IMPROVING CLASSIFICATION BY REJECTION OF HIGH EPISTEMIC UNCERTAINTY POINTS.

Tim Willaert
Student ID: 20011759

Promotor: Prof. Dr. Willem Waegeman
Tutor(s): Ir. Drs. Thomas Mortier

A dissertation submitted to Ghent University in partial fulfilment of the requirements for the degree of Master of Science in Statistical Data Analysis (Computational Statistics track).

Academic year: 2020 - 2021

GHENT UNIVERSITY

IMPROVING CLASSIFICATION BY REJECTION OF HIGH EPISTEMIC UNCERTAINTY POINTS.

Tim Willaert
Student ID: 20011759

Promotor: Prof. Dr. Willem Waegeman
Tutor(s): Ir. Drs. Thomas Mortier

A dissertation submitted to Ghent University in partial fulfilment of the requirements for the degree of Master of Science in Statistical Data Analysis (Computational Statistics track).

Academic year: 2020 - 2021

GHENT UNIVERSITY

ACKNOWLEDGEMENTS

I would like to thank my tutor Thomas Mortier and promotor Willem Waegeman for their time and advice, and for interesting discussions about some of the topics addressed in the thesis. I would also like to thank Thomas for some useful suggestions and references in preprocessing the image data (transfer learning, variational auto encoders,..).

Apart from that I would also like to thank my family (especially Aline, Janna and Chris) for their support and patience while I was combining the master with my job. I also want to thank my supervisor at work Lore, for flexibility in the planning of my time off from work to finish the thesis, after the covid-restrictions called for some last-minute change of plans.

FOREWORD

Describe own contributions, data, and confidentiality of data.
The data used is either simulated or completely public. For the image datasets the sources are quoted in the relevant section, and the creation of the simulated datasets in appendix E and section 3.3 is also described there or in the corresponding appendix. Regarding what contributions are my own, the first chapter is by large based on or at least inspired by the literature. Section 1.1 arose from an attempt to come up with a unified definition of the uncertainties, in terms of tangible quantities, namely observed and predicted targets, that can be compared between very different models irrespective of the parameters they use or don't use. Certainly if we want to arrive at a decomposition into two non negative and non-overlapping complementary parts, it seems necessary that both contributions share the same 'units' i.e. live in a same space and can be added and compared to each other, which seemed impossible when epistemic uncertainty is defined in terms of model parameters.

I should mention in particular [1], [2] ,[3], [4] (and [5], [6] for the section on BNN) as sources for the theoretical chapter, and [4] and [2] as sources of inspiration for the experiments in appendix E. For the results in chapter 3 we wanted to combine methods from [7] and [8].

Notation

I've tried to favor readability by not pursuing utter consistency in notation throughout the thesis. Vector arrows are written only when we want to stress the vectorial nature or when necessary to avoid ambiguity. For example, we use $\vec{p}(x)$ in appendix E to collect the conditional probability $p(i|x)$ of each class $i = 1, \dots K$ at that point in feature space in a single vector . Here x is not always written out explicitly as a vector, but $\vec{p}$ is so we don't confuse it with the marginal probability $p(x)$ at that same point, which is $p(x) = \sum_i p(i, x) = \sum_i p(i)p(i|x) = \sum_i p(i)(\vec{p}(x))_i$

In general, capital letters will denote random (scalar) variables, small letters (with an arrow only when necessary to avoid confusion and when the distinction is relevant) vectors. For example, the i-th observation for the random variable Y is often one hot encoded into y_i, where the i still denotes the observation number, and has components $y_{i,j}$ which are 0 or 1 depending on whether the category of Y_i is class j or not, $y_{i,j} = \delta_{Y_i,j}$. (We sometimes use the Kronecker symbol $\delta_{i,j}$ as a shorthand for $\mathbb{1}_{i=j}$).

Parameters are usually denoted by greek letters, and their vectorial nature is implicit unless stated otherwise. We will often denote predicted values by hats, and averages by overbars. For matrices/tensors I use doublestroke (or mathbbol) font, and for lists of parameters (like neural network weights), or datasets consisting of a set of points each with a feature vector and scalar outcome) I use mathcal, ie somewhat cursive.

Abbreviations

- GMM: Gaussian Mixture Model
- (B)NN: (Bayesian) Neural Network
- ID/OOD: In/Out of Domain
- D.E.: density estimation
- RV: Random Variable
- PDF : probability distribution function: probability density. (not the cumulative distribution!)
- KL: Kullback-Leibler divergence : an asymmetric 'distance' measure (the quotes because symmetry is required to speak of proper distance) between two probability distributions, defined as $KL(p(x)||q(x)) = \int_{x\in\mathcal{X}} \mathrm{d}x p(x) \log \frac{p(x)}{q(x)}$ where the integral can also be replaced by a sum for a discrete state space $\mathcal{X}$.
- JSD: Jenssen Shannon Divergence: Symmetrized version of the Kullback Leibler divergence that takes an arbitrary number of probability vectors, and computes their KL divergences wrt to the average probability vector. Note that this can be rewritten as the average entropy minus the entropy of the average distribution.

ABSTRACT

The initial objective of this work was to use the method of conformal cautious classification to eliminate points of high epistemic uncertainty (on the edges of each class) in order to improve overall classification performance on the remaining points. However, we soon found that this approach is limited, as it neglects other sources of uncertainty (aleatoric, model variance, and also bias for restrictive models) by assuming that points at the edges of each class are those with highest uncertainty, while this needn't be the case. This approach can be very useful (as we try to illustrate in section 3.3) in situations where distributional uncertainty is dominant, and other components of epistemic uncertainty such as bias and model variance are less important. But one of the main problems is that a larger uncertainty, if it is epistemic in nature, needn't translate to a lower accuracy. Especially for artificial problems where the test data is not fundamentally different from training data, this approach will typically not be rewarded at all in terms of accuracy, as it will typically be the spatial inhomogeneities in aleatoric uncertainty that will determine what happens to accuracy when we reject the peripheral points. But that by no means implies that we were any less uncertain about our extrapolations a priori. In more realistic situations, where the test set can only be approximated by a training set collected under different circumstances, it may be crucial to avoid overconfident predictions.

Because of this limitation to distributional uncertainty on the one hand, and because the method above assesses only overall performances (while predictive uncertainty can depend strongly on the location in feature space where we find an instance) on the other, afterwards we also tried to visualise the different components of uncertainty in each point of feature space, be it for a low dimensional toy example. We will draw ensembles of training sets and study the variability in predictions for different methods of density estimation. I will try to briefly present and comment some of the results from each part.

The outline of the thesis is then as follows: In chapter 2, we introduce some of the tools used from the literature. The following chapter (which has been transferred eventually to appendix E) presents some of those visualisations we tried after the results from the conformal classification turned out to be of limited use. Namely, for a simulated data set in two dimensions where each class consists of a Gaussian mixture, we create maps of what could be candidate indicators of epistemic, aleatoric and total uncertainty. Finally, the last chapter is concerned mainly with the application of the method of cautious conformal classification to a few publicly available image datasets. At the end of that chapter we also experiment with the exclusion of points of high aleatoric uncertainty.

CONTENTS

Contents vi

1 Introduction 1

1.1 Motivation . . . 1

1.2 Outline of the thesis . . . 2

2 Methods 3

2.1 Epistemic and Aleatoric uncertainty . . . 3

2.1.1 Introduction . . . 3

2.1.2 Frequentists uncertainty decomposition . . . 4

2.1.3 Bayesian uncertainty decomposition . . . 9

2.2 Bayesian Neural Networks . . . 16

2.2.1 Introduction . . . 16

2.2.2 Neural Networks . . . 16

2.2.3 Bayesian Neural networks . . . 18

2.3 Cautious classification . . . 22

2.3.1 Motivation . . . 22

2.3.2 Conformal Prediction . . . 23

2.3.3 Using cautious classification to exclude points of high epistemic uncertainty . . . 28

3 Results 30

3.1 Density Estimation . . . 30

3.1.1 Mclust For Gaussian Mixtures . . . 31

3.1.2 Non-parametric approaches (KDE , kNN) , and PPGGMGA . . . 32

3.2 Performance measures . . . 34

3.3 Detecting distributional uncertainty from the rejection curve. . . . 38

3.4 Application to Image Data . . . 39

3.4.1 Using existing state of the art pre-trained feature extractors. . . . 40

3.4.2 Results for image data: MNIST , FashionMNIST and Cifar 40
3.4.3 Rejecting also points with high aleatoric uncertainty 45

Conclusions **48**

References **50**

Appendix A Mclust for Gaussian Mixture Models **A-1**

Appendix B Detecting distributional mismatch **B-1**

B.1 Simulated Gaussian data with varying fractions of new points, and varying overlap. B-1
B.2 Real datasets B-3

Appendix C Application to Image Data: preprocessing and feature selection **C-1**

C.1 Transfer Learning and VGG16 . . . . . C-1
C.2 Obstacles to using VGG16 as-is C-3
C.3 Adaptation of VGG16 . . . . . C-4
C.4 Data Splitting C-6
C.5 PCA for the 512 standard-VGG channels C-7

Appendix D Figures other image datasets. **D-1**

D.1 Cifar D-1
D.2 Mnist D-1
D.3 Fashion (with Aleatoric) D-7

Appendix E Quantification of epistemic and aleatoric uncertainty **E-1**

E.1 Introduction E-1
E.2 Uncertainty Measures E-4
E.3 Results Mixture data E-9
E.4 Discussion E-20

CHAPTER 1
INTRODUCTION

1.1 Motivation

In this thesis we will take a closer look at different components of predictive uncertainty. This distinction is important because the best way to reduce such uncertainty depends on its nature. A use case could be for example, a biologist that intends to identify bacteria using MALDI-TOF mass spectroscopy. He would like to know whether the main source of uncertainty is due to doubt between very similar subtypes of a same genus (aleatoric), due to the limited number of examples of a certain subtype (epistemic), or because the bacterium is simply unlike any other seen before (distributional). When we find uncertainty to be dominated by the so-called aleatoric component, there is no point in collecting more training instances to learn from. Also widening the hypothesis space and training models with more parameters is completely useless in this case, as we can be sure that we cannot do better with the data at hand unless we add new, independent, more predictive features.

Another important source of uncertainty, though harder to quantify, is that that comes with the extrapolation of distributional assumptions beyond the original domain of the training data. The danger of basing conclusions on $p(y|x)$ in such a situation is that we are doing classification 'by elimination', classifying an instance (sometimes with very high confidence) into the least unlikely class, however unlikely that class may be. Especially in certain safety critical applications, it can be preferable to flag an instance as 'unlike anything we've seen before' (eg. to be analysed in more detail, or manually) rather than classifying it into the least unlikely class without any further verification. For the same reasons, it can also be valuable to know when several classes have a large probability, rather than choosing the most likely of them and ignoring how likely this actually was.

Conformal prediction offers a way out of this by determining for each class individually how safe it is to exclude a certain class for a given instance, allowing one to attain a predefined confidence level. In [7] a method is put forward to achieve this confidence level in a distribution free and finite sample fashion, and we will apply this method to a few publicly available image data sets. To do so we have to pre-process the data to convert it from pixel-wise RGB values into a more manageable number of more meaningful features, for which we then need to perform a density estimation in many dimensions. We will try several parametric and non-parametric methods for this, and compare the results.

By building a simple generative classifier and studying the evolution of a series of performance measures with this confidence level, we hope to find out when exclusion of the most anomalous points can be beneficial to overall performance. Because we will find this method especially suited for so-called distributional uncertainty and less for situations where other contributions are also important, we also add some experiments focusing more on the quantification of those other contributions in each point of predictor space.

1.2 Outline of the thesis

In the first chapter (2), we will discuss how the different contributions of uncertainty are defined in the literature, and try to find some common ground between them in order to work towards a more general definition. We see that it is not so easy to do this in a sense that is free of all context, and propose to use the bias-variance-noise decomposition as a blueprint. However that definition is only of theoretical interest, as it requires the true population parameters and predictive distributions of $\hat{Y}$. A more pragmatical definition comes from the Bayesian literature, and can be used if we can generate a sample of parameter values.

We will also argue there and in the next chapters why we think it is hard to avoid translation of uncertainties about parameters to corresponding uncertainties about predicted targets, if one is to compare models with very different parametrisations, or epistemic to aleatoric uncertainty in each point. This chapter furthermore draws some attention to the context of Bayesian Neural networks where most of the literature on this topic comes from, and finally we devote a section to the definition and interpretation of the conformal prediction framework in the context of cautious classification in section 2.3.

Then we had a chapter (eventually transferred to Appendix E) that was not part of the original assignment, but originated as a result of some experiments focusing on the quantification of the impact of sample variance in the training set on our predictions throughout feature space for generative models using different types of density estimation. We think these fill a void that would otherwise be left by using the method of cautious classification, because the latter is especially useful in settings where we can expect a significant dataset shift (difference between training and test data), that we unfortunately rarely had in our rather pedagogic datasets. In this App. E we also try to quantify uncertainties on an instance-by-instance basis, i.e. at a given location in feature space. Because the method we apply in the last chapter produces only overall performances, it seemed like an interesting addition to visualise also the variation of uncertainties throughout feature space (for a low dimensional toy dataset) in this separate experiment. By looking at the evolution as the training set grows, we can focus in particular on the effect of model variance.

The last chapter contains the results of the main topic, namely how the rejection of points of high distributional uncertainty could influence classification performance. We also add an additional experiment to compare with the effect of rejecting also points of high aleatoric uncertainty, which appears to be much more beneficial for accuracy (as can be expected, since only aleatoric uncertainty has a direct link to accuracy).

Finally, a note about the appendices: In the applications we obtain a lot of figures, and it would be impossible to include these in the main text given the space constraints[1]. More details on some of the other sections (eg. the VGG network, and the Mclust package) can also be found each in a separate appendix.

[1] Even for the appendix, it would be too much to include all figures, so I have uploaded three zipped folders: one with all maps, organised by dataset, by density estimation method, by uncertainty measure and for each size of the training set, and another with all performance measures for each of the image datasets in different settings. A third folder contains some figures related to detection of new classes in the presence of bias and overlap that were left out of Appendix B.

CHAPTER 2

THEORETICAL CONTEXT

2.1 Epistemic and Aleatoric uncertainty

2.1.1 Introduction

Although I certainly hope to clarify this distinction more precisely over the coming pages, it is difficult to give a quantitative definition of these concepts that is really independent of the descriptive model we choose. For example, whether it is Bayesian or non-Bayesian, (non-)parametric, regression/classification, generative/discriminative, and so on.

Informally speaking, we could say that **epistemic uncertainty** (=uncertainty **about the model**) is uncertainty that could in principle be reduced by sampling infinitely densely the entire feature space.
In contrast, there's also uncertainty that can only be removed, if it can removed, by collecting additional features. That is intrinsic or **aleatoric** (=uncertainty about the data).

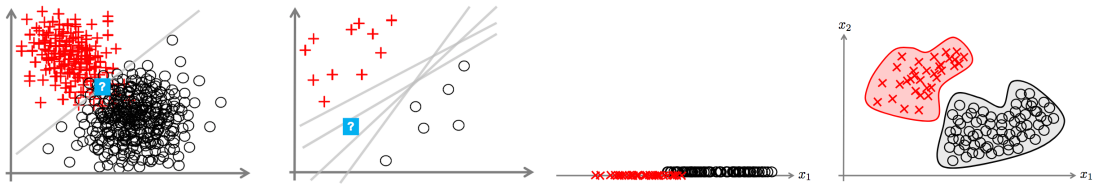


Figure 2.1: From left to right: In the first figure, in spite of knowing the optimal class boundary, we have aleatoric uncertainty about the actual label at the point marked '?' because classes overlap in that region. In the second figure, we are also unsure about that optimal boundary, and this epistemic uncertainty could be reduced if we had more data in that boundary region.
In the third and fourth figure we see how adding features can reduce aleatoric uncertainty, because the class overlap could disappear. Figures taken from [1].

The figures 2.1 (taken from [1]), illustrate the two types of uncertainty. In the first figure from the left we see how, even knowing the optimal decision boundary (=the Bayes classifier), there still remains aleatoric uncertainty about the labels because the classes overlap. In the second figure we see how epistemic uncertainty on the other hand arises due to uncertainty about the location of that optimal boundary itself.

The 3rd and 4th figure on the right illustrate how the aleatoric uncertainty (overlap) depends on the predictors that are used. In general, adding features can convert part of the aleatoric uncertainty into epistemic uncertainty, as adding dimensions means new parameters are to be learnt.

In regression, and for squared loss, a similar distinction would be that between the epistemic uncertainty as expressed in the confidence band, and a prediction band which represents the total uncertainty. The latter is then the combination of this epistemic uncertainty about the true line and an additional aleatoric uncertainty (noise) due to the spread of outcomes around the true line.
There, aleatoric and epistemic uncertainty can easily be expressed on the same scale as the target y, because an uncertainty in parameters θ is readily translated to a corresponding uncertainty in outcome using $\hat{y} = f(\hat{\theta}, x)$ (where the hats denote predictions).

But for classification we don't always model $\hat{y}$ directly as a function of the parameters, and it can be more ambiguous to translate the uncertainty about parameter values into a corresponding uncertainty about target values[1]. This is however necessary if one hopes to obtain a decomposition of the total uncertainty as a sum of these two components, or to compare one component to another in a given point of feature space x.

Perhaps it's best to start with the very familiar shape these concepts take in some particular situations. We will first discuss the bias-variance-noise decomposition for squared loss in univariate regression in a frequentist setting, which can be probably be seen as the simplest decomposition of uncertainties, where additivity is guaranteed by the law of total variance. Next we briefly point out some complications that arise in the classification context, to arrive at a unified description in terms of an arbitrary loss function. This also reveals how the parallel with bias-variance-loss is not limited to the particular case discussed before, but might serve as a blueprint to define the uncertainties in a more general fashion. Finally, we also point out some differences between the frequentist and Bayesian description.

2.1.2 Frequentists uncertainty decomposition

Regression

Considering for a moment the very simplest situation of univariate regression of Y on X with squared loss, we will immediately arrive at the decomposition of the *mean squared error* $MSE^2 = E\big[(\hat{y} - y)^2\big]$ in terms of *bias*, *variance*, and *noise*. While it's natural to think of epistemic and aleatoric uncertainties in this context, it is important to stress that this is already a specific case, and interesting to note that bias, variance and noise can also be defined in a much more general setting, be it without the simple additivity that we have grown accustomed to for the squared loss. More on that in a moment. First let's consider in more detail the case of squared loss. In the Bias-Variance decomposition the respective contributions then correspond to squared errors or variances themselves[2]:

We leave the conditioning on X as implicit for notational convenience, keeping in mind that these quantities have a local meaning in each point of feature space X. The LHS involves a double expectation, a different one of which remains in each term on the RHS (cfr comments

[1] For example in logistic regression, we model the logit $\frac{p(y=i)}{1-p(y=i)}$ as a linear function of predictors, and then predict $\hat{y}(x) = \text{argmax}_y\, p(y|x)$.

[2] If we bring the bias to the LHS we have 3 variances: total-bias=$Var(Y - \hat{Y}|X)$, epi=$Var(\hat{Y}|X)$ alea=$Var(Y|X)$. But for unbiased models we can consider this a decomposition of total variance into model variance and noise.

below).

$$MSE^2 = E\left[\left(\hat{Y} - Y\right)^2\right] = \underbrace{\left(E(\hat{Y}) - E(Y)\right)^2}_{Bias^2} + \underbrace{E\left[\left(\hat{Y} - E(\hat{Y})\right)^2\right]}_{Variance} + \underbrace{E\left[(Y - E(Y))^2\right]}_{Noise} \tag{2.1}$$

$$= \underbrace{\text{Bias}^2 + \text{Var}(\hat{Y})}_{\text{Epi}} + \underbrace{\text{Var}(Y)}_{\text{Alea}}$$

Each term on the RHS stems from a different kind of error/limitation:

1. First, the bias stems from having too limited a hypothesis space: Given certain features, our hypothesis space does not accommodate models that are flexible enough for the expected prediction $E(\hat{Y})$ to coincide perfectly with the expected outcome $E(Y)$ in all points X [3].
   Here the former expectation, $E(\hat{Y})$, is over all possible training data sets $\mathcal{D}$ of size N while the latter expectation ($E(Y)$) is over the true distribution of Y at that X.

2. Second, the variance stems from the finite size of our training data set. Sample variance causes differences in the optimal model parameters that are selected, which leads in turn to variance in the predictions by the model[4]. In general adding more parameters will require also more data to keep the variance in the predictions $\hat{Y}$ in check, and a tradeoff between flexibility and robustness has to be found that minimizes the total prediction error.

3. Finally, the intrinsic noise stems from not having the right features (if such even exist) to discriminate between different outcomes Y at an exact same input value X. It is the degree of variation in Y that is left unexplained when using only these features, even with unlimited sample size and the wildest of hypothesis spaces, because no deterministic function of two equal X can explain why different Y's are observed. This component has expectation zero, because any nonzero expectation would be part of the bias and could be adressed by a more complicated function of X.

Only the third and last contribution is what we consider aleatoric uncertainty, and the first two fall under the nomer epistemic uncertainty. If we restrict ourselves to unbiased models, we can conclude that epistemic uncertainty is the component that shrinks when the sample size increases, while aleatoric uncertainty is what remains in that limit[5].

Note that, **in contrast** to the situation for a **Bayesian model** (cfr. next section), the **uncertainties here are not conditional on an already observed dataset** $\mathcal{D}$. Rather, an expectation over all datasets of a given size N (sampled from the joint distribution for some fixed true value of the model parameters θ) was required to arrive at the expression for bias $[\mathrm{E}_{P(Y|X)}(Y) - \mathrm{E}_{P(\mathcal{D})}(\hat{Y})]^2$ and model variance $\mathrm{Var}_{P(D)}(\hat{Y}) = \mathrm{Var}_{P(\hat{\theta})}[f(\hat{\theta}, X)]$.

[3] think of approximating a parabola by a straight line: the sign of the bias changes twice.

[4] In case of non parametric models this may get a local meaning as well, otherwise it is still possible that a given uncertainty in parameters has a very different impact on predictions in one point of feature space than in another. Uncertainty about a slope would have a larger impact far away from the center of the data cloud; for example.

[5] Note how this defines aleatoric uncertainty as a constant, and means total approaches this constant from above as epistemic shrinks. While we are used to having such a constant aleatoric component σ^2 as a lower bound for the total predictive uncertainty, we will see that this view becomes untenable in situations where the conditional variance $\mathrm{Var}(Y|X, \theta)$ depends also on θ, as for classification. While we can still write down a law of total variance, none of the terms will be constant, and this definition of aleatoric uncertainty loses its familiar meaning when also epistemic uncertainty is present.

Because in frequentist statistics the observed dataset $\mathcal{D}$ uniquely determines the fitted model parameters $\hat{\theta}_{MLE}$ and predictions $\hat{Y} \equiv f(\hat{\theta}_{MLE}, X)$, the variability here only comes from differences in the finite dataset $\mathcal{D}$ that is drawn. In the next section, we turn things around and model the reduction of prior uncertainty about model parameters by the observation of one particular dataset $\mathcal{D}$. The uncertainty about these model parameters that still remains after observation, is then epistemic.

Classification

Another example of the distinction between epistemic and aleatoric uncertainty, in classification, could be how performing thousands of coin tosses to establish that the coin must be very close to fair ($\theta \approx 0.500$ to great precision) wouldn't help you one bit in predicting the next outcome. The fact that it would have helped you quite a lot when you'd found $\theta \leq 0.001$ because the variance in Y for known θ is $\theta(1-\theta)$ shows how the value of θ determines the aleatoric uncertainty in Y. There is thus a kind of hierarchy where we need epistemic uncertainty to be low, $\hat{\theta} \approx \theta_0$ before even being able to quantify the amount of aleatoric uncertainty. Whether the latter is also low then depends not on the precision with which we know θ but on its actual value. This is however perhaps a feature that is to some extent typical of classification with a bi- (or multi)nomial likelihood, where the variance $p(1-p)$ depends directly on the parameter p that also determines the expected value $\mathrm{E}(Y)$, thereby tying mean and (co)variance to one another.

Namely, in the previous (regression) example, we were lucky that the noise level did not depend on the model parameters, as we had $Y \sim N(\beta.X, \epsilon^2)$ with ϵ independent of X and Y. We could consider heteroscedastic scenarios for regression as well, but I think it is an important point that this is not something optional for classification: it will never be possible to think in terms of 'confidence band' plus independent noise, because the uncertainty in the mean implies uncertainty about the noise by itsself, as the noise depends on the mean. Already for this very simple classification example, both epistemic and aleatoric uncertainty depend on the true value θ_0 of θ. In light of the above discussion, it feels more appropriate to discuss this in the next section in a Bayesian context, where we will derive a PDF $p(\theta|D)$ for the true parameter value θ given the observed data and a prior probability $p_0(\theta)$.

For now it is only important to mention that we will typically not express the uncertainty about a variable with a discrete state space in terms of a variance, but rather in terms of the Shannon/Gibbs entropy $H(Y) := -\sum_{y \in \mathcal{Y}} p_y \log p_y := H(p(Y))$[6]. In information theory $-\log p_i$ is a measure of surprise upon observing state i (unprobable means large surprise), that arises from the demands that it be positive and additive for independent events ($p(x,y) = p(x)p(y) \Rightarrow \log p(x,y) = \log p(x) + \log p(y)$). We can then see $H(p(Y))$ is the expected amount of information we'll obtain when making an observation and learning the state, or alternatively, the uncertainty we had before we made the observation. This is the uncertainty associated with the true distribution of Y, without any model being involved.

[6] Here (in exception to our notation conventions) the lowercase letters y denote a particular value of the random variable (RV) Y, and run over its state space $\mathcal{Y}$. With a slight abuse of notation for the sake of clarity, we will use here the same symbol H to denote entropy explicitly as a function of the probability distribution $p(Y)$ of a discrete random variable Y, as to denote it as function of that random variable Y, because we will usually tend to write out explicitly the probability distribution as an argument for H instead of the random variable.
A second notational shortcut we will use sometimes, in the case of a random variable Y with a discrete state space $i = 1, \ldots, K$, is to collect the probabilities $p(y)$ for each state y of Y in a vector $\vec{p}$, where the i-th component then contains $p(Y = i)$. That way, we abbreviate the entropy of Y to $H[\vec{p}]$.

Training a model consists in specifying a loss function $L(y, \hat{y})$ which depends on the model parameters through the predictions $\hat{y} = f(\theta, x)$, and then minimizing the summed loss $L(\theta) := \sum_i L(y_i, \hat{y}_i(\theta))$ between predictions $\hat{y}_i$ and observations y_i. For a classification model with mutually exclusive classes, we typically use cross entropy loss, which corresponds to the (negative) multinomial loglikelihood, $\log L(y_i, \hat{y}_i) = \sum_{i,j} y_{i,j} \log \hat{y}_{i,j}$, where the class y_i has been one-hot-encoded[7] and $\hat{y}_i$ is the prediction for that one hot encoded vector, i.e. the vector of predicted class probabilities in the point x_i.

Here we recognize the cross-entropy $S(\vec{p}, \vec{q}) = -\vec{p}.\log \vec{q}$, (logarithm applied element-wise) between two discrete probability vectors $\vec{p}$ and $\vec{q}$. Up to a term $H(p)$, this is just the relative entropy $-\sum_i p_i \log \frac{q_i}{p_i}$ which is a discrete-space analogon of the so-called Kullback-Leibler divergence, an often used 'distance' measure between two probability distributions. That means that the parameters θ that minimize this loss function, in the absence of bias, would be the ones for which we predict in each point x_i of the training set the true class probability $p(x_i) = E(Y(x_i)$ at that point of the training set.

To end this section, we first attempt at a more general definition of epistemic and aleatoric uncertainty that unifies the classification and regression cases in terms of a general loss function, to point out why this is more complicated.

General setting

Defining overall uncertainty may be done in different ways. In general we will want to know how much the observed (y) and predicted ($\hat{y}$) outcomes could typically differ. We'll need to choose some metric on $\mathbb{R}^n$ (where $y, \hat{y} \in \mathbb{R}^n$) to express how big a deal we consider each possible discrepancy between y and $\hat{y}$ to be. This metric is called a **loss** function, $L(\hat{y}, y)$, and we usually use a different one for classification, where y, $\hat{y}$ are discrete and bounded, than for regression, where y, $\hat{y}$ are continuous and unbounded. If we have a distribution over y, $\hat{y}$, we could then quantify uncertainty in terms of expected loss between y and $\hat{y}$[8].

Domingos et al. review in [9] the attempts to generalize the well-known bias-variance decomposition (with its important implications for model building) to other loss functions. They find that attempts starting from additivity of bias,variance and noise often end up with these notionas having a different meaning, while starting from a consistent definition implies non-additivity in general. Loosely speaking: if we replace the variances (MSE loss between observed resp. predicted quantities and their expectation) in (2.1) by a different loss function, we will no longer have an equality. The paper discusses some (loss-dependent) weights that can be used for each term so that the equality is restored (for particular loss functions) while each term maintains its meaning.

The reason we mention this paper here, is because it shows us how a decomposition into two positive, non-overlapping contributions is not possible without changing the meaning of those concepts when the loss used is not MSE (eg for classification). But that also means that simply defining epistemic uncertainty as the difference of total and aleatoric in such

[7] $(\vec{y}_i)_j = 1$ if y_i is of class j and 0 for other j

[8] We need to distinguish here already the Bayesian framework where we are decomposing the uncertainty of the **prediction** in a new point given a training set $\mathcal{D}$, not the expected loss between observed y and predicted $\hat{y}$. We will come back to this in subsection 2.1.3, after introducing very briefly the main concepts of Bayesian statistics.

a case will in general not result in the generalised model variance (as defined here below) that resulted from this difference in the squared loss case.

Even if we can not expect both contributions to add up to the total uncertainty in general, this definition would at least allow us to compare their size in a given point, which was not so obvious if the epistemic uncertainty were defined in terms of uncertainty about θ. Such a comparison can be important to decide on the measures best taken to further reduce the uncertainty in a given region, as these depend on the nature of the dominant contribution.

First we define the following quantities (conditioning on x left implicit everywhere):

- the (Bayes)-**optimal** prediction $y_o = \text{argmin}_t \, \mathrm{E}_{p(y|x)}[L(y,t)]$ is the one that would minimize the expected loss with respect to the observed outcomes y. It is the **'central' outcome**, where central reduces to mean, median and mode for squared, absolute and 0/1 loss respectively.
- the **main** prediction (cfr. [9]) $y_m = \text{argmin}_t \, \mathrm{E}_{p(\mathcal{D})}[L(y',t)]$ is the one that would minimize the expected loss with respect to the predictions $\hat{y}$ produced by different datasets $\mathcal{D}$ of fixed size. This is the **'central' prediction** in the same sense as before.

In terms of these quantities, we can then express the bias, variance and noise as follows:

- The **expected loss of a prediction wrt the main prediction** $E[L(\hat{y}, y_m)]$, then measures the uncertainty in the prediction. This is what we'll call the **model variance** and is part of **epistemic** uncertainty. It shrinks as the number of (independent) training instances grows[9].
- The **expected loss of the outcome wrt to the optimal prediction** $E[L(y_o, y)]$ on the other hand, measures the **noise** or **intrinsic variance** present in the observed outcome at fixed $\vec{x}$. This is what we'll refer to by **aleatoric** uncertainty[10].
- Finally, the **loss between optimal and main prediction**, $L(y_o, y_m)$ measures the **bias**. (Note that this is often called squared bias, but here we have not yet explicited the loss function to be squared loss and we prefer to give bias the same dimensions of a loss.) This also pertains to the **epistemic** uncertainty[11].

Only for squared loss do the above 3 contributions nicely add up to the expected[12] total loss between true and predicted value $L(y, \hat{y})$.

We think the above definitions of noise and model variance would be good candidates for a model-independent definition of aleatoric resp. epistemic uncertainty. But that means we have to give up the additivity, and furthermore the definition is only of theoretical interest, as we don't usually know the distribution of observations and predictions. In practice we will have to use a more pragmatical, ensemble-based and additive definition from the Bayesian

[9] Note that the expectation here is with respect to the distribution of all datasets of fixed size that could be drawn from the true joint distribution $P_N(\mathcal{D}) = \prod_{i=1}^{N} P(\vec{x}_i, Y_i)$.
[10] Here the expectation is with respect to the conditional distribution $P(Y|\vec{x})$
[11] And here the expectations (one of each) are inside the loss function
[12] with respect to **both** distributions this time

literature, but we want to stress that those are different from the familiar notions of noise and model variance given above.

Finally, there is also an out of domain (OOD) or **distributional** component in epistemic uncertainty ('dataset shift'), which is probably the hardest one to quantify, as it arises because we can't be sure that the model learnt on in-domain (ID) data extends arbitrarily far outside of that domain (OOD). This is most obvious when we use a parametric model, and all observations, however far away they may be, are used in determining the best parameter values. When these parameter values are then used to predict the behaviour also in very different regions, this entails the assumption that the same values remain valid everywhere, and that is a big assumption. That is, this **distributional uncertainty arises only when we assume a description** (or elements thereof, like the values of certain parameters etc.) **learnt in one region can also be applied in regions where it is yet to be confirmed.**

Here we see how for flexible (unbiased) models distributional uncertainty could also be **seen as a more local extension of the model variance** component: When we consider each region to have its own local parameter values, evolving smoothly over space, then a **sufficient training data density** will be required to have a reasonably precise idea of the values in that region, i.e. **to limit local model variance**[13]. A model that is to be free of distributional uncertainty should base its prediction on enough local training data and the lack of that also implies the lack of certainty about the parameters in that region.

This brings us quite naturally to considering non parametric descriptions (Gaussian Processes, Kernel Density Estimation, k-Nearest Neigbours,..) as a way to mitigate this form of uncertainty[14]. For these models with locally learnt 'effective' parameters, (an increased) model variance appears to replace the distributional uncertainty that came with the (now relaxed) distributional assumptions. We will come back to this in Appendix E.

While these models have smoothness parameters (which may still imply some degree of distributional uncertainty or bias), they don't assume a single distribution has global validity, and that was just the cause of distributional uncertainty[15].

2.1.3 Bayesian uncertainty decomposition

[13] When bias and noise are defined in each point $\vec{x}$ and can be expressed in terms of observation y and prediction $\hat{y}$ only, without appeal to θ and its PDF, then why should we not attempt the same with model variance? It seems preferable over a quantification in the entirely unobservable θ space, as a reparametrization might still give the same predictions but correspond to a very different (hypervolume in/dimension of) parameter space. For example in a high dimensional setting, or in the presence of strong collinearity, uncertainty about θ needn't imply uncertainty about $\hat{y}$, prediction <> inference. Therefore I think uncertainty in $\hat{y}$ rather than in $\hat{\theta}$ would be the least ambiguous way to define model variance. Cfr also introduction Appendix E. If we want to add up epistemic and aleatoric to total uncertainty, they have to have to live in the same space: y and $\hat{y}$ have the same units, whereas y and θ do not (necessarily).

[14] For example, Gaussian processes incorporate the local data density into the uncertainty in their predictions and the model will properly account for the uncertainty out of domain because the mean is a local parameter that is influenced only by its correlation with neighbouring points.

[15] While using non parametric methods allows us to give more importance to nearby data, for non Bayesian models we still have to allow the model not to to predict anything in certain points, in order to really avoid far away points from being able to influence predictions. We will discuss such a method, cautious conformal classification, later in this chapter. Otherwise, the model could still base it's prediction on far away points when no closer ones are available (eg kNN classif.). Bayesian models resort to prior information when no nearby data is available.

Bayesian Models

The Bayesian paradigm is intertwined with the definition of conditional probability

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)P(A)}{\sum_{a\in\mathcal{A}} P(B|A=a)P(A=a)} \sim P(B|A)P(A) = L_B(A).P(A) \quad (2.2)$$

where the sum runs over the entire state space $\mathcal{A}$ of the random variable A, and L_B is the likelihood of parametervalues B after having observed A.
While frequentists do not deny the validity of this expression per se, they operate under the assumption that certain quantities (the underlying unobservable parameters that describe the distribution of the data, like a true population mean etc.) can only appear on the right hand side of the conditioning bar |. They use the word 'likelihood' to stress how this conditional probability is no longer a probability when seen as a function of the variable being conditioned upon.

A Bayesian on the other hand, will not hesitate to use (2.2) to move just about any piece of information he/she pleases to the left hand side of that conditioning bar, and can assign a probability to just about anything, but needs to specify the a priori probability of the true parameter values $p(A)$ to do so. This is the object that the frequentist cannot deal with. The Bayesian, while agreeing that a prior can be somewhat awkward to decide on, insists that this a problem of luxury, because at least now we are able to specify our prior (lack of) knowledge, rather than assuming all parameter values to be a priori equally likely, as a (non-penalised) MLE estimate does.

On top of that, in the frequentist case, once we determine the most likely values of the parameter, the likelyhood of the data will become very small when we have more and more data, because hardly any assumption is ever exactly true. When you have an initial distribution over parameter values, and update this as data keeps coming in, you will not necessarily end up discrediting your original hypothesis as not being true to arbitrary precision. So working with a distribution is also a convenient way of keeping track of a range of viable alternatives.

Say we have decided upon a model hypothesis M_1, which states that data is distributed according to some (unknown) member of a parametric family f_θ indexed by a parameter vector[16] θ and also decided upon a prior $p_0(\theta)$. Then we use (2.2) to obtain

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p_0(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p_0(\theta)}{\int \mathrm{d}\theta\, p(\mathcal{D}|\theta)p_0(\theta)} \quad (2.3)$$

Here the denominator is at times an irrelevant normalisation constant, but not always so! Under this model assumption, the likelihood $p(\mathcal{D}|\theta)$ takes a certain form, and the quantity $p(\mathcal{D}) = \int \mathrm{d}\theta p(\mathcal{D}|\theta)p_0(\theta)$ in the denominator, called the evidence, will have a particular value. Had we chosen another model hypothesis, the likelihood would have been different, and we would have a different $p(\mathcal{D})$. That is because this is in fact not $p(\mathcal{D})$, but really $p(\mathcal{D}|M_1)$ resp $p(\mathcal{D}|M_2)$, and for one model it is more likely to observe the data than for another. This means we can use the ratio $\frac{p(\mathcal{D}|M_1)}{p(\mathcal{D}|M_2)}$ to answer the question which model (M_1

[16] we will drop the vector arrow for notational convenience here, but unless stated otherwise θ represents an arbitrary number of parameters in what follows

or M_2) gives the best description of the data, as the likelihood $l_{\mathcal{D}}(M_1) = p(\mathcal{D}|M_1)$ will be larger accordingly. Again, we could specify a priori probabilities for M_1 and M_2 to decide which model is more credible *after* seeing the data, i.e. which has the larger $p(M_i|\mathcal{D})$. The factor $\frac{p(\mathcal{D}|M_1)}{p(\mathcal{D}|M_2)}$ by which the a priori ratio $\frac{p(M_1)}{p(M_2)}$ is multiplied to get

$$\frac{p(\mathcal{D}, M_1)}{p(\mathcal{D}, M_2)} = \frac{p(M_1|\mathcal{D})p(\mathcal{D})}{p(M_2|\mathcal{D})p(\mathcal{D})} = \frac{p(M_1|\mathcal{D})}{p(M_2|\mathcal{D})}$$

is called the Bayes factor, and it is the amount by which observing $\mathcal{D}$ changes our opinion about the plausibility of M_1 and M_2, respectively. But this is just the ratio of the denominators in (2.3) for the different models M_1 and M_2.

Now, once the posterior distribution (2.3) is known, we can predict any quantity that depended on the true value of θ by weighing all viable alternatives each with their respective probabilities and integrating out θ. For example, the probability of observing a value y' at a new point x' is given by the *predictive* posterior

$$p(y'|x', \mathcal{D}) = \int \mathrm{d}\theta \; p(y'|x', \theta)p(\theta|D)\mathrm{d}\theta \tag{2.4}$$

Uncertainty Decomposition

In Bayesian statistics, by considering the **predictive posterior** as the mixture (2.4) of components of fixed θ (each having only aleatoric uncertainty), we can decompose the variance as the average variance 'within' each component (aleatoric), plus the variance stemming from the mixing 'between' these components (epistemic). Observing data is then supposed to 'thin' the mix of components, while the aleatoric variance in the surviving components remains. Starting again from a law of total variance for the predictive posterior (2.4) and using the subscript $p(\theta|\mathcal{D})$ to denote expectations / variances over the model posterior, we have

$$\mathrm{Var}(y'|x', \mathcal{D}) = \mathrm{E}_{p(\theta|\mathcal{D})}[\mathrm{Var}(y'|x', \theta)] + \mathrm{Var}_{p(\theta|\mathcal{D})}[\mathrm{E}(y'|x', \theta)] \tag{2.5}$$

Here we see how the variance in the outcome y' at a new point x' can be decomposed into a sum of **unexplainable** variance (variance that remains in spite of θ being known: aleatoric), and **explainable** variance (variance in y that would disappear if only we would know the exact value of θ: epistemic). In terms of information, say we knew the slope and intercept of a line (i.e. θ), we still don't know the residual in each observation. So part of the info in the data y was not in the parameters θ, and that part is aleatoric.

So the aleatoric component is defined here as the variance remaining if we assume that the right parameters are used (the expectation of a conditional variance in the first term of the RHS of 2.5), and the epistemic is the variance of the posterior mean (the second term in the RHS of 2.5)[17].
Note how this is different from the situation we had before, because the uncertainty now no longer refers to an expected discrepancy between what's to be observed and what will

[17] In section 2.2.3 we mention in more detail how this can be estimated in practice in Bayesian Neural Networks.

be predicted[18], but to the spread of predicted values[19]. What used to be a distribution of predictions (one for each training set) is now a single predicted distribution for one dataset. The aleatoric component now also changes as our knowledge $p(\theta|\mathcal{D})$ does.[20].

In the above we were able to use the law of total variance because we considered variances as a measure of spread. However, when we consider entropy instead, we do not have such an additive 'law of total entropy'. In the BNN literature, the total and aleatoric uncertainties are then still defined in analogy with the above (i.e. marginal and conditional entropy), but the epistemic component is then defined simply as the difference between them. While this no longer allows us to interpret the epistemic uncertainty as an entropy by itself, it becomes the **mutual information** between the outcome and the parameters. (cfr. for example [2] and [10]). Leaving out the conditioning on x:

$$H(Y) = E_\theta[H(Y|\theta))] + (H(Y) - E_\theta[H(Y|\theta)]) := H(Y|\Theta) + [H(Y) - H(Y|\Theta)] \tag{2.6}$$

where we have now also written Θ to denote that this is a random variable and that an averaging E_θ over its distribution is implicit. To see that this still has some meaning apart from just the difference between total and aleatoric uncertainties, keep in mind how $H(Y)$ expresses the amount of information we expect to learn from observing Y, and likewise for the joint information $H(Y, \Theta)$. Using the definitions (and writing integrals over all values θ of Θ as unspecified sums for readability), we will write the information in Y that was not already in Θ as the difference $H(Y, \Theta) - H(\Theta)$:

$$\begin{aligned} H(Y, \Theta) - H(\Theta) &= -\sum_{\theta, y} p(y; \theta) \log p(y; \theta) + \sum_\theta p(\theta) \log p(\theta)) \\ &= -\sum_{\theta, y} p(y; \theta) \log \frac{p(y; \theta)}{p(\theta)} = -\sum_{y, \theta} p(\theta) \big[p(y|\theta) \log p(y|\theta) \big] \\ &= E_\theta \big[H(Y|\theta) \big] = H(Y|\Theta) := Aleα = H(Y; \Theta) - H(\Theta) \end{aligned} \tag{2.7}$$

Since this is just how we defined the aleatoric uncertainty (information in the labels that wasn't already in the model parameters), and because we identify the total uncertainty in the outcome Y with $H(Y)$, the remainder $H(Y) - H(Y|\Theta)$ in (2.6) must be epistemic (by definition for now). Because of (2.7) we now see that this can also be written as

$$Epi \equiv Tot - Aleα = H(Y) - E_\theta H(Y|\theta)) = H(Y) - H(Y|\Theta) = H(Y) - H(Y, \Theta) + H(\Theta) := I(Y; \Theta) \tag{2.8}$$

or the part that we counted double by adding the information in Y and that in Θ, ie their overlap in information. This is called the **mutual information** between Y and Θ. The last expression shows us how this definition means more than simply total minus aleatoric:
The **epistemic** uncertainty is **the part of information in the parameters that was not**

[18] before we decomposed that expected discrepancy between observed and predicted into an expected discrepancy between observed and central observation, and another between prediction and central prediction, where central observation would coincide with central prediction for an unbiased model.

[19] which here results from the expected spread of the likelihood on top of the spread in the model posterior.

[20] Previously, for the frequentist case, the aleatoric uncertainty was constant, depending only on the true parameter value θ_0 (namely $\mathrm{Var}(Y|\theta_0)$ for some unknown θ_0). However, we should add that the frequentist definition is not very useful because we don't know the value of θ_0. If we did know it, there would be no epistemic uncertainty, and the expression conditional on an unknown θ_0 is therefore more of a total uncertainty than an aleatoric). In practice, when we estimate θ_0 eg. through a sample of θ values obtained by sampling methods, a conditional average has to be taken, and we will eventually end up in a situation quite similar to the one decribed here for a sample from a model posterior.

yet contained in the observed data, i.e. still to be revealed. This is exactly what we were after, as it expresses what still remains to be learnt about θ after our observations.

Finally, to allow a 'loss'-function[21] independent characterization of how epistemic uncertainty is defined in Bayesian statistics, we can also rewrite (2.8) as follows:

$$Epi \equiv Tot - Alea = H(Y) - E_\theta H(Y|\theta) = H[E_\theta(Y|\theta)] - E_\theta[H(Y|\theta)] = [H, E_\theta](Y|\theta) \tag{2.9}$$

In the last step I just wrote this as a commutator $[X, Y] := X \circ Y - Y \circ X$ of two operators because this is in a sense also what we find from the law of total variance when we decompose the predictive variance in an epistemic and aleatoric term:

$$\begin{aligned} Epi = Tot - Alea &= Var(Y) - E_\theta[Var(Y|\theta)] \\ &= \mathrm{Var}[E_\theta(Y|\theta)] - E_\theta[\mathrm{Var}(Y|\theta)] \\ \Rightarrow Epi &= [Var, E_\theta](Y|\theta) \end{aligned} \tag{2.10}$$

In both cases it means the epistemic uncertainty is found as a commutator of our measure of spread and an expectation over model parameters working on the conditional distribution. This is of course because of the way Bayesians obtain the predictive posterior from the model posterior and likelihood. But that means for any concave spread measure of the class probability (like differential entropy or variance), Jenssen's inequality guarantees us a decomposition in two mutually exclusive and nonnegative terms, and **the epistemic uncertainty becomes the amount by which the expected spread, involved in the aleatoric term, underestimates the spread of the expected distribution (involved in the total uncertainty)**. Note how this implies both aleatoric and epistemic uncertainty now depend on our acquired knowledge $p(\theta|\mathcal{D})$, which is very different from the situation for the regression with MSE example.

Let's make this more concrete for our earlier single parameter example of a Bernouilli trial where the true proportion θ_0 is unknown but a dataset is available to estimate it, with Y heads found in n coin flips. We will use Haldane's prior, the $\lim_{\epsilon \to 0} \mathrm{Beta}(\epsilon, \epsilon)$ distribution, $p_0(\theta) \sim B(\theta; 0, 0)$. This is the most noninformative prior distribution for the binomial proportion, in the sense that its posterior mean always coincides with the MLE estimate (even for small n) and its posterior variance is larger than for any other (conjugate) prior. Note that it corresponds to a flat uniform prior on the log-odds scale[22]. While this is an improper prior by itself, it yields a proper posterior $B(Y, n-Y)$ provided at least one success and one failure have been observed, $0 < Y < n$.

Keeping in mind the expression for the expectation and variance of the beta distribution[23], we find for the variance of this posterior $B(Y, n-Y)$

$$\mathrm{Var}(\theta) = \frac{Y(n-Y)}{n^2.(n+1)} = \frac{\hat{\theta}(1-\hat{\theta})}{n+1}$$

[21] 'uncertainty measure' would be more appropriate, a measure of the spread in a distribution like entropy, variance, 1-maxprob, or any other concave measure vanishing on the edges of the simplex. This concave function will be referred to as spread' from now on

[22] For a comparison of Jeffreys, Haldane and uniform prior in this context, see for example ref. Mu Zhu & Arthur Y. Lu (2004) The Counter-intuitive Non-informative Prior for the Bernoulli Family, Journal of Statistics Education, 12:2, , DOI:10.1080/10691898.2004.11910734

[23] Namely, $B(a, b)$ has a variance $\frac{a.b}{(a+b)^2(a+b+1)}$ and expectation $\frac{a}{a+b}$

with $\hat{\theta} \equiv \frac{Y}{n}$ the expected value of θ under the posterior, which (for this prior) is also the MLE estimate, i.e. the observed binomial proportion in our dataset. We see that the epistemic uncertainty comes very close to the 'classical' plug-in estimate for the variance on the binomial proportion based on n observations, $\frac{\hat{\theta}(1-\hat{\theta})}{n}$, as it is also obtained from the MLE and curvature of the likelihood (which is not so surprising as we took such a minimal prior).

What's more, when we combine the aleatoric uncertainty with the epistemic to find a total uncertainty, this turns out to be equal to the most naïve estimate we could have made, plugging the estimated parameter in the expression $p.(1-p)$ for the variance of a Bernouili variable with chance of succes p, i.e. $\hat{\theta}(1-\hat{\theta})$. Namely for the prediction of the next toss ($n = 1$) we have:

$$\begin{aligned} \mathrm{Var}(Y) &= E_\theta[\mathrm{Var}(Y|\theta)] + \mathrm{Var}_\theta[E(Y|\theta)] = \mathrm{E}_\theta[\theta(1-\theta)] + \mathrm{Var}_\theta[\theta] \\ &= \mathrm{E}(\theta) - \mathrm{E}(\theta^2) + \mathrm{Var}(\theta) = \mathrm{E}(\theta) - (\mathrm{Var}(\theta) + [\mathrm{E}(\theta)]^2) + \mathrm{Var}(\theta) \\ &= \mathrm{E}(\theta).[1 - \mathrm{E}(\theta)] := \hat{\theta}.(1-\hat{\theta}) \end{aligned} \tag{2.11}$$

Here the hat refers to the posterior expectation, which for this prior coïncides with the sample estimate (or the MLE). Note that whatever prior we choose, because of the cancellation of $\mathrm{Var}(\theta)$ in the above RHS, the total variance will always be $\hat{\theta}(1-\hat{\theta})$ with $\hat{\theta}$ the expectation of the posterior over θ. (But only for this prior does that coïncide with the sample mean).

Because we found the variance of this posterior to be $\frac{\hat{\theta}(1-\hat{\theta})}{n+1}$, we find the epistemic uncertainty to take up $\frac{1}{n+1}$ (and the aleatoric $\frac{n}{n+1}$-ths) of total uncertainty in this example. Aleatoric uncertainty (being n times epistemic) immediately dominates as n increases.[24].

In classification, variances are bounded and entropies are more commonly used as uncertainty measure, but in either case the epistemic part is defined as the amount by which the expected conditional uncertainty underestimates the uncertainty of the expected conditional (which is also the marginal) distribution. At first sight, looking at 2.11 it seems very odd that the total uncertainty is just given by what appears to be the aleatoric uncertainty for one particular $\hat{\theta}$ that is our best guess for the true population value θ_0. But because aleatoric and total uncertainty must by definition coïncide as we get more and more data and the posterior narrows around the true θ_0, this must actually always be the case. Since entropy is also concave, we visualise this for both cases as in Figure 2.2:

While Jenssens inequality guarantees us that our estimate for aleatoric uncertainty from this sample, $\overline{f(\theta)}$ (with f concave), will not be larger than our estimate of total uncertainty from this sample ($f(\bar{\theta})$, with $\bar{\theta} = \frac{1}{N}\sum_{i=1}^{N}\theta_i$), this estimate for total uncertainty $f(\bar{\theta})$ can both over- or underestimate the true population value $f(\theta_0)$[25].

There are in fact two distributions at play: that of of our posterior mean, and that of the individual θ_j drawn from our posterior. Increasing the size N of the ensemble narrows the distribution of the ensemble average $\overline{\theta}$ around the posterior mean $\hat{\theta} = E(\theta)$. But this does not narrow the distribution of θ_i around $\overline{\theta}$, i.e. does not decrease epistemic uncertainty. For

[24] Note the similarity with the example mid p.66 in Gelman [11] for the variance of the predictive posterior for a normal distribution with noninformative prior. This is nothing exceptional, even for our prediction and confidence intervals in linear MSE regression, we have $\sigma_{\hat{y}}^2 = \sigma^2 + \frac{\sigma^2}{n}[1 + \frac{(x'-\bar{x})^2}{\sigma_x^2}]$, i.e. apart from an x dependent factor, epistemic is also n times smaller than aleatoric.

[25] It will be larger or smaller depending on whether our sample average lies closer or further from uniform probability than the true value θ_0 respectively.

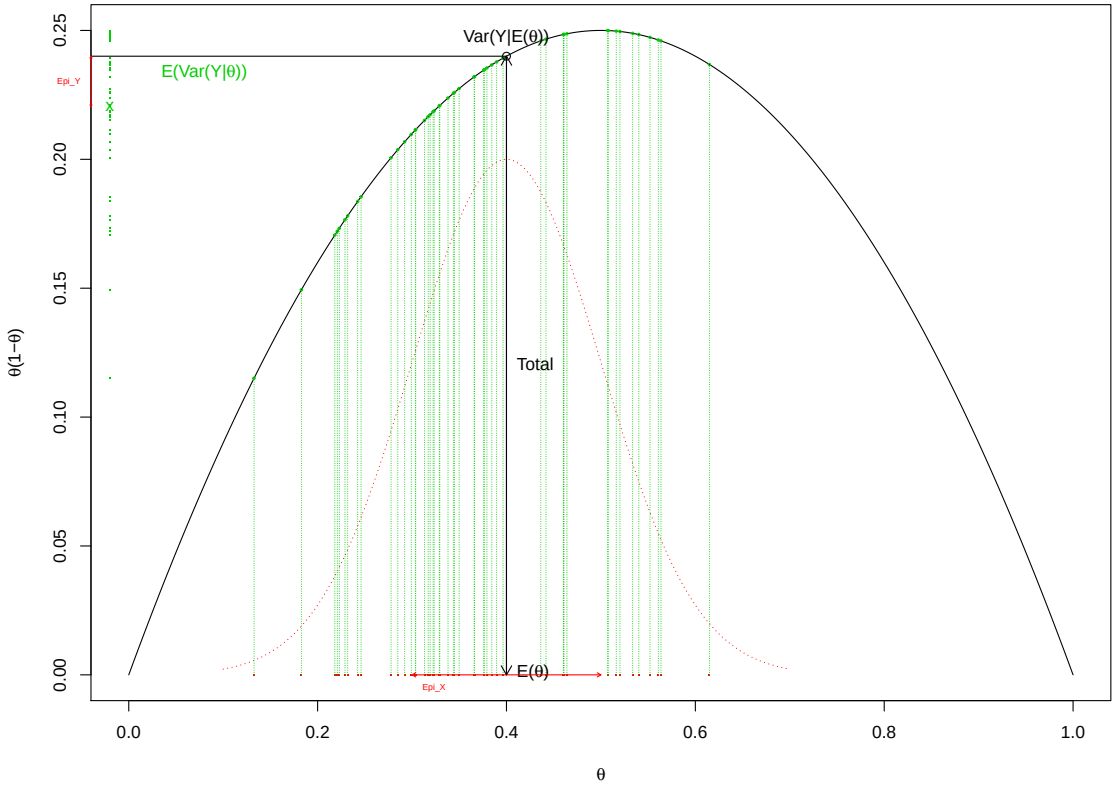


Figure 2.2: The epistemic uncertainty about the true parameter value θ_0 on the abscissa implies an underestimation of the total uncertainty $f(\theta_0)$ on the y axis: When we have a sample of θ_j or a distribution over parameter values $p(\theta)$, then Jenssen's inequality for concave functions (like $f(x) = x(1-x)$ or $-x\log x$) guarantees us that the average of the function values underestimates the function of the average value: $\overline{f(\theta)} <= f(\overline{\theta})$, where $\overline{f(\theta)} = \frac{1}{N}\sum_{j=1}^{N} f(\theta_j)$ (or $\int \mathrm{d}\theta\ p(\theta)f(\theta)$) and $\overline{\theta} = \frac{1}{N}\sum_{j=1}^{N}\theta_j$. But in 2.8 and 2.9 we see that the average of the function values estimates the aleatoric uncertainty, and the function of the average estimates the total uncertainty. The estimate for the epistemic uncertainty defined as their difference is then the amount by which the sample of green y-values underestimates the y-value of the sample average (Black arrow). Cfr also main text.

the latter we have to increase the size of our training set, so both the distribution $p(\theta)$ of the individual θ_i and that of their average $\overline{\theta}$ narrow around $E(\theta)$[26]. Our estimate of aleatoric uncertainty then approaches that of total uncertainty, which in turn also approaches $f(\theta_0)$.

This example[27] which should be the simplest possible decomposition for classification, illustrates the complications that are typical for classification, where (co-)variance and mean cannot be modelled independently and where variances are bounded.

Traditionally we might have expected aleatoric uncertainty to be constant, depending only on the true distribution at X, and the total uncertainty to approach it from above as the

[26] which is also θ_0 when the sampling/posterior mean is unbiased

[27] is a bit special because here the feature space collapsed to a single point, and there is only a single parameter θ_0 to be learnt, so we can now compare global epistemic and aleatoric uncertainty as if we would have done locally in a problem where relevant features did exist. But the same situation would persist in models with predictors, with the single parameter θ_0 resp. θ replaced by the true resp. predicted class probability in that point. (For example $Softmax(f(\theta_0, X))$ resp. $Softmax(f(\theta, X)$ in the case where θ represents weights in a neural network ending in a softmax layer.)

epistemic uncertainty decreases. But at least in the frequentist setting[28] , the total uncertainty has the constant expectation $f(\theta_0)$, and can be brought arbitrarily close to it when we use large enough ensembles of training sets. The reason is probably that the $\overline{\theta}$ in the estimation of total uncertainty draws from all the training sets in the ensemble simultaneously, while aleatoric uncertainty and its complement use only data from one training set at a time, and therefore estimate uncertainties for a dataset of given size. The result is that total uncertainty has constant expectation (namely the limit of aleatoric for $\theta \to \theta_0$), and defining epistemic uncertainty as the complement of aleatoric implies that decreasing epistemic uncertainty by narrowing down the plausible range of θ, converts it into increased aleatoric uncertainty. This is also what we find in Appendix E when we draw maps, and suggests that **this definition is useful to get an indication of epistemic uncertainty, but the aleatoric and thus also total uncertainties it implies for finite datasets should probably be taken with a grain of salt.**[29]. The epistemic uncertainty expresses the granularity by which the other two can be quantified, and probably only their (equal) limiting value has meaning.

2.2 Bayesian Neural Networks

In the past couple of years, many interesting PhD theses ([5],[6],[12]) have been written on this topic, along with some older theses in BNN's (eg. [13]). It would be impossible to attempt the same level of detail as one can find in the combined body of these works, given the limited time span, and my limited knowledge of this topic and ML in general, and I will only try to present in this section some of the ideas presented there.

2.2.1 Introduction

To get the epistemic component, we ideally require a probability distribution over the parameter values. Therefore, it is perhaps not too surprising that Bayesian Neural Networks (BNN's) have attracted a lot of attention in the context of uncertainty quantification. Using this approach, point estimates for parameter values are replaced by posterior probability distributions conditional on the observed data, which is of course very convenient to express our uncertainty about those parameters given the data.

We introduce very briefly first what a neural network is, before coming back to BNNs. This is done in the following subsection.

2.2.2 Neural Networks

A neural net consists of a set of inputs (the 0-th layer, if you like), a series of hidden layers (the actual layers $1, \ldots, L$), and an output layer $L+1$. Each layer consists of identical nodes,

[28] When we draw ensembles of training sets from the correct likelihood, I think we'll have $E(\hat{\theta}) = \theta_0$. However, for a Bayesian posterior based on a single small dataset and a non-informative prior, I don't think we can assume that the posterior mean $E(\theta)$ necessarily coincides with the population value θ_0. An informative prior correctly incorporating model variance might fix this, but this seems like information we don't usually have a priori.

[29] i.e. they differ from the familiar notions introduced in section 2.1.2

and each node applies a nonlinear *activation function* (common to that layer at least) to a linear combination of the outputs of the previous layer.
That nonlinear function could be the logistic function (sigmoid, tαnh,..), it could be a rectifier, ($x \rightarrow \max(0, x)$), or simply the identity in some layers. Of course, adding more nodes gets us a linear combination of these nonlinearities, and especially adding more layers quickly increases flexibility. Even a single layer is sufficient to arbitrarily approach any continuous function provided the number of nodes is allowed to be arbitrarily large[30]. But that complexity can be expressed more efficiently in a few layers of much smaller size.

The number of nodes in each layer can be different, and because the number of connections becomes quickly very large for deeper networks, often specific architectures limiting the number of connections (eg. Convolutional networks, CNNs) are considered. But for the purpose of clarity, lets consider a fully-connected simple feed-forward network, where each node receives an incoming connection from all nodes of the preceding layer, and the strength for the connection coming from the node i in layer $l-1$ into node j in the l-th layer is represented by a weight $w_{j,i}^{(l)}$. Each node also has a bias ($b_j^{(l)}$ for the j-th node in layer l), which can be thought of as the strength of an incoming connection from an imaginatory node in the previous layer that is always outputting the constant value +1. That way we don't have to distinguish between weights and biases.
Lets use n_l to represent the number of nodes in layer l. Each node receives an input that is a sum of all incoming (weighted) inputs (including the bias), $z_j^{(l)} = \sum_{i=1}^{n^{l-1}} w_{j,i}^{(l)} x_i^{(l-1)} + b_j^l$, applies a certain *activation function* $x_j^{(l)} = f^{(l)}(z_j^{(l)})$ to it, and then sends the result on to the nodes in the next layer in a similar fashion.

Needless to say that these models are very flexible (will have vanishing bias), but tend to overfit (i.e. have very large variance) if not properly monitored (early stopping) or regularized (dropout, bayesian nets,weight decay,..).

Dropout consists of randomly deactivating each node with a given probability. This forces the network not to 'put all its eggs in one basket', and will tend to lead to more stable predictions, and thus less overfitting. We learn the weights to do their job without relying too much on the presence of any single other node. Note that the weights obtained in this fashion have to be multiplied by the dropout probability at test time in order not to 'overpredict' when no units are dropped.The dropout mentioned here is not to be confused with so called Monte Carlo dropout, to which we come back later, and occurs at test time.

Weight decay simply adds a regularizing term to the loss function, balancing accuracy with small weight size, where we can choose between L_2 or L_1 penalties as in ridge or lasso regression.

As for any other model, training consists in specifying a loss function suitable to our priorities, ie our performance metric, and finding the values for the model parameters that minimize this loss. A typical component of the loss function would be the log-likelihood of the observed outcome (given the parameters take certain values), perhaps balanced with some penalty to reward certain smoothness or sparsity properties.

[30] for a visual explanation, see for example Michael Nielsen's introduction http://neuralnetworksanddeeplearning.com/chap4.html

A standard way to ensure we predict a normalised probability vector over the K classes is to 'squash' the last layer through a *softmax* function, $\hat{\vec{y}} = \text{Softmax}(\vec{z})$, which is nothing more than a normalised exponential $\hat{y}_j = \frac{\exp z_j}{\sum_{j=1}^{K} \exp z_j}$.

While this is an obvious generalization of a sigmoid/expit function, it's worth stressing that this transforms **a whole vector of K inputs into K outputs**, i.e. **does not fit the picture** I gave before, **of a scalar activation function that works inside a single node** and transforms 1 (linear combination) input into 1 output. This is an important difference because it does not allow us to calculate the output in a single node without knowing the others. That being said, if we wish our classes to be mutually exclusive, we will need some kind of interaction between the different outputs, and this is a pretty standard way of transforming K independent 'logit' outputs into K components of a single probability vector.

2.2.3 Bayesian Neural networks

The loss function we wish to optimize in a classical network typically consists of two parts. The first part, the likelyhood, has to assure our model explains the observed training outputs as good as possible. Then there is usually a second, regularizing term, that is important for neural networks especially, and has to keep them from overfitting.

The addition of such a penalty term in the loss function is equivalent to placing a prior over the weights and maximizing the log posterior (=log-likelihood+log prior + C) instead of choosing the MLE estimate that maximizes the log likelyhood alone. Indeed, the extra log prior term for a normal prior takes precisely the quadratic form of an L2-penalty ('weight decay'). This is just like doing ridge regression is in fact a MAP approximation to a Bayesian linear regression model (cfr for example final part of section 6.2.2 in [14]).

So, while the Bayesian approach will also provide us with regularization, it is not just limited to the MAP point estimate, rather we can use it to obtain a **full posterior distribution over the weights given the data**. Since this quantifies the uncertainty we still have left about the true parameter value given some data, and that is just what epistemic uncertainty is about[31] (whether you translate it into a corresponding predictive uncertainty in y or not), this is the true reason for our interest in this approach in this context of this uncertainty quantification.

For the Bayesian network, we would like to find posterior distributions over weights $\mathcal{W}$ rather than point estimates. Remember we can obtain the posterior probability for $\mathcal{W}$ using $p(\mathcal{W}|\mathcal{D}) = \frac{P(D|\mathcal{W})p(\mathcal{W})}{p(\mathcal{D})}$, and then the predictive posterior distribution at a new input $\vec{x}'$ is given by

$$p(Y'|\vec{x}', \mathcal{D}) = \int \mathrm{d}\mathcal{W} p(Y'|\vec{x}', \mathcal{W}) p(\mathcal{W}|\mathcal{D})$$

[31] well, that depends: we used to capture sample variance as well, while now we are working with uncertainty for 1 particular data set.. So unless we manage to incorporate the uncertainty due to the different training sets that might have been drawn in our prior, we will not be taking it into account! But capturing the range of possible parameter values a priori in the prior requires prior knowledge: when we simply use a completely noninformative prior, that will get overwhelmed by even the smallest amount of data. As the weight prior and data each get in the posterior is proportional to their respective precision (inverse variance), the effect of the noninformative prior is negligible compared to that of the likelihood even for small datasets. That means we are not taking into account model variance in such a situation. Cfr also footnote 25 on p. E-22.

.
With the enormous number of weights to be determined in a neural network, the integral $p(\mathcal{D}) = \int d\mathcal{W}\ p(\mathcal{D}|\mathcal{W})$ in that denominator is taken in a space of enormous dimension, so evaluating it's integrand on a fine enough grid of points to do the normalization by brute force is also out of the question. Then we will really need to know the actual value of the normalization constant in the denominator whenever we want to get more than just a maximum a posteriori (MAP) point estimate for $\mathcal{W}$. For a very simple model where the likelihood is conjugate to the prior we can calculate this analytically, but in general this will be a very complicated multimodal landscape in a huge number of dimensions, so that even sampling the posterior is not without its challenges. This is the field of **Markov Chain Monte Carlo (MCMC)** methods, which is a subject in its own but beyond our scope here.[32] Once one has a sample of the posterior on $\mathcal{W}$, we can use that sample to replace the integrals over the posterior by sample averages from that ensemble.

Another approximative method consists of approximating the true posterior by selecting the closest member[33] from a preselected family using **variational inference**. This turns the integration into a minimization of an objective function.
Say we index our family of functions $q_\theta(\mathcal{W})$ by the parameter(s) θ[34], we find *that* member $q_{\theta^*}(\mathcal{W})$ of the family that minimizes the Kullback Leibler divergence[35] to the true posterior. We can then use (samples from) $q(\theta^*)$ instead of the true posterior to calculate (or sample from) the predictive posterior for Y'. For that we need to find the solution to

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathrm{KL}[\,q_\theta(\mathcal{W})||p(\mathcal{W}|\mathcal{D})] \tag{2.12}$$

with the 'exterior' KL divergence

$$\mathrm{KL}[\,q_\theta(\mathcal{W})||p(\mathcal{W}|\mathcal{D})] \equiv \int d\mathcal{W} q_\theta(\mathcal{W}) \log\left[\frac{q_\theta(\mathcal{W})}{p(\mathcal{W}|\mathcal{D})}\right] \tag{2.13}$$

This is called the 'exterior' divergence because $\mathrm{KL}(q||p)$ diverges if q isn't absolutely continuous wrt the true posterior, i.e. if q wouldn't vanish at some point where p does. Since the support of q has to at least include the support of p, the approximation will envelop the true distribution. In the same fashion we can see why $\mathrm{KL}(p||q)$ would have the true distribution enveloping the approximation. For a more detailled discussion of the (dis-)advantages of the more general α-divergence, which simply combines interior resp. exterior with weights α and $1-\alpha$ respectively, and of the effect the choice of α implies on the approximation made, we refer to for example the Ph.D. thesis of Stefan Depeweg, [6]

Now it turns out (e.g. [5]) that minimizing KL is equivalent to maximizing a lower bound for the log denominator of (2.3), which is referred to in the literature as the 'ELBO' (Evidence

[32] Basically a Markov chain is constructed where the transition probability satisfies detailed balance wrt to the distribution we want to sample from. Because this implies that the stationary distribution for this system will be the sought-for pdf, after a sufficient 'burn-in' period, the process will be visiting the state space with probabilities proportional to the equilibrium probabilities of this chain, i.e. the sought for probabilities. Key assumption is ergodicity so it does not get 'stuck' in a subspace: There shoudnt be any isolated islands in the state space that are not connected to each other.
[33] But how close that closest member actually is, and how good the approximation is a whole other matter..
[34] typically a list of many parameters, but we will drop arrows/cursive for readability
[35] an (asymmetric) 'distance' measure between two continuous pdf's.

lower bound), that means we want to **find the θ^* for which $q_{\theta^*}(\mathcal{W})$ maximizes**

$$E_{q_\theta(\mathcal{W}} \log P(Y'|x', \mathcal{W}) - KL(q_\theta(\mathcal{W})||p(\mathcal{W})) \leq \log P(Y'|x') \tag{2.14}$$

Notice that the KL term is now wrt to the prior, and this acts as a regularizer: while the first term rewards explaining well the data (it is the expected log-likelihood), the second tries to keep q as close to the prior as possible.

A few years ago it was realized by Y.Gal & co that variational inference for a particular family of multimodal posteriors could be approximated by using an extremely simple ad hoc trick, that they call MC dropout. The particular family of multinomial distributions is obtained by replacing the optimal weights vector for a node by a mixture of two Gaussian components with near-zero variance: one with mean at the previous optimal location for the weight, the other in zero.[36] With a product of such a mixture for each node, one can approximate complicate multimodal posteriors, and the sample from the posterior is simply obtained by performing drop out when predicting outputs, i.e. at test time.

In contrast, traditional dropout, consists in bagging predictions from an ensemble of decorrelated nets obtained by randomly eliminating different nodes during training time. During test time, weights are not dropped[37].

Papers

Here we mention a few more key results from papers that were recommended and references therein that seemed relevant.

- Kendall and Gal [3] apply MC dropout to tasks in computer vision, but this paper was less interesting for us because it actually focuses more on regression. There is also a brief section discussing options to apply this to classification, but there they discuss segmentation, ie pixel-by-pixel classification, which is actually not our main concern here. (We want to assign a class to an image as a whole instead).
- Furthermore, according to paper by Kwon et al that followed it [4], they did not approach the case of classification in a very good way, as they modelled the mean and variance of outcomes in the logit space (i.e. on the pre-softmax outcomes) to translate this into a regression problem which they adressed in more detail before. But this means they model mean and covariance independently, using separate output nodes, while the mean and covariance of a multinomial are inseparably linked to each other.

  Kwon et al. address this concern by producing an ensemble $\vec{p}_t, t = 1, \ldots, T$ from the (post-) softmax predictive posterior (using MC-dropout), i.e. the predicted class probabilities. The aleatoric uncertainty we had in equation 2.5 can then be estimated as the ensemble average of the covariance of a multinomial with probability vector p_t, that is

[36] They had to use Gaussians of variance of the order of machine precision because a Bernouilli, being a discrete distribution, has diverging KL divergence wrt to any continuous distribution. But in practice this just amounts to setting the weights from that node zero with given probability, ie MC dropout.

[37] only rescaled by multiplying all weights obtained during training with the 1-dropout probability that was used during training. (e.g. [15] section on dropout, or [16] p11 for linear activations). This is to compensate for the fact that now all nodes are present simultaneously.

$\frac{1}{T}\sum_t[\text{diag}(\vec{p}_t) - \vec{p}_t \otimes \vec{p}_t]$. The epistemic uncertainty is the MC sample variance of the p_t, $\frac{1}{T}\sum_t \text{E}[(\vec{p}_t - \text{E}(\vec{p}_t))^{\otimes 2}]$, where $\otimes$ denotes the Kronecker product, ie if A_i and B_j denote component i resp. j of vectors A and B, the matrix with components $(A \otimes B)_{ij} = A_i B_j$. This also means we only need K nodes and no additional steps in this approach.

- Then there is a paper [2] that describes the state of epistemic and aleatoric uncertainty in any point of feature space by means of a Dirichlet distribution. This Dirichlet has a $(K-1)$-dimensional simplex $\sum_{i=1}^{K} p_i = 1$ as domain, where each point corresponds to exactly 1 possible softmax outcome (a vector of K class probabilities). So it can serve as a distribution over multinomials, and in fact it is the multinomial generalization of the beta prior/posterior for the binomial. Just like for the beta, the relative proportions of the components of its parameter vector $(\alpha_1 : \alpha_2 : \ldots : \alpha_K)$ determine the expected value of the multinomial's class probabilities $\vec{p}$, while the overall scale of the parameters $(\sum_i \alpha_i)$ determines its precision (inverse variance). And just like for the beta prior, the observed numbers of instances of each class can simply be added to the parameters $\vec{\alpha}$ of a Dirichlet prior, to obtain those of the resulting posterior $\vec{\alpha}$.

  So, the parameter vector of this posterior distribution can characterize the complete state of uncertainty by its expected value and precision, which are represented by the location on (alea), and spread over (epi) the simplex respectively (cfr fig 2.3). (To describe distributional uncertainty, in [2] the distinction is made in between the distribution of the train data (described by parameters θ) and that of test data (described by parameters μ), so they can ask μ to be determined by θ for in-domain data but rather completely flat for OOD data.

  What is somewhat special, to accomplish this, they explicitly train the network, not to simply maximize the likelyhood of the observed training data, but rather to minimize the KL distance simultaneousy to a flat posterior over the whole simplex for OOD inputs, and to a sharp distribution (centered on the point that would maximize likelihood) for I.D. inputs.

  While it is a bit subjective to choose the data that should produce a completely flat prior (noise, datasets differing completely or only slightly,..), their approach is guaranteed not to make any overconfident predictions out of domain. But this is perhaps a bit too harsh[38].

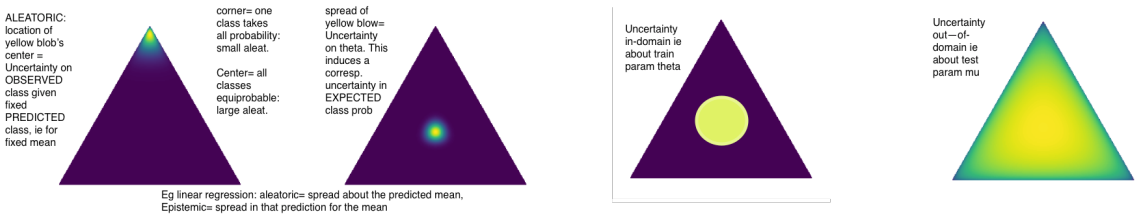


Figure 2.3: Visualisation of epistemic and aleatoric uncertainty from [2]

[38] One gets the impression that the OOD samples that are used might have a large effect, and can't help but wonder what the model will predict for a noisy specimen of an existing class. It is after all impossible to train the network on everything that is not an existing class, and the danger exists that the model will learn to tell ID and OOD apart based on irrelevant artefacts that differ from dataset to dataset but have no relation to the objects represented. Whether data is OOD is to some extent relative, dependent on the smoothness we assume, as no dataset is ever dense in space, there are always voids/lacunas. For data still showing some resemblance to the original data, a larger spread around the prediction seems more reasonable than a completely flat posterior.

They arrive at this from a Bayesian viewpoint, but because the uncertainty is quantified in terms of the observed pattern of predicted class probabilities, it seems of secondary importance whether this pattern arises due to uncertainties in prior and given the data (ie in a Bayesian setting), or due to variation in the data used for training (frequentist). Therefore I think we could quantify the epistemic uncertainty in $p(Y|\vec{x})$ in a frequentist setting in a similar fashion, for example by bootstrapping the training data to produce an ensemble of predicted class probabilities at each point, and looking at the parameters of the Dirichlet distribution that best describes the observed ensemble at each point. This is what I tried to do eventually, and an onset of this approach is presented in the next chapter.

2.3 Cautious classification

2.3.1 Motivation

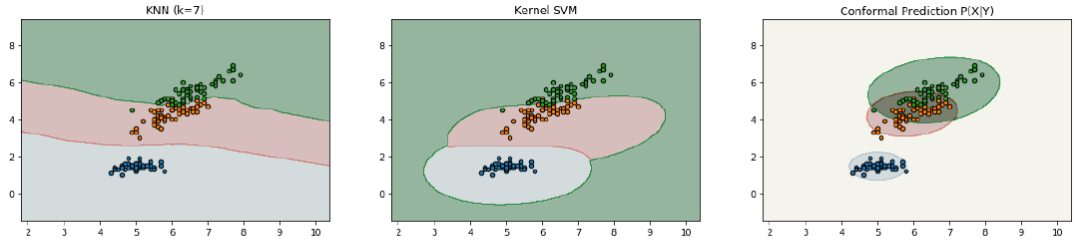
When doing classification, one typically only takes into account the *relative* chance that an instance at a given point X comes from class i, as compared to that for the other class(es) $j \neq i$. But that way, we ignore the knowledge we have about the marginal $p(X)$, which is a common denominator for all classes i in $P(y = i|X) = P(X|y = i).P(y = i)/P(X)$. Up to some potential weighting factors $P(Y = i)$, this means we simply select the class with the highest (weighted) conditional PDF $P(X|Y = i)$, *however small that may be*.

This makes sense when we have confidence that our model assumptions, such as the classes known to exist, hold throughout the region of feature space in which we wish to classify. As probabilities are normalized, all other classes being even much more unlikely implies that it is likely to be of class i by elimination. But in a setting where one can't exclude the existence of a so far undetected class, such conclusion would no longer be justified[39]. Worse yet, provided the other classes are orders of magnitude less likely to be found there, we would arrive at this conclusion with very high confidence. Furthermore, the classification in regions where there is no data will be highly unstable for flexible models, as adding a single point can lead to very different predictions far from the training data.

Likewise, on the other end of the spectrum, we may have regions where multiple classes are very plausible, and simply classifying each point into its most likely class obscures the fact that this is only the model's best single guess. Even if the fitted model perfectly describes the truth, if $K >> 1$ our best guess will typically still likely be wrong.

As illustrated in Figure 2.4, both of the above points imply that there is a very large uncertainty outside of the training data domain, or in regions of overlap, but this uncertainty is often not at all apparent from a traditional classifier. Certainly in safety-critical applications, we need our classifier to tell us which predictions we can trust and which are little more than a wild guess.

[39] Even without the possibility of other classes, it is often imprudent to assume that the functional relationships observed in a limited region of feature space are applicable throughout the whole space. So even without invoking the potential existence of other classes, we can't be sure our estimated probabilities far from the original domain are correct.



(a) (b) (c)

Figure 2.4: Classification boundaries for different methods for the Iris dataset. For the conformal prediction method (c) (with $\alpha = 0.05$) the overlapping areas are classified as multiple classes and white areas are classified as the null set. For the standard methods (a-b), the decision boundaries can change significantly with small changes in some of the data points and the prediction cannot be justified in most of the space. (Figure and caption taken from [7].)

So, it could be useful if we could somehow take into account that the prediction made in some regions is only one of several plausible classes, or that none of the known classes are actually plausible at a certain location.[40].

A way to accomplish this is by allowing the prediction of a **set of labels**. By having the liberty to decide for each class individually whether it is in or out of the predicted set, the reasoning 'by elimination' disappears, as the interdependence between the prediction for each class vanishes.

Now, we can model the shape of the region in which class c is amongst the predictions independent of data points for other classes, using only the data from class c in the estimation of $P(X|Y = c)$. Because of this independence, we are now able to guarantee a confidence level: for each class i, the chance of excluding a point that truly belonged to that class is at most a given value α_i, the conformal confidence level.

A more practical advantage of this independence is also that class densities can be estimated in parallel, and when new class(es) are added at some point, one only has to estimate the density for the new classes to update the classifier, the old classes remain unaffected. Note however that when the number of classes is large and the amount of data per class small, sharing certain parameters between classes is a common way of reducing the variance on the estimates of those parameters (eg. in LDA we have a shared covariance matrix). In this situation the above remark about independently estimating only the new class no longer holds[41].

2.3.2 Conformal Prediction

Here we will use a **distribution free** approach that constructs prediction *sets* to guarantee having the right class amongst the predicted set with a predetermined confidence level

[40] For a discussion on some of the literature on these so called possibility sets, see the tutorial [1]

[41] In that sense using GMM density estimation differs from [7] which mentions only non-parametric methods for density estimation. There each class can always be trained independently, but they will have a much larger variance when the number of instances per class becomes small. It should also be added that the proof of the finite sample conformal guarantee they refer to, in [17], also assumes a KDE density estimator.

$1-\alpha$. Guaranteeing this level is possible because we can have as many classes in the prediction as needed to achieve that level, even all known classes if necessary.

Say the chance of falsely excluding a point from its true class i is α_i, potentially differing from class to class. Even when we don't know the true class i of a point, we can still safely say the chance α of falsely excluding it is bounded by the largest of the individual chances: $\alpha = \max_{i \in \{1 \ldots K\}} \alpha_i$.[42].
That means we can now focus on making sure that the chance of rejecting a point *given* that it really comes from class i is at most α_i. For that we will use the method of *conformal prediction*, which I outline briefly now.

Firstly, we require the choice of a *conformal score*. This is a function $\psi(z, \mathcal{D})$ of an arbitrary point $z = (x, y)$ and an (unordered) data *set* $\mathcal{D} = \{(x_i, y_i), i = 1 \ldots N\}$ that reflects in a way the dissimilarity of the new point w.r.t. a dataset D, as will become clear from the examples below.

For each of the points (x_i, y_i) in the extended set $\mathcal{D}' = \{\mathcal{D}, z\}$, the number $R_i \equiv \psi[(x_i, y_i), \mathcal{D}']$ should express in some way how bad that point fits in with the group (residuals, distances, dissimilarity indices, etc), the actual value not being as important as the relative ordering. Here the $N+1$-th point $z = (x_{N+1}, y_{N+1})$ is an arbitrary one in the space (X, Y), ie this defines the score with respect to that dataset $\mathcal{D}$ everywhere.

The fraction of $N+1$ points in this extended set having a score at least as extreme[43] as the new point is then used as an empirical p-value for the hypothesis that the point is just like any of the others (in some application dependent sense), $\pi(x, y) = \frac{1}{N+1} \sum_{i=1}^{N+1} \mathrm{I}(R_i \geq R_{N+1})$. This definition assures that a dataset of N points only allows us to draw conclusions at a significance of $\pi \geq 1/(N+1)$.

That means, all points in space that are 'odder than the oddest point in the dataset' (in terms of having a higher conformal score), would receive this p value $1/(N+1)$.
Provided our dataset is large enough however, $\alpha.(N+1) >> 1$ (with α the desired confidence level), we can then select from the $N+1$ scores the lowest one that still keeps at least a fraction 1-α of the conformal scores below it[44]. Depending on the context, many choices are possible for ψ, and the examples will best clarify the meaning of this procedure.
Consider for example detecting outliers in linear regression? Well, the residuals of each point wrt to the linear relation fitted to ALL $N+1$ points are a sensible indicator on the basis of which we might flag a point as anomalous. Therefore, such a residual could be a good choice of conformal score in that situation[45]. In our density estimation context on the other hand, it makes sense to consider the PDF fitted to all points of one class, evaluate it in

[42] This is not an instance of multiple testing as the classes are mutually exclusive, we can make 1 error at most. When the point pertains to some other, yet unknown, class, we can't even make any errors, even if we fail to reject it from the known classes. Remember that failing to reject a wrong class is not punished here.

[43] depending on how scores evolve as points become more dissimilar, without loss of generality we'll suppose they become larger here, but for cases where that isn't the case we can adapt our score so it is. For example in case of a density they decrease as they become further apart and we could, instead of using the density as score, use $-\log$ density to change this around.

[44] This is the j-th highest score, where j is defined by $j < \alpha.(N+1) < j+1$, i.e. $j = \lfloor \alpha.(N+1) \rfloor$.

[45] Even though in practice we typically use a more sophisticated method that also takes into account the leverage due to the location of X values and leaves out the point under scrutiny in linear regression outlier analysis. But that's because there we also need to know how much a point affects the resulting line in order to decide whether to exclude it or not. Here we only need an *ordering*, and this simpler order statistic would do fine.

these same points, and use that empirical distribution of PDF values to flag some points as more outlying than others.

Because z has to be included in the extended dataset $\mathcal{D}'$ used to calculate all scores, the first N scores $R_1, \ldots, R_N$ also have to be recalculated each time the new point z changes, which is computationally prohibitive in many situations. We can use the original dataset to fit the density $\mathcal{D}$ instead of $\mathcal{D}'$, but then the confidence level is guaranteed only asymptotically as the number of points in each class approaches infinity, and under the additional assumption that our density estimator is consistent. However, an alternative way to recover our finite sample confidence level, is to use a separate data set for determining the threshold than for learning the density. This is what we will do here.

So we use as cutoff the α quantile of the empirical distribution of the fitted training density evaluated at validation points. That is, **the largest validation density that still keeps at most a fraction α of densities in validation points below** or equal to it:

$$\hat{t}_i \equiv \sup_t \left[t \quad | \quad \frac{\sum_{j \in V | Y_j = i} I(\hat{p}(\vec{x}_j | Y_j = i) \leq t)}{n_{V,i}} \leq \alpha \right]$$

where in the sum j runs over *validation* points with class i, of which there are $n_{V,i}$, and the hats refer to a PDF estimated only on *training* points with class i.

Note that this 'right' quantile definition $p \rightarrow \sup\{x | F(x) \leq p\}$ is different from any of the 9 choices available in R's 'quantile' function, for which a quantile is never smaller than the minimum of the set. Here however, once α drops below the inverse of the number of validation points in that class, we will accept the whole feature space in that class, because there are no points we can safely exclude at the requested confidence level with a validation set of that size.[46] That is why on plots versus α, in regions where either $\alpha.n_{V,i}$ or $(1-\alpha).n_{V,i}$ are not much larger than 1, the results will be based on the behavior of a few validation points only, and we should be careful not to overinterpret any trends in those regions.

It is because of this conservative definition of the quantile that we can keep our (distribution free) conformal guarantee even for finite sample sizes, and not just as some large sample approximation! The price we pay for this guarantee, however, is that the accepted regions for each class may become too wide to be useful. (This is probably why the approach does not give good results when the model assumptions are not satisfied, because the classes become too wide when there is too much bias.)

Now, we will need to estimate the density to be used in this conformal score. It turns out that the classifier depends only on the *ordering* of the predicted densities, not on their actual values. As the validation points are fixed, the ones that have the lowest densities and are rejected will stay the same if the ordering does, and so will the class boundary.

In conclusion: the PDF is fitted on a training set (for that class), then **that density is evaluated in all validation points** (of that class), and the α-th **empirical quantile of the *validation* PDF values will be used as the threshold**. All test **points with a predicted density below this threshold are then rejected from this class.**

[46] Since the fraction of validation points for which the density isn't larger than the smallest validation density is already $1/n_{V,i}$, there are no validation points for which this fraction is even smaller. Then the supremum is over an empty set ($-\infty$), and all points in feature space have a PDF larger than that.

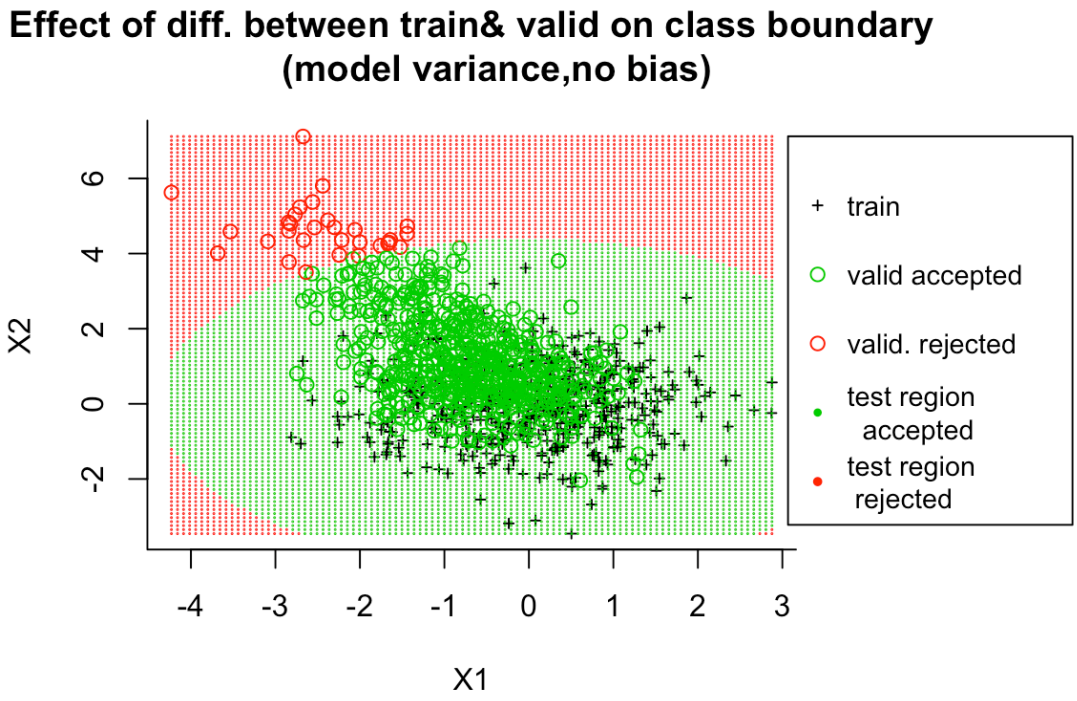


Figure 2.5: Extreme where the difference between validation and training data is gravely exaggerated to show how the region that is eventually accepted is more cautious thanks to the splitting, also in other directions than the original discrepancy between train and validation.

Because train and validation data come from a simple (stratified) random split,any difference in distribution between them is due to model variance. Any difference between the fitted training density and the training data on the other hand, is due to model bias. So the difference between validation data and fitted training density contains both contributions, and evaluating the latter at the former will in fact create a 'buffer' against epistemic uncertainty of both types. Namely, as we can see in Figure 2.5, the validation cloud will be trimmed by a fraction alpha by some training contour, yielding an asymetric region of accepted validation data. For a new (test) point, the region of acceptance would then be that *whole* same ellipse. So, the acceptance region has the symmetry of the trained distribution, but its scale is increased[47]by any discrepancy between trained PDF and actual data cloud due to bias or model variance:

- Finite sample size will cause the centers of training and validation PDF to be spread around a common true center, and the distance between them will make the average Mahalanobis distance from validation points to the training center typically larger (and PDF values lower). The α-th quantile will then also be lower, and the accepted region wider, not only in the direction of the difference between train and validation centers, but in all directions where the trained PDF decreases. **The shape of the accepted region is already fixed by the symmetry of the training density contours, only it's scale will be increased by any discrepancy between train and validation sets**, as shown in Figure 2.5.

[47]at least wrt to the most compact of train and validation cloud separately

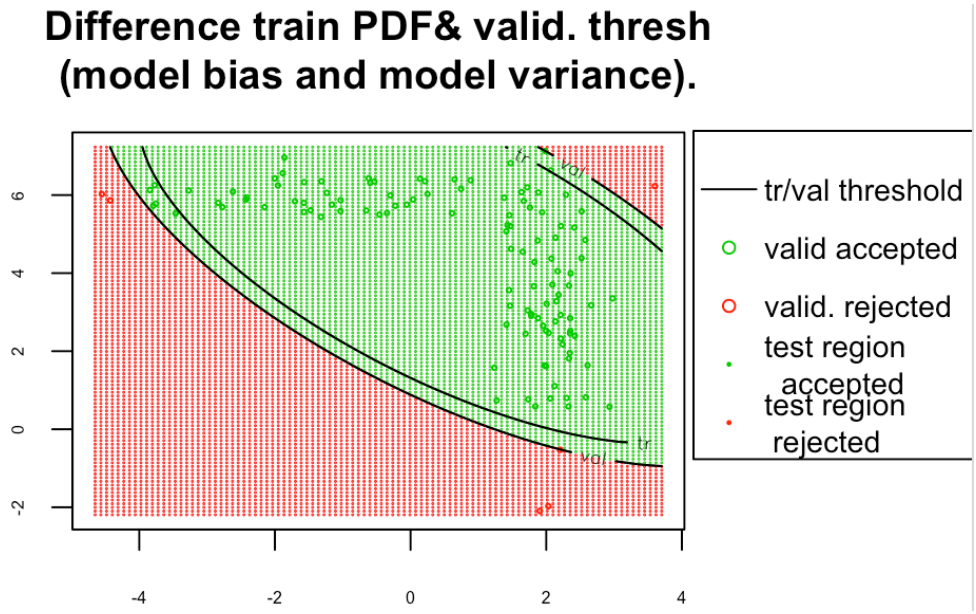


Figure 2.6: Example with data for each class being drawn from a 2 component mixture (a single class is shown) , while a single Gaussian component PDF was fitted for that class. Due to model bias, the lower part of the ellipse is also accepted in the class, in spite of not having any training data there. Due to model variance, the ellipse chosen eventually is still wider than it would have been for the training data alone. So for the training data, even less then a fraction α will be rejected.

- On the other hand model bias would also cause the fitted center of the training distribution to move away from that of the true PDF, but in some particular direction, to try and make up for the distorted shape of training density contours. (Think of a Gaussian placing itsself at the center of what is in reality a 2 component gaussian mixture).
  In that case, the validation data, which will be distributed according to the more complicated shape, will contain an underrepresentation of high PDF values (at the estimated center between the components) and an overrepresentation of lower ones (in the two actual centers). Again that will make for a lower threshold and wider class boundaries. The deviation will be in a certain direction, but the accepted regions always have the symmetry of the models from the hypothesis space we use.
- As a third potential contribution to epistemic uncertainty, one sometimes studies cases where test distributions are explicitly different from train and validation data because they are collected in different circumstances. Our cautious approach would simply reject these regions, but **only provided our density estimate works well**. As we can see on Figure 2.6, using a parametric approach in the presence of bias can lead to large regions where $p(x)$ is estimated to be larger than the threshold, although they are clearly OOD. Therefore I think we cannot rely on this feature unless we use an unbiased distribution free density estimation method.

The bottom line is that any other distribution than the true distribution will result in lower thresholds than if the dataset had also been the most dense where the predicted density was highest. The fact that this extra data splitting guarantees the confidence level for finite samples is proven more formally in the references in [7][48].

[48] eg. [17] for the finite sample, distribution free guarantee, but there it is assumed KDE is used (with some conditions on sufficiently small bandwidth), so I'm not sure this applies for GMM.

Finally, it is important to stress that this method of conformal classification is completely independent from the focus we lay here on $p(x|y)$ (and to which we refer as 'cautious'). Namely, before that, existing conformal methods focussed on the more traditional $p(y|x)$ instead. The great advantage of using $p(x|y)$ however, is that it allows us to tell apart these very different regions of uncertainty: those where the null set should be predicted (epi) and those where multiple classes are to be predicted due to strong overlaps (alea).

For, when we use $p(y|x)$ in a conformal scheme, the idea is to exclude instead a class $y = i$ for a point x when its probability $p(y = i|x)$ is smaller than a given threshold. But in the context with potentially hundreds or thousands of classes ($K >>$) for which this method is primarily devised, that would mean predicting also the null set in regions of maximal aleatoric uncertainty, $p(y = i|x) \sim 1/K << \quad \forall i = 1 \dots K$, even when $p(x)$ (to which $p(y|x)$ is oblivious) is very high in that region![49] So we see how using $p(x|y)$ to build our conformal score instead allows us to distinguish regions where all classes are equally likely and we predict the full level set, from those where no class is plausible and we predict the null set, because nothing sensible can be said with certainty.

We remind that the method is distribution free at finite sample, and it manages to do so because it cannot make errors on test points that do not truly belong to the class in question. That allows us to assume a new test and $n_{V,i}$ (known) validation points are IID, and in that case the chance to find the new point 'odder than the oddest of $n_{V,i}$ validation points', is simply 1 out of $n_{V,i}$+1.

2.3.3 Using cautious classification to exclude points of high epistemic uncertainty

We hope to see that excluding points with very low $p(x)$ will increase our performance (the precise form of which is to be defined further on). The hope is that perhaps these points were not accurately described by the model fitting the majority of the data, and abstaining from prediction for them is the right way to signal that we are seeing something unfamiliar to the model there. Note that this remains true even if it does not improve performance: Even if the data behaves the same way out of domain as it did in domain, we have no means of checking this, and in safety critical applications it can be preferable not to predict than to make uncertain predictions. So even though a cautious approach will only be rewarded when the real relationship is more complex than what the model suggests at the border of the domain, in some applications it can still remain a necessity regardless.[50].

The original objective set out for the thesis was to visualise the dependency of the traditional accuracy and some custom performance measures for set valued prediction on this confidence level α for different datasets, using normal densities/mixtures with varying degrees of complexity for each class. It was not the intention to select an optimal α, as no cost of rejection of a data point relative to missclassification was specified. Because the method

[49] In my own personal opinion, this is a rather artifical argument: how likely is it that thousands of classes would all be predicted to be equally likely? As K becomes large, the fraction the K-dim. simplex that is near the edges/corners will approach 1. So the chance that not any of K classes has a significantly larger than average probability becomes virtually zero. On the other hand, the fact that $p(y|x)$ is oblivious to $p(x)$, and we don't want our method to be, seems more than convincing enough.

[50] unless we manage to encode the larger cost of missclassification properly in the perfomance measure in those applications as well.

appeared to be fit mostly for distributional uncertainty, I also used it to test in some simple situations whether this method was able to reject unknown classes (cfr. section 3.3). To end this chapter, we review here the three types of epistemic uncertainty and how the cautious method could or could not detect them:

- First, there is the possibility that test and train data stem from different populations. The method of cautious classification is well suited to address this form of epistemic uncertainty especially. **A large rejection rate on the test set could be an indication of this.** We will look a little more into this in section 3.3.
- Second, it is possible that the density we learnt from the training points doesn't properly represent their true distribution. This could happen for instance when we select our density from a parametric family not flexible enough to represent the training data. The 'difference' between the prediction of the best[51] member of that family, and that of this true model, is the model bias. Here notably, our bias could stem from the fact that we assumed the density was a Gaussian or a mixture of Gaussians (the latter having less bias than the former). This component does not depend on the training set D nor on the test set (as these are averaged out), but only on the choice of hypothesis space. For flexible enough models, it will vanish, but then we will typically have larger variance (see below), so a trade-off is to be found there.
  Since this must show up in the discrepancy between the training points' distribution and the fitted training density, it's presence could probably be revealed be using distributional tests. Using non-parametric density estimation (or more flexible parametric models) could also strongly reduce this component (at the cost of an increased variance).
  We will ignore this component most of the time, because there isn't really any immediate link with the methods we consider, **but** it should be noted that it **may be very significant when we use normal classes with only 1 component**.
- Finally, even if the model inferred from our training data will on average (over all possible training sets D) be correct, there will still be deviations due to the particular realisation of training set D we based our inference on, and this is called model variance. **A large difference between the rejection rate on the training set (if we were to score that afterwards) and the nominal rejection α (on the validation set) could be an indication of this.**

Because the cautious method appears a little less suited for the last two contributions, I have done some separate experiments in Appendix E,where we turn off model bias and distributional uncertainty by drawing data from classes that are truly mixtures of gaussians and estimate densities using a mixture with the right number of components. We can then focus on the model variance uncertainty component as a function of sample size specifically.

[51] ie expected, over all training/test sets

CHAPTER 3
RESULTS

3.1 Density Estimation

In this chapter we present the results from what was originally the topic, trying to increase performance by eliminating the most anomalous points of each class. Because there are many more technical points we couldn't treat in detail in the previous chapters, we will try to cover them here from a more practical angle in the first sections. This includes for example the density estimation, and the feature extraction method we used to convert the image data into more traditional data. We also have to decide on the performance criteria we will consider.

For an unbiased model, the epistemic uncertainty is due to what we called model variance (sample variability due to a finite amount of training data) and distributional uncertainty (O.O.D. generalization of a model learnt I.D.). But this distinction between ID and OOD is rather fuzzy, as even non-parametric approaches have a characteristic length scale over which we reasonably assume the behaviour not to change dramatically, like a bandwidth (KDE), or length scale (GP). Without such assumptions of smoothness, the domain would be limited to those exact points where we have made training observations. Therefore, it is clear that the domain boundary is not something sharp, but rather a gradual decrease in data density, and it would seem that $p(x|y)$ can be a good proxy for distributional uncertainty at x, and that the method of cautious classification (which completely rejects points when $p(x|y)$ is low for all y) is well suited to exclude points where such uncertainty is largest. This is what we will try in this chapter.

Although we have neglected it most of the time, we should stress the importance of the unbiasedness in this reasoning: for the points where the predicted density is lowest to actually be the most anomalous ones, we need our model to be unbiased. When the training density does not properly match the underlying distribution, the effects can be dramatic, and the danger of this happening is larger for parametric models that make stronger distributional assumptions. For non-parametric models, with enough data we can at least expect to find the right training density, and this method would correctly identify all anomalous points. But we will need a lot of data to counter the increased model variance (that comes with the increased flexibility and hence lower bias of such models), and because they base their predictions only on nearby data, this can again be translated to having a sufficient training data density $p(x|y)$ for each class.

Therefore, while the scope of the uncertainties captured by this method may differ somewhat[1], it seems that rejecting regions where the observed training density was too low could be a useful way to increase the confidence in our predictions. This is what is proposed in [7], where they favor KDE for theoretical considerations and kNN for practical implementations. Ironically, while these methods are the ones that should allow a better assessment of total epistemic uncertainty (i.e. including distributional) in each point of feature space, they will typically have much higher uncertainties because of the increased model variance.

Because that lower model variance is expected to be important in higher dimensions (and a number of other reasons discussed in the following section), we started with Gaussian Mixture Models for density estimation. Later we added also kNN and KDE and experimented with a method called Projection Pursuit Gaussian Mixture Models Genetic Algorithms [18] (PPGMMGA from now on). We now discuss this density estimation in a bit more detail.

3.1.1 Mclust For Gaussian Mixtures

Because high-dimensional density estimation is far from a trivial topic, and the quality of our density estimation is key[2] to the remainder of our results, there is an important choice to make here. On the one hand, we argued that using the PDF as a proxy for epistemic uncertainty means we preferably use a DE method that is sensitive to the local data density, like KDE, kNN,.. On the other hand, we know we need a **generative** model. Some models work very well in high dimensions but simply classify, and these were not useful, since we require not only $p(y|x)$ but also an explicit form of $p(x|y)$. Of course working with a regular grid is not feasible in dozens of dimensions, so the most attractive is a PDF for which we have a parametric form, allowing a swift and precise evaluation of both $p(x|y)$ and $p(y|x)$ in any point of feature space.

Gaussian Mixture Models (GMM) (e.g. [19]) are also useful for anomaly detection, as demonstrated in [8] where they use a neural network with radial basis functions to learn the parameters $(\vec{\mu}_i, \mathbb{\Sigma}_i),\ i = 1, \ldots, K$ of a mixture of K multivariate (diagonal) normals, imposing a penalty on the total volume (apart from a typical weight decay penalty). They can then flag points with low density as anomalous.

Because of the similarity with our situation, the idea was that I could start with a Gaussian mixture or even normal distributions (LDA/QDA) to estimate the density of each of our classes separately. Because of the potential for generalisation towards multiple components, the option to fully control the covariance structure, and the selection of model complexity based on B.I.C., I arrived at the ***Mclust*** package for R. The features of this package that are relevant to us are introduced very biefly in Appendix A.

[1] i.e. for parametric models, we would be neglecting bias and model variance this way. Because of that I also tried some experiments that focus more on model variance, but in much lower dimension, in Appendix E.

[2] Actually when we look at the results, these turn out remarkably stable throughout different DE methods. The authors of [7] already remarked that only the ordering is important, but it still seems surprising that this does not differ much more between methods. Perhaps the fact that we have only a global picture of the performance on the dataset as a whole also hides more regional differences.

3.1.2 Non-parametric approaches (KDE , kNN) , and PPGGMGA

The assumption of normality in high dimensions is very restrictive, much more restrictive than marginal normality of all those predictors. In spite of that restrictiveness, even LDA can have a large number of parameters when D becomes large (cfr. Table A.1). Even more problematic in my opinion is **the extreme non-robustness** of fitting a single Gaussian: there will always be outliers, and because these large deviations will dominate the (quadratic) loglikelihood, they will have a large effect on the result. Ironically enough, we intend to discard just those points in a later stage that are now going to determine our result! Because of these, and arguments mentioned before, we also tried non-parametric density estimation.

k Nearest Neigbour Density Estimation (kNNDE)

Was the method recommended in [7]. The method estimates the density based on the number of neighbours it finds within a given (Euclidean) distance. While results in low dimensions sometimes stay somewhat below other methods, this method appeared to be more robust to increasing the number of features.

KDE

The paper on cautious classifications [7] also applies KDE in 2048 dimensions, but unfortunately does not specify which method they use for this in practice. There are many packages for kernel density estimation in R. The most commonly used one (also used in ggplot) is '*ks*', but the manual there states it should be used only up to 6 dimensions. Although it appeared to work fine in 8 and 10 dimensions as well, it wasn't clear if we could rely on the results given that warning.

Eventually[3] we ended up with the function 'kepdf' from package *pdfCluster*, which could be used in arbitrary dimensions, and this seemed to give comparable results to mixtures. This package also allows us to choose an 'adaptive'[4] bandwidth (which becomes larger in regions where the density is lower), with a parameter α that controls the flexibility with which this happens ($\alpha = 0$ corresponds to fixed bandwidth). We tried the default value $\alpha = 0.5$ as well as 0.2 and 0.8, and compare this to a fixed bandwidth (which still use different values in different dimensions, but the same for each observation). We tried both the Gaussian and t7 kernel but differences were small and only Gaussian kernels are shown on the figures.

Because it seemed impossible to start tuning over all possible fixed bandwidth matrices in 10 dimensions (even if only diagonal ones), to look at the effect of bandwidth, we eventually extracted the optimal fixed bandwidth, and multiplied it by constant scale factors, to make also curves for 0.2, 0.8, 0.9, 1.1, 1.2 and 5 times this value. (For 0.9 and 1.1 results are usually comparable, but for 0.2 and 5 times the optimal bandwidth, the results are far worse than for all other models (KDE and non KDE) considered. In the 2D maps from Appendix E

[3] I considered a few other packages specifically for KDE in high dimensions but they were either difficult to understand or produced disappointing results.

[4] cfr. [20], sec 5.3.

we also tried using some extremely small and large fixed values (0.1,0.1) and (1,1) , to see the effect of the bandwidth, and there we can see how much this affects the results.

Projection Pursuit Gaussian Mixture Models Genetic Algorithm

Projection Pursuit (originally proposed by Kruskal [21] and succesfully implemented by Tukey and Friedman in [22] [5]) is an unsupervised dimension reduction method, which finds the most 'interesting' directions, in a sense that can be problem dependent, by maximizing some objective function (the index) that is calculated for each possible direction to project on. Without making use of the labels, this method often manages to achieve impressive class separation in only 2 dimensions, where techniques like PCA or LDA (which even uses labels) fail to do so.

The author of the Mclust package developed a version of this algorithm called Projected Pursuit Gaussian Mixture Models Genetic Algorithms (PPGMMGA) [18], which fits a GMM and finds those directions in which the largest deviations between the projected GMM (still a GMM) and a Gaussian is observed. The reasoning is that directions in which the projected clusters are normal are to be expected, and that those where this is not the case are therefore those that contain the most information[6].

The negentropy is used as a measure to quantify the departure from normality, and because this projection index remains equal for a Gaussian of given dimension in any basis, it can be used for a fair comparison of the deviations from normality between all subspaces of a same dimension. By maximizing this index we can find directions in which the departure is largest, to use them for visualisation.

Originally I had assumed that this method would then fit a GMM model in the projected space[7], which should be less sensitive to outliers and to overfitting[8]. However it turns out the densities returned by this method are fit in the original space before projection. In that sense it is just a version of GMM that does trimming, and class-wise standardisation of the data first[9] [10].

[5] A very transparent 2-page introduction may also be found for example at https://sites.stat.washington.edu/wxs/Visualization-papers/projection-pursuit.png.

[6] A normal distribution is after all the shape white noise takes (by definition of course, but there is a reason we define it like that), and the normal distribution maximizes entropy (or minimizes information) for a given variance.

[7] What is a bit confusing is that it stores an object 'call' which explicitly shows a densityMclust call using 'Z' (the scores, which are 2D only) as data, so it appeared as if the model was fit there.

[8] While the projection of the GMM in the original high dimensional space will also be a GMM in the low dimensional space, I think it will be a different one from the one we would have obtained by first projecting and then fitting, as outlying values in directions with projection zero would vanish, and a 10dimensional outlier that had a tremendous effect on the likelihood might have much less of an effect in 2D.

[9] Sphering (ie also decorrelating the data so it also has zero correlations is customary for PP but but NOT done in the ppgmmga package as explained in the documentation

[10] Still, I think we could perfectly extract the low dimensional features, and do the density estimation ourselves over there. We could then project any test point onto the (different) subspace of each class to determine low dimensional densities for each class. While we wont be able to get the marginal density at that point (which we need for $p(y|x)$) because we can't add up $p(x|y)$ that live in different subspaces, this would still allow us to reject points that are unlike any known class, as we do in the cautious approach. Due to time constraints I have not tried this anymore.

3.2 Performance measures

We would like to use the method of cautious classification introduced in Section 2.3 to exclude the most anomalous points, where distributional uncertainty is larger. One might hope that we will see the accuracy on the remaining (Non-Rejected, NR) points increase initially with increasing conformal confidence level α[11] and then reach a plateau when all the 'abnormal' points are excluded, so that this would allow us to choose a good α value that discards the 'difficult' points and keeps the rest[12]. Not only would this require the 'consumption' of our test set, it also hides the assumption that the hardest points to classify are always the peripheral ones, while I'm not sure this is always appropriate, for the following reasons mainly:

- The identification of (un)certainty with (in)accuracy is based on an aleatoric view of uncertainties. For these, being less certain really means that there is more chance our prediction is wrong, because they are based on $p(y|x)$.

  For epistemic uncertainty however, we can not really make this link, as this concerns how sure we are of our *model*, but another model doesn't necessarily make wrong predictions. That is why we tried to express these uncertainties in terms of variability in predictions, either due to the different parameter values learnt from different training sets for a same model (model variance), either from different models (distributional uncertainty).

  So the fact that distributional uncertainty is large in the periphery does not guarantee us that these points will necessarily be misclassified more often, it merely means we shouldn't trust the extrapolation blindly. Whether this will also be rewarded in terms of accuracy will depend on the complexity of that particular true distribution[13]. Not seeing the accuracy increase when we reject peripheral points, doesn't mean that we were any less uncertain a priori.

- Even if we consider it a given that our density estimation is unbiased and robust to the outliers it is supposed to detect, and that epistemic uncertainty is higher in the peripheral regions, then this would still only give us an **ordering**, without any clue about how large the uncertainty actually is.

- Last, this assumes aleatoric uncertainty isn't larger at the center than at the periphery. This is immediately confirmed using a toy '+' shaped dataset where both classes consist of a narrow elongated ellipse with common center but with perpendicular long axis, a template case for QDA (cfr. fig 3.1).

[11] The fraction of data trimmed (from the validation set)

[12] That such an approach can be indeed useful to detect large distributional uncertainty (new classes etc.) is discussed in section 3.3.

[13] As this uncertainty expresses the chance of our *model* assumptions remaining accurate, to quantify this a Bayesian could perhaps also specify a prior over all such possible distributions in and out-of domain, and calculate a posterior probability that the actual data distribution continues the same behavior also O.O.D. But that would probably involve shifting the problem to a whole new set of higher level assumptions. An alternative, that we have tried, is to absorb this uncertainty in the model variance component by using a more flexible class of models and letting the training data decide, so it becomes part of model variance. (cfr also discussion App. E.)

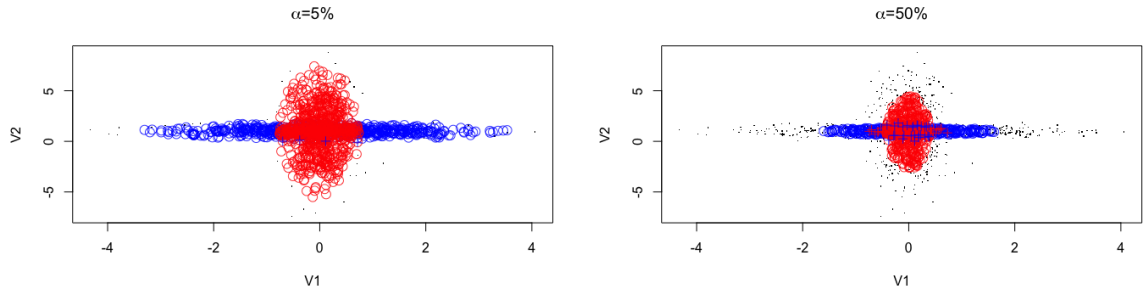


Figure 3.1: Effect of increasing the conformal confidence α on a toy dataset. In black dots we have points rejected from all classes, and hollow circle(s) represent predictions. If the true label is not amongst the set, we plotted a '+' in the true color inside the circle with predicted color(s).
This illustrates how we should not expect this method to always yield a higher accuracy when the most anomalous points are excluded. The assumption that the points in the highest density region of each class are the easiest ones to classify requires, for starters, that the aleatoric uncertainty is homogeneous.

As the conformal confidence level α increases, a larger fraction of remaining points comes from the difficult-to-classify center, and performance will decrease. This is of course because of the overlapping highest density region (HDR) of both classes.

While this is an extreme example, it does show that merely using overall accuracy cannot reveal whether the excluded points were those with highest epistemic uncertainty, because we still have the aleatoric uncertainty as well. Therefore, we would **ideally like to 'switch off' the aleatoric uncertainty by allowing set-valued predictions**. We then consider a prediction as accurate when the true label is among the predicted set. First we will have to introduce some notation, so we can define our performance measures.

Notation and Definitions

- Tr, V and T are train, validation and test sets respectively.
- N_T the number of points in the whole test set.
- K the number of classes (in the training set).
- D The number of features, i.e. the dimension in which we have to do D.E.
- Y_i is the true label of observation i. Its feature vector is $\vec{x}_i$.
- We reserve the notation $\hat{Y}_i$ for the **classical** (single valued) prediction, i.e. the class with largest conditional probability which for balanced classes is the also the one with largest density:
$$\hat{Y}_i = \underset{c=1\ldots K}{\operatorname{argmax}}\, p(c|\vec{x}_i) = \underset{c=1\ldots K}{\operatorname{argmax}}\, p(\vec{x}_i|c)$$
- S_i refers to the set of labels that were accepted for an observation i, ie the **set valued** prediction at point $\vec{x}_i$. Therefore $|S_i|$ is the cardinality of that set, ie the number of classes the point belongs to (which depends implicitly on the confidence level α).

- NR stand for the collection of non-rejected points, $\{i \in T \quad | \quad |S_i| > 0\}$, ie the points accepted in at least 1 class. When this is used as a subscript, for performance measures, it means that only points with nonempty prediction set are considered in the numerator and denominator.
  We use $|NR|$ as a shorthand to denote the number of non rejected points (which is just $N_T.(1-RR)$), as we can then write such averages over NR as $\frac{1}{|NR|}\sum_{i\in NR}(\dots)$

Then, we'll usually consider the evolution of the following quantities as a function of the confidence level α or rejection rate RR:

- RR: The Rejection Rate i.e. the fraction of points belonging to no class at all.

$$RR = \frac{1}{N_T}\sum_{i\in T}\mathbb{1}_{\{|S_i|=0\}}$$

- $Prec$: The precision, defined as the total fraction of all used **labels** that is correct. Writing Pred_j for the set where j is (amongst) the prediction(s), and PPV_j for the positive predictive value of class j[14]:

$$\begin{aligned} Prec &= \sum_{j=1\dots K}\frac{|\text{Pred}_j|}{\sum_{k=1\dots K}|\text{Pred}_k|}PPV_j = \frac{\sum_{i\in NR}\mathbb{1}_{\{Y_i\in S_i\}}}{\sum_{k=1\dots K}|Pred_k|} \\ &= \frac{\#\ correct\ labels}{\#\ labels} = Fraction\ of\ correct\ labels \end{aligned}$$

  where we used the fact that only 1 label can be correct per observation to replace $\sum_j\sum_{i\in\text{Pred}_j}\mathbb{1}_{\{Y_i=j\}}$ by $\sum_{i\in NR}\mathbb{1}_{\{Y_i\in S_i\}}$.

- AccNR: The (set-valued) accuracy amongst non-rejected points:

$$Acc_{NR} = \frac{1}{|NR|}\sum_{i\in NR}\mathbb{1}_{\{Y_i\in S_i\}}$$

  Observations with only wrong labels contribute a zero, and points without labels (i.e. rejected) are discarded from the average in numerator AND denominator.

- AccTradNR: This is the traditional accuracy, but only amongst non-rejected points: the prediction $\hat{Y}_i$ at point $\vec{x}_i$ is the class $c = \text{argmax}_c\, p(c|\vec{x}_i)$, and we compare that to the true label. The fraction of points where these are equal is AccTradNR. This depends on the α because the set NR over which the average is taken does.

$$AccTradNR = \frac{1}{|NR|}\sum_{i\in NR}\mathbb{1}_{\{\hat{Y}_i=Y_i\}}$$

Caveats We mentioned $AccNR$, the set-valued accuracy (is the true label among the set?) on all non-rejected points, as a way to neutralize aleatoric uncertainty. However in practice, because this quantity starts out in 1 at $\alpha = 0$ by definition[15], it necessarily decreases initially.

[14] i.e. the fraction of Pred_j that is truly class j, so $PPV_j = \frac{\sum_{i\in\text{Pred}_j}\mathbb{1}_{\{y_i=j\}}}{|\text{Pred}_j|}$

[15] as we include the whole feature space in each class at $\alpha < \alpha_- = 1/n_{V,i}$ with $n_{V,i}$ the number of validation points in that class. This is because of the conservative definition of the quantile that is needed to guarantee

For a dataset where we have little or no class overlap (or outliers in the validation set) and the density estimation goes well, we can expect this quantity to take a sharp drop within the first few elementary 'steps' $\Delta\alpha = \alpha_- = \frac{1}{n_{V,i}}$, towards the value of AccTradNR at that point, and then follow the evolution of AccTradNR closely. When there is more aleatoric uncertainty however, AccNR will not fall as deep as AccTradNR (predicting several labels remains advantageous), or when there are more outliers the drop will be spread to higher α. Therefore AccNR can remain significantly larger all the way up to $\alpha = 1$. The Cifar dataset shows a clear example of this, cfr last section. (Note that this is not necessarily due to aleatoric uncertainty alone, if the density estimation is really off, we might probably see something similar.)

In other set-valued prediction problems (eg [23]), scaling the accuracy of each prediction by the set size used provides a useful measure that combines accuracy with precision. But in contrast to the situation there, where we can predict one or several labels and rejection consists in predicting all K labels, here we can also reject by predicting the null set, and that will happen for more and more points as $\alpha \to 1$. Because this measure always starts out at $1/K$ (since initially each class spans the whole space), and we will typically doubt between a few classes at most and not all K of them, the increase in the inverse set size always overpowers the decrease in accuracy as we reject more and more labels. This is essentially because there is no cost for completely rejecting (i.e. predicting the empty set) an observation, and both accuracy and precision can be made high by rejecting all but a few easy points. If we require a trade-off between precision and accuracy to select an α value, we can use an $F(\beta)$ measure using set-valued-accuracy and -precision (FracCorLabs),

$$F_\beta = (1+\beta^2)(\frac{1}{Prec} + \frac{\beta^2}{AccNR})^{-1}$$

typically with $\beta > 1$ in this setting where accuracy is favored over precision. This should allow us to make a good choice for α (but requires 'burning' a separate part of the test set in the process). I have excluded these figures from the final report due to space constraints, but they are available in the uploaded folders[16].

Because of these problems with the set valued measures, we eventually focused most on the evolution of *AccTradNR* as more and more points get rejected. This is the accuracy of the traditional classifier $\hat{y} = \text{argmax}_i\, p(y = i|x)$ for those points that do not get rejected by the set valued classifier. Here, as α increases, only the set NR changes, and we can see whether we're doing good by rejecting. While this measure suffers less from the artefacts of predicting sets, it only tells us whether the rejected points were missclassified more often than average, not whether this is due to aleatoric or epistemic uncertainty.

While the cautious classification approach turns out less suited for a quantification of the uncertainty of each type in general, the rejection curve RR vs α we obtain from this method can be a very useful tool for signalling large *overall* distributional uncertainty.

our confidence level. It is not guaranteed to end up in 1 as $\alpha \to 1$, but unless the very centers of each class still overlap (or the estimated HDR are really off), it will also tend to 1 on that side. There the performance is based on only a few remaining points and becomes very erratic.

[16] Without choosing an α measure, we can also use them to judge the merit of the cautious approach: when we see a sharper peak at lower α for a fixed value of β in one set than in another, it means the first dataset probably benefits more from the exclusion of a small fraction of anomalous points than the other. Often we see indeed a very sharp maximum at the lowest α values.

Figure 3.2: Rejection Rate vs α for different fractions of new points (fracnew), and different degrees of overlap (sdval). A horizontal line indicates the fractions of new points. Left: model correctly allows different covariances ('VVV') ie QDA. Right: model (incorrectly) assumes equal covariances ('EEE') ie LDA. The simulated dataset used is shown in B.1.

3.3 Detecting distributional uncertainty from the rejection curve.

By construction we reject at most a fraction α of validation points from each class. Then the fraction of points that is rejected from *all* classes (the rejection rate RR) cannot be more than α **for the validation set**. So, when in the test set we observe RR to increase more steeply than α, we can suspect an important difference between the test set and train/validation, like a new class, or a large distributional uncertainty (dataset shift). Note that we do not need to use the labels of the test data for this.

On the other hand, we could also score our original training set, and because the validation was obtained by a mere random split, we will have no such distributional uncertainty here. Also, here we have the labels, so we can compare classwise rejection rates[17] in the training set to their nominal value α. While we expect them to be slightly lower in the training set, a large difference would also point to large model variance[18]. That means when we have $RR \approx \alpha$ on the training set but $RR >> \alpha$ on the test set, we can almost be sure[19] that this has to be due to a difference in distribution between train and test.

When overlap is not too large we can see in Figures 3.2 and 3.3 for example, as one would hope, that the Rejection Rate immediately jumps to the fraction of new points, at very low thresholds. If we observe a graph of this type, it would be a strong indication that we have unidentified classes. We did a few very simple simulations to see to what extent this remains the case when there is some overlap or the distributional assumptions are not satisfied. Because of space constraints (and because the results are not so surprising) we place them in the Appendix B and present here only the conclusions.

[17] For the test set we don't want to use labels, so we considered points rejected from all classes instead.
[18] Model bias will typically not induce such a difference as can be seen from figure 2.6, where the training contour includes a lot of extra space due to model bias, but the difference in width between the train and validation quantiles comes from model variance alone.
[19] unless the test set is really small so the rejection curve is very coarse

Figure 3.3: Rejection Rate vs α for different fractions of new points (fracnew), and different degrees of overlap (sdval). A horizontal line indicates the fractions of new points. On the left we have the case where the known classes are fitted by 2 component mixtures (ie the truth), on the right the case where they are described by a single component each, ie in the presence of severe bias. The simulated dataset used is shown in B.1.

Conclusion The results suggest we could use $\frac{RR(\alpha_-)}{\alpha_-}$ as an indicator for the presence of new classes or large dataset shift[20]. Here $\alpha_- = \frac{1}{n_{c,V}}$[21] stands for the smallest meaningful α value for that class c, and corresponds to rejecting only points with a predicted density smaller than the lowest density observed for a point of that class in the validation set.

We conclude from the experiment in the Appendix that reliably detecting **which points are anomalous and which aren't becomes more difficult in situations where the model is completely misspecified or aleatoric uncertainty is larger, but even then we would typically still get a strong signal that something is wrong simply by looking at the rejection curve**. Therefore, this could be a useful first check to perform before we can even try to extrapolate inference from the training set towards the test set.

Big caveat here, I think, is that we used a large number of points in each cluster, so the smoothness of the curves is mostly thanks to that. If we have much smaller datasets, curves will be much coarser, and RR increases by visible jumps. That makes it a lot harder to recognize deviations from $y <= x$ at low α, and the larger model variance will also disturb this pattern for smaller datasets.

3.4 Application to Image Data

We first applied the method to a few simpler toy datasets, like the mfeat dataset from the UCI ML repository, where the macroscopic properties of the images are already summarized for us in features that can be used separately with eg. supervised feature selection. The eventual goal however was to apply the method to datasets of images, with in the order of 30x30 pixels with each 1(gray) or 3 (RGB) channels. So these have about 800 to 3200 numeric features (each containing an integer in the range [0,255]). Of course, these features are not meant to be used directly, as they have only a very local meaning. To turn this

[20] It is best to look at low α values since we dont know how many points are from new classes or how strong the dataset shift is. If we have large datasets (which is also necessary to get smooth rejection curves) it may be better to use a multiple of this value, to protect against outliers in the validation set playing a too large role.. But that would be at the cost of decreasing the sensitivity to small groups of anomalous points in the test set.

[21] with $n_{c,V} = n_V/K$ the number of validation points in class c, $n_{c,V} = \sum_{i \in V} \mathbb{1}_{Y_i = c}$.

into a manageable number of more meaningful features, we had to do some preprocessing typical for image data. This is discussed next.

3.4.1 Using existing state of the art pre-trained feature extractors.

We intended to apply an off-the-shelf feature extractor (VGG), which was trained to recognize low-level features like edges, shapes, colors on a much harder data set with thousands of classes, to our image datasets. This convolutional neural network, and the adaptations we made to apply this to our datasets, are discussed in Appendix C, and here we will present only a few of the results.

3.4.2 Results for image data: MNIST , FashionMNIST and Cifar

Introduction

We tried 3 publicly available image datasets, Cifar-10, FashionMNIST and MNIST. They each consist of 60.000 training (50.000 for FashionMnist) and 10.000 test images of 28^2 to 32^2 pixels, divided equally over 10 distinct classes. The main difference lies in what the classes represent, the number of channels (gray/RGB), the amount of preprocessing required, and the accuracy we can achieve. Fashion-MNIST is included in the keras library in R, Cifar-10[22] and MNIST [23] can be downloaded online. We will briefly discuss results for all 3 datasets, but except for FashionMNIST, we defer the figures to Appendix D due to space constraints.

The situation for each of the three datasets is quite different, and Fashion (with its baseline accuracy of about 92%) keeps the middle ground in the sense that MNIST is too easy (accuracy starts out at 98%, even with very simple classifiers and few features), while Cifar is a lot harder (only 72.5% for the best of the simple architectures we considered.)[24]. Cifar is also the only dataset for which we don't really see any improvement when we use the cautious method to exclude peripheral points[25].

The interpretation of the evolution of accuracy with the conformal confidence level α is a bit speculative when all those uncertainties are simultaneously at play, as we don't even know how well our density estimation went[26], and aleatoric uncertainty is hard to grasp while epistemic uncertainty remains significant.

We have defined several quantities in Section 3.2, some of them more a diagnostic tool than an actual performance measure (think of mean inverse set size, and rejection rate),

[22] http://www.cs.toronto.edu/~kriz/cifar.html

[23] https://pjreddie.com/media/files/mnist_train.csv

[24] Comparing this to a random accuracy of 10%, that is not extremely bad either, but the problem is that we need epistemic uncertainty to be really small before we can properly quantify aleatoric uncertainty, and for Cifar it is probably a bit too high.

[25] There the aleatoric uncertainty is probably much higher, because that picture changed a lot when we also exclude points of high (estimated) aleatoric uncertainty. Then we see the accuracy increase from 70% to about 90% when we remove less than half of the points.. This is discussed in the next subsection, and here we'll focus on rejecting only distributionally uncertain points.

[26] there could be bias, model variance, changes in accuracy upon increasing α will also be caused by spatial inhomogenity in the aleatoric uncertainty.. The difficulty of quantifying each type of uncertainty from these figures is the main reason we did a separate exercise focusing on quantification of uncertainties of each type in Appendix E, because it seems hard to extract such information from the curves we have here.

and we would now like to deduce as much as we can about the uncertainties of each type using these probes. I have left out the curves for the F_β measures in an attempt to reduce the figure load a bit more, but they can be found in an online Appendix for several values of β. Their use is probably limited to choosing a value for α[27], or less drastically, to see how beneficial application of this 'filter' for distributional uncertainty can be for a given dataset and a given 'cost' of rejection (as determined by β): A sharp peak at low α value suggests that rejecting a small fraction of points might really benefit performance, so in these cases I would say that the method seems more promising than it does for datasets where we have a broader peak smeared out towards higher α[28].

Fashion

On this set we obtain about 94% accuracy when using the mini-VGG (more like 92.5% for the 2 block architectures used eventually), and about 91% after substituting the dense layers by our cautious framework. In Figures 3.4-3.5 we show results when we use the cautious classification to exclude only the lowest density points.

Here we can make the following observations:

1. The fact that $RR \approx \alpha$ very closely suggests that the test set is not very different from the training data. This also seems to be confirmed by the fact that AccTradNR is so high at $\alpha = 0$.

2. The fact that AccNR doesn't decrease[29] very much when α is raised also suggests that aleatoric uncertainty is quite low in this dataset: even when the mean inverse set size becomes 0.95 (which would correspond to about 1 point in 10 with a second label, (9.1+1.1/2)/10=0.95), we only lose about 4% in AccNR. That suggests we have only a small fraction of points for which the 'primary' label is not correct. Especially when we compare AccNR here with what we see for Cifar, where it keeps on dropping steadily as α increases, this suggests overlap for fashion is much smaller.

3. But because $1 - AccNR$ (after the initial drop from 1) is still only about half of $1 - AccTradNR$ (eg at $\alpha = 0.2$) , there must be at at least some aleatoric uncertainty: the set-valued prediction scores a bit better when a few extra labels are allowed. Once α becomes larger than say 60%, the set-valued and traditional accuracy are virtually identical, and it appears the aleatoric uncertainty has become very small for the points that remain.

4. So in conclusion we could say that nearest neighbour models (NN9) and adaptive KDE (for low confidence level α, but that is the region we care most about) are the best

[27] again, contaminating our test set in the process
[28] Of course we have to compare $F(\beta)$ for the same β value..
[29] That decrease is inevitable, as it necessarily starts out at 1 by definition in $\alpha = 0$.

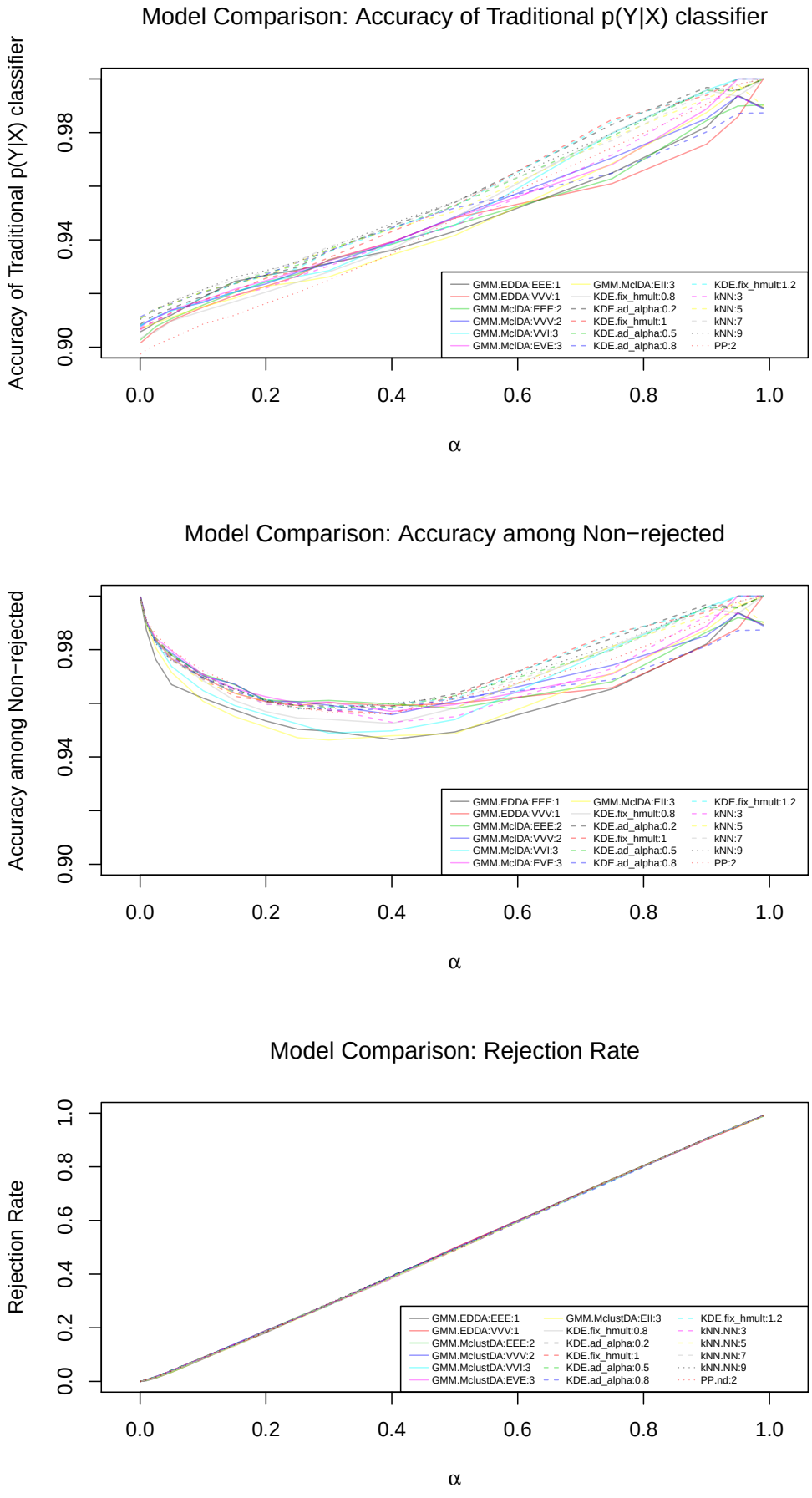


Figure 3.4: Top row: traditional accuracy.
Second Row: accuracy of set valued classifier.
Bottom: Rejection Rate (from all classes)..

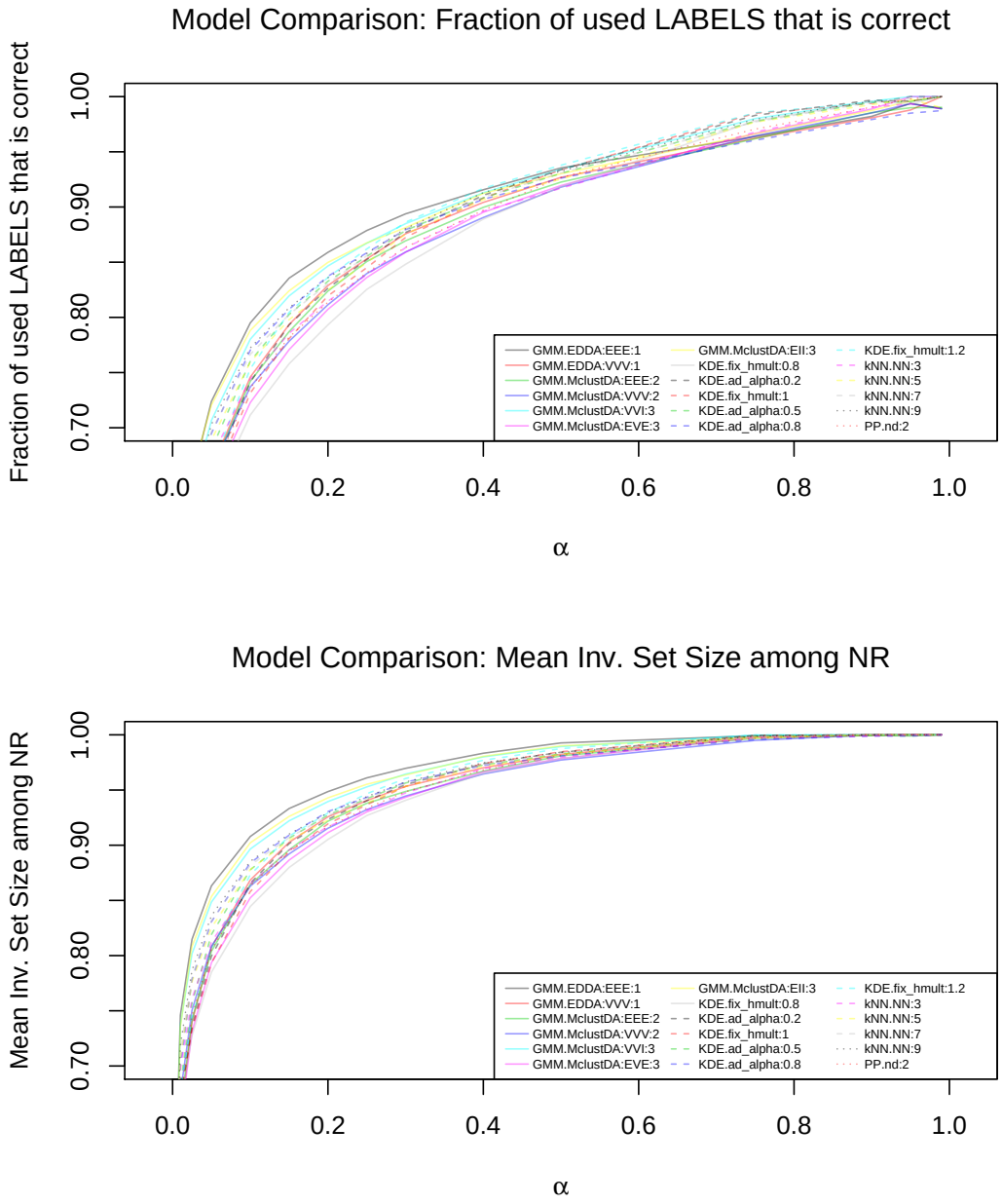


Figure 3.5: Top row: Precision (traditional, i.e. fraction of labels used that is correct). Bottom: Mean inverse setsize amongst NR points.

models here in 8 dimensions[30], and that there is a little aleatoric but not so much epistemic uncertainty for this case[31].

5. The fact that overlap is quite small (confirmed by precision measures mean inverse setsize and Fraction of Correct labels that approach 1 already for low α values) appears to indicate that the density estimation went well for this dataset. Otherwise, in case of large bias there we usually accept way too large regions in each class, and overlap tends to be bigger and stay until higher α values).

6. In conclusion we could say that cautious classification was not really needed here because of the high baseline accuracy, but that it still has a beneficial effect to exclude the most peripheral points, i.e. that these are somewhat harder to predict here. But the increase is also slow and steady. if the errors were due to different behaviour on the outskirts of the domain, we would see a sudden jump at small α values. Here the improvement is quasi-linear over the whole range, which suggests that the exterior points are somewhat harder but errors are also made in more central regions. But the fact that AccNR eventually ends up in 1 again also means that the very centers are corectly classified. (In contrast, for cifar, this is clearly much less the case.)

For Cifar and Mnist we refer for the same figures to the Appendix D.

Cifar

Here we have a very different situation than for FashionMNIST.

- We see a much lower baseline accuracy that does not improve at all when peripheral points are dropped from each class.
- In combination with the fact that the set-valued accuracy drops quite fast as the number of labels is reduced, this is already quite suggestive of much higher aleatoric uncertainty (or of large bias causing bad class centers).
- The fact that Figure D.3 in the Appendix shows how the situation clearly improves when the most aleatorically uncertain points are also removed as α increases, confirms this.
- Precision measures like mean inverse set size show that the predicted sets are now much larger even until α is very high.
- The fact that the fraction of correct labels increases even slower than the mean inverse set size appears to indicate that, as we reject labels, we are also rejecting correct ones, and the overlap is genuine, not just due to very biased density estimates.

[30] but I should add that this can change completely when we use a different number of hidden features. While the accuracy of the VGG model does not change that much, and the best accuracy here also remains quite similar, the precise order between the models does change a lot when we make different choice for the number of features. Some methods (eg NN) will cope better with increasing number of dimensions, and for example the precision of the LDA model was one of the worse in 32D, while it becomes one of the best here in 8D. (It would also be natural that deviations from normality are much larger in 32D than in 8D.) But this is probably more a property of the DE method than of this dataset.

[31] To support this we refer also to the next section where we reject also aleatorically uncertain points

- Even the rejection rate (with $RR < \alpha$ at first) appears to indicate that some of the points rejected from their true class are not entirely rejected yet because they remain in other class(es) for a while. That would give a slightly slower RR (and strongly decreasing AccNR) initially.

To what extent might a bad density estimate really be to blame for this? We have seen in Section 3.3 how bias due to a too simplistic density model can also cause large (non-genuine) overlaps (i.e. in *predicted* densities), which cause $RR < \alpha$ initially and a slight upwards curl at larger α.

- From the fact that the fraction of correct labels in Figure D.2 reaches only .7 when α becomes 1, whereas the inverse set size has reached 1 at that point, it appears that the central points are often wrong, so we have at least for some classes overlap in the predicted class center. The question is whether this is only the *predicted* center, or also the center of the true distribution?
- But the fact that even the VGG had only about 72%, suggests we have more genuine aleatoric uncertainty, at least with the features we have extracted here.
- I would also expect that the non-parametric density estimates would differ more from that for the GMM if the density estimation was to blame entirely for this.
- Finally, again the fact that the picture improves so much when we reject also points where the predicted aleatoric uncertainty is high (cfr. D.1.2), suggests that these were often misclassified and the apparent overlap was genuine.

Recapitulating, we don't really see that the accuracy for Cifar increases when the peripheral points are left out, and this could mean that the points in the periphery of classes are not the hardest ones, for example because the test set shows no unexpected behaviour while the overlapping regions are more central to certain classes.

Mnist

This dataset is a bit too easy to be really interesting, so will not discuss the results here. A few remarks are given in Appendix D where the figures can also be found.

3.4.3 Rejecting also points with high aleatoric uncertainty

Because of the lack in improvement when rejecting exterior points for Cifar, and because it seemed obvious that the previous approach will only focus on a certain type of uncertainty, we later tried to reject also the most aleatorically uncertain points, and the difference on all datasets is spectacular. In part this is understandable as aleatoric uncertainty has the advantage of using immediately the (predicted) class probabilities that are linked directly to accuracy (in contrast to epistemically uncertain points)[32].

[32] But, for those to be also the ones that are misclassified most often, and for their exclusion to benefit accuracy, we still need our estimated densities and class probabilities to be more or less correct.

For a fair comparison with the curves from the previous section it is also important to place the rejection rate RR on the abscissa and no longer α (or at least to use the RR versus α plot as a translation key, so we compare performances at a same RR), since we will now reject more points at a same α.

All figures can be found in Appendix D, organised by dataset. Here we will only discuss how we defined that aleatoric uncertainty. There are several options:

1. First option is to use the definition we already mentioned, of rescaled Bayes error $A = \frac{K}{K-1}(1-\max_y p(y|x))$. That ranges from 1 to 0 as the probability of the chosen class (that which has the highest probability) ranges from a random $1/K$ to 1. In order to combine this approach with the rejection of exterior points discussed before, we would like a criterion where the fraction of points rejected due to aleatoric uncertainty also starts from zero at $\alpha = 0$ and gradually increases with α. That way we could compare the figures with the previous ones to see the additional effect of these rejections. We could either reject points with uncertainties above a certain absolute value, or a certain fraction of points with highest uncertainties:

   (a) One possibility was to re-use the α values used for the cautious method as cutoffs for $1-A$, rejecting points for which $A > 1-\alpha$. But, that way, we don't reject much points, as we don't usually doubt between all K classes but between 2 or 3 at the most, and A is then much smaller than 1 for $K = 10$. We could try obtaining a more even spread by using instead $A \geq 1 - \sqrt{\alpha}$, or even a higher root[33], but eventually we start getting a slight jump at $\alpha = \alpha_-$ due to the very few points that did have higher aleatoric uncertainties now getting rejected even before we start. It is therefore better to use the next option if we really insist on spreading the aleatoric rejections more evenly over the α range[34].

   (b) A second option was to reject a *fraction* α of points, i.e. those with A above the $1-\alpha$ quantile of A values. In analogy with the OOD rejections we could determine the $1-\alpha$ quantile of A values in the validation set, and use that as a cutoff for test set A values. Or we could exclude directly the fraction α of those test points with highest A values. We dismissed the last option as it does not seem reasonable to have the rejection of a test point depend on the rest of the set being scored, we might even score one test point at a time.

2. Second option, instead of using A is the entropy of $p(y|x)$, i.e. $H(x) = -\sum_y p(y|x) \log p(y|x)$ as a measure of aleatoric uncertainty. Because I had no idea of a sensible absolute cutoff here, I immediately considered the $1-\alpha$ quantile of the valdiation entropies, and

[33] When doubting between 2-3 classes at most we'll rarely have $p_{\max} < 0.4$, meaning $A < (10/9).0.6 \approx 65\%$ always. That would mean we reject no points at all until $\alpha > 0.35$. Using the square root, this is shifted to about $\alpha = 0.12$, and using the 4th or 8th root even to 1.5% and 0.02% respectively.

[34] It is also questionable if that is at all desirable: It would seem preferable to have rejections based on an absolute measure of aleatoric uncertainty rather than rejecting some fixed fraction irrespective of the actual uncertainty at those points. The main reason for doing this is that, **without resampling methods or posterior distributions, we have to use rather arbitrary measures for these uncertainties that are not directly comparable to one another, and we can only compare the importance of each uncertainty through their effect on the accuracy**. Because these measures and the way we threshold them as α runs through its range have a large effect on the number of points being excluded at a given α, it **seems necessary to have a comparable number of points excluded due to each type in order to compare the the effects this has on the performance**. This is **only in order to compare the 'strength' of aleatoric and epistemic uncertainty**. A better effect on the accuracy could be obtained by excluding simply the same total fraction of points with highest *total* uncertainties, and there would be no reason why these would consist for half of the worse points from each group.

rejected test points with an entropy above that. This is the definition that was eventually used to make the figures. The advantage of this measure is that it is sensitive to the uncertainty about all classes, not just the most likely one, but the disadvantage is that immediate link to accuracy we had with (the unscaled) A is now gone.

The points selected for rejection due to aleatoric uncertainty can slightly overlap with those rejected due to OOD uncertainty, but this will not be much typically, as points in several classes typically don't get rejected from all of them that easily. Usually we will see a total rejection of about 2α for small α (which is important to take into account when comparing with figures from the previous section).

Cifar

This is the dataset where the effect is most spectacular, and this is probably because the aleatoric uncertainty was highest here. (We need to be careful not to attribute a faster increase to better rejections, when there are simply more at same α. But when the accuracies increase not only faster but also reach higher values, we do have an improvement.)

Noting the difference between FracCorLabs and AP[35] (for example at $\alpha = 0.2$ we have FracCorLab barely 0.45 but AP about 0.65), we can conclude that we are still using a lot of labels for points that aren't even correct. Those labels don't make any difference in AP (0/2=0/1) but they do in FracCorLab (where the denominator increases but the numerator doesn't). The fact that we have nearly $RR = 0.4$ at $\alpha = 0.2$ shows how the most aleatorically uncertain points are not OOD points, as we reject 2α points here, so there is hardly any overlap between the α-fraction of most OOD and that of most aleatorically uncertain ones.

Fashion

Now we can achieve 98% set-valued accuracy by rejecting 20% of points (which occurs just after $\alpha = 0.1$ this time). But for the traditional accuracy, the effect is larger: rejecting those 20% brings accuracy to almost 97%, and rejecting 40% to 98%. We can see here that after that point, traditional and set-valued accuracy nearly coïncide, so we have properly eliminated aleatorically uncertain points.

The fact that we have $RR \approx 35\%$ at $\alpha = 0.2$ while we used to have $RR = \alpha$ for the distributional rejections alone previously appears to indicate that the 20% most aleatorically uncertain points now have at least some (about 25%) overlap with the 20% most OOD points, so the most peripheral points are now at least as aleatorically uncertain as the more central ones.

MNIST

We have left this dataset out of the analysis due to space constraints, but a few remarks about the results are included in Appendix D where the figures are shown.

[35] This is a quantity we have dropped elsewhere because it rarely added something. This measure counts 0 for a set that doesn't contain the true label, and the inverse setsize for one that does, $AP = \sum_{i \in NR} \frac{\mathbb{1}_{y_i \in S_i}}{|S_i|}$.

CONCLUSIONS

From the last chapter (which gave the thesis its name) we conclude that the fact that epistemic uncertainty has no immediate link to accuracy keeps us from being really able to judge the merits of the cautious approach by its effect on performance. While we expect the method to be most valuable in situations where we have a large dataset shift, i.e. a large systematic difference between training and test data, we usually have no way of knowing beforehand whether this will be the case, and this makes the rejection curve of the cautious approach an important sanity check before we try to extrapolate our inference from the training set to the test set.

When additionally the other forms of uncertainty are small so that the distributional uncertainty is the dominant component, the cautious approach can be used to reject instances on an individual basis, which can be very useful in certain safety-critical situations (think self-driving cars, medical diagnosis, ...). In those cases, it may be much better to reject certain instances so they can be classified manually or so the decision can be postponed until more information is avalaible (eg. slowing down the car until the unknown object is closer, or taking an additional MRI scan, etc). For such an individual instance-by-instance rejection, it appears however necessary that bias and model variance are also small, because we really need our density estimation to be of high quality. But even when they are not, a strong deviation of the rejection curve from the line $RR = \alpha$ of the cautious classifier can still be an important indicator of dramatic *overall* distributional mismatch. This could signal for example that there remain thus far unidentified classes, or that the distribution of some of the known classes is more complicated (eg. mixture) than assumed so far.

We confirm also that the rejection of points with high aleatoric uncertainty typically has a more beneficial effect on accuracy, and given the above this comes as no surprise: Aleatoric uncertainty directly refers to uncertainty about the correctness of a particular *prediction*, while epistemic uncertainty actually refers to the uncertainty we have about the correctness of our *model*, either about the functional form of the model (=bias when in domain, distributional when out-of-domain), or about the best values of the parameters for a given functional form (=model variance). Because it is hard to quantify to what extent another model would give different predictions, the effect of epistemic uncertainty on accuracy is much harder to grasp.

From the theoretical chapter we conclude that one of the main obstacles complicating the separation of epistemic and aleatoric uncertainties in classification is the fact that both the covariance and mean of a multinomial are linked to the same class probability vector[36] and any attempt at studying these independently (eg by considering pre-softmax outcomes as target as in [3]) will ignore important correlations between them. In this respect the criticism made by Kwon et al in [4] on the work of [3] might be very important. However, the hierarchy that results when we don't ignore these correlations also brings certain difficulties: We started out in terms of noise and model variance, with an aleatoric uncertainty

[36] i.e. $E(\vec{y}) = \vec{p}$ and covariance $[Var(\vec{y}|\vec{p})]_{i,j} = p_i.\mathbb{1}_{i=j} - p_i p_j$

that only depends on the true distribution and not of our knowledge of it, and a total uncertainty that approaches this constant from above as the epistemic component shrinks. But the definition we had to use in practice (because the true and predictive distributions are not usually known) turns out quite different, and has an aleatoric uncertainty that is expected to increase as we narrow down the possible range $p(\theta)$ of model parameters so the epistemic component decreases (which is also confirmed by the maps in Appendix E).

I think this discrepancy stems from one of the key differences between the regression with constant noise context and classification: In the former case, the (aleatoric) noise σ^2 is actually a lower bound for the predictive variance, and therefore we can indeed expect total variance to decrease towards this (constant) aleatoric component as the dataset becomes denser. For classification however, $\theta_0.(1-\theta_0)$ is by no means a lower bound for the variance, and due to Jenssens inequality for concave functions (like $x(1-x)$), we can even expect the aleatoric uncertainty (not defined as $\theta_0(1-\theta_0)$ but as $E[\theta.(1-\theta)]$ <= $E(\theta).[1-E(\theta)]$) to increase as $p(\theta)$ concentrates in probability around θ_0, while the total uncertainty $E(\theta).[1-E(\theta)]$ could go either way[37]. This appears to indicate that only the epistemic component retains its familiar meaning[38], while the values of aleatoric and total uncertainty are 'hazy', only measurable up to a certain precision, and this precision is just that epistemic component. Once this vanishes, the haziness disappears and the latter two become equal. The definition we used assigns each of the three contributions a well determined value also for finite datasets, but apart from ensuring the correct limiting behaviour and the nonnegativity of the epistemic (model variance) component, these values for aleatoric and total uncertainty probably don't have much meaning as long as the epistemic component remains.

In the second chapter, we found that our measure of epistemic uncertainty appears to recognize the domain only for very flexible models, which is probably because in those cases, the model variance has absorbed most of what was originally distributional uncertainty. While such an approach might capture the previously unquantifiable uncertainty on the edge of the domain, it is almost certain to result also in a larger model variance with**in** the domain, and this may be very disadvantageous to model performance. It thus appears that we are faced with a choice between a better model that ignores part of the uncertainty OOD, and another that may do a better job at being aware of its uncertainty OOD, but might perform much less in-domain and, in most situations good performance in-domain will be even more important than reliable uncertainties for extrapolation OOD[39]. For smaller datasets, it may then be preferable to keep in mind an unquantifiable uncertainty beyond the domain in exchange for lower uncertainties in-domain. We also see that both measures we used to quantify the aleatoric uncertainty agree very well with one another. But because of the way they are defined, they also depend strongly on the density estimation used and the epistemic uncertainty that comes with it, in line with what we wrote for the first chapter. Finally, we also have to conclude that the measures we constructed based on Dirichlet priors, in an attempt to recognize the distributional uncertainty beyond the domain also for models with stronger distributional assumptions, don't really give the hoped-for results.

[37] depending on whether $E(\theta)) \to \theta_0$ approaches from the side of the uniform $\theta = 1/K$ or from the edges $\theta = 0, 1$.
[38] which appears to be confirmed also by the equivalence of this definition with an often-used measure of spread between the ensemble of class probabilities, the JSD.
[39] unless perhaps we can still act upon that uncertainty to reduce it.

BIBLIOGRAPHY

[1] Eyke Hüllermeier and Willem Waegeman. “Aleatoric and Epistemic Uncertainty in Machine Learning: A Tutorial Introduction”. In: *CoRR* abs/1910.09457 (2019). arXiv: 1910.09457. URL: http://arxiv.org/abs/1910.09457.

[2] Andrey Malinin and Mark Gales. “Predictive Uncertainty Estimation via Prior Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/3ea2db50e62ceefceaf70a9d9a56a6f4-Paper.pdf.

[3] Alex Kendall and Yarin Gal. *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* 2017. arXiv: 1703.04977 [cs.CV].

[4] Yongchan Kwon et al. “Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation”. English. In: *Computational Statistics and Data Analysis* 142 (Feb. 2020). Publisher Copyright: © 2019 Elsevier B.V. Copyright: Copyright 2019 Elsevier B.V., All rights reserved. ISSN: 0167-9473. DOI: 10.1016/j.csda.2019.106816.

[5] Yarin Gal. “Uncertainty in Deep Learning”. PhD thesis. University of Cambridge, 2016.

[6] Stefan Depeweg. “Modeling Epistemic and Aleatoric Uncertainty with Bayesian Neural Networks and Latent Variables”. PhD thesis. TU München, 2019.

[7] Yotam Hechtlinger, Barnabás Póczos, and Larry Wasserman. *Cautious Deep Learning*. 2019. arXiv: 1805.09460 [stat.ML].

[8] Mehran H. Bazargani and Brian Mac Namee. “The Elliptical Basis Function Data Descriptor (EBFDD) Network: A One-Class Classification Approach to Anomaly Detection”. In: *ECML/PKDD*. 2019.

[9] Pedro Domingos. “A Unified Bias-Variance Decomposition and its Applications”. In: *In Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, 231–238.
A more detailed version of the paper can be found on the homepage of the author at this url: URL: https://homes.cs.washington.edu/~pedrod/bvd.pdf..

[10] Stefan Depeweg et al. *Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning*. 2018. arXiv: 1710.07283 [stat.ML].

[11] Andrew Gelman et al. *Bayesian Data Analysis*. 3rd ed. Chapman and Hall/CRC, 2013. DOI: https://doi.org/10.1201/b16018.

[12] Adam Cobb. “The Practicalities of Scaling Bayesian Neural Networks to Real-World Applications”. PhD thesis. Oxford University, 2020.

[13] Radford Neal. “Bayesian Learning for Neural Networks”. PhD thesis. University of Toronto, 1994.

[14] Gareth James et al. *An Introduction to Statistical Learning: with Applications in R*. English. Springer Texts in Statistics. Springer New York, 2013. ISBN: 978-1-4614-7137-0. URL: http://doi.org/10.1007/978-1-4614-7138-7.

[15] Stanford University. *CS231n: Convolutional Neural Networks for Visual Recognition*. [Online. Accessed 2021]. URL: https://cs231n.github.io/convolutional-networks/.

[16] Beate Sick. *Connection Dropout and Bayesian statistics*. [(Online, Accessed 2021)]. URL: https://tensorchiefs.github.io/bbs/files/dropouts-brownbag.pdf.

[17] Jing Lei and L. Wasserman. "Distribution-free prediction bands for nonparametric regression". In: *Quality Engineering* 60 (2014), pp. 109–110.

[18] Luca Scrucca and Alessio Serafini. "Projection Pursuit Based on Gaussian Mixtures and Evolutionary Algorithms". In: *Journal of Computational and Graphical Statistics* 28.4 (2019), pp. 847–860. URL: https://doi.org/10.1080/10618600.2019.1598871.

[19] Geoffrey J. McLachlan and David Peel. *Finite Mixture Models*. International series of monographs on physics. Wiley, Oct. 2000. ISBN: 978-0-471-00626-8.

[20] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall, 1986.

[21] Joseph B. Kruskal. "Toward a practical method which helps uncover the structure of a set of observations by finding the line transformation which optimizes a new "index of condensation"". In: *Statistical Computation*. Ed. by Roy C. Milton and John A. Nelder. Academic Press, 1969, pp. 427–440. ISBN: 978-0-12-498150-8. URL: https://www.sciencedirect.com/science/article/pii/B9780124981508500240.

[22] J. H. Friedman and J. W. Tukey. "A Projection Pursuit Algorithm for Exploratory Data Analysis". In: *IEEE Trans. Comput.* 23.9 (Sept. 1974), pp. 881–890. ISSN: 0018-9340. URL: https://doi.org/10.1109/T-C.1974.224051.

[23] Thomas Mortier et al. *Efficient Set-Valued Prediction in Multi-Class Classification*. 2020. arXiv: 1906.08129 [cs.LG].

[24] L. Scrucca. "Dimension reduction for model-based clustering". In: *Statistics and Computing* 20.4 (2010). arXiv:1508.01713, pp. 471–484. DOI: 10.1007/s11222-009-9138-7.

[25] L. Scrucca. "Graphical tools for model-based mixture discriminant analysis". In: *Advances in Data Analysis and Classification* 8.2 (2014). arXiv:1508.01695, pp. 147–165. DOI: 10.1007/s11634-013-0147-1.

[26] Halima Bensmail and Gilles Celeux. "Regularized Gaussian Discriminant Analysis Through Eigenvalue Decomposition". In: *Journal of the American Statistical Association* 91.436 (1996), pp. 1743–1748. ISSN: 01621459. URL: http://www.jstor.org/stable/2291604.

[27] Charles Bouveyron and Camille Brunet. "Model-Based Clustering of High-Dimensional Data: A review". In: *Computational Statistics and Data Analysis* 71 (Jan. 2013), pp. 1–27. DOI: 10.1016/j.csda.2012.12.008.

[28] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[29] Rohit Thakur. *Step by step VGG16 implementation in Keras for beginners*. [Online. Accessed 2021.] URL: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c.

[30] Jason Brownlee. *How to Use The Pre-Trained VGG Model to Classify Objects in Photographs*. [Online. Accessed 2021.] URL: https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/.

[31] The TensorFlow Authors and PBC RStudio. *VGG16 and VGG19 models for Keras*. [Online. Accessed 2021.] URL: https://tensorflow.rstudio.com/reference/keras/application_vgg/.

[32] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009. URL: http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

[33] Donald B. Rubin. "The Bayesian Bootstrap". In: *The Annals of Statistics* 9.1 (1981), pp. 130–134. DOI: 10.1214/aos/1176345338. URL: https://doi.org/10.1214/aos/1176345338.

APPENDIX A

MCLUST FOR GAUSSIAN MIXTURE MODELS

Mclust uses the EM algorithm (starting from some initial guess based on a hierarchical procedure) for model based clustering, assuming that the data is composed of a mixture of multivariate normal distributions (GMM, For Gaussian Mixture Model). We can then restrict covariances at two levels: either between classes when there is 1 cluster per class, using modeltype='EDDA' (for EigenValue Decomposition Discriminant Analysis, or within classes when each class is a mixture by itself, using modeltype='MclustDA'[1].

Mclust offers 14 different possible restrictions that can be placed on cluster covariances, in terms of common or independent cluster oriëntation, shape, and scale. These correspond to $\frac{D(D-1)}{2}$, $D-1$ and 1 degrees of freedom respectively. These can either be eliminated (eg diagonal covariances), shared between K classes, or fully independent. The '**EVI**' model for example is then the model with common cluster volumes ('**E**qual'), independent shapes ('**V**arying'), aligned with coördinate axes ('**I**dentity'). The 'EDDA' model with K components and 'EEE' structure for example then corresponds LDA, while 'EDDA' with 'VVV' produces QDA. We can also assume a mixture for each class using the MclustDA option, and specify for each class separately the maximal number of components and covariance restrictions to apply to clusters within that class.

In figures (A.1) we provide a table of the properties and the corresponding degrees of freedom for each of these 14 options. There we see clearly how even the d.o.f. of an LDA model with its common covariance can explode if D becomes too large, and that most can be won by restricting the oriëntation, for which the number of parameters increases quadratically with the dimension D. So it really depends on whether we have very large K, or D , or both, what we can afford to fit. By default, *Mclust* tries a whole range of complexities and selects the one from that set that optimizes some criterium balancing model parsimony with likelihood. For that criterium one can choose between the beforementioned BIC and the ICL (Integrated Completed Likelihood). According to the documentation, BIC tends to favor models that approximate the data well but are a bit simpler than reality (stresses parsimony), while ICL will not be as easily satisfied with a simpler approximation (stresses likelihood). That would mean ICL is better for inference and BIC for prediction, so here we

[1] The MclustDA function is actually meant for producing also cluster/class membership probabilities $p(y|x)$, but in the process it fits an independent density model in each class. We will only extract those densities and ignore the classification-related outputs, so we can just use one call to MclustDA instead of K calls to Mclust.. The advantage of this is that we can then also use their dimension reduction technique MclustDR on an MclustDA object to plot all classes and density contours on a same low dimensional figure.

will let Mclust select the model with the highest BIC (from those submodels considered that converged).

Important to keep in mind however, is that Mclust will implicitly use a simpler submodel if the more general one failed to converge. This convergence is in part stochastic[2], and although underdetermined models (with more parameters than observations) are seen to converge much less frequently, they still can (even though there is no regularization by default). This is probably because (unlike eg. least squares) the EM algorithm doesn't have to invert any matrices to find an exact and unique solution, it is an iterative procedure, that remains dependent on the initial condition[3].

Finally, this package also offers dimension reduction and visualisation methods (discussed in the papers [24],[25]) For a single component per class, and a common covariance matrix between classes, this dimension reduction would reduce to Fisher canonical (LDA) coordinates, according to Proposition 1, p.5 in [24]). If we have different covariances, and/or several clusters per class, it amounts to a generalisation thereof that also takes into account directions where the variances are very different between clusters. If I understand correctly [24], it finds directions which maximize some combined variation in the cluster means and in the cluster covariances. So, directions can be important either because they separate well the cluster means, or because they have a significant difference in cluster dispersion, or a little of both. One can set the relative importance to be given to the contribution from differences in covariance and that from differences in mean respectively via the parameter λ, which is 0.5 by default. We have set $\lambda = 1$ to focus on separating cluster means when we produce these graphs[4].

[2] For example, on occasions we saw the VVE model with 1 and 3 components converge, and that with 2 components not, in the same run on the same data.

[3] While there may be an infinite number of solutions that describe the training set equally well in such situations, it seems unlikely that they will all describe the test set equally well, so even if convergence is not always an issue, overfitting is still likely to occur in such a situation. Therefore I think it is preferable that a unique global optimum at least exists, even if we have no guarantee of finding it. (On that note: Mclust uses a procedure (called hierarchical agglomerative clustering) to determine the initial condition for its EM algorithm that ought to improve its chances of ending up near the global optimum.)

[4] I have not included such graphs in the final report, but they can be convenient to visualise the clusters identified by Mclust, and I think we could in principle also use the basis vectors of the reduced spaces to project the results for non-GMM density estimation on.

Model	Name	Nb. of parameters	$K = 4$ $p = 100$
$[\lambda_k D_k A_k D_k^t]$	VVV	$(K-1) + Kp + Kp(p+1)/2$	20603
$[\lambda D_k A_k D_k^t]$	EVV*	$(K-1) + Kp + Kp(p+1)/2 - (K-1)$	20600
$[\lambda_k D_k A D_k^t]$	VEV	$(K-1) + Kp + Kp(p+1)/2 - (K-1)(p-1)$	20306
$[\lambda D_k A D_k^t]$	EEV	$(K-1) + Kp + Kp(p+1)/2 - (K-1)p$	20303
$[\lambda_k D A_k D^t]$	VVE*	$(K-1) + Kp + p(p+1)/2 + (K-1)p$	5753
$[\lambda D A_k D^t]$	EVE*	$(K-1) + Kp + p(p+1)/2 + (K-1)(p-1)$	5750
$[\lambda_k D A D^t]$	VEE*	$(K-1) + Kp + p(p+1)/2 + (K-1)$	5456
$[\lambda D A D^t]$	EEE	$(K-1) + Kp + p(p+1)/2$	5453
$[\lambda_k B_k]$	VVI	$(K-1) + Kp + Kp$	803
$[\lambda B_k]$	EVI	$(K-1) + Kp + Kp - (K-1)$	800
$[\lambda_k B]$	VEI	$(K-1) + Kp + p + (K-1)$	506
$[\lambda B]$	EEI	$(K-1) + Kp + p$	503
$[\lambda_k \mathbf{I}_p]$	VII	$(K-1) + Kp + K$	407
$[\lambda \mathbf{I}_p]$	EII	$(K-1) + Kp + 1$	404

Parameterisations of the within-group covariance matrix $\boldsymbol{\Sigma}_k$ for multidimensional data available in the **mclust** package, and the corresponding geometric characteristics.

Model	$\boldsymbol{\Sigma_k}$	**Distribution**	**Volume**	**Shape**	**Orientation**
EII	$\lambda \boldsymbol{I}$	Spherical	Equal	Equal	—
VII	$\lambda_k \boldsymbol{I}$	Spherical	Variable	Equal	—
EEI	$\lambda \boldsymbol{A}$	Diagonal	Equal	Equal	Coordinate axes
VEI	$\lambda_k \boldsymbol{A}$	Diagonal	Variable	Equal	Coordinate axes
EVI	$\lambda \boldsymbol{A}_k$	Diagonal	Equal	Variable	Coordinate axes
VVI	$\lambda_k \boldsymbol{A}_k$	Diagonal	Variable	Variable	Coordinate axes
EEE	$\lambda \boldsymbol{D}\boldsymbol{A}\boldsymbol{D}^\top$	Ellipsoidal	Equal	Equal	Equal
EVE	$\lambda \boldsymbol{D}\boldsymbol{A}_k\boldsymbol{D}^\top$	Ellipsoidal	Equal	Variable	Equal
VEE	$\lambda_k \boldsymbol{D}\boldsymbol{A}\boldsymbol{D}^\top$	Ellipsoidal	Variable	Equal	Equal
VVE	$\lambda_k \boldsymbol{D}\boldsymbol{A}_k\boldsymbol{D}^\top$	Ellipsoidal	Variable	Variable	Equal
EEV	$\lambda \boldsymbol{D}_k\boldsymbol{A}\boldsymbol{D}_k^\top$	Ellipsoidal	Equal	Equal	Variable
VEV	$\lambda_k \boldsymbol{D}_k\boldsymbol{A}\boldsymbol{D}_k^\top$	Ellipsoidal	Variable	Equal	Variable
EVV	$\lambda \boldsymbol{D}_k\boldsymbol{A}_k\boldsymbol{D}_k^\top$	Ellipsoidal	Equal	Variable	Variable
VVV	$\lambda_k \boldsymbol{D}_k\boldsymbol{A}_k\boldsymbol{D}_k^\top$	Ellipsoidal	Variable	Variable	Variable

Figure A.1: Names and number parameters for each model as a function of the dimension (named p here), the number of clusters K, and the model restrictions. For more information and a comparison between EDDA and RDA, see for example [26] and [27], from which these tables are taken.

APPENDIX B

DETECTING NEW CLASSES (OR OTHER DISTRIBUTIONAL MISMATCH) IN THE PRESENCE OF BIAS AND OVERLAP

Figures 3.2-3.3 showed how the RR immediately ($\alpha < 0.1\%$) jumps to the fraction of new points, because the class is different enough to be rejected at very high confidence levels (low α). This suggests using $\frac{RR(\alpha_-)}{\alpha_-}$ as an indicator for the presence of new classes. In the remainder of this appendix we have some experiments intended to see to what extent this remains true in the presence of more bias and overlap.

B.1 Simulated Gaussian data with varying fractions of new points, and varying overlap.

We simulated data from 5 bivariate normals, of which the means where randomly drawn integers in $[-5, 5]$. The covariances of the 4 known (ie passed during training) classes were fixed, and controlled by a common scale parameter $\sigma_k = 2$. The shapes were as follows: For the first two known classes I fixed the covariance at $\mathrm{diag}(\sigma_k^2, 0, 0, \sigma_k^2/2)$ and for the other two at $\mathrm{diag}(\sigma_k^2/2, 0, 0, \sigma_k^2)$.
For the unknown class we have a spherical covariance $sd^2.\mathbb{1}$, with increasing values for $sd = 1, 2, 3$. The appropriate model was therefore QDA, and I compared the case where a QDA model ('VVV') was fit, and that where an LDA ('EEE') was fit, to check if the picture changed dramatically in the presence of bias. The number of points in the 4 known classes was fixed (and large, to eliminate model variance), and that in the unknown class varies from .1% to 25% of the test set. A horizontal line indicating that fraction of new points is always shown in the RR vs α plots (eg fig. 3.2). The data is shown in fig. B.1 (right), only there the known clusters are already combined two by two (as explained in the text further). Here $\alpha_- = \frac{1}{n_{c,V}}$[1] stands for the smallest meaningful α value for that class c, and corresponds to rejecting only points with a predicted density smaller than the lowest density observed for a point of that class in the validation set[2].

So, we see that when there is more overlap, the jump gets smeared out somewhat more, as does it when there is bias due to incorrectly specified covariance, but this does not appear to dramatically change the picture for RR vs α. In the graphs for the accuracies (in the

[1] with $n_{c,V}$ the number of validation points in that class c, $n_{c,V} = \sum_{i \in V} \mathbb{1}_{Y_i = c}$.

[2] If extreme outliers in the validation set (eg. generated by errors of some kind) are possible, or if the validation set is large enough, it would be preferable to choose a larger α that is based on more than a single validation point. But that would be at the cost of decreasing sensitivity to small groups of anomalous points in the test set.

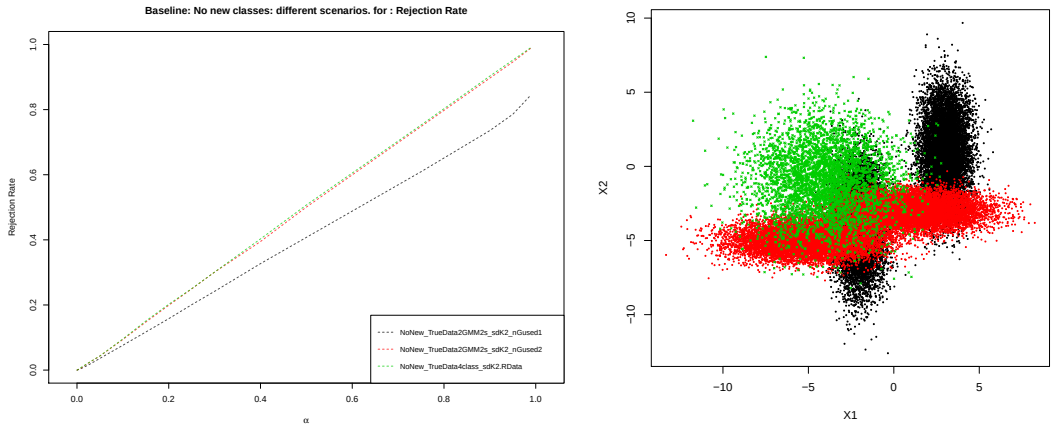


Figure B.1: Rejection Rate vs α for different baselines withouw new classes: the model with 4 classes estimated using QDA, the case of 2 GMM's with 2 clusters each, estimated by a GMM allowing 2 components, and the same estimated by QDA, ie allowing only 1 cluster per class. We don't see such a sudden jump in RR as we do for the figures with new classes. On the right we show the data used in the previous figures, with largest overlap (sd=3) and fracnew=25%. The figure on the left then corresponds to an absent green cluster.

online appendix) the effect is more pronounced, as could be expected. That seems to make RR vs α the preferred tool for this task.

In a next stage, we assign the 4 'known' classes to two mixtures of 2 components each, and compare the results when a 2 component mixture was fit to those when a single normal per class was fit, i.e. in the presence of significant bias (incorrect number of components). The main difference we see (fig. 3.3) in such a case is that the model is unable to reject some of the points until the very last moment, causing the slight upward curl as α approaches 1. That can also be understood from fig. 2.6 where the fact that the closest validation point is quite far from the center of the trained PDF means a large region of space is never rejected.

Finally, we also make the figure B.1 (left) for the case where no new classes are present, as a baseline, for comparison. Here again, we see the lower overall and final rejection rate (because of bias) and upwards curl at the end (because of the increased overlap due to bias) for the case with most severe bias. Apart from that though, behaviour is nicely 1:1.

Big caveat here, I think, is that we used a large number of points in each cluster here, so the smoothness of the curves is mostly thanks to that. If we have much smaller datasets, curves will be much coarser, and RR increases by visible jumps. That makes it a lot harder to recognize an anomalous deviation from $RR \leq \alpha$ at the lowest α.

On the right of fig. B.1, we also show the original data: 4 known clusters (and their attribution to two times 2 mixtures in red and black dots), as well as the new class for the particular case $sd = 3$,$fracNew = 0.25$ in green crosses.

While the jump for large fractions of points from another class was to be anticipated, it seemed less obvious beforehand that even in the presence of significant bias, and for significant overlap, the signature of new classes remains clearly present, provided the fraction of new points is not extremely small. While this little experiment is too limited to draw any

general conclusions, it suggests that this rejection curve can provide clues about bias on the one hand and new components on the other.

What this method will not tell us however, is whether all rejected points have something in common (eg form an unknown cluster), or whether they are scattered all over the place (as might for example be the case when only the variance is much larger in the test set than in training). It merely signals the fact that something is different in the test set, and this difference could even still be due to model variance. But because that also plays between train and validation set (which come from a random stratified split), we propose to look also at the training rejection rates. These would typically be lower than nominal, (because the density was tailored to the training set, the densities in training points would typically be higher than in an independent validation set). But if the difference is small, we could suspect that model variance is likely not large enough to account for a much larger rejection rate in the test set[3].

While all of this is probably rather obvious, this is probably the context where the method of cautious classification could play its largest role. It would be fairly easy to implement a filter removing abnormal points before going over to the actual model, but we have to be careful not to do this when we are not completely sure how reliable our density estimate is. In situations where we are less sure of that, the method could still be useful to flag for an overall distributional mismatch, without trying to reject individual points. Finally, while this method can help with the detection of distributional mismatch, we should also keep in mind that not all distributional mismatch will show up using this method. So RR not being dramatically larger than α by no means guarantees that the test set fits the distribution learnt from the training data (as rejections are conservative).

B.2 Real datasets

The question remains to what extent this remains true for more realistic datasets that are only approximately described by the density we use.
In that respect the following anecdote: to start in lower dimensions, I initially used the mfeat dataset (from the ML repository at UCI), selected some 5-10 features, and left out the 10th class during training. When I classified a test set with all 10 classes, each and every one of the instances from the class left out, resulted in 'NaN' density for all 9 known classes. The reason is simple: R only used the default precision (16 digits) at that time, and to produce $p(y|x)$, we have to normalize $p(y,x)$ by dividing with $p(x) = \sum_c p(c,x) = \sum_c \exp\log p(c,x)$. Because of this sum, we have to translate logdensities to densities again.
But for high dimensional gaussians, the density decreases very fast for outliers, and in practice this simply returns 0 for each $p(x,y)$ and thus for $p(x)$. Later I increased precision, but this shows how the fact that densities decrease so fast in higher dimensions can in fact be an advantage to flag any potential outlier. At the same time, it also means that any outliers present during training will have a tremendous effect on the likelihood in high dimensions, which is why a more robust density estimation method should probably be considered.

[3](provided train, validation and test have comparable and not too small sizes so deviations due to model variance between any two sets would be of a same order)

APPENDIX C

APPLICATION TO IMAGE DATA: PREPROCESSING AND FEATURE SELECTION

C.1 Transfer Learning and VGG16

The VGG16 convolutional neural network developed by the Visual Geometry Group at Oxford [28] was one of the best performing models[1] in the ImageNet 2014 challenge. What's more important, this architecture was shown to generalize also quite well to other settings ('transfer learning'). Because of this, the whole network was made available by the creators, to serve as an off-the-shelf feature extractor in other problems.

The idea behind transfer learning is that the first layers of this architecture merely convert information at the pixel level to slightly more macroscopic information. Especially for images, it is not unreasonable to assume that the conversion of pixelwise information to lines, edges, shapes and so on, could remain useful for recognizing and distinguishing other types of objects than the ones used for training.
The last, fully connected layers (which contain most of the parameters, over 10^8 weights in the case of VGG16) and the softmax layer (which depends on the number of classes to distinguish) then use these more meaningful features to do the classification. So, to take advantage of this, we can take the pre-trained network, and chop off and retrain the fully connected layers, with the last softmax layer now having the appropriate number of classes for our new problem.

In its default form, it takes a picture of at least 32x32 pixels, and 3 color channels (RGB) in each pixel, and produces 512 features for each subpatch of size 32x32 of the original picture (fig. C.1). These are then usually fed to a few densely connected layers for the actual classification. Those densely connected layers are also the more intensive part of the training, with hundreds of millions of weights to be found. But we 'cut that off', and keep only the pretrained feature extractor, which we can then retrain for our situation, with only 10 classes, and using much smaller dense layers so it can run in 5-30 minutes[2] on a typical laptop. We review here briefly what this architecture does[3] before explaining how we modified it in the next section.

[1] 1st for the localization and 2nd for the classifciation track.
[2] depending on the other settings/parameters
[3] see also for example the Stanford University online notes on convolutional networks at [15] , or for a hands on introduction to using VGG16 for transfer learning [29] and [30]. For the R documentation see [31].

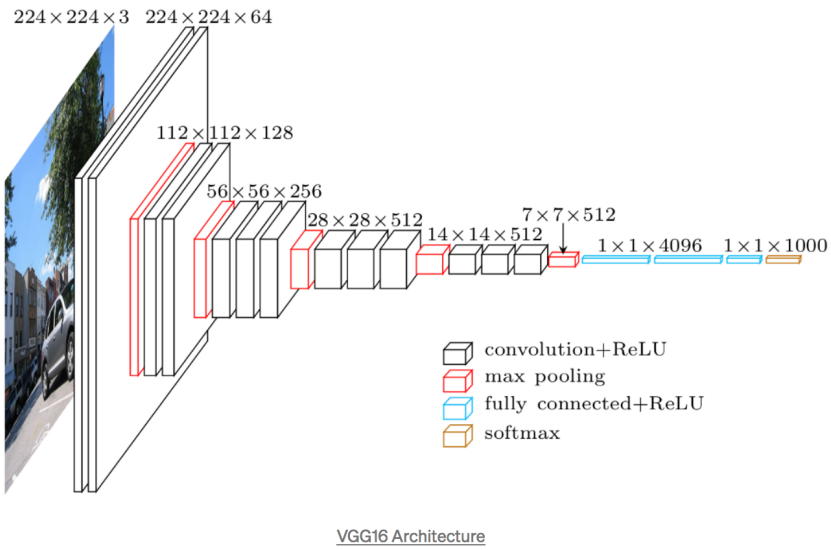


Figure C.1: The architecture of the VGG16 convolutional network. As larger and larger patches of the original image get summarized in each 'node', the number of channels increases, from 3 to 64 to 128 etc. The fully connected part is the hardest to train, but also the one that is most application dependent.

VGG16 consists of 5 blocks, in each of which we have 2 convolutional layers, and a maxpool layer. Those convolutional layers greatly reduce the number of parameters as opposed to dense layers by allowing 'pixel'[4] (i,j) in layer L+1 to receive input only from the pixels within a small window of size NxN centered on that same pixel (i,j) in the previous layer L+1. For example, when we use a window of size 3x3, then the pixel on position (x=4,y=5) in the 2nd layer will receive inputs only from pixels in the window (x=3..5,y=4..6) in the first layer.

There are some additional parameters like padding (which determines what we do when the active window reaches passed the borders of the original image), and stride, (which determines how fast the filtering window sweeps over the previous layer image). The latter codetermines in how many windows each original pixel participates.

Here we focus on the situation where the stride is 1 and the outer layers are padded by 1 row and column so the convolutional layers do not change the width and height of the image for a window size 3. As for the final dimension: in each input pixel we have 3 colors (channels). As we go to the first block, each 'pixel' or cell now has 64 channels that each summarize the information in the channels on that and the adjacent pixels of the previous layer in different ways.

That describes a single convolutional layer. Now, each block consists of 2 such layers followed by a pooling layer. The latter shrinks the image's width and height by a factor 2, by summarizing each 2x2 block in a single 'pixel'. To avoid losing too much information there, the number of channels is doubled subsequently for the next block, which will be 2x2 times smaller. There are different ways to 'pool' , i.e. summarize such a 2x2 (here) window in a single 'pixel'. VGG16 uses 'max'-pooling, which simply keeps the maximum of the values observed over the 4 cells. It does this for each of the channels separately.

[4] 'Pixel' is a tempting word here because it corresponds to one fixed small region of the image, but its perhaps not the best terminology as it no longer has 3 RGB values but 64 or more features, and in every next block it will summarize the info in a 2x2 times larger subregion of the original image.

Now that we have described a block, it suffices to say that VGG16 has 5 such blocks[5]. After these 5 blocks, both VGG16 and VGG19 then add 3 dense layers to come to a total of 16 resp 19 layers (pooling not included as these dont have any parameters), to predict their data. While that part of the model is less transferable to other datasets (and very expensive to train), the 5 blocks we just described are known to be useful for feature extraction on image datasets, as-is, ie with the weights publicly available.

It was therefore a natural step for the image datasets that we considered, which have about a thousand pixels in RGB, to try to use this approach to get a more manageable number of features, that are hopefully predictive enough to be able to replace those dense layers of the network by a generative model. After all if the network is able to classify well based on those inputs, these should contain the necessary information in one form or another.

C.2 Obstacles to using VGG16 as-is

- One of the limitations of the approach as is, is that it requires at least 32x32 images, while we also had datasets with smaller images. We could cut off the last block and use only 4, but then it might be better to retrain, as the current weights were obtained for 5 blocks.
- Another problem seemed to be that standard, it produces 512 channels in the last layer[6]. If the idea is to retrain the dense layers following these blocks, then starting with 512 channels and adding only a single not too small dense layer before the softmax, would already result in more weights in that layer alone than we have training instances in all classes combined. That is not supposed to be a problem in neural networks, thanks to regularization and data augmentation[7], but we have none of these, as the off the shelf-VGG didn't include dropout [8].

  Part of the point of transfer learning is just that we use the pretrained weights as-is so we don't overfit when using a network with many more weights on a small dataset. As I originally intended to do that, I didn't have an augmented data set, and I also feared a bit that doing this might make the epistemic uncertainty (model variance component) depend most of all on the amount and quality of data augmentation instead of on the original data set, which seemed undesirable.
- Because density estimation in high dimensions is not an easy task, and we only have a few thousands points in the training set of each class, any gain in predictive potential by adding features could soon be overpowered by the unevitable decrease in D.E. quality for a dataset of fixed size when the dimension increases. I also had a hard time

[5] As a minor detail, in the last 3 blocks it uses 3 consecutive convolutional layers, so the total is 2x2+3x3=13 convolutional layers. There are variants (VGG19 has 5 blocks with 2x2+3x4=16 convolutional layers), but the pattern stays similar.

[6] per 32*32 patch the original image has, but that's just 1 here

[7] increasing the number of training instances by creating many rotated, stretched,.. etc copies of each original instance. Because these all get the same label, the net will learn to ignore such differences and focus on what really matters in it's prediction of the class.

[8] With most of the lecture coming from the Bayesian Neural Network corner, one tends to forget that VGG is NOT such a net. We could perhaps have regularized by adding a quadratic weight decay term to the cross entropy loss, which would have amounted to MAP with a Gaussian prior over the weights. But with the performance for 2 out of the 3 data sets above 90% right from the start, and a better VGG performance not necessarily equalling a better result for our generative D.E. model, a small potential gain in (neural network) performance didn't really seem to justify considering more complicated architectures and/or higher dimensional feature spaces.

at first finding a KDE method that worked in more than about 10 dimensions[9], and the GMM has to be extremely restricted as well in hundreds of dimensions. Therefore we aim for a few dozen features at the most.

- Then there is the activation function of the last dense layer. The standard VGG method uses activation functions like tanh or relu which produce strongly non-normal features. That's not really a problem when they are inputs to fully connected (FC) layers, but more so if the goal is to fit a multivariate normal for each class. In that case a tanh/ relu activation didn't seem to be the best choice for the layer that produces the features[10].

C.3 Adaptation of VGG16

We use tensorflow with Keras in R to train a model via cross-entropy on the validation set, with Adam optimizer, an initial learning rate of .001[11] (and twice that for dropout), 12 epochs but with early stopping when the performance did not improve more than 0.5% during 5 epochs[12]. The code for this and some helper functions is included in the online code folder. For the reasons discussed in more detail in the previous section, we then adapted the VGG architecture in the following ways:

- We removed one block to be able to use it for smaller images, and because the performance did not suffer at all, we kept removing as much blocks as possible without hurting performance too much, and ended up with 2. That means we can use it for smaller images and train faster.
- We experimented with the activation functions, as it was obvious that relu or tanh activations in the last layer would not nearly produce normal features (cfr fig. C.2 on the right). We ended up using tanh in the first and ReLu in the second layer of each block, and linear in the last (and often only) dense layer before the softmax layer.
- The most important improvement consisted of greatly decreasing the number of channels used. As I didn't do any data augmentation and have much less images, our situation is going to be more sensitive to overfitting. I think this could be why a 'mini-VGG' appeared to perform better.

Before changing the architecture, we tried simply reducing the number of features coming out of the last dense layer before the softmax, but that was less succesfull:

[9] for example the *ks* package used often in R is limited to 6 dimensions according to the manual, probably because they try to estimate a very flexible kernel (eg full-covariance Gaussian) or use a grid. To do this in very high dimensions we probably need very restricted (eg coördinate aligned)) kernels.

[10] This became obvious from a few plots in arbitrary coordinate planes, showing for example most of the data points smeared out against the positive X and Y axis for relu activation in the last layer (see right side of fig. C.2), or in the 4 corners of the $[-1, 1]^2$ square (for tanh).

[11] We set a so-called callback 'reduce learning rate at plateau' which reduces the learning rate when performance levels off, but in retrospect this was never used because the default patience of 10 epochs was much larger than the early stopping patience (cfr below).

[12] In retrospect, this early stopping tolerance is way too crude, especially for datasets with very high baseline performance like Fashion and MNIST. These values were chosen initially when working with Cifar which had much lower performance, because we were only after approximative performances, and have to compare many different architectures so speed is an issue. In retrospect, this could be the reason why we don't see much differences between architectures: if the improvement for the best models is much slower from that point onwards, training will stop too early and fail to recognize that these are better architectures.

- One of the suggestions was PCA, and a screeplot and the cumulative variance for the 512 features produced originally are found in figure C.2 (left& middle), but there we see that at the 'knee' of the screeplot, at about 18 components, we still have only 45% of (**input**!) variance, and it takes up to 24,117 and 221 components respectively to get 50,80 or 90%. On top of that, this gives us no guarantee whatsoever about the fraction of **output** variance this will capture.
- I also considered supervised feature selection (using JMIM and RF importance scores again like for the toy datasets), but the results were rather disappointing, perhaps because they were not meant to be used only partially.
- I also tried adding more dense layers to gradually decrease the number of features, but results were much worse with 2 dense layers before the softmax than with 1. This is almost certainly because the network had too many weights to be used without regularization. Still, adding dropout did not seem to change this picture dramatically.

I think we can understand the above as follows: when we have 64,128, and eventually 512 channels, any trends that the network picks up are divided over (encoded in) those features, and by selecting only a handful of them eventually we will throw away most of what the network has learnt. So when the final goal is to represent all trends in only 10 features, it is better to start with less channels right away, and train the network accordingly so it can focus on the main trends only from the start and encode as much information as possible in a number of features comparable to the number that we will eventually use.

One of the risks was that the architecture optimal for our VGG performance would produce features that are much less optimal for our generative model. After all, we rely upon the features having a normal, (or simple gaussian mixture) density, and certain choices (relu activation etc) may perform better in a VGG but worse for the VGG with last layer replaced by our discriminant-analysis-like model.. To protect us from this risk, we monitor aside from the VGG performance also the test accuracy of the VGG with the last layer replaced by an LDA model. While our model is not limited to LDA, we can reasonably hope that models where both the VGG and LDA performed well on a test set, will also do well with our generative model.

Ideally we would cross validate the performance of the cautious classifier (for some choice(s) of α) for several types of density estimation methods on the features produced by each possible VGG architecture, to choose the architecure, i.e. proper model tuning. But this would be very time consuming, and at the time, because of the satisfactory performance (+90% on test set for 2 out of the 3 datasets), we considered the question of the best architecture more or less closed, and focused on the remainder of the method. Much later we came back to do an a posteriori sensitivity check to see if we could justify our initial choice,and because both the performance of the VGG and of the VGG with last layer replaced by LDA remained comparable for (some) of the much simpler architectures[13], I eventually used the following parameters:

[13]We ran the VGG and VGG+LDA for an extensive grid of combinations of numbers of channels in blocks 1 and 2, number of nodes in the last (pre-softmax) layer, activation functions, with and without a thrid block with several possible # channels, with/without extra dense layer(s), and with different dropout rates in the dense and/or convolutional layers, for all three datasets, to compare performances VGGval,VGGtest, LDAval, LDAtest in a large table. However, we later found that the variation between different repetitions for the exact same settings (±0.5%) was quite large in comparison to the differences between different settings (perhaps in part due to the crude tolerance for early stopping). Namely we find a continuum of performances with dozens of settings within

- Cifar: 64 channels both in blocks 1 and 2, and **16** neurons in the last layer.
- Fashion: 64 resp. 128 channels, and **8** neurons in the last layer.
- Mnist: 32 and 64, and only **6** neurons.

In conclusion, rather than modifying the VGG and assuming weights trained in one architecture are still usefull in a different one, or augmenting the dataset itself (and changing model variance that way), or retraining such a large network without proper regularization, it seemed safer to retrain a network that was similar in spirit but much simpler: less blocks, less channels and smaller dense layers, so it would not overfit when used with only 3000 training points per class.

C.4 Data Splitting

Recapitulating from beginning to end we have:

- We split the training data a first time in train(I)+valid(I), and monitor the performance on valid(I) of the model trained on train(I) to avoid overfitting. After the model is determined, we feed all instances of train, validation and test to the network and replace the original features x by the outputs x' of the last layer before the VGG's softmax.
- We then apply cautious classification to this new dataset: instances of train(I) and valid(I) are recombined, shuffled, and resplit in train(II)+valid(II). The density $p(x'|j)$ in the space of new features is learnt on train(II), and the density thresholds t_j (the α quantile of the set $\{p(x'|j)\}|x' \in V_j\}$) for each class are determined from valid(II).
- The test set is now scored by evaluating the training density $p(x'|j)$ (learnt on train(II)) for all $j = 1 \ldots K$ classes in the test point, and comparing it to the threshold t_j for the same class learnt from valid(II). We retain in the predicted set for a test point at x only those classes for which $p(x'|j) > t_j$.

Because the class-wise density estimation in the space of features produced by the VGG's last layer is an unsupervised task, we did not think it required a completely independent validation set[14]. The test set is kept apart at all times however.

only 0.5% of each other, while the variation between runs for a same setting is also of that order. The tables can therefore not be used to determine the 'best' architecure, and were left out. On top of that, the 'best' VGG/LDA choice may not even correspond to the best choice for our VGG+cautious classification based on GMM/KDE/NN/.. model. We therefore selected eventually the simplest model that still had comparable (ie within .5%) VGG and VGG+LDA performances.

The only thing we can conclude from the comparison, is that results are remarkably stable over architectures. Apart from the crude early stopping criteria, this may in part be due to the conformal classification, which could hide large differences in precision behind more or less comparable accuracies!

The authors of [7] stress how results depend only on the correct **ordering** of predicted probability densities. While the differences in VGG/VGG+LDA accuracy are surprisingly small, there are probably larger regional differences, and we should also stress that the best D.E. method for the features produced by one architecture may differ a lot more from that produced by another, and any conclusion about the best DE method for a certain dataset can change completely when another architecture is chosen.

[14] However, once we start making decisions based on these test performances, like the number of components to use in the GMM, or the type of density estimation to use, we would need another separate set to get unbiased performance estimates. We can still assume that the decisions we make are the best ones, but the actual value of the performance of the 'winning' option will then be biased upwards, as we selected the one having maximal performance.

C.5 PCA for the 512 standard-VGG channels

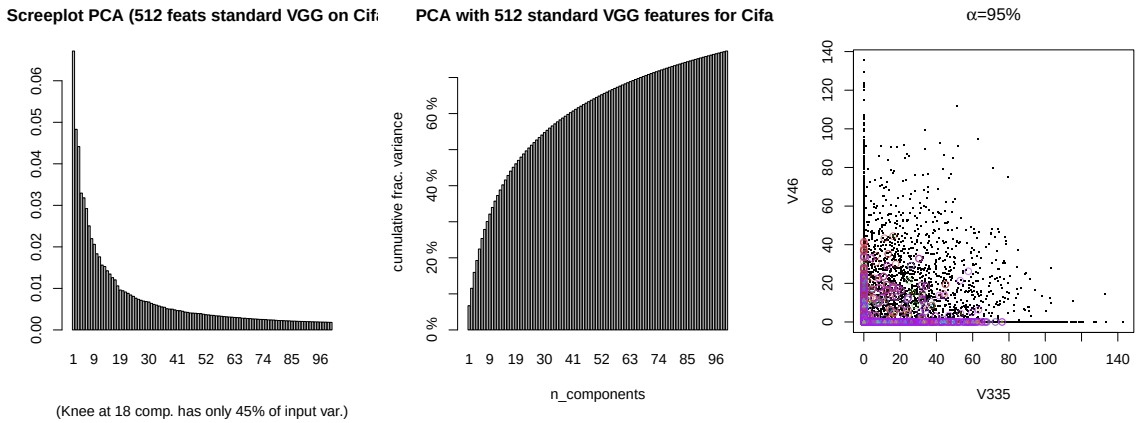


Figure C.2: Left: Screeplot. Middle: cumulative variance versus ncomponents. Right: Example of outputs for relu activation, even at very high thresholds we can hardly reject points when we assume normal classes with such a dataset. (Colors denote points accepted in different classes).

APPENDIX D

FIGURES OTHER IMAGE DATASETS.

D.1 Cifar

D.1.1 Cifar: Cautious classification only

First results leaving out only peripheral points of each class: Figures D.1-D.2, discussed in section 3.4.2.

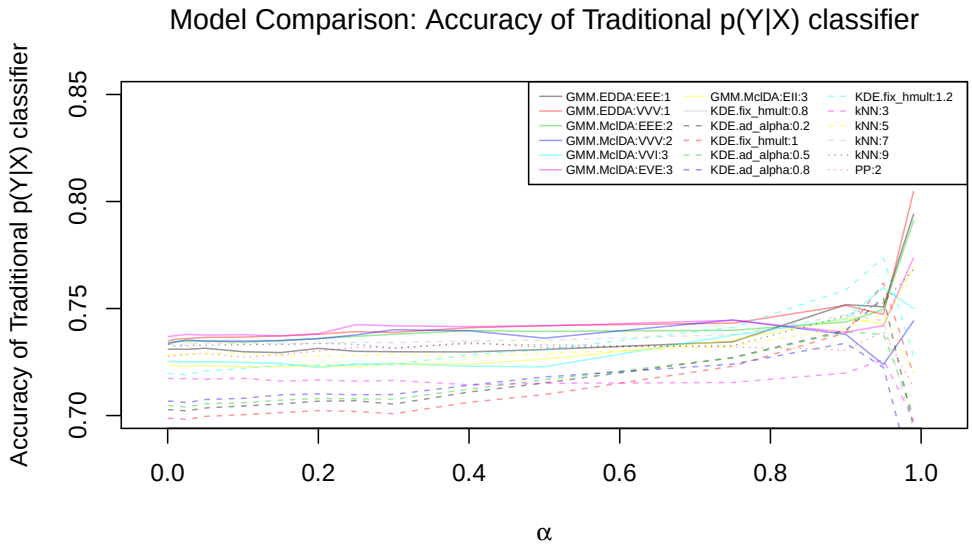
D.1.2 Cifar: Rejecting also points with aleatoric uncertainty

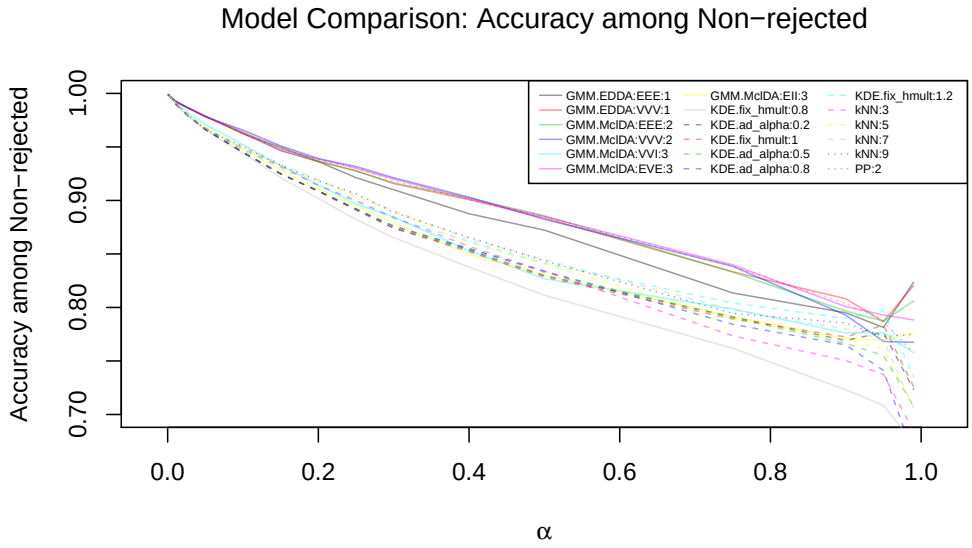
Figures D.3-D.4, are discussed in section 3.4.3.

D.2 Mnist

D.2.1 MNIST:Cautious classification only

First results leaving out only peripheral points of each class: Figures D.5-D.6. We see that rejecting about 20% of points (RR and α are quasi interchangeable) brings accuracy from 97.5% to about 99.75%, or decreases by 10 fold the error rate. After that improvement is much slower, so in some situations, it could be interesting to exclude a small fraction of points to achieve much higher accuracies. We also see how the precision reaches very high values (98-99%) as soon as 1-2% of points are rejected. So the classes appear well separated.





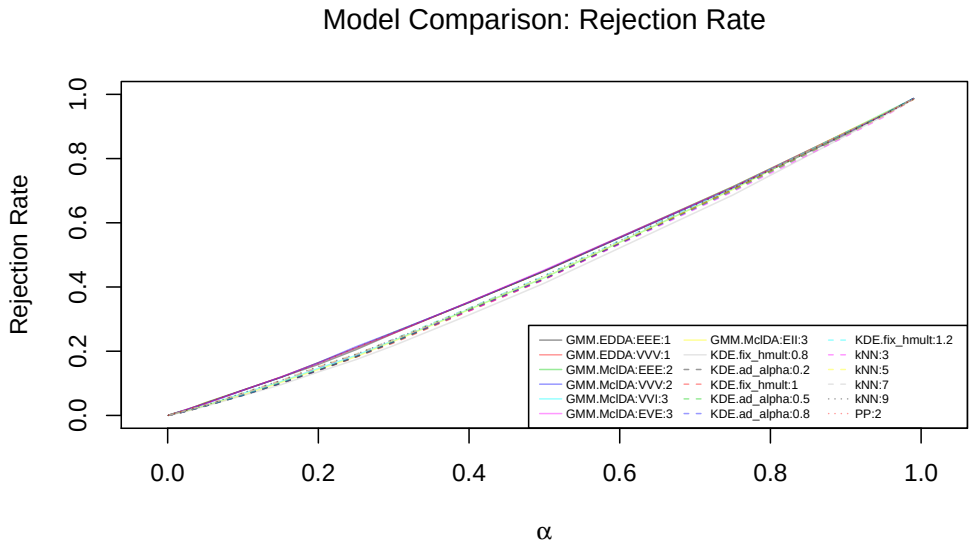
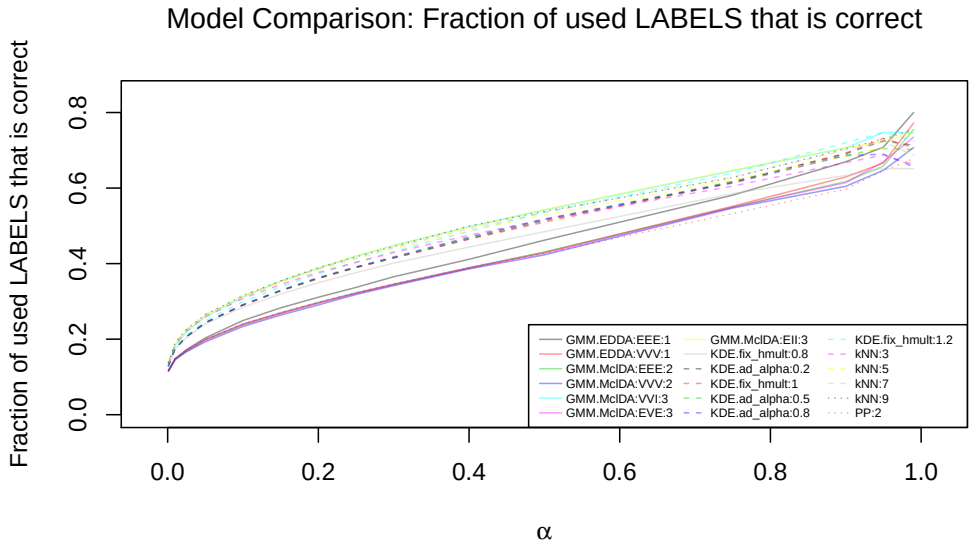


Figure D.1: Results Cifar, Part1. Top row: traditional accuracy. 2nd row: accuracy of set valued classifier. Third Row: Rejection Rate RR (from all classes).



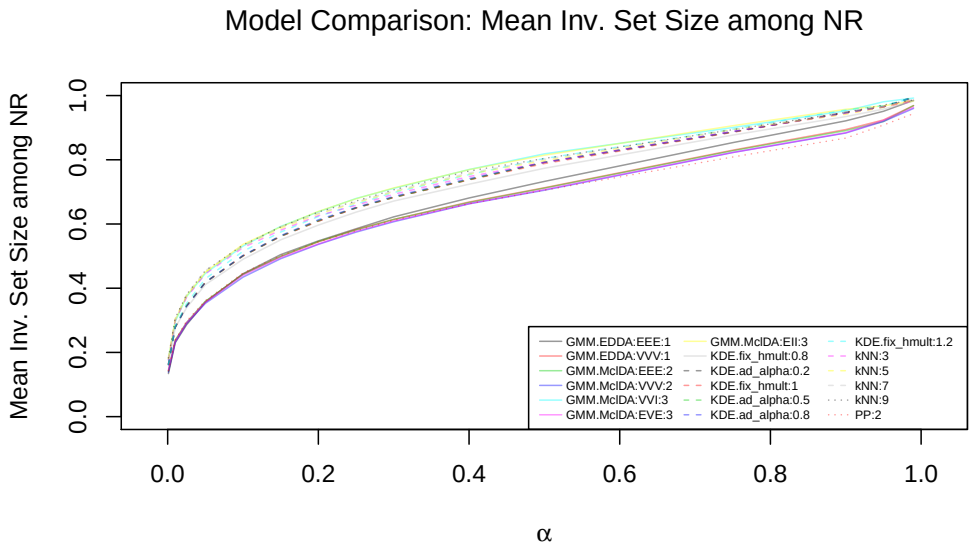
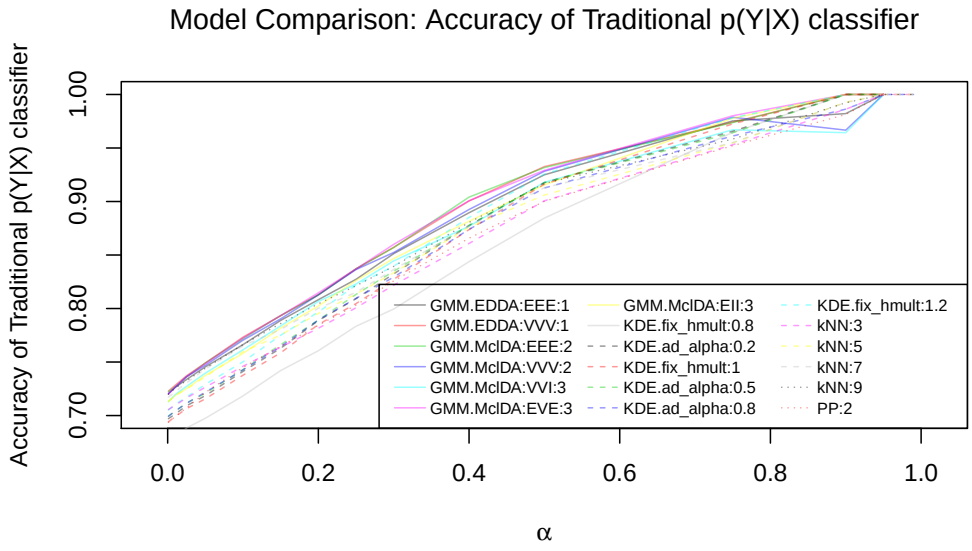
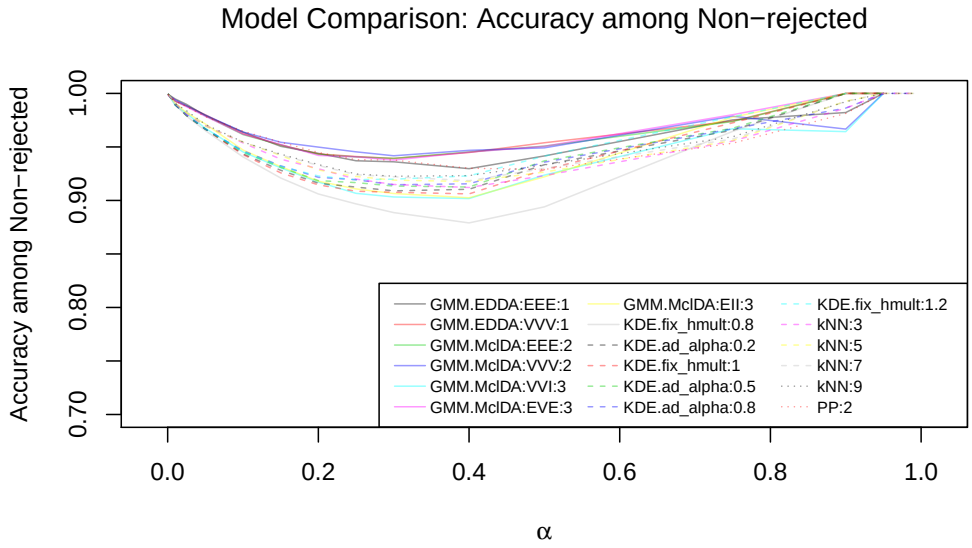


Figure D.2: Results Cifar , part 3. Top Row: 'Traditional' precision, i.e. fraction of labels used that is correct. Bottom: Mean inverse set size among NR.





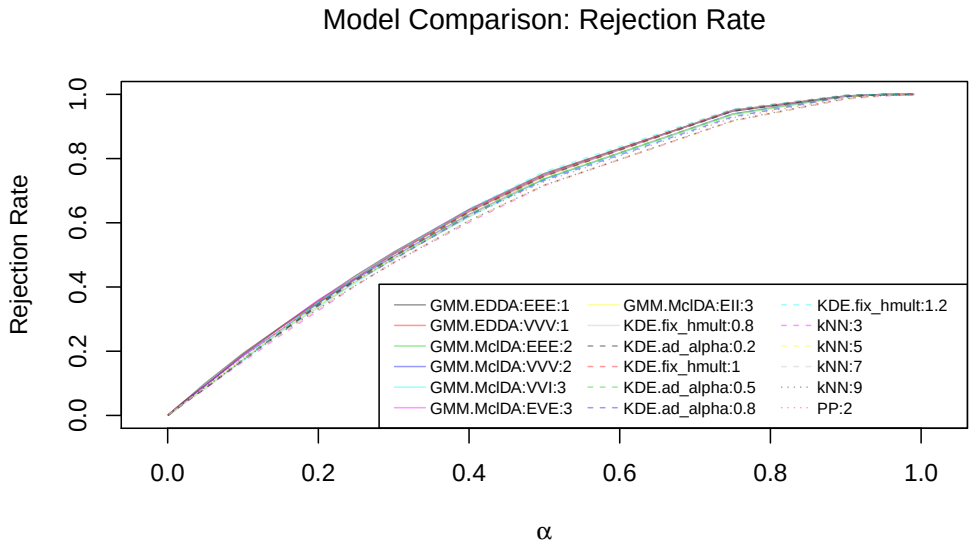
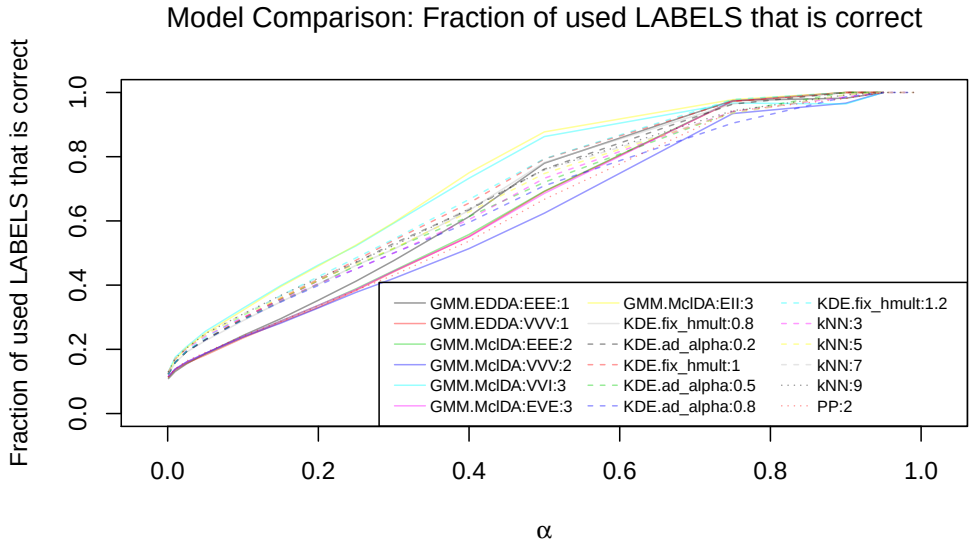
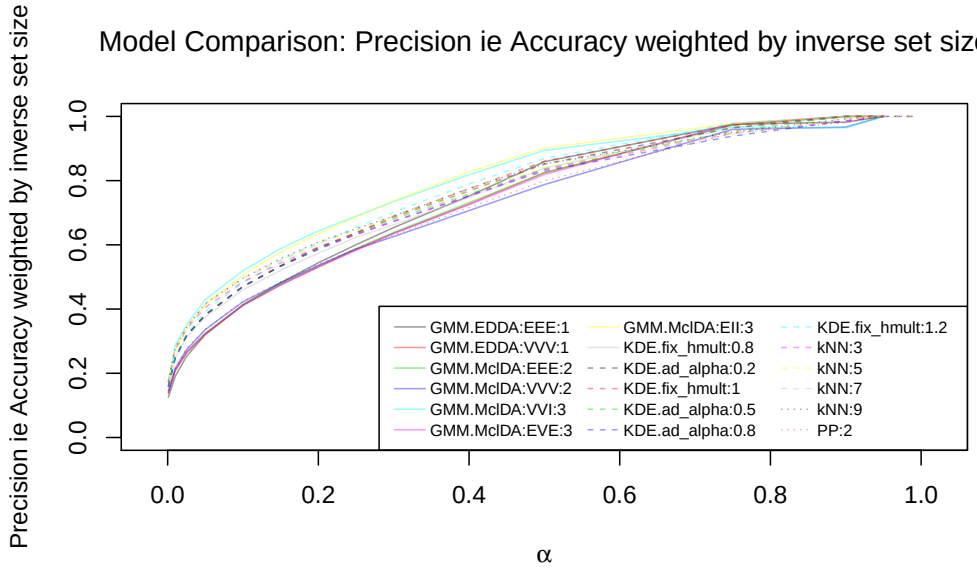


Figure D.3: Results Cifar, Part1. Top row: traditional accuracy. 2nd row: accuracy of set valued classifier. Third Row: Rejection Rate RR (from all classes).





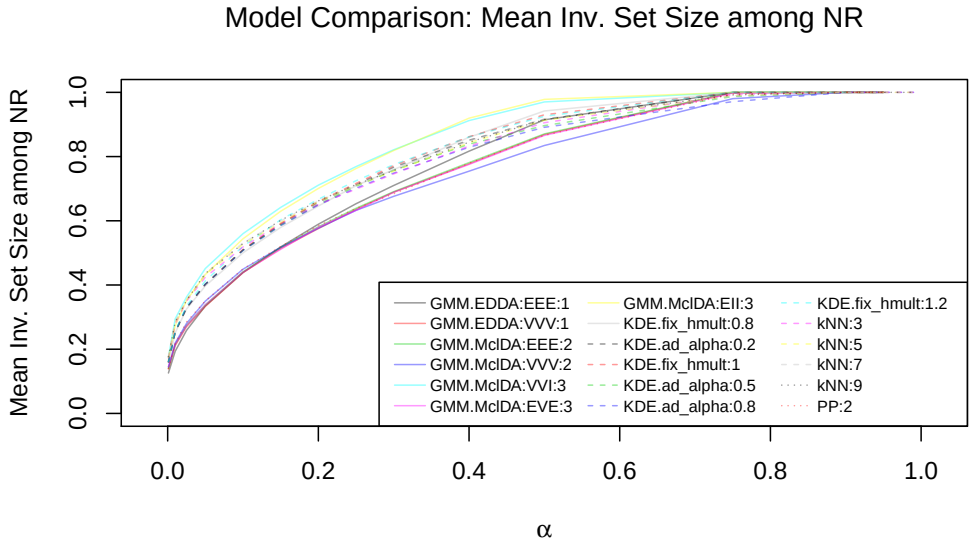
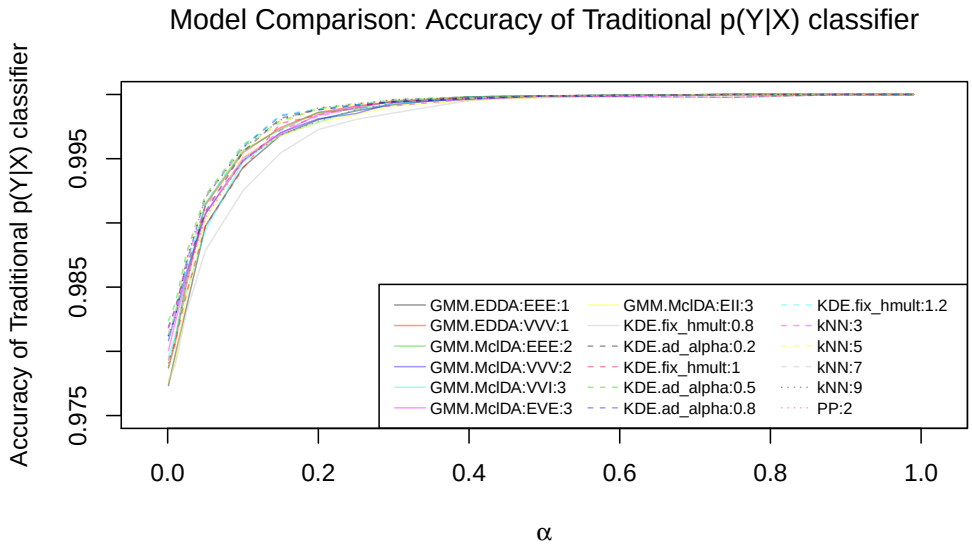
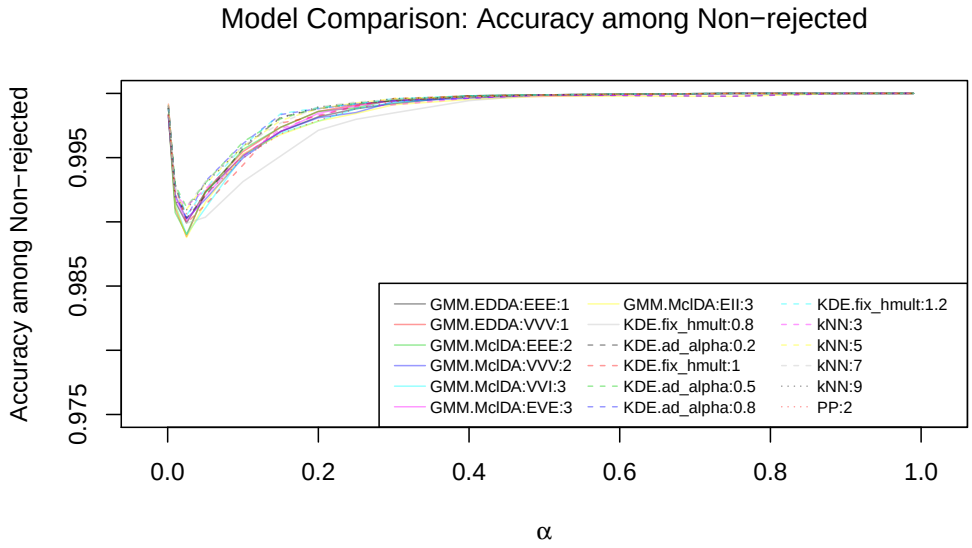


Figure D.4: Results Cifar , part 3. Top Row: 'Traditional' precision, i.e. fraction of labels used that is correct. Second Row: Set valued 'precision-accuracy' AP (cfr footnote 35 on 47). Bottom: Mean inverse set size among NR.





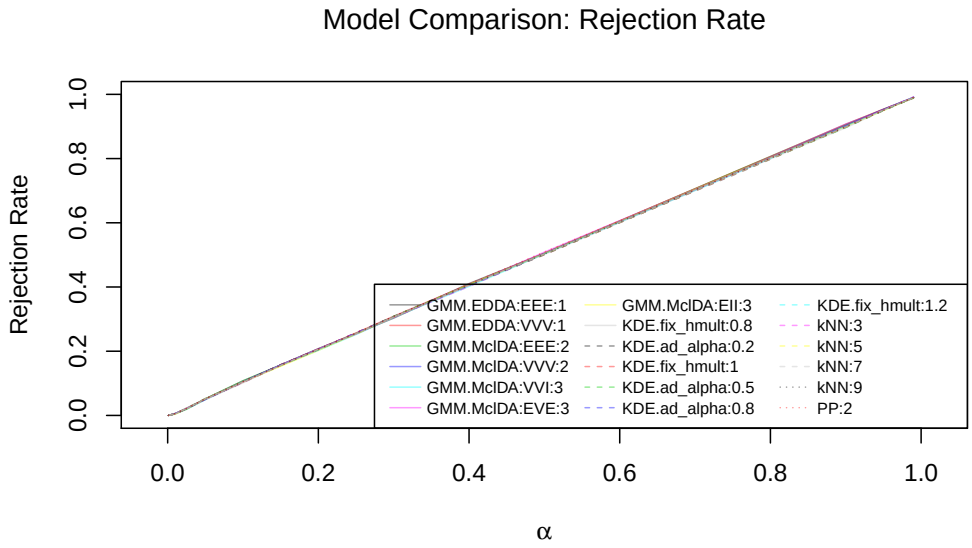


Figure D.5: Results Mnist, part1. Top row: traditional accuracy. Second row: accuracy of set valued classifier. Third Row: Rejection Rate (Fraction of test points rejected from all classes)

D.2.2 MNIST:Rejecting also points with aleatoric uncertainty

Figures D.7-D.8. Here, rejecting only a few percent of points now brings the traditional accuracy to about 99.7%, so it appears we are correctly identifying the hardest points, which leads us to believe that the density estimation models (nearly all of them) do a good job. Probably that means the true distribution is quite simple, otherwise I would expect the differences between GMM's and non parametric methods to be larger.
The fact that we have RR clearly less than 2α right from the start (about 25-30% at $\alpha = 0.2$ whereas it used to be 0.2 when we rejected only OOD points), suggests the most overlap now occurs more in the periphery than in the center. That appears to be confirmed by the very sharp increase of accuracy and precision after rejecting the most exterior points.

D.3 Fashion (with Aleatoric)

The results for the standard cautious classifier are in the main text (section 3.4.2), and here in figures D.9-D.10 we add those when aleatorically uncertain points are rejected as well. These are briefly discussed in section 3.4.3.

D.3.1 Fashion: Rejecting Also Points with Aleatoric uncertainty.

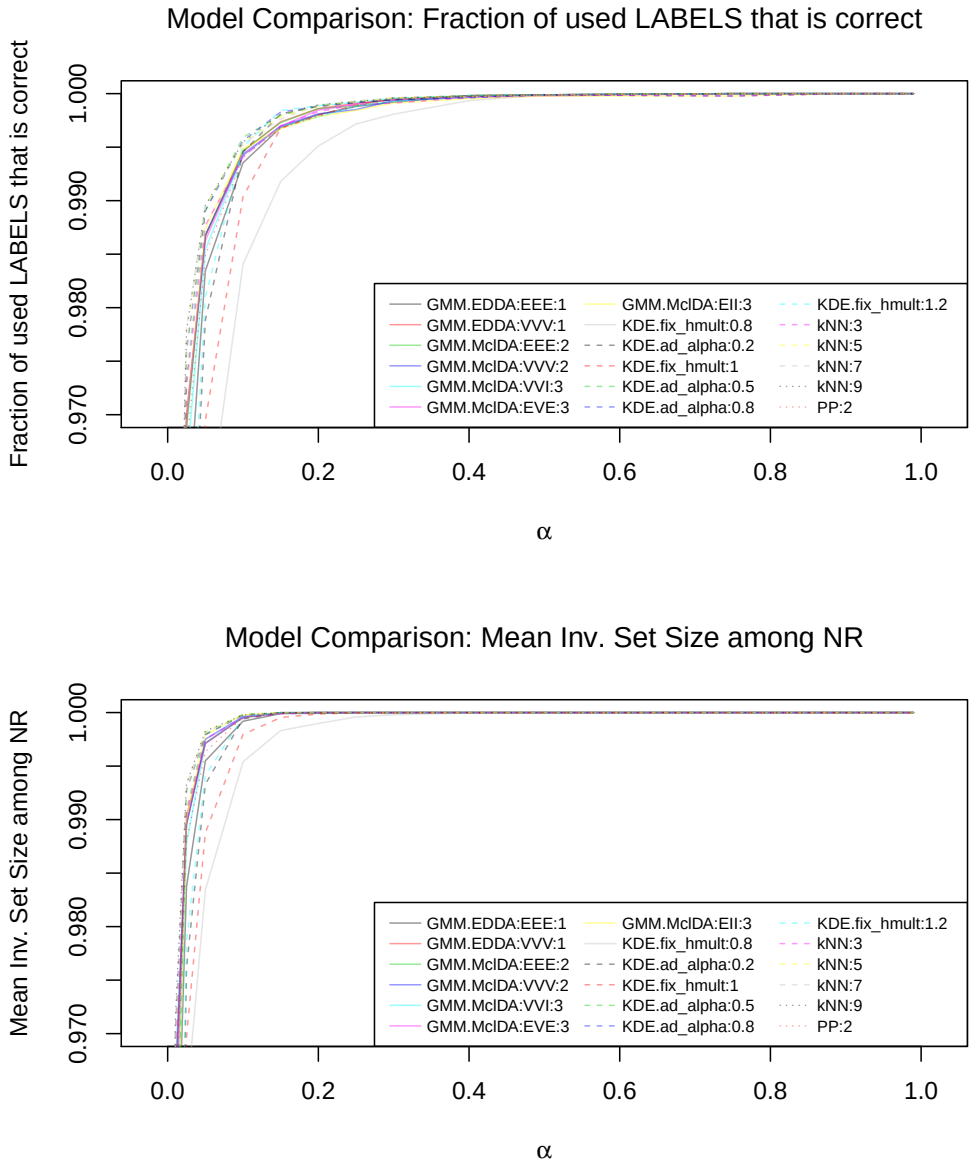
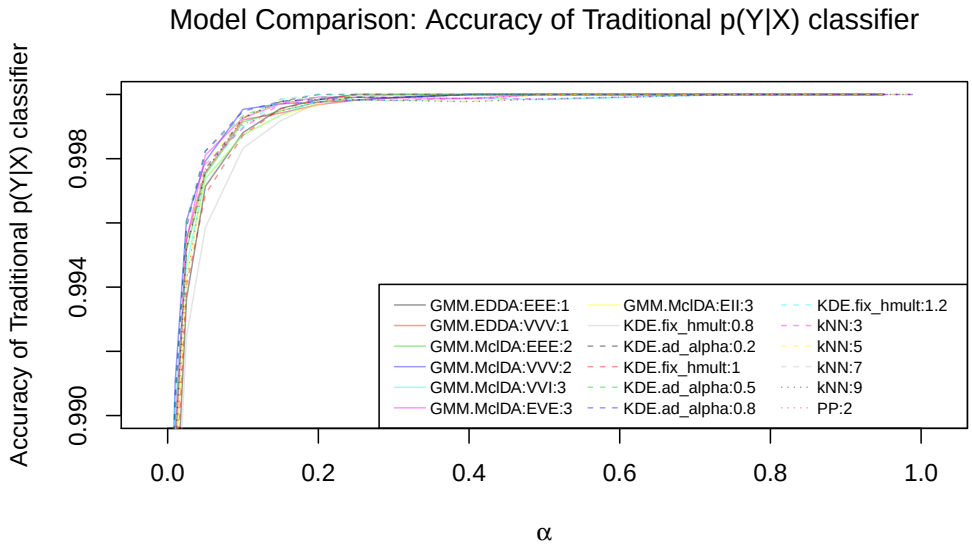
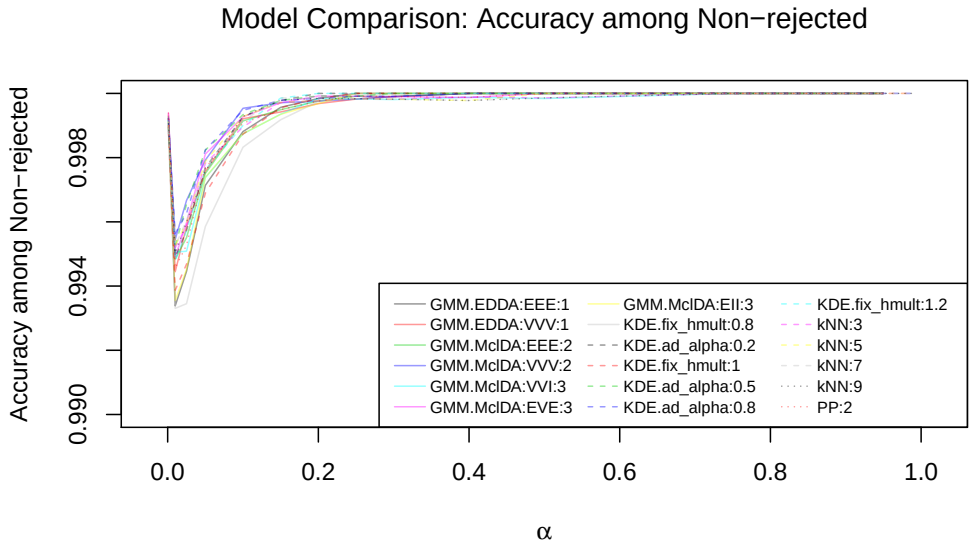


Figure D.6: Results Mnist, part3. Top Row: 'Traditional' precision AP (fraction of all labels used that is correct). Bottom: Mean inverse setsize amongst NR points.





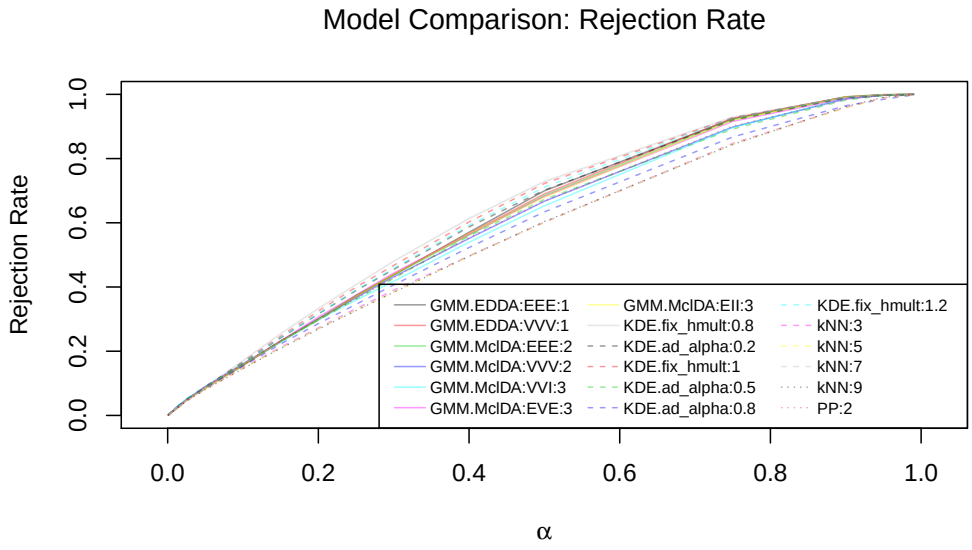


Figure D.7: Results Mnist, part1. Top row: traditional accuracy. Second row: accuracy of set valued classifier. Third Row: Rejection Rate (fraction of test points rejected from all classes).

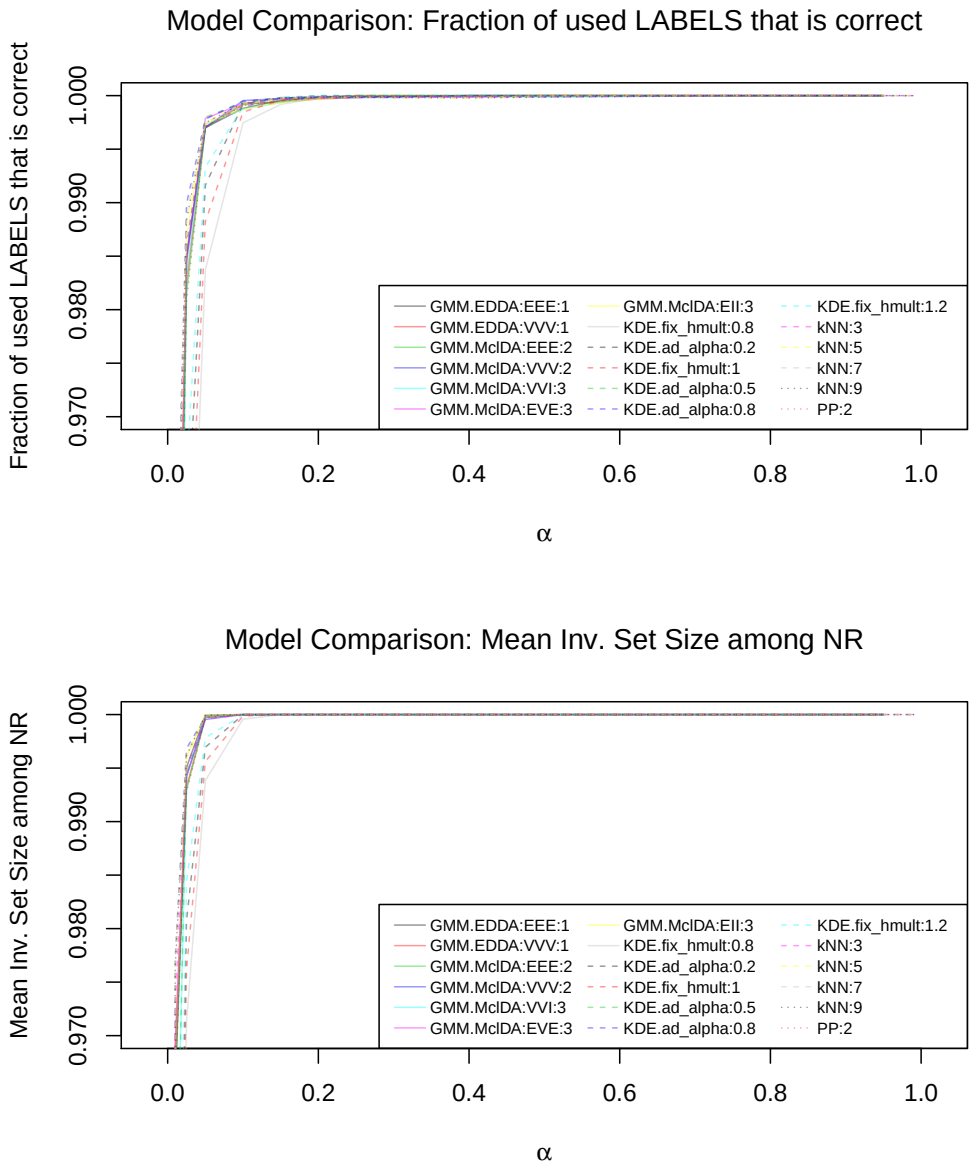
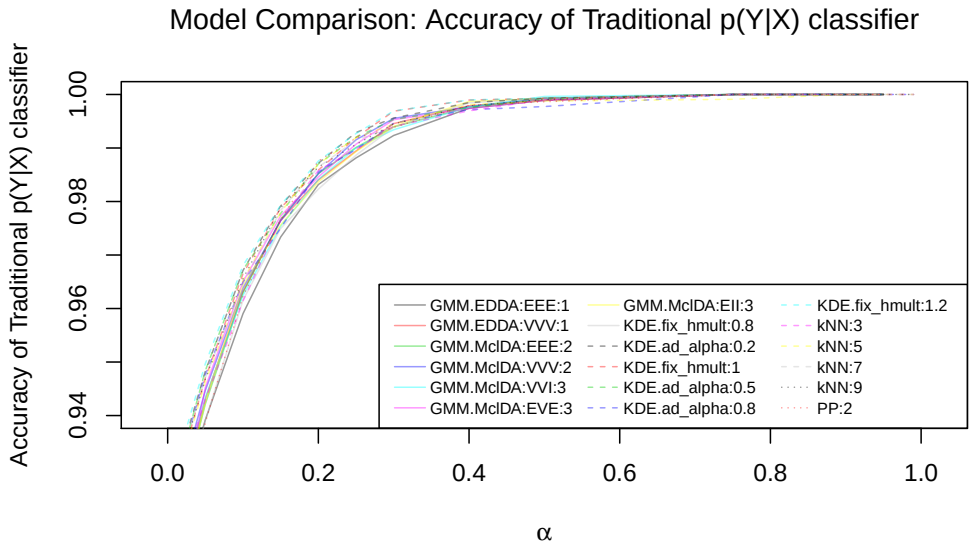
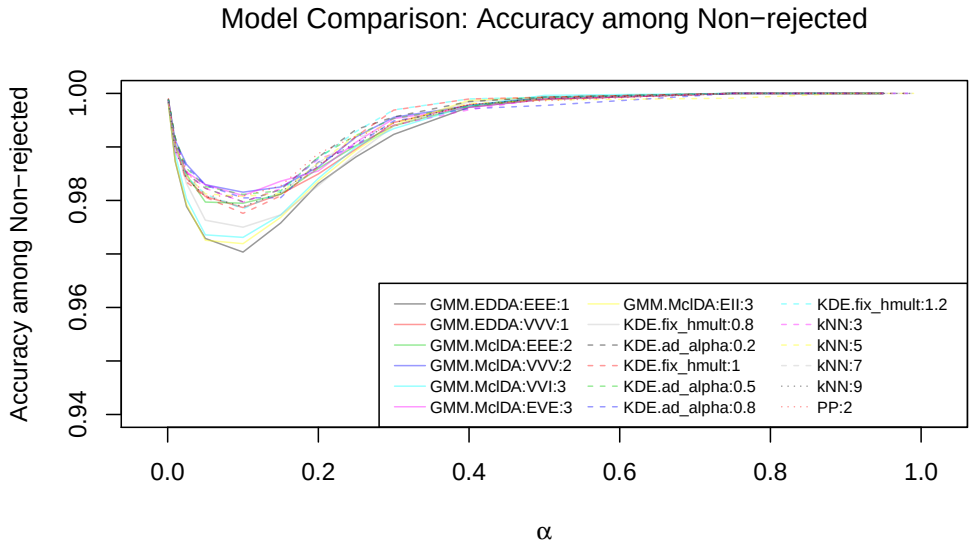


Figure D.8: Results Mnist, part3. Top Row: 'Traditional' precision AP (fraction of all labels used that is correct). Bottom: Mean inverse setsize amongst NR points.





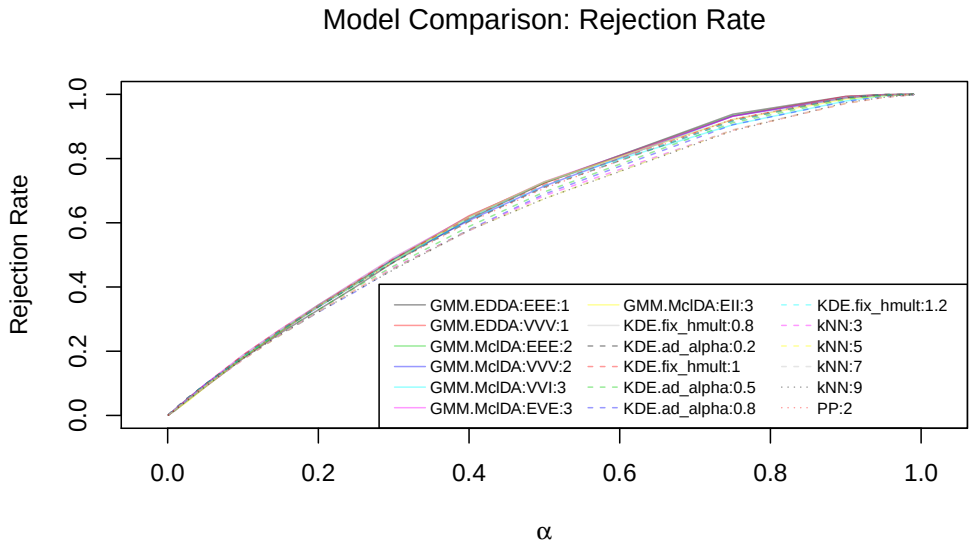


Figure D.9: Fashion: Top row: traditional accuracy. Second Row: accuracy of set valued classifier. Bottom: Rejection Rate (from all classes).

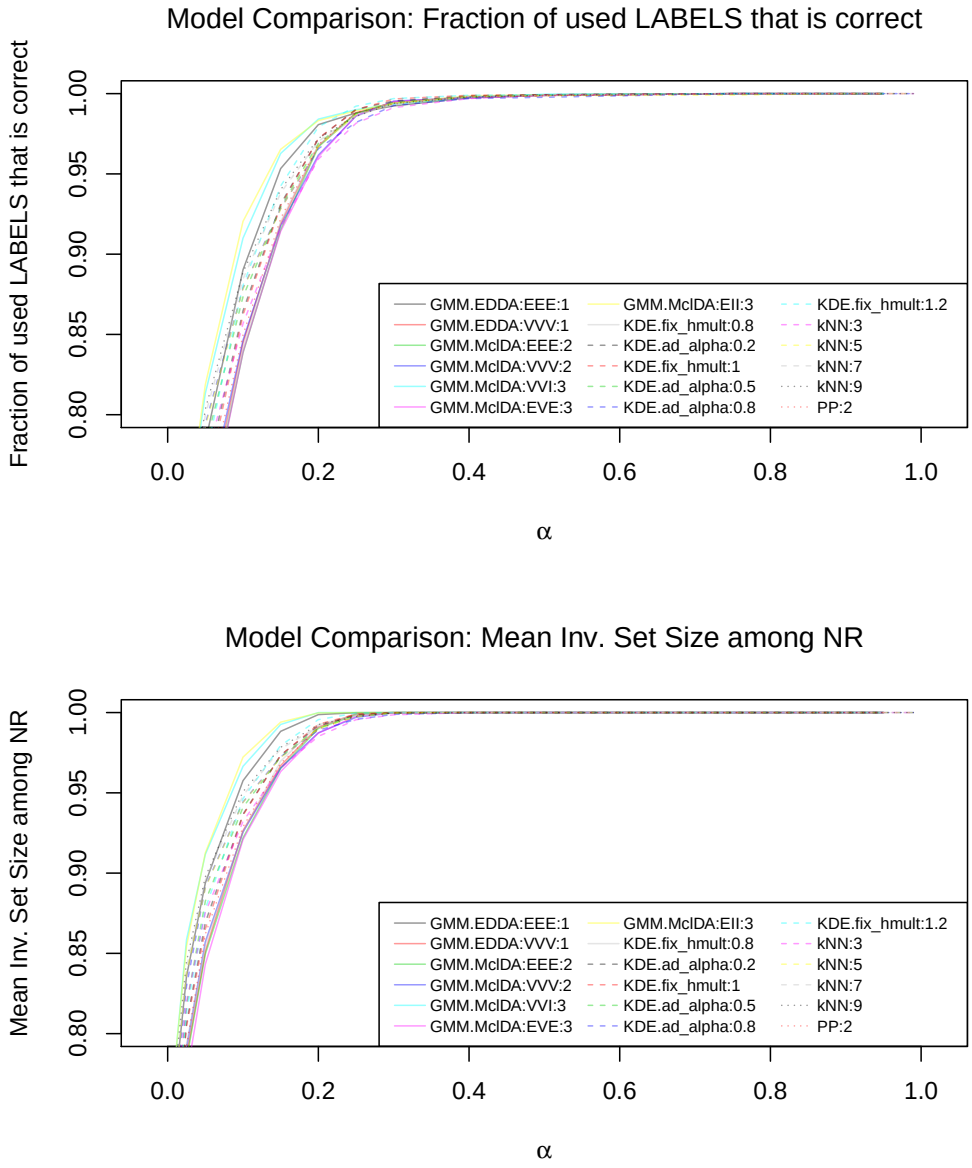


Figure D.10: Fashion: Top row: Precision (traditional, i.e. fraction of labels used that is correct). Bottom: Mean inverse setsize amongst NR points.

APPENDIX E

QUANTIFICATION OF EPISTEMIC AND ALEATORIC UNCERTAINTY

E.1 Introduction

The method proposed in section 2.3 and to be applied in chapter 3 can be a useful way to reduce distributional uncertainty, but it unfortunately does not allow one to quantify the uncertainty of different types in a given test point. It is also limited in the sense that it heavily relies on our density estimates (at least on the ordering between densities at different locations), and this can be problematic when there is a large bias or model variance. Ideally, we would like to know what the predominant nature of uncertainty is at each point, so we can take the appropriate action (collect more data, reject a point, manual classification, get additional features etc..). It is also obvious from what preceded that aleatoric and distributional uncertainty vary strongly between points, and this is no different for the effect of model variance on our prediction[1].

Traditionally, however, epistemic uncertainty is seen as uncertainty about parameter values, and it can indeed not be seen independently from the model that is chosen. In an inference context, the uncertainty about those parameters is a crucial datum. But when we are doing prediction, I would think that the uncertainty about parameters only matters through the variability in predictions it implies. Without this 'translation' of uncertainty into what is actually observable, it would also be impossible to compare uncertainties between models with different parametric shape, or even between epistemic and aleatoric uncertainty, for predictions live in the same space as the targets, but not so for the parameters. Unlike in inference, in prediction these parameters are mostly an approximate hypothetical construct of ours, and a simple reparametrisation of the model (eg embedded in a more general one with extra parameters, which might then be irrelevant, underdetermined or correlated) would completely change the dimension and volume of the region of parameter space where the solution lives, while it might have little to no effect on our predictive uncertainty. Therefore the volume occupied by plausible solutions in that parameter space seems of little use as a practical predictive uncertainty, and we have chosen here to see epistemic uncertainty as the variability in *predictions* produced by either different parameter values in a same model (model variance), or by different models (distributional uncertainty)[2].

[1] Think for example of the different effect of a same uncertainty about a slope in linear regression (or about a covariance matrix in density estimation) for predictions further away from the center of the training data.

[2] Capturing the latter without a posterior over all plausible distributions seems hard, but we can transfer this distributional uncertainty largely to the model variance component by considering non parametric models which hardly make any distributional assumptions. The increased variability of these models under different training sets will then capture part of the uncertainty we used to have about the validity of our parametric model on the edge of the domain.

By sampling several training sets of given size from a distribution (or bootstrapping from a 'mother' dataset), and fitting the model for each of them, we can in each point get an ensemble of predicted conditional densities $p(x|i)$, marginal densities $p(x)$, and ultimately also predicted class probability vectors $\vec{p}(x) \equiv p(i|x)_{i=1,\dots,K}$[3].
That way we hope to quantify how variable the quantity that determines our decision, $\vec{p}(x)$, actually is in each point of feature space, regardless of model parameters, and compare this between a parametric and a non-parametric model. We will then look at different ways to extract the uncertainty of each type from these ensembles, and visualise how these measures evolve as the training set size increases.

By looking at $\vec{p}(x)$ as the output of a first 'phase' of the model, we are hierarchically splitting the uncertainties: The model that has $\vec{p}$ as output contains only what was our epistemic uncertainty about Y[4]. Aleatoric uncertainty on Y has now been detached, to a second 'phase' of the model, which takes a particular realization $\vec{p}_j(x)$ as input and predicts the class label $Y(x)$. The variability of the *prediction* is now zero, as the predicted label $Y = \mathrm{argmax}\,\vec{p}_j(x)$ follows deterministically from the class probability, and only the outcome Y varies: this is pure aleatoric indeed. Because aleatoric uncertainty is what remains in spite of knowing the correct class probability, we can for each realization j separately treat the $\vec{p}_j(x)$ received by this second 'phase' as correct, and calculate the variance/entropy/Bayes error/.. of Y accordingly. A simple ensemble average over $j = 1 \dots K$ then gives us our aleatoric uncertainty at x.

Remember we are not minimizing the cross entropy to a series of observed outcomes, but estimating class densities, and the distribution of predicted $\vec{p}(x)$ will inherit its variability from the method of density estimation we use: a flexible non-parametric model that is unbiased will typically have more variable predictions for the class probability vector than one that makes strong distributional assumptions. But in the latter the uncertainty about the validity of our distributional assumptions goes uncaptured[5].

Now we can focus on the first part of the model, which produces the ensemble of densities. We have two possibilities: we could do as suggested above, and model the effects of sample variance through the distribution of MLE estimates it induces for $\vec{p}$ at each x. Or, we could explicitly learn a *distribution* of class probability vectors at each x, starting from a prior distribution over class probability vectors (eg Dirichlet) at x, and updating it using the previously-mentioned set of MLE estimates for $\vec{p}$ as 'observations' of our target (which is now $\vec{p}$). Such a model no longer makes definite predictions for $y(x)$, but explicitly tries to describe the epistemic uncertainty due to model variance at each point. This is also what is done in [2] to reduce the model's confidence in regions where not much training data was observed, and we have tried to use something similar here, by assigning more weight to 'observations' where the data density is higher. We will come back to this in a moment.

In light of what we discussed for the cautious classification however, let's keep in mind that although $p(y|x)$ seems more interesting because it's what we use to make decisions, we are

[3] the vector arrow will hopefully prevent confusion with the marginal density $p(x)$: the vector $\vec{p}(x)$ has for its i-th component the probability of finding class i at location x.

[4] indeed, by letting the training set size become infinite, $\vec{p}$ will have a well determined value in each point of its domain, so there is none of the original aleatoric uncertainty here.

[5] That weakness is also the strength of the parametric model: its distributional assumptions will yield more stable predictions where there isn't much data, so if we are really sure our distributional assumptions continue to hold in those regions, we will be able to do a lot more with less data.

already using certain model assumptions (such as the non-existence of other classes) when we convert K density estimates $p(x|i)$ into $\vec{p}_i \equiv p(i|x)$. If we are interested in the cautious classification alone, the purest thing to do might be to study the variability of the estimated density $p(x|i)$, That way we would disentangle the uncertainty in the density estimation of each class from that of the others[6].

For that quantity it could also be feasible to obtain an analytical expression in the case of a single normal component per class, and we first want to outline a potential approach for this in the next paragraph, because an analytical result for the distribution of $\log p(x|i)$ in any fixed point x (not part of the sample) could be very useful in higher dimensions where working on a grid is prohibitive[7].

Outline of an analytical approach for Gaussian classes

Since we know the sample mean $\hat{\vec{\mu}}$ and sample precision $\hat{\mathbb{\Sigma}}^{-1}$ of a set of n points drawn from a normal with given population mean $\vec{\mu}_0$ and precision $\mathbb{\Sigma}_0^{-1}$ to be normal-inverse Wishart distributed[8], we should be able to get[9] also the distribution of the *estimated* Mahalanobis distance $m_i(X) = -\frac{1}{2}(X-\hat{\mu}).\hat{\mathbb{\Sigma}}^{-1}.(X-\hat{\mu})^T$ to any fixed grid point X from that of $\hat{\mathbb{\Sigma}}^{-1}$ and $\hat{\mu}$. That would allow us to immediately draw from a distribution of logdensities (=Mahalanobis distances) for each class at X (rather than having to draw an ensemble of datasets and refitting those densities each time).

However, it turns out that sampling from such a Wishart distribution is done in practice by drawing normal vectors and getting their scatter matrix. Also, because we are interested in eventually obtaining also the distribution of $p(y|x)$ (which depends on that of all K class densities and of their sum), it seems better to proceed by sampling right from the start. In that case the reason to restrict ourselves to a single normal component per class also disappears, and we will therefore draw datasets from a gaussian mixture for each class. This is recapitulated schematically in Algorithm (1). Now we describe the particular mixture data set we created for this experiment.

Experimental setup

We consider 3 classes in two dimensions. In a first scenario they all consist of a single normal distribution with equal covariance matrices and the 3 class centers equidistantly along one of their principal axes (chosen as x axis). There we know more or less what would like to find for the uncertainties, and this serves mostly as a 'calibration' exercise.

In the second, more interesting situation, 3 classes consist of respectively 6, 3 and 1 normal components. The former has 6 randomly-drawn centers in the region [-5,5] *[0,10], the

[6] In that case, a relevant uncertainty might be extracted from an ensemble of densities $p(x|i)$, and a given α. The number of times the point is rejected from the class could be considered as binomial with probability r, and for example $H(r)$ or $r(1-r)$ might then express some uncertainty about whether to accept the point in that class.

[7] Although we could limit the number of points where we produce ensembles of predictions to those in the test set if necessary, we worked on a grid here mostly because we are also interested in visualising the spatial dependence of uncertainty measures.

[8] It helps to think of the one dimensional equivalent: the distribution of a sample variance s^2 estimated from n observations **given the true population variance** σ_0^2 is scaled χ^2. The Wishart is the multivariate generalization of the χ^2, and likewise it describes the sampling distribution of the observed covariance matrix given some true population covariance.

[9] perhaps through a matrix version of the deltamethod, or via the distribution of the eigenvalues and vectors of $\hat{\mathbb{\Sigma}}^{-1}$, in the worst case through numerical integration.

Algorithm 1: Producing an ensemble of predictions to visualise uncertainty.

for *n in trainingset sizes* **do**
  for *j in 1:N_{sim}* **do**
    Draw training set D_j of n independent samples from the distribution (or bootstrap from the mother dataset).
    for *i in 1:K classes* **do**
      Estimate density (from sufficiëntly flexible hypothesis space) for class i.
      for *x in testgrid/-set* **do**
        Evaluate that density in each test point and store this as the density for class i in testpoint x during the j-th realisation for total training set size n.
      end
    end
    for *x in testgrid/-set* **do**
      Add up the values of $p(x|i)$ to $p(x)$. Use the marginal proportions $p(i)$ of each class in D_j to obtain $\vec{p}_i(x) \equiv p(i|x) = \frac{p(x|i).p(i)}{p(x)}$
    end
  end
  -In each test point we now have ensembles of $p(x|i)$, $p(x)$ and $p(i|x) \equiv \vec{p}(x)$.
  -Calculate from those our uncertainty measures at x for training set size n.
  -Produce contour plot of these measures (or their logarithm) at set size n.
  - In a classification setting: reject x where the uncertainty is too high.
end
Animate the evolution of uncertainty measures with n.

second has it's 3 cluster centers in (-3,0), (0,0) and (0,3), and the third a single cluster with center in (-4,-2). For the 6 component mixture, we have 3 very different subgroups of two clusters each with equal covariance. The 3 groups differ in orientation of principal axes and their aspect ratio long to short axis. The second class has the identity matrix as covariance in all 3 clusters, and the third class diag(0.2,0.8).

The dataset is shown on the left in Figure E.1. For the details on the methods of density estimation, we refer to chapter 3. We have determined densities here with unrestricted Gaussian Mixtures ('VVV' with 6-3-1 components respectively) and KDE (both with fixed and adaptive bandwidth). Before proceeding with the results and their discussion, we discuss a series of candidate-measures for different contributions to the uncertainty.

E.2 Uncertainty Measures

Aleatoric uncertainty Because the aleatoric uncertainty is determined by the value of the class probability (rather than its distribution), and we now have a whole ensemble of those, we will always have to take ensemble averages to come up with a single value. It is also this averaging that assures us we end up below the total uncertainty.

1 - maxprob: If we take the error rate $1 - \max_y p(y|x)$ at x as a measure of uncertainty, and average that over our ensemble, this will be smaller than the error rate of the average $p(y|x)$, because maximum and average do not commute[10].

In the multiclass setting, we could also ask the question if a model with a Bayes probability $\max_y p(y|x)$ of say 0.5 is any less uncertain when we have 100 classes then when we have 2 classes. The obvious answer seems 'yes because it is a mere random guess in the last case, while it is one class that has as much probability as all other 99 combined in the first'[11]. That could lead us to scale this quantity by its maximal value $1 - 1/K$ and use

$$A(x) \equiv \frac{1}{N_{sim}} \sum_k \frac{K}{K-1} (1 - \max_y p(y|x))$$

as definition for aleatoric uncertainty in the maps (and in section 3.4.3). The advantage is that it has a clear interpretation, ranging from 0 for pure random to 1 for totally sure. Disadvantages are that it doesn't distinguish how the remaining probability is divided over the other classes, and that it does not readily compare to epistemic uncertainty.

Expected Variance or Expected Entropy A quantity that takes into account all components of the class probability vector, is the discrete entropy. We could then consider

$$EH(x) = \overline{H[\vec{p}(x)]} := \frac{1}{N_{sim}} \sum_{j=1}^{N_{sim}} H[\vec{p}_j(x)]$$

the ensemble average of discrete entropies of the $\vec{p}_j(x)$ in our ensemble[12].

Note how for both max probability and entropy, the ensemble averaging apparently done only to get a single value is in fact crucial to guarantee that aleatoric uncertainty will not be larger than total uncertainty, and that epistemic uncertainty remains positive (cfr discussion preceding fig 2.2).

Epistemic Uncertainty For the epistemic uncertainty, it seemed reasonable to use a form of average difference between any two draws in the ensemble. Because the 'distance' between 2 class probabilities is often expressed in terms of their relative entropy (or KL divergence), to capture this spread I initially considered an average of all pairwise relative entropies between any 2 realisations in the ensemble. After a suggestion from my tutor, it turned out much more efficiënt to consider the relative entropy between the members of the ensemble $\vec{p}_k$ and their average $\bar{\vec{p}}$, called the Jenssen Shannon Divergence (JSD). This JSD takes the whole set of N_{sim} vectors at once and calculates a symmetrized version of the Kullback-Leibler divergence, with the huge advantage that the support of the components p_i needn't coincide for all realisations, since $p_{j_i} > 0$ for at least one realisation j already

[10] Taking the average of the maximal components will be larger than the maximal component of the average, so the aleatoric uncertainty will be smaller than the total uncertainty obtained as the error rate of the ensemble averaged prediction.

[11] But in terms of error probability, there is really no difference, so if we want a clear link between uncertainty and accuracy, it is better not to scale.

[12] At first sight a decomposition of total uncertainty into two parts using variances also seems interesting, but keep in mind that y is a one hot encoded vector, and its components are completely correlated (cfr. also remark later on the covariance matrix of the distribution of $\vec{p}$).

implies $\overline{p}_i > 0$ as the components are nonnegative[13].

$$JSD(x) = \frac{1}{N_{sim}} \sum_{k=1}^{N_{sim}} KL(\vec{p}_k(x)||\bar{\vec{p}}(x)) = \frac{1}{N_{sim}} \sum_{k=1}^{N_{sim}} \sum_{i=1}^{K} p_{ki}(x) \log \frac{p_{ki}(x)}{\bar{p}_i(x)}$$
$$= H[\frac{1}{N_{sim}} \sum_{k=1}^{N_{sim}} \vec{p}_k(x)] - \frac{1}{N_{sim}} \sum_{k=1}^{N_{sim}} H(\vec{p}_k(x)) = H(\bar{\vec{p}}) - \overline{H(p)} \tag{E.1}$$

where $\bar{\vec{p}} = \frac{1}{N_{sim}} \sum_{k=1}^{N_{sim}} \vec{p}_k(x)$ is the sample mean of the realisations and $KL(\vec{p}||\vec{q}) = \sum_i p_i \log \frac{q_i}{p_i}$. Note how this intuitive definition can be rewritten as a commutator of entropy and ensemble averaging, which makes it actually coincide perfectly with the one used in the Bayesian setting, (entropy expected distribution minus expected entropy of the distribution.)

Distributional Uncertainty Finally we also wanted an indication for OOD, because the above forms of epistemic uncertainty do not appear to capture this always well (depending on how flexible the DE method is I think).
For that, there are also several possibilities. For a start, in some cases (eg when we use KDE with very small bandwidth), large fractions of the grid where we evaluate densities have density zero up to very large precision. There is always one step where we have to exponentiate the logdensities to get the marginal density, and in those cases, far away from the class centers one can get $p(y|x) = NaN$ because $\sum_i p(x|i) = p(x) = 0$. So that means that in some cases, the OOD part of the grid is already rejected by numerical difficulties.

- One option consisted in plotting only points where this didn't happen, so the corners of the domain are sometimes gone. This is one way of dropping OOD points , but is quite arbitrary since it depends on the number of digits we use in our calculations.
- An alternative would be setting a threshold for the marginal $p(x)$ (eg .1% quantile of the validation points' marginal densities). Similar for $\max_y p(x|y).p(y)$. Still, that isn't perfect either, because densities don't take into account how much points we had, or how flexible the model was (correlation lengths etc), and I would think these will play a role in how far we can safely extrapolate our model. I would think we have more information about the tails when we have more points to learn from, but again this will be so dependent on whether we're working with (non)-parametric models, and the smoothness we assume etc, that it is out of scope here. I finally settled for $\max_i n_{tr}(x,i) := \max_i p(x|i).N_{tr,i} = \max_i p(x,i).N_{tr}$, the expected number of training points per unit volume of the dominant class in that region. If that is low, it means we are really out of domain for **all** classes. I've experimented a bit with other quantities based on the relative variance of density estimates as well, but I think the above one is probably a more convincing choice.

$\vec{p}(x)$ as target

When we explicitly model the distribution of $\vec{p}(x)$, and treat that as our target, for which our ensemble of class probability vectors is a sample of 'observations', our original epistemic

[13] When comparing between all realisations, the relative entropy would be undefined in each point x where one of the realisations $\vec{p}_j(x)$ has a zero component $p_j(x)_i = 0$.

uncertainty on Y will be captured by the total uncertainty in this distribution. Here we will use a Bayesian approach.

Due to lack of prior information, we start for each point in feature space from the most non informative conjugate prior over class probabilities

$$P_0(\vec{p}(x)) \sim \lim_{\alpha \to 0} \mathrm{Dir}(\alpha.\vec{1})$$

Now, remember that the parameter of the posterior Dirichlet distribution is obtained from that of the prior by simply adding up the vector of class counts observed in the dataset [14]. Since each point of feature space had its own Dirichlet distribution, we need to add up the counts that occurred exactly at that point x. It then seems reasonable to **smoothen out our observed training data so we observe at each point x the local *density***

$$n_{tr}(x,i) \equiv N_{tr} p(x,i) \equiv \lim_{\mathrm{d}^D x \to 0} \frac{N_{tr,i} \in [x, x+\mathrm{d}^D x]}{\mathrm{d}^D x}$$

of training *points* of each class. If we use that as pseudocounts, we can simply add up the vector $n_{tr}(x,y)$ of joint training data densities[15] to the prior parameter $\alpha = \vec{0}$ and conclude that $N_{tr}.p(x).\vec{p}(x)$ is actually the parameter vector of a posterior Dirichlet in the point x. That will also make the originally independent distributions at each x now smoothly correlated to that of neighbouring x. That way, we obtain for each realisation $\vec{p}_j(x)$ in our original ensemble a Dirichlet distribution with parameter $\vec{\alpha}_j(x) := N_{tr}.p_j(x).\vec{p}_j(x)$.

Why this detour? Since the $\vec{p}_j(x)$ in the ensemble are normalised probability vectors, $\sum_i (\vec{p}_j(x))_i = 1 \ \forall j, \forall x$, **without this reweighting the ensemble averaged $\vec{\alpha}(x)$ would have had exactly the same concentration parameter[16] at each point x**. That is of course not what we want, since we do not have the same amount of data at each of these x. If we would like to rely mostly on nearby data, it makes sense to attach more belief to predictions in regions where more data was observed[17]. Apart from that, we need counts to update our Dirichlet's parameters, and not class probabilities.

We should also stress that the resulting $\alpha(x)$ is only determined up to a multiplicative constant because we can choose the unit of volume in feature space. The absolute value of the uncertainty thus obtained will be without meaning, but we are mainly interested in how it varies between different x[18].

Now all that remains is to quantify the epistemic uncertainty about p in terms of the parameters of this Dirichlet distribution. We could again look at ensemble averaged conditional uncertainties and the uncertainty of the ensemble averaged distribution but here we will

[14] Just like for the beta distribution of which the Dirichlet is a generalisation to more than two classes, we have $\vec{\alpha}' = \vec{\alpha} + \vec{n}$:

$$P(\vec{p}|\vec{n}) \sim P(\vec{n}|\vec{p}) P_0(\vec{p}|\vec{\alpha}) \sim \prod_i (p_i)^{n_i} p_i^{(\alpha_i - 1)} = Di(\vec{\alpha} + \vec{n})$$

where we have used the Dirichlet density for the prior $Di(\vec{x}|\alpha) = \prod_i x_i^{\alpha_i - 1}$. Here we chose $\vec{\alpha} \to \vec{0}$, so $\vec{\alpha}' \to \vec{n}$

[15] i.e. count densities, not probability densities

[16] $\alpha_0 := \sum_{i=1}^{K} \alpha_i$ which determines the spread of a Dirichlet (when the relative proportions $\tilde{\alpha}_i := \frac{\alpha_i}{\alpha_0}$ are kept fixed)

[17] Of course, we need our density estimate to describe the *training* density well for this to work, but that should be the case without bias. This could help our parametric model be more cautious with its class probability output where it predicts low densities. Additional assumption is that there are indeed only K classes.

[18] Perhaps we could find a way to choose also this overall constant by some calibration, but we have not pursued this any further.

focus immediately on the latter, i.e. the total uncertainty about $\vec{p}$, because all of it still corresponds to epistemic uncertainty on our original target Y[19].

(co-)variance (determinant) We could look at the variance of the Dirichlet distribution with ensemble averaged α, but a complication here is that this is really a **covariance matrix, because this is a distribution over vectors**, and we should then consider **determinants** if we want to extract a single measure of the spread in all classes combined. But, the probability vector being normalised, the K**-th component of** $\vec{p}$ **is completely determined by the first** $K-1$. We therefore tried **excluding this last component** and considering the covariance determinant of the remaining $K-1$ as uncertainty measure. However, even then the resulting measure has the unwanted feature that it becomes quasi zero in all x where at least one class can be excluded (or has constant probability)[20].

Therefore we have eventually considered the differential entropy as an alternative.

Differential Entropy As mentioned above, we eventually consider the differential entropy of the Dirichlet distribution with the ensemble averaged parameter:

$$Epi(x) = \mathcal{H}(Di(\bar{\vec{\alpha}}(x)) = \mathcal{H}(Di[\frac{1}{N_{sim}}\sum_{j=1}^{N_{sim}}\vec{\alpha}_j])$$

with $\bar{\vec{\alpha}}$ the ensembled averaged parameter $\bar{\alpha} = \frac{1}{N_{sim}}\sum_{j=1}^{N_{sim}}\vec{\alpha}_j$.

Results

Because of the vast amount of graphs involved, I will upload them in a folder with the thesis, and include in the appendix only the maps for one small, one intermediate and one large training set size for a selection of a few choices of density estimation methods and for some of the potential uncertainty measures that were discussed previously.

We show the result for small, intermediate and large training set size. We made figures for $\log N = \log 30, \ldots, \log 2000$ in 12 equidistant steps, and $N = 5000$, for a variety of KDE and GMM models. **The ones we show here are** $N_{tr} = 30$**,**250 **and** 5000 **points in the whole training set**. Going much further than a few thousands points becomes slower and less interesting, and in general the transients play out in this region. Of course in 10D a dataset of 5000 points is not large, but in 2D having a few hundreds points per cluster should be more than enough.

First we have the figures for the 'calibration' data set (in the online appendices) to confirm whether these quantities express more or less what we expect, and then in section E.3 those for the actual dataset. We will zoom out because of our particular interest in what

[19] Remember how we had split off the aleatoric uncertainty on Y as the part that works conditional on a particular $\vec{p}_j(x)$, and are focussing here on describing the epistemic uncertainty about Y as total uncertainty about $\vec{p}$.

[20] For example, in a 100 class problem, even if the first 98 classes' probabilities are extremely variable between realisations, it suffices that the 99th probability always has the same value (eg 0) to make the covariance determinant vanish. That seems suboptimal, because we think such a situation should receive a large uncertainty, but a 98th dimensional object can never have nonzero volume in 99dimensions.

happens when we are further away from the training data. We end by discussing the results in section E.4.

E.3 Results Mixture data

First we'll show a picture E.1 of the full dataset, from which subsets of increasing size are drawn many times, and of contours of the 'true' Bayes error, based on the probability distribution from which we have drawn the data.

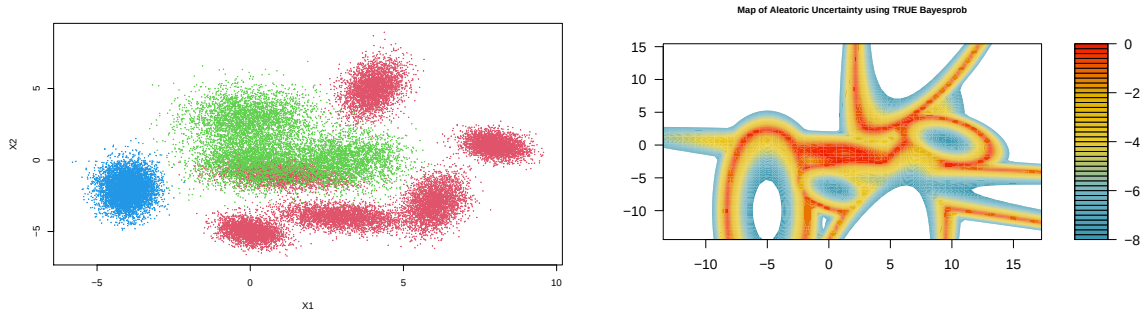


Figure E.1: Left: 'Mother' dataset drawn from a known distribution, with 6000 points in the blue cluster, and 3000 in each of the 3 green and 6 red clusters. Note that one of the red clusters is completely hidden between the green ones. That means the high aleatoric uncertainty regions should not just be a filament along the class boundaries, but feature an additional massive spot in that region of strong overlap. Right: Because we know the true densities, we can draw the true contours of constant $p(y|x)$, and here we show (on a logscale) the contours of constant rescaled bayeserror A. First of all, note the different scales on the axes: we will always consider this larger region so we can see what happens away from the data as well. We stress again the contours shown are not the densities $p(x|y)$, but those of $\max_y p(y|x)$. For example the single cluster class in the left lower corner occurs a bit to the right of the center of the ellipse at that place. Because that class has such a small standarddeviation, it is overrun by the central clusters again at the extreme left of the domain. Therefore the lower left corner would be a region of very large distributional uncertainty, as the dominant class is predicted solely based on the observed distribution elsewhere.

E.3.1 Aleatoric Uncertainty

Using rescaled 1-max probability

We start with aleatoric uncertainty, defined as ensemble-averaged rescaled classification error $A = \frac{1}{N_{sim}} \sum_k \frac{K}{K-1}(1 - \max_y p(y|x))$. Because we have the true class probabilities in this simulation, we can also plot the true Bayes error for comparison (see earlier fig. E.1). Only for GMM and a model with reasonably large fixed bandwidth does this appear to coincide well:

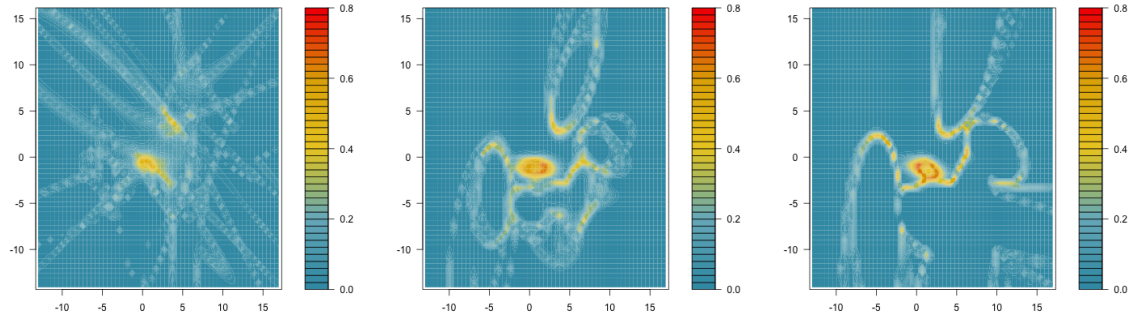


Figure E.2: Aleatoric uncertainty (as rescaled 1-maxprob), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is GMM, with 6-3-1 clusters.

GMM model: VVV631 : fig. E.2

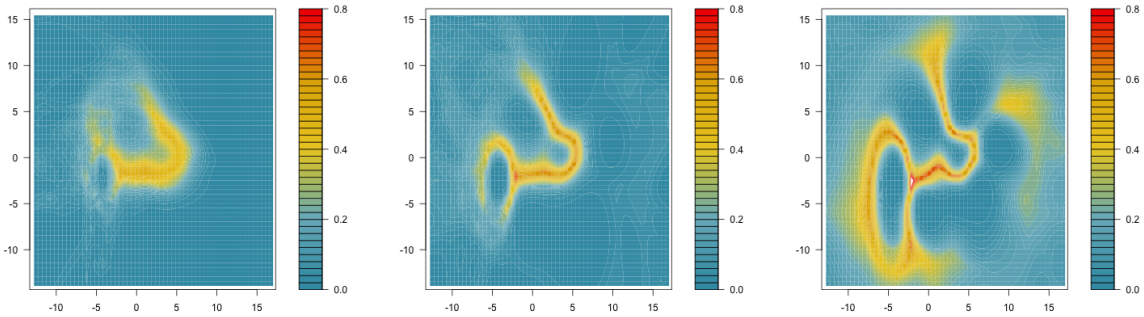


Figure E.3: Aleatoric uncertainty (as rescaled 1-maxprob), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with adaptive bandwidth, $\alpha = 0.5$ (default).

With KDE model adaptive bandwidth, default $\alpha = 0.5$: figure E.3

With KDE model adaptive bandwidth, $\alpha = 0.8$: fig. E.4 Notice how the uncertainty really recognizes the domain here:

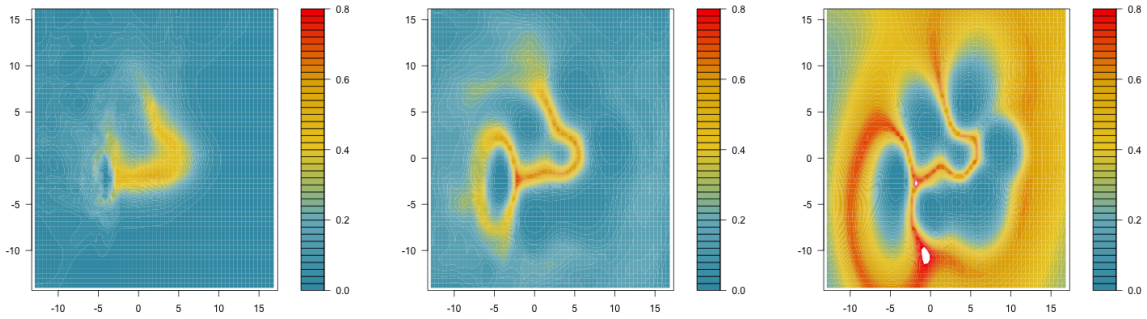


Figure E.4: Aleatoric uncertainty (as rescaled 1-maxprob), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with adaptive bandwidth, $\alpha = 0.8$.

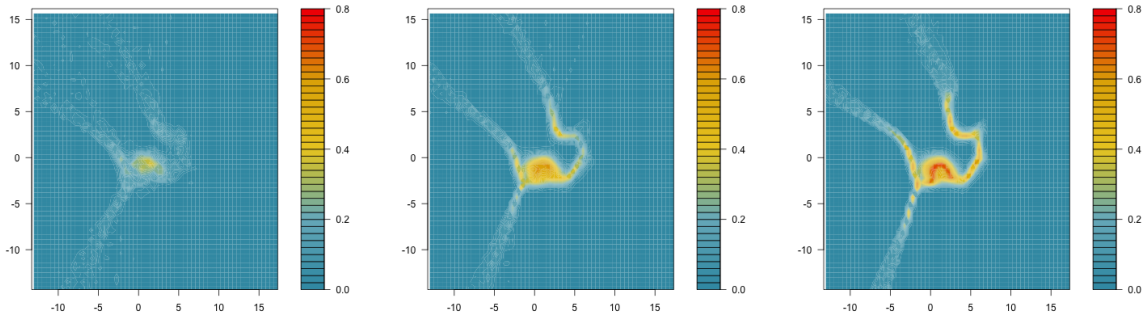


Figure E.5: Aleatoric uncertainty (as rescaled 1-maxprob), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with fixed bandwidth, $h = (0.5, 0.5)$.

Using KDE with fixed large bandwidth (0.5,0.5) : fig. E.5

KDE with fixed very small bandwidth h=(0.1,0.1) : fig. E.6 Even though the bandwidth doesnt seem extremely small, it is obviously too small, because this is the only setting where we get nonsense for figures. Also, it is the setting where large part of the grid (OOD) simply returns numerical errors, due to all densities being 0 up to very high precision.

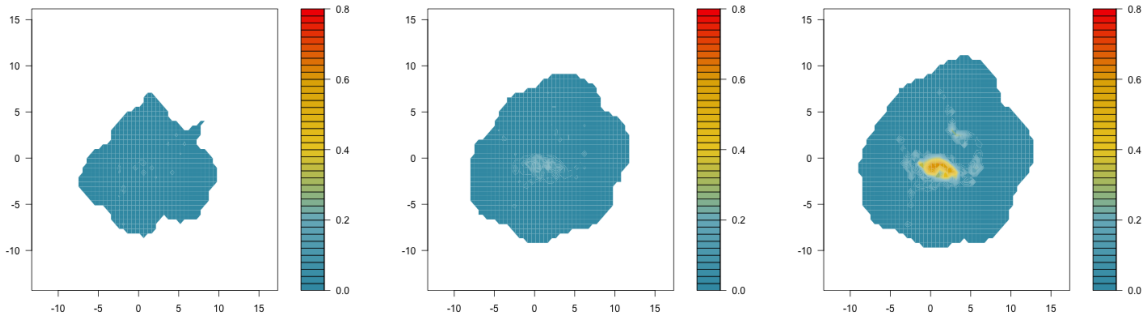


Figure E.6: Aleatoric uncertainty (as rescaled 1-maxprob), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with fixed bandwidth, $h = (0.1, 0.1)$.

Using ensemble averaged entropy of the class probabilities

And now the same for the aleatoric, defined as ensemble-averaged discrete entropy of the categorical distribution in each point.

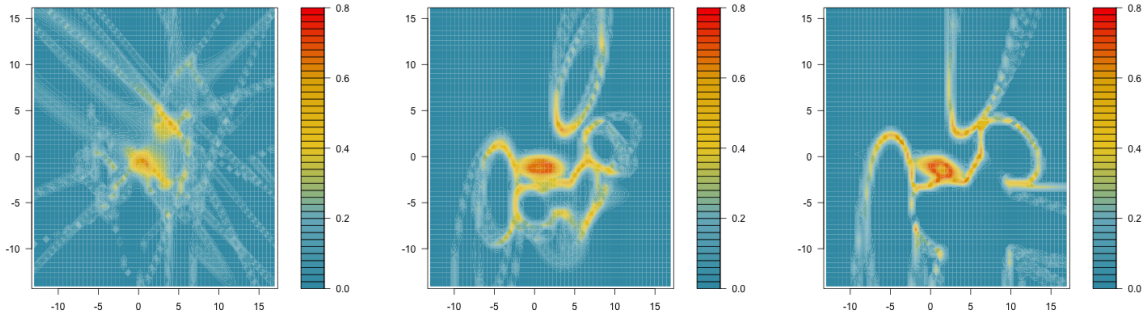


Figure E.7: Aleatoric uncertainty (as ensemble averaged entropy), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is GMM, VVV with 6-3-1 clusters.

With QDA model (EDDA VVV1) : fig E.7

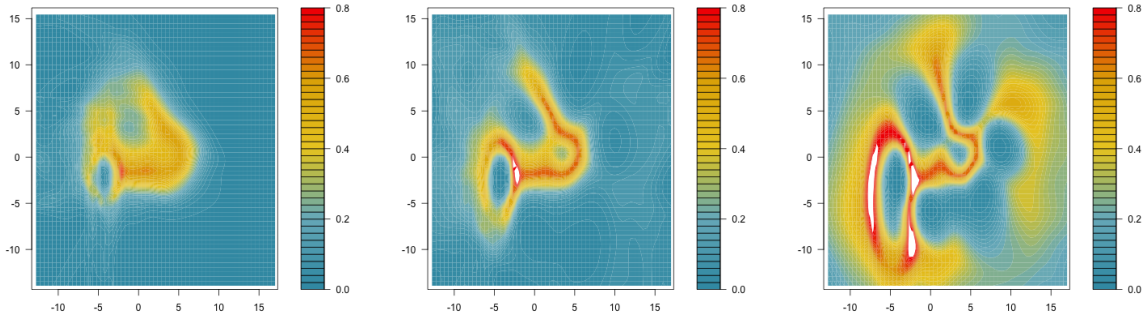


Figure E.8: Aleatoric uncertainty (as as ensemble averaged entropy), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with adaptive bandwidth, $\alpha = 0.5$ (default).

With KDE model adaptive bandwidth, default $\alpha = 0.5$: fig E.8

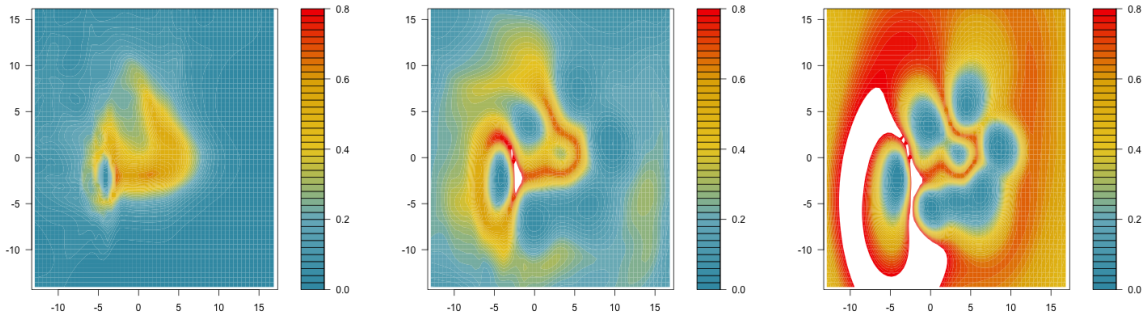


Figure E.9: Aleatoric uncertainty (as as ensemble averaged entropy), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with adaptive bandwidth, $\alpha = 0.8$. Note how the white region is slightly of scale, but I wanted to keep same scale as in the other two graphs to facilitate comparison.

With KDE model adaptive bandwidth, $\alpha = 0.8$: fig. E.9

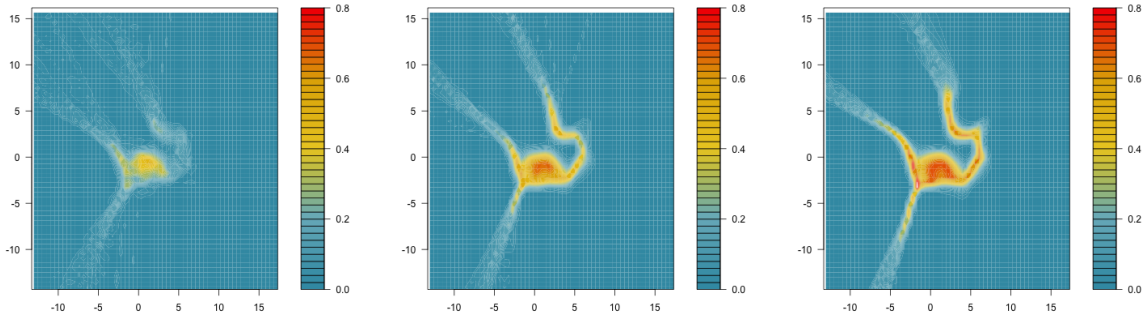


Figure E.10: Aleatoric uncertainty (as as ensemble averaged entropy), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with fixed bandwidth, $h = (0.5, 0.5)$.

Using KDE with fixed large bandwidth (0.5,0.5) : fig. E.10

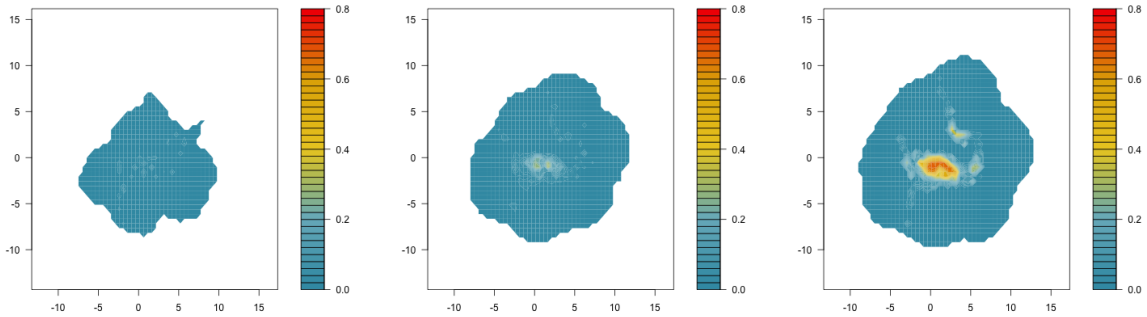


Figure E.11: Aleatoric uncertainty (as as ensemble averaged entropy), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with fixed bandwidth, $h = (0.1, 0.1)$.

KDE with fixed very small bandwidth h=(0.1,0.1) : fig. E.11

E.3.2 Epistemic Uncertainty

This quantity is most sensitive to the model of density estimation (eg KDE with small bandwidth vs GMM etc), which seems quite natural given the large differences in model variance between them.

Epistemic Uncertainty Using JSD

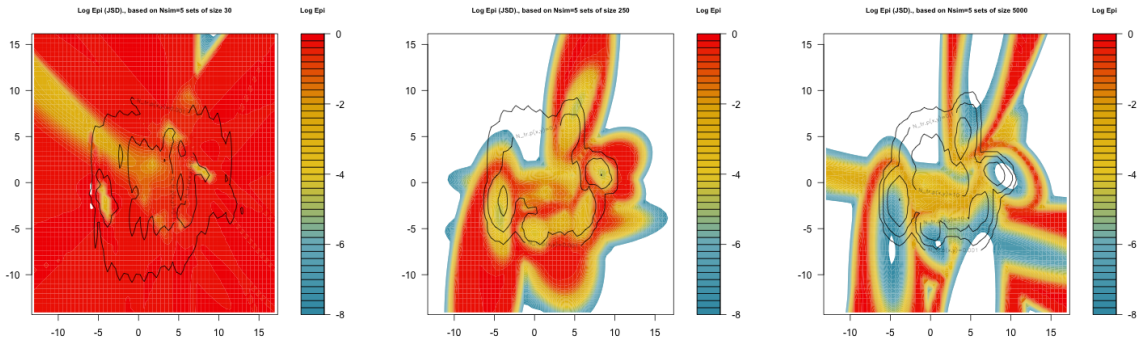


Figure E.12: Log Epistemic uncertainty (based on JSD), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is GMM, VVV with 6-3-1components. Here the white zones with blue edges go below scale. The contours show are those of $\max_y N_{tr}.p(x, y)$, i.e. the count density of the dominant class (for $10^{-3}, 10^{-1}, 10$), in between two contours this increases by a factor 100.

QDA model : fig E.12

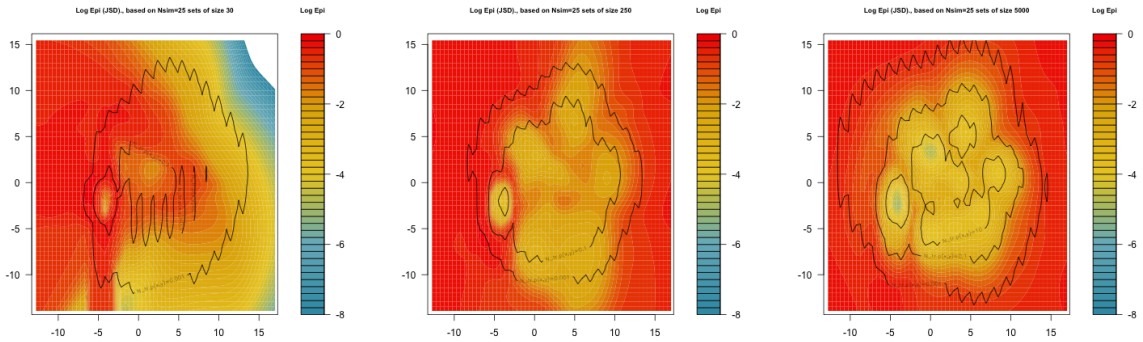


Figure E.13: Log Epistemic uncertainty (based on JSD), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, adaptive bandwidth, gaussian kernel, and $\alpha = 0.5$ (the default).

KDE adaptive bandwidth, default ($\alpha = 0.5$) : E.13

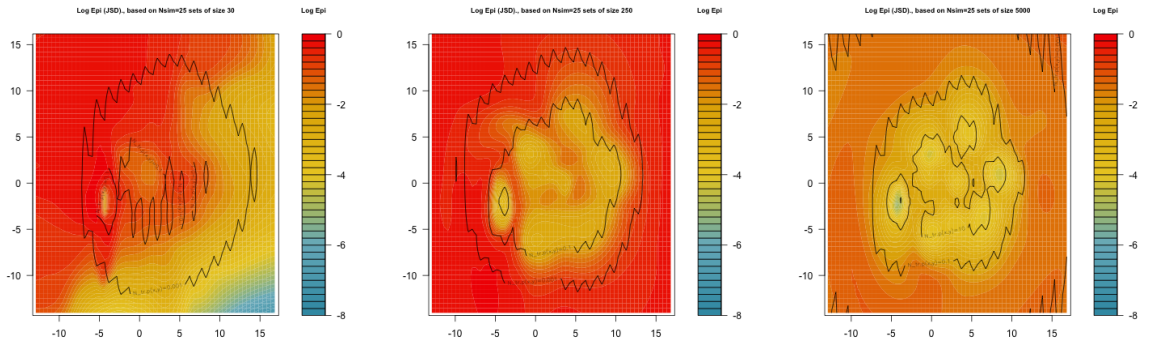


Figure E.14: Log Epistemic uncertainty (based on JSD), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, adaptive bandwidth, gaussian kernel, and $\alpha = 0.8$.

KDE adaptive, $\alpha = 0.8$: E.14

KDE fixed large bandwidth, h=(0.5,0.5) : E.15 Here we go below scale on the blue edge. (I kept the scales equal in different figures for comparison).

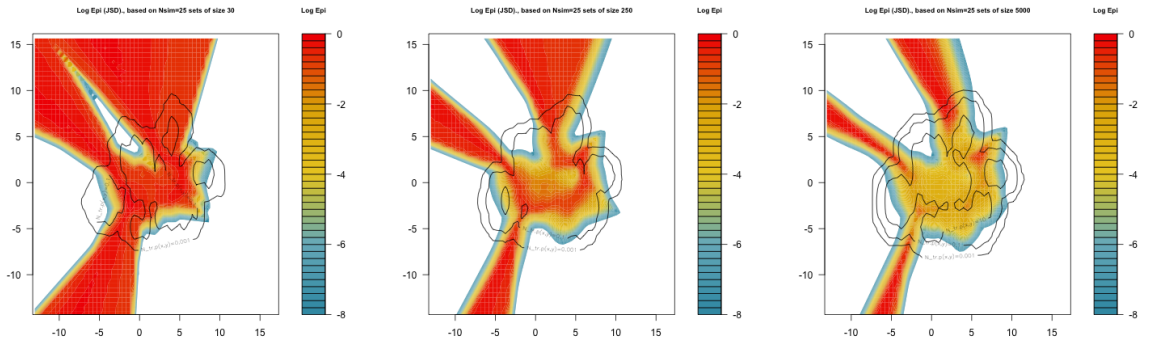


Figure E.15: Log Epistemic uncertainty (based on JSD), from $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with fixed bandwidth, $h = (0.5, 0.5)$.

KDE with fixed very small bandwidth h=(0.1,0.1) : E.16 Remember here large part of the grid (OOD) simply returns numerical errors, due to all densities being 0 up to very high precision. Inside the domain, on the blue edge we go below scale.

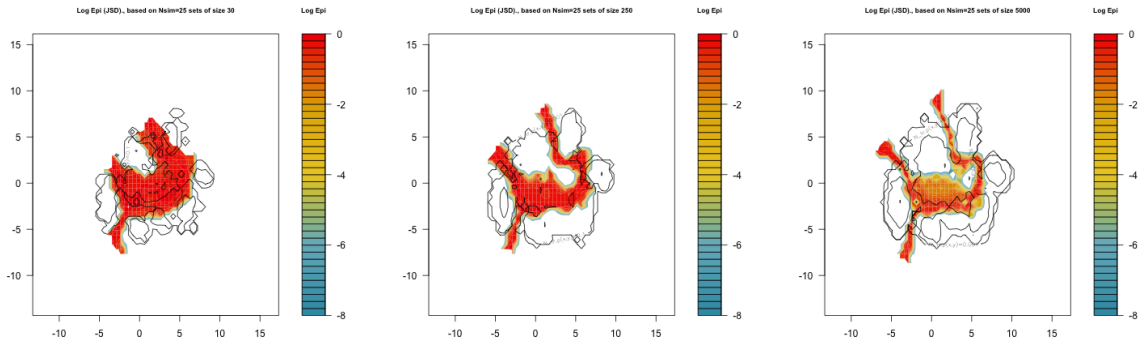


Figure E.16: Log Epistemic uncertainty (based on JSD), from on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, with fixed bandwidth, $h = (0.1, 0.1)$.

Epistemic uncertainty using differential entropy of the Dirichlet for p

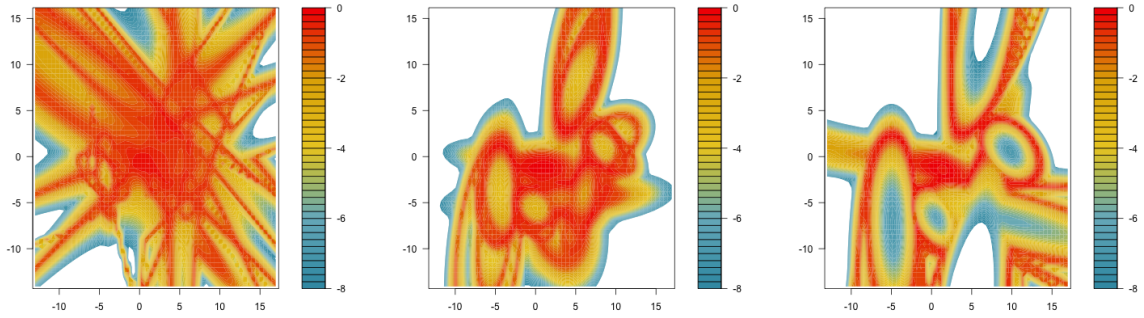


Figure E.17: Log Epistemic Uncertainty (based on differential entropy of the Dirichlet with ensemble averaged parameters, based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is GMM, with 6-3-1 complelety unrestricted clusters.

GMM model: VVV631 : E.17

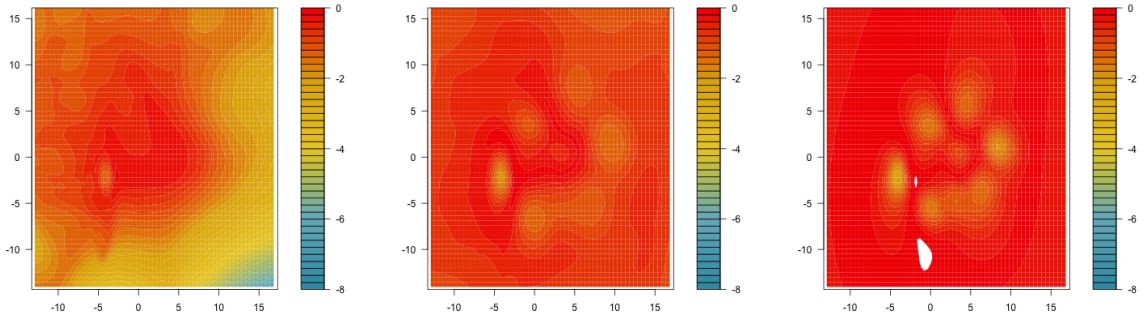


Figure E.18: Log Epistemic Uncertainty (based on diff entropy of the Dirichlet of the Dirichlet with ensemble averaged parameters, based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, adaptive bandwidth, gaussian kernel, and $\alpha = 0.8$.

KDE, $\alpha = 0.8$: E.18

Epistemic uncertainty based on Covariance determinant of the expected Dirichlet distribution

Note that here we are now considering total uncertainty about p as a target. I think that should then match the epistemic uncertainty about Y as a target.

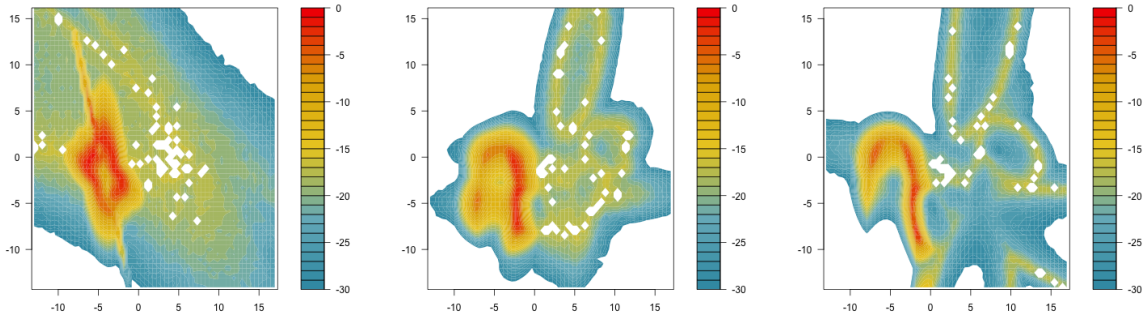


Figure E.19: Log Epistemic Uncertainty (based on Dirichlet), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 30.000 points that plays the role of the actual distribution. Model used is GMM, one cluster per class, VVV, that is QDA simply.

For QDA model : figure E.19

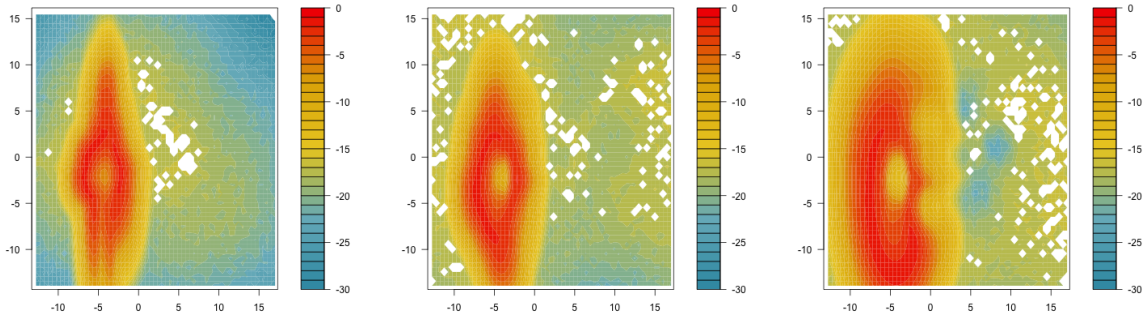


Figure E.20: Log Epistemic Uncertainty (based on Dirichlet), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, adaptive bandwidth, gaussian kernel, and $\alpha = 0.5$ (the default).

For KDE adaptive (default $\alpha = 0.5$) : figure E.20

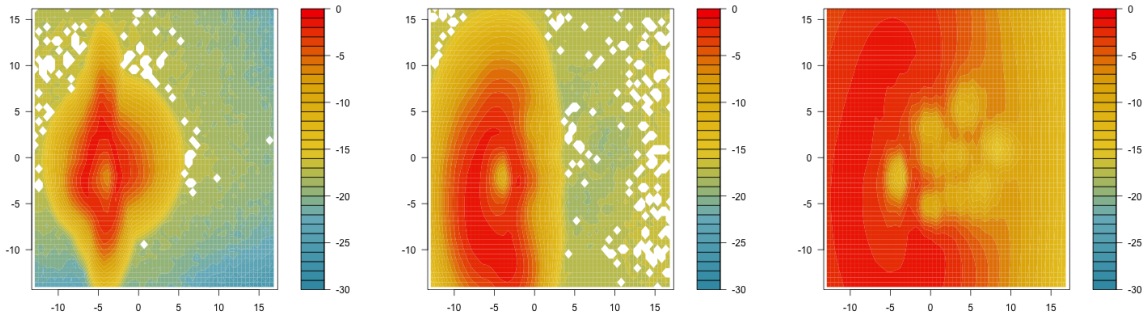


Figure E.21: Log Epistemic Uncertainty (based on variance of the expected Dirichlet), based on $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, adaptive bandwidth, gaussian kernel, and $\alpha = 0.8$.

KDE adaptive, $\alpha = 0.8$: E.21

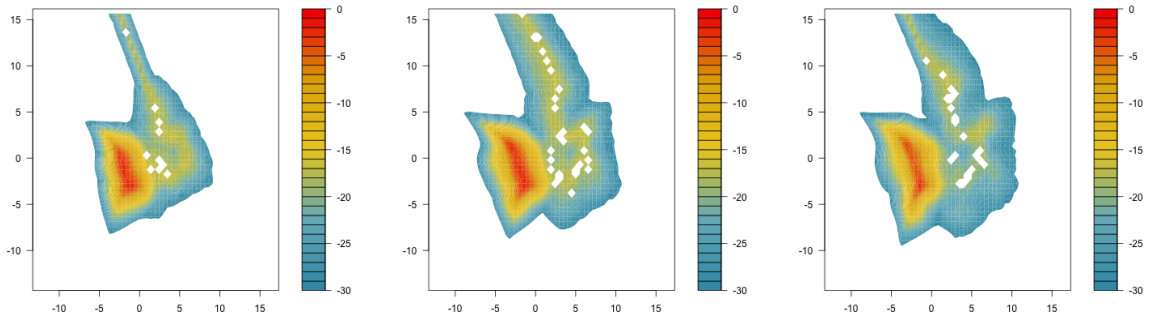


Figure E.22: Log Epistemic Uncertainty (based on variance of the expected Dirichlet), from $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, fixed large bandwidth $h = (1, 1)$, gaussian kernel

KDE fixed bandwidth, large $h = (1, 1)$: figure E.22

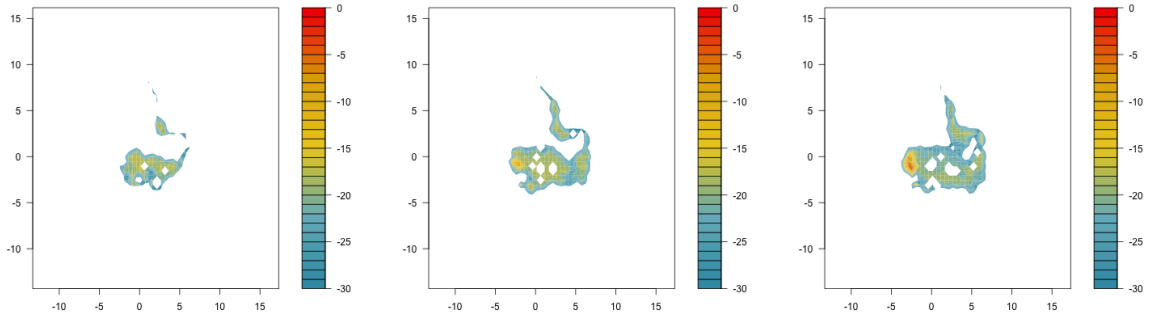


Figure E.23: Log Epistemic Uncertainty (based on variance of the expected Dirichlet), from $N_{sim} = 25$ draws of a dataset of given size N_{tr} from a very large set of 33.000 points that plays the role of the actual distribution. Model used is KDE, fixed large bandwidth $h = (0.1, 0.1)$, gaussian kernel

Using KDE: fixed bandwidth (small) : figure E.23

E.4 Discussion

E.4.1 Simpler (LDA-like) dataset

What we can see clearly (in the online appendices for the LDA folders in maps/LDA/...) already for the simpler dataset, is how estimates are more variable for KDE then for GMM (but less so for the fixed bandwidth version). As we expect, the aleatoric uncertainty is quite similar (at least for the LDA-like data), but it becomes a bit hazy if the dataset size becomes small. After all it is also estimated from an average over the ensembles. The

epistemic uncertainty is more dependent on the DE method, which should not surprise us that much either.

We also get the impression that the adaptive KDE with high α succeeds best in learning the domain. That is also what we'll find for the mixture dataset and seems reasonable as it is also the most variable model, and its larger variance in OOD regions is because we have made less assumptions (which bring along that distributional uncertainty). But that means we face the choice between **a model with higher model variance that does a better job being aware of its uncertainty OOD, or a model with lower model variance that can have large distributional uncertainty without even being aware of it?**[21].

The GMM (=QDA for the simpler dataset) models on the other hand are very confident everywhere except for regions of large aleatoric uncertainty, although they properly appear to recognize the large epistemic uncertainty when there isn't enough data to determine the parameters of each cluster. Once we have enough data to learn the covariance of each cluster, they are very confident away from the domain.

It is also clear that the fixed small and large bandwidth that we tried there as well (which were completely arbitrary, just one very small and one very large) give the worse results, but one should take into account that using the optimal choice for the fixed bandwidth selected by *kepdf* produces much better results (cfr online appendix).

E.4.2 Actual (mixture) dataset

Aleatoric

Only for the aleatoric uncertainty do we have the option to compare to the result for the true distribution, and for the other measures it appears hard to know what we should find. We know the epistemic component should depend on the D.E. used, as that co-determines our uncertainty (eg. for more flexible models, model variance will increase and the chances of having wrong distributional assumptions OOD will decrease). But it was surprising to see that also for aleatoric uncertainty results are so different between different D.E., because this was much less the case for the simple 'LDA-like' dataset.

Presumably, the larger model variance component for some D.E. methods causes more variation in each ensemble, thereby causing the ensemble averaged conditional uncertainty to underestimate the total uncertainty more (cfr figure 2.2 and remarks there). When we use an ensemble of many small training sets, the average probability $\vec{p}$ draws from all those sets, and will approach its correct population value, and the total uncertainty its constant limiting value[22]. It then seems as if, by insisting on additivity even when one of both quan-

[21] The answer probably depends on the context in which we are working: When there exist possibilities to intervene manually (eg collect extra data, manual decision,..) for regions of large uncertainty, we might prefer to know when we are very unsure because we can still do something about it. But in situations where we can not do anything about the uncertainty, the model predicting well in-domain is likely to be even more important than knowing how well it generalizes beyond the domain. It would then seem preferable to reduce variance using distributional assumptions, bearing into mind that any OOD extrapolation ignores the distributional uncertainty.

[22] provided the expectation over all training sets gives us the right result, $E(\theta) = \theta_0$, but I think that should be the case when training sets are sampled from the correct likelihood (or approximately so when they are bootstrapped from a large enough 'mother sample'). For the Bayesian scenario where samples are drawn from a single model posterior conditional on one small dataset on the other hand...

tities becomes ambiguous as long as the other is nonzero, a larger epistemic uncertainty can imply a lower aleatoric uncertainty[23], and the two are no longer independent.

While the aleatoric uncertainty depends strongly on the D.E. method chosen (and thus on epistemic uncertainty), we have a very good agreement over all D.E. methods between aleatoric uncertainty based on max probability and that based on ensemble averaged entropy. Compare figures E.3 with E.8 for example.

Epistemic

Because the definition in terms of a commutator of expectation and entropy arose in a Bayesian context, where one works conditionally on a single dataset, there were some doubts as to whether it was really justified to use this in this different (sampling) setting. Even if bootstrap is considered 'poor mans Bayesian' in the Elements of Statistical Learning [32] (which dedicates section 8.4 to this connection[24])[25], it appears strange to apply those formulas in a context where the dataset is not fixed but being sampled. It was also not clear what we should expect of our epistemic uncertainty, apart from being positive and decreasing in expectation everywhere. Those properties were guaranteed here by the definition, but it seems reassuring that an attempt at a more intuitive definition, in terms of average KL distance between realisations (JSD), turns out to be entirely equivalent to the criterion used there, and we therefore think that we can use this JSD criterion to measure the spread of a set of class probabilities regardless of the setting in which they arose.

The advantage of this quantity (as to compared for example that based on a Dirichlet distribution), is that it is really the complement of one our of criteria for aleatoric uncertainty (the ensemble averaged conditional entropy), so that we can compare them to each other and as a fraction of total uncertainty. That is, not only relative differences by location or training set size, but also its absolute value is meaningfull, which was not always the case for other measures.

[23] Note that the definition used for epistemic uncertainty here, as the mutual information (total-alea), turned out completely equivalent to an independent 'distance' measure (JSD) between members of an ensemble of predicted class probabilities. When we had defined aleatoric uncertainty in terms of noise however, this interdependence of epistemic and aleatoric would probably not have arisen. But such a definition is only useful when we have the true distribution of $P(y|x)$ and if that were the case, the epistemic uncertainty has already vanished.

[24] as follows: " *Hence we might think of the bootstrap distribution as a "poor man's" Bayes posterior. By perturbing the data, the bootstrap approximates the Bayesian effect of perturbing the parameters, and is typically much simpler to carry out.* " For a slightly more detailed discussion and a link between the two in terms of the Bayesian bootstrap, see http://www.sumsar.net/blog/2015/04/the-non-parametric-bootstrap-as-a-bayesian-model/ and references therein, e.g. [33].

[25] I found this correspondence less evident because the expected posterior probability of θ when estimated from an unknown dataset generated by θ_0 in the Bayesian approach appears to be quite different: the unconditional probability assigned to θ (averaging out all possible data sets we might have conditioned on in a Bayesian analysis), given that the true parameter is θ_0 should be given by:

$$p(\theta|\theta_0) = \int p(\theta|\mathcal{D}|)p(\mathcal{D}|\theta_0)\mathrm{d}\mathcal{D} \sim \int p(\mathcal{D}|\theta)p(\mathcal{D}|\theta_0)\mathrm{d}\mathcal{D} = \int L_{\mathcal{D}}(\theta))L_{\mathcal{D}}(\theta_0)\mathrm{d}\mathcal{D}$$

where chose a flat prior so the model posterior and the likelihood are proportional. This says that **those parameter values that share a large likelihood with θ_0 on many datasets, will be most likely**. **On the contrary, when we follow our sampling approach, we will find that $p(\theta|\theta_0)$ only has support for parameter values θ that are actually the exact MLE of a dataset that could reasonably be generated by θ_0**. That makes me think that there must at least be some differences, and the Bayesian approach should result in a smoother and broader range of a posteriori likely θ's because it also allows values of θ that are not the MLE of any dataset that could be drawn from the likelihood at θ_0,

The question remains whether the JSD really represents model variance or also distributional uncertainty. The fact that it remains low for less flexible models suggests it contains only model variance, but it should be noted that we can convert distributional uncertainty into model variance by using a more flexible family of distributions. Distribution-free models would have much larger model variances at the edge of the domain, and this would show up in the JSD. They have also greatly reduced the risk of making wrong assumptions beyond the domain, so they appear to have incorporated the uncertainty that used to be distributional into their model variance. Effectively, for the more flexible adaptive KDE models, we see a large uncertainty in all directions away from the domain, which gives the impression that we needed to convert our uncertainty into a model variance (variance in predicted distribution given certain distributional assumptions) in order to pick it up using the JSD. This also appears to be confirmed by the fact that a less flexible KDE model, with fixed, large bandwidth, maintains much lower JSD beyond the domain in regions where it is confident about its predictions.

We should also keep in mind that we are not training using cross entropy to observed labels here, but merely using the variation in predictions that follow from our density estimation. Therefore, if we make strong distributional assumptions, the predicted densities will not be very variable where one class dominates, and the JSD is low. That means it ignores the fact that we don't know if our assumptions still hold in that region.

Because it appears that the JSD only takes into account the variability of our predictions, and not the validity of the assumptions we made, we will underestimate the uncertainty for models with strong distributional assumptions. Therefore we wanted to see if we can resolve this issue of overconfident predictions far away from the data by modelling the distribution of $p(y|x)$ explicitly and imposing a flat prior on the predicted class probabilities. That way the model should predict all classes to be equally likely unless there is really enough evidence to the contrary.

Dirichlet based uncertainties

covariance determinant We first studied the covariance determininant, but even after dropping one completely collinear component, that quantity remains very small unless the probability of **all** classes varies substantially between realisations, which seems to make it less interesting as a measure of uncertainty.
From a practical point of view, because the third class (single cluster) has very low probability in all realisations on most of the grid (due to its small variance), this quantity becomes very small in those grid points, and calculating determinants with extra precision, for many realizations and over a whole grid simply becomes too slow[26].

When we look at those maps, we see indeed that for the GMM's this quantity only becomes high around the single cluster class, in the only region where the probability of the third (single cluster) class changes fast. On the boundaries between two classes it becomes slightly larger than minimal, but for all the rest of the domain, and especially beyond the domain in regions where one class is dominant, it is very small.

[26] Eventually we calculated these with standard precision, but this gives some small artefacts on the graph (figures E.19 ,E.18), probably due to working with very small numbers that cause this strictly positive quantity to become negative.

For adaptive KDE models on the other hand we again appear to learn the domain quite well and have very large uncertainty everywhere else, but that was already the case for our JSD measure. Therefore, this quantity is not the tool for distributional uncertainty one might have hoped it to be.

Differential Entropy Here we used the differential entropy of the Dirichlet distribution with ensemble averaged parameter. It is again somewhat disappointing to see that also this quantity increases in all directions beyond the domain only for the flexible KDE models, while it remains small in certain directions far away from the domain for GMM. Again it appears this quantity will not protect us from the overconfidence of models with strong distributional assumptions beyond the domain. In [2] they used this differential entropy to detect distributional uncertainty, be it in a very different setting where they explicitly trained a network to result in a flat posterior for a series of 'non-training' datasets. Here we only want to attenuate the predictions of our model when those predictions were not backed up by sufficiently dense training set at that point, but we see the uncertainties remain low for the less flexible models in regions beyond the domain where only one class is considered possible.
It's not completely clear what could be the reason for this. Perhaps an increase of the concentration parameter $\alpha_0 = \sum_i \alpha_i$ for Dirichlet distributions with parameter vectors $\vec{\alpha}$ where a single component is many orders of magnitude larger than the others does not cause a comparable increase in entropy as it does for a more balanced parameter vector[27]. While the JSD seemed to do quite well in capturing the model variance component, this additional attempt of incorporating also the distributional uncertainty for models with strong distributional assumptions by means of this Dirichlet posterior therefore seems less of a success.

[27] the analytical expression for the differential entropy of a Dirichlet is not so transparent: It is clear that the entropy of the Dirichlet with $\vec{\alpha} \sim \frac{\alpha_0}{K}\vec{1}$ increases with decreasing α_0, but I'm not sure whether this is also the case when $\vec{\alpha}$ has many near zero components, after all such a distribution has a much more limited spread as it constrains $\vec{p}$ to an edge or corner.