illustrate this hierarchical model graphically in figure 13. If we assume d and y_o are statistically independent of one another, then we can drop the dependence on y_o from terms that depend on y_t as well i.e.: $P(d|y_o, y_t, \tau) = P(d|y_t, \tau)$.

We can do this since we assume that y_t represents y_o exactly. We then end up with the result shown in equation 22.

$$P(\tau|d, y_o) \propto P(\tau) \int dy_t P(d|y_t, \tau) P(y_o|y_t, \tau) P(y_t|\tau) \tag{22}$$

We assume the measurement errors associated with d and y_o are Gaussian, which makes the probability terms $P(d|y_t, \tau)$ and $P(y_o|y_t, \tau)$ Gaussian likelihoods. We further assume a flat prior on the distribution of true values of y_o, $P(y_t|\tau)$. These assumptions allow us to perform the integration analytically. We can now generalise this method to work for n training/known objects.

A.2 Modelling n training instances

Here we begin with Bayes' theorem in the same manner as before. The class probability is now dependent on the unknown object d, as well as n known objects y_o^i where $i \in [1, n]$:

$$P(\tau|d, y_o^1, ..., y_o^n) \propto P(d, y_o^1, ..., y_o^n|\tau) P(\tau) \tag{23}$$

Now we introduce n latent variables y_t^i, with $i \in [1, n]$, such that $y_t^i, ..., y_t^n$ are the true values corresponding to each object y_o^i. By marginalising over these latent variables, we arrive at equation 24.

$$P(\tau|d, y_o^1, ..., y_o^n) \propto P(\tau) \int dy_t^1 ... dy_t^n P(d, y_o^1, ..., y_o^n, y_t^1, ..., y_t^n|\tau) \tag{24}$$

We can apply the product rule, where we group all variables y_t^i together. We can do this since $P(A, B, C) = P(A, B|C)P(C) = P(A|B, C)P(B|C)P(C) = P(A|B, C)P(B, C)$.

$$P(\tau|d, y_o^1, ..., y_o^n) \propto P(\tau) \int dy_t^1 ... dy_t^n P(d, y_o^1, ..., y_o^n|y_t^1, ..., y_t^n, \tau) P(y_t^1, ..., y_t^n|\tau) \tag{25}$$

Now we apply product rule again. We also skip a step we showed in section A.1, where we drop the dependence of $y_o^1, ..., y_o^n$ on $P(d|y_o^1, ..., y_o^n, y_t^1, ..., y_t^n, \tau)$. This does not apply if observations of different objects are correlated, but for clarity, we ignore this possible source of correlation.

$$P(\tau|d, y_o^1, ..., y_o^n) \propto P(\tau) \int dy_t^1 ... dy_t^n P(d|y_t^1, ..., y_t^n, \tau) P(y_o^1, ..., y_o^n|y_t^1, ..., y_t^n, \tau) P(y_t^1, ..., y_t^n|\tau) \tag{26}$$

In addition to assuming the objects y_o^i are statistically independent of one another, we assume they are independent of the true values y_t^j, where $j \neq i$ as well. This means that $P(y_o^1, ..., y_o^n|y_t^1, ..., y_t^n, \tau) = P(y_o^1|y_t^1, \tau) ... P(y_o^n|y_t^n, \tau)$. We can therefore express equation 26 as follows:

$$P(\tau|d, y_o^1, ..., y_o^n) \propto P(\tau) \int dy_t^1 ... dy_t^n P(d|y_t^1, ..., y_t^n, \tau) \prod_{i=1}^{n} P(y_o^i|y_t^i, \tau) \prod_{i=1}^{n} P(y_t^i|\tau) \tag{27}$$

Now we need to find a way to calculate the value of $P(d|y_t^1, ..., y_t^n, \tau)$. The probability of observing object d given one y_t is a simple likelihood. We use a kernel density estimate for the probability of observing d given a number of variables y_t^i. This means that $P(d|y_t^1, ..., y_t^n, \tau) \equiv \frac{1}{n} \sum_{i=1}^{n} P(d|y_t^i, \tau)$. Now we have:

$$P(\tau|d, y_o^1, ..., y_o^n) \propto P(\tau) \int dy_t^1 ... dy_t^n \left[\frac{1}{n} \sum_{i=1}^{n} P(d|y_t^i, \tau) \right] \prod_{i=1}^{n} P(y_o^i|y_t^i, \tau) \prod_{i=1}^{n} P(y_t^i|\tau) \tag{28}$$

$P(y_t^i|\tau)$ is simply a prior on the true y_o^i measurements. Since we don't aim to place any constraints on the intraclass variability of class τ, we choose a flat prior here ($P(y_t^i|\tau) = 1$). It is important to note that the choice of an improper prior on hierarchical parameters can introduce biases on parameter estimates (Gull (1989) and Gelman (2006)). One could easily select a more informative prior if there was additional information about the data available. For ease, the prior should be analytically integrable with the form of the probabilities $P(d|y_t^i, \tau) P(y_o^i|y_t^i, \tau)$. It would however be possible to resort to numerical integration if this were not the case. Here we assume uncorrelated Gaussian measurement error in the determination of the values of $P(d|y_t^i, \tau)$ and $P(y_o^i|y_t^i, \tau)$, so equation 28 can be expressed as the following:

$$P(\tau|d, y_o^1, ..., y_o^n) \propto \int_{-\infty}^{\infty} dy_t^1 ... dy_t^n \left[\frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_d} \exp\left(-\frac{1}{2} \left(\frac{d - y_t^i}{\sigma_d} \right)^2 \right) \right] \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_{y_o^i}} \exp\left(-\frac{1}{2} \left(\frac{y_o^i - y_t^i}{\sigma_{y_o^i}} \right)^2 \right), \tag{29}$$

where σ_d is the uncertainty on d and $\sigma_{y_o^i}$ is that on the i'th training point y_o. Note that we can switch the sum and integral, and then evaluate the integral of the product of $n+1$ Gaussian likelihoods. However, when we evaluate this integral, it simplifies in the case where the set of y_t^i are uncorrelated. The simplification also means we can integrate over dy_t^i instead of $dy_t^1...dy_t^n$.

$$P(\tau|d, y_o^1, ..., y_o^n) \propto \frac{1}{n}\sum_{i=1}^{n}\int_{-\infty}^{\infty} dy_t^i \frac{1}{\sqrt{2\pi}\sigma_d}\exp\left(-\frac{1}{2}\left(\frac{d-y_t^i}{\sigma_d}\right)^2\right)\frac{1}{\sqrt{2\pi}\sigma_{y_o^i}}\exp\left(-\frac{1}{2}\left(\frac{y_o^i-y_t^i}{\sigma_{y_o^i}}\right)^2\right) \tag{30}$$

Now it is easy to complete the square and do the integration analytically. In equation 31 we simply state the result:

$$P(\tau|d, y_o^1, ..., y_o^n) \propto \frac{1}{n}\sum_{i=1}^{n} P(\tau_i)(2\pi\sigma_d\sigma_{y_o^i})^{-1}\left[\frac{\pi}{\frac{1}{2}(\Gamma_d+\Gamma_i)}\right]^{1/2}\exp\left(-\frac{1}{2}\left(\Gamma_d d^2 + \Gamma_i {y_o^i}^2 - \frac{(\Gamma_d d + \Gamma_i y_o^i)}{\Gamma_d+\Gamma_i}\right)\right) \tag{31}$$

where $\Gamma_d = \frac{1}{\sigma_d^2}$ and $\Gamma_i = \frac{1}{\sigma_{y_o^i}^2}$.

Equation 31 only handles objects with a single datapoint. For objects with multiple datapoints, we introduce $P(\tau|\{d_j\}, \{y_{o,j}^1\}, ..., \{y_{o,j}^n\})$, where it is a product of the probabilities for the single datapoints assuming they are uncorrelated. This means we have:

$$\begin{aligned} P(\tau|\{d_j\}, \{y_{o,j}^1\}, ..., \{y_{o,j}^n\}) \propto \frac{1}{n}\sum_{i=1}^{n} P(\tau_i)\prod_{j=1}^{m}(2\pi\sigma_{d_j}\sigma_{y_{o,j}^i})^{-1} \\ \times \left[\frac{\pi}{\frac{1}{2}(\Gamma_d+\Gamma_i)}\right]^{1/2}\exp\left(-\frac{1}{2}\left(\Gamma_d d_j^2 + \Gamma_i {y_{o,j}^i}^2 - \frac{(\Gamma_d d + \Gamma_i y_{o,j}^i)}{\Gamma_d+\Gamma_i}\right)\right) \end{aligned} \tag{32}$$

where $\Gamma_d = \frac{1}{\sigma_{d,j}^2}$ and $\Gamma_i = \frac{1}{\sigma_{y_{o,j}^i}^2}$, and $\{d_j\}$ and $\{y_{o,j}^i\}$ are all sets over the index j.

We use equation 32 in our evaluation in section 4.

B Correlated data

It is possible to take correlated data into account in the formalism we've developed, if it is known how the data are correlated. Taking into account correlated data allows for better classification accuracy. Here we will talk about two types of correlation. Firstly, correlations may exist between different features in the same instance. Here we'll refer to this as intra-instance correlation. Then we have correlations between different instances, which we refer to as inter-instance correlation. The following subsections show where accounting for these correlations would enter the formalism we developed in sections A.1 and A.2.

B.1 Intra-instance correlation

Intra-instance correlations often arise in science. These are also the type of correlations we assess in section 5.3, albeit without correctly accounting for the correlated nature of the data in order to test the robustness of our formalism. Here we show how one would correctly account for correlated Gaussian noise using BADAC.

Continuing from equation 28, we get the following result if we assume that $P(d|y_t^i, \tau)$, $P(y_o^i|y_t^i, \tau)$ and $P(y_t^i|\tau)$ are normalised Gaussian distributions, and statistically independent of one another (we explore what happens when this is not the case in section B.2):

$$P(\tau|d, y_o^1, ..., y_o^n) \propto P(\tau)\int dy_t^1...dy_t^n\left[\frac{1}{n}\sum_{i=1}^{n}P(d|y_t^i,\tau)\right]\prod_{i=1}^{n}P(y_o^i|y_t^i,\tau)\prod_{i=1}^{n}P(y_t^i|\tau) \tag{33}$$

$$P(\tau|d, y_o^1, ..., y_o^n) \propto P(\tau)\frac{1}{n}\sum_{i=1}^{n}\int dy_t^i P(d|y_t^i,\tau)P(y_o^i|y_t^i,\tau)P(y_t^i|\tau) \tag{34}$$

Evaluating $P(d|y_t^i,\tau)$ when there is correlated Gaussian noise is conceptually simple through the introduction of a covariance matrix C_d.In practice, estimating C_d can be difficult. One approach is to parametrise the covariance matrix and marginalise over these nuisance parameters. Assuming we know the covariance matrix, we can evaluate $P(d|y_t^i,\tau)$ for the i-th instance as follows:

$$P(d|y_t^i,\tau) = (2\pi \det |C_d|)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}{\Delta_d^i}^T {C_d}^{-1} \Delta_d^i\right) \tag{35}$$

where $\Delta_d^i = (d_1 - y_{t,1}^i, \ldots, d_m - y_{t,m}^i)^T$, and C_d is the covariance matrix of the observed instance d, where d has m features. In the same way, we can evaluate $P(y_o^i|y_t^i,\tau)$ for the i-th instance as follows:

$$P(y_o^i|y_t^i,\tau) = (2\pi \det |C_{y_o^i}|)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}{\Delta_{y_o^i}}^T {C_{y_o^i}}^{-1} \Delta_{y_o^i}\right) \tag{36}$$

$\Delta_y^i = (y_{o,1}^i - y_{t,1}^i, \ldots, y_{o,m}^i - y_{t,m}^i)^T$ and C_y^i are the n intra-instance covariance matrices of the observed training instances y_o^i.

B.2 Inter-instance correlation

A possible source of correlations is that the training curves y_o^i could be statistically dependent on one another. This would mean that for a measurement of an object y_o^i, another object y_o^j might be scattered in a correlated or anti-correlated way relative to y_o^i. Here we will call this type of correlation inter-object correlation. Possible causes of this type of correlation in a physical system would be biases introduced by measurement equipment, or observing/environmental conditions. What this would mean for the formalism in sections A.1 and A.2, is that we would not be able to assume $P(d|y_o^i,...,y_o^n,y_t,...,y_t^n,\tau) = P(d|y_t^i,...,y_t^n,\tau)$. It is not trivial to evaluate $P(d|y_o^i,...,y_o^n,y_t^i,...,y_t^n,\tau)$ sensibly, which is why we assume the dependence of d on $y_o^i,...,y_o^n$ is negligible in equation 5.

Inter-object correlations also enter into our formalism in equation 27. Here we assumed that $P(y_o^1,...,y_o^n|y_t^1,...,y_t^n,\tau) = P(y_o^1|y_t^1,\tau)...P(y_o^n|y_t^n,\tau)$. If one were interested in taking these correlations into account, the correct expansion would be:

$$P(y_o^i,...,y_o^n|y_t^i,...,y_t^n,\tau) = P(y_o^1,...,y_o^n|y_t^1,...,y_t^n,\tau) \tag{37}$$

$$P(y_o^i,...,y_o^n|y_t^i,...,y_t^n,\tau) = P(y_o^n|y_t^n,\tau)\prod_{i=1}^{n-1} P(y_o^i|y_o^{i+1},...,y_o^n,y_t^i,...,y_t^n,\tau) \tag{38}$$

If we assume a Gaussian form for the likelihood as in equation 29, and further assume no intra-instance correlations (no $x - x'$ correlations), then equation 38 simplifies to:

$$P(y_o^i,...,y_o^n|y_t^i,...,y_t^n,\tau) = (2\pi \det |C|)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(\Delta^T C^{-1}\Delta\right)\right) \tag{39}$$

where $\Delta = (y_o^1 - y_t^1, \ldots, y_o^n - y_t^n)^T$ and C is the covariance matrix.

This type of correlation could arise for example, if multiple training data are observed with the same observing factors, leading to correlated errors.

C Metrics for Algorithm Evaluation

In this section we outline the metrics we use in section 4.2 to quantify algorithm performance for both classification and anomaly detection. The choice of a metric is important since inappropriate metrics can give very misleading results. In our case we want metrics that are insensitive to class imbalance (since anomalies are assumed to be rare).

While one of the metrics considered, the AUC (section C.1), uses the probability of belonging to a particular class, the other metrics discussed require a strict classification. In all cases, we take the class with the highest probability to be the algorithm's classification.

C.1 Area Under the Curve

The Area Under the Curve (AUC) metric is suitable for gauging performance in any binary classification problem. It is found by calculating the area under the Receiver Operating Characteristic (ROC) curve for the class of interest. The ROC curve is found by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR). In order to do this, some sort of probability of belonging to the class of interest is required for the objects being classified.

The AUC has a possible range of $[0, 1]$, where 0 represents all true objects belonging to the class of interest being given lower probabilities than other objects, and 1 represents all true objects belonging to the class of interest being given higher probabilities than other objects. An algorithm performs perfectly under this metric with a score of 1, and the worst possible performance is 0. Random guessing for classification would on average yield an AUC score of 0.5 for either balanced or unbalanced data.

Note that the AUC of the ROC curve is known to be problematic for imbalanced data (of which anomaly detection is the extreme case) Davis and Goadrich (2006). As a result we also consider the MCC and RWS scores.

C.2 Matthews Correlation Coefficient

The Matthews Correlation Coefficient (MCC) (Matthews 1975) is another metric for assessing the performance of machine learning algorithms on binary classification problems. It is useful for measuring algorithm performance on datasets with unbalanced classes. For example, an algorithm would score 99% accuracy on a dataset with 99 inliers and one outlier if it predicted only inliers. The same algorithm would get an MCC score of 0, where -1 is the worst possible score, and 1 the best.

The MCC is defined as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{40}$$

where TP is the number of true positives, TN the number of true negatives, FP the number of false positives and FN the number of false negatives in a test dataset.

In this section we only use the MCC score to measure performance in anomaly detection, not classification, and the positive class refers to the anomaly class.

C.3 Accuracy

We use the overall accuracy in computing classification performances only (not anomaly detection) in section 4.2. This is the number of correctly classified objects divided by the total number of objects.