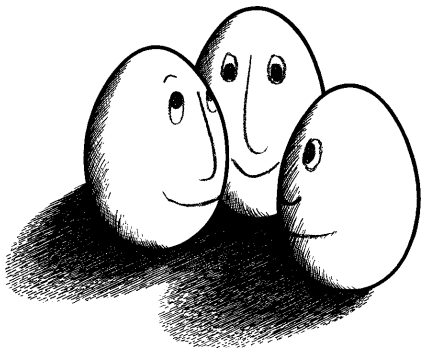
technische universität
dortmund

Master's Thesis

Continuous Deconvolution of Probability Density Functions in Cherenkov Astronomy

Maik Schmidt
December 2019



Supervisors:
Prof. Dr. Katharina Morik
Mirko Bunse (M.Sc.)

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS VIII)
https://www-ai.cs.tu-dortmund.de

Contents

Chapter 1

Introduction

Deconvolution is the non-parametric estimation of the probability density function of a target variable. When the variable cannot be measured directly, it has to be reconstructed from contaminated observations. A popular application of deconvolution is the analysis of γ particles in Cherenkov astronomy. Upon entering the atmosphere, the particles induce air showers. Imaging Air Cherenkov Telescopes (IACTs) measure Cherenkov light emitted by the air showers. To reconstruct the distribution of the particle energy from the measured signal, deconvolution algorithms are used. A precise estimate of the density helps physicists in verifying theoretical models and scientific assumptions.

The *Dortmund Spectrum Estimation Algorithm* (DSEA) [36, 37] formulates the problem of deconvolution as a multinomial classification task where the relation between latent and measured quantities is learned from training data by a classifier. A recently improved version of DSEA, DSEA$^+$ [8], achieves state-of-the-art performance on deconvolution tasks. Since we are interested in the density of a continuous target variable (the particle energy in the IACT domain), it may be more suitable to use regression techniques instead of classifiers.

The present work is based on DSEA$^+$ and extends it by the use of regression methods. These will allow the prediction of real-valued target variables. As the energy of γ particles is continuous and binning introduces discretization errors, direct predictions of the energy is more precise and can lead to an increased accuracy of the deconvolution. To estimate the distribution of the predictions, kernel density estimation (KDE) is used. KDE places a basis function on each observation, then all basis functions are summed up to obtain the estimation. This will allow for the estimation of a continuous distribution. Existing methods that estimate continuous densities revolve around fitting functions to the probabilities assigned to the bins. However, fitting a continuous function to a handful of values conceals the level of detail with which the estimate was fit to the data. The consideration of all predictions for the density estimation leads to a more granular deconvolution and prevents discretization errors.

1.1 Goals and Outline

The goal of the present thesis is to extend DSEA by the use of regression methods. The current specification of the algorithm does not allow their use yet. To overcome this, we explore appropriate regression techniques that meet our requirements. To fully adapt DSEA to regression tasks and estimate continuous densities, the dependency on binning the data is removed. Ways to incorporate a continuous density estimator are explored. Finally, we evaluate the presented methods and compare the results to DSEA as the developed algorithm should at least match the discrete estimations of DSEA in quality. The thesis is structured as follows:

- **Chapter 2: Deconvolution.** This chapter opens with applied deconvolution in Cherenkov astronomy to explain the basics of deconvolution. Existing approaches are described afterwards. These are *Iterative Bayesian Unfolding* (IBU), *Regularized Unfolding* (RUN) and eventually DSEA and DSEA+. At the end of the chapter, modifications are listed that are necessary for DSEA to estimate continuous densities.

- **Chapter 3: Regression Analysis.** We begin with the basics of regression analysis. The specific requirements for a regression method that are necessary for an iterative deconvolution are addressed. These are the possibility to train from weighted data and the output of uncertainties. In the course of the chapter the linear regression, kernel ridge regression, Gaussian processes, random forests and Mondrian forests are presented with special attention to the requirements. The methods are evaluated at the end of the chapter.

- **Chapter 4: Density Estimation.** We briefly discuss the current density estimator, the histogram used in DSEA, before moving on to kernel density estimation. Options to embed the uncertainties returned by the regression method in the kernel density estimation are investigated. The options comprise weighted and variable bandwidth KDE.

- **Chapter 5: Continuous Spectrum Estimation Algorithm.** This chapter presents the developed algorithm. Beforehand a number of changes to DSEA are covered, namely changes to the weight update, the adaptive step size and an optional calibration of the predictions. The calibration is deemed useful in some configurations.

- **Chapter 6: Evaluation.** The algorithm is evaluated on different data sets using various combinations of the aforementioned regression methods and density estimators. Lastly, we explore how the distribution calibration and the variable bandwidth KDE incorporating the uncertainties can improve the results. The deconvolution results of CSEA and DSEA are compared with regard to their quality.

- **Chapter 7: Conclusion.** The last chapter concludes the work with a brief summary of the most important aspects. It also gives an outlook on possible improvements and use cases of the algorithm.

Chapter 2

Deconvolution

2.1 Deconvolution in Cherenkov Astronomy

The *First G-APD Cherenkov Telescope* (FACT) [1] and the *Major Atmospheric Gamma Imaging Cherenkov Telescopes* (MAGIC) [45, 9] are two telescope systems in the Imaging Atmospheric Cherenkov Telescope (IACT) domain for measuring cosmic showers caused by the entrance of γ particles into the atmosphere. By interacting with particles in the atmosphere, they emit Cherenkov light. The Cherenkov light is measured by the telescopes (see figure 2.1).

One goal of analyzing the measured data is to draw conclusions about the γ particles. Deconvolution can help to achieve this. Deconvolution estimates the distribution of a relevant quantity $\boldsymbol{Y}$. In the IACT domain, this is the energy of the γ particles. Since this distribution cannot be measured directly, it must be reconstructed from observable quantities $\boldsymbol{X}$, the measurements of the telescopes:

$$g(x) = \int_Y R(x|y) f(y)\, \mathrm{d}y. \tag{2.1}$$

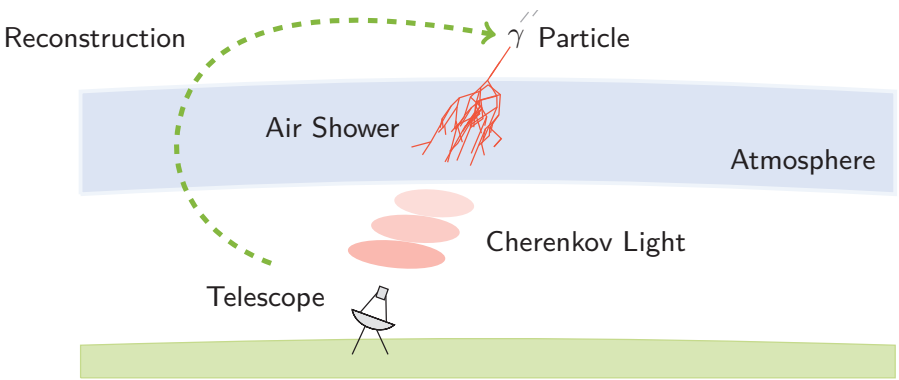


Figure 2.1: Reconstruction of the distribution of γ particles from measurements of the telescope [4, 7].

The relationship between the observations and the target quantity is modeled by a Detector Response Function. $R(x_i, y_i)$ represents the probability for the target value y_i given that the observation is x_i. The interaction between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is too complex to be encoded manually in a function. Therefore, R is estimated using simulated data. The observations are obtained by the convolution of R and the target $\boldsymbol{Y}$. Of interest is the distribution $f(y)$, the distribution of particle energy. To get $f(y)$ the model has to be inverted. Therefore, this procedure is called deconvolution (or unfolding). In the continuous case there is no closed form for the calculation, which is why there are different methods to iteratively adjust f to g and R. Classical deconvolution methods discretize the measurements and then solve the discretized problem. DSEA uses an alternative approach that incorporates a classifier to estimate a discrete density. At the end of the chapter necessary steps are presented to extend DSEA by a regression method. In this way the estimation of a continuous density and thus the solution of the continuous problem (2.1) becomes feasible. For a more detailed overview of the introduced deconvolution methods, please refer to [7].

2.2 Classical Discrete Deconvolution

Classical deconvolution methods such as *Iterative Bayesian Unfolding* (IBU) and *Regularized Unfolding* (RUN) solve a discrete reformulation of the continuous problem (2.1):

$$\boldsymbol{g} = \mathbf{R}\boldsymbol{f}. \tag{2.2}$$

Discrete probability densities $\boldsymbol{f}, \boldsymbol{g}$ and a matrix $\mathbf{R}$ as Detector Response Function are used for this purpose. In the IACT domain, the target quantity (the energy γ) is one-dimensional and can easily be discretized into equidistant intervals. There are several options to cluster the usually multidimensional observations into a finite number of subspaces to get a discretized one-dimensional representation. Decision trees showed promising results achieving such a clustering [5]. By interpreting the leaves of the tree as clusters the examples are assigned a cluster depending on the leaf they end up. Decision trees are covered in chapter 3.5.

The Detector Response Matrix indicates the relationship between the observations and the target quantity which is estimated using training data. Each component of the matrix is assigned the relative frequency $N_{j,i}$ of the respective observation x_j at a fixed target value y_i:

$$R_{j,i} = \frac{N_{j,i}}{\sum_{j'} N_{j',i}}.$$

This matrix is better conditioned when the computed clusters correlate with the target quantity. This explains the advantage of the supervised clustering with decision trees over classical clustering algorithms that disregard the target variable.

To get the optimal solution for $\boldsymbol{f}$ in (2.2) one could simply invert $\mathbf{R}$ analogue to the procedure of fitting linear models (see chapter 3.2). This leads to the solution with minimal sum of squares of the residuals:

$$\hat{\boldsymbol{f}} = \mathbf{R}^{+}\boldsymbol{g}.$$

$\mathbf{R}^{+}$ denotes the pseudo inverse, which exists even in the case that $\mathbf{R}$ is singular. Unfortunately, this solution has a very high variance. Also, the compliance with the Kolmogorov axioms is not guaranteed, so that the calculated vector does not have to specify a valid probability distribution. In addition, this procedure does not prevent a strong dependence on the distribution of the training data, as these determine the Detector Response Matrix $\mathbf{R}$. Since this solution is unfavorable, a solution is approached iteratively by reducing the influence of the distribution of training data in each iteration. This will introduce some bias in exchange for a lower variance. *Iterative Bayesian Unfolding* and *Regularized Unfolding* are two popular methods that accomplish this. They are described in the following.

2.2.1 Iterative Bayesian Unfolding

Iterative Bayesian Unfolding [10, 11] is an algorithm for the iterative deconvolution of discrete probability densities. Bayes' theorem is applied repeatedly so that the solution is updated in each iteration. Starting from a prior distribution (e.g. uniform distribution) the updates simply follow Bayes' theorem:

$$\mathbb{P}(y_i|x_j) = \frac{\mathbb{P}(x_j|y_i)\mathbb{P}(y_i)}{\mathbb{P}(x_j)} = \frac{\mathbb{P}(x_j|y_i)\mathbb{P}(y_i)}{\sum_{i'}\mathbb{P}(x_j|y_{i'})\mathbb{P}(y_{i'})}.$$

The conditional probabilities of $\boldsymbol{X}$ given $\boldsymbol{Y}$ are taken from the Detector Response Matrix and the previous estimate is inserted for the probabilities of the target variable:

$$\hat{\boldsymbol{f}}_i^{(k)} = \sum_j \hat{\mathbb{P}}(y_i|x_j)\hat{\mathbb{P}}(x_j) = \sum_j \frac{R_{j,i}\hat{f}_i^{(k-1)}}{\sum_{i'} R_{j,i'}\hat{f}_{i'}^{(k-1)}} g_j.$$

Recall that the observations were clustered and $\boldsymbol{g}$ is the resulting vector containing the probabilities of each cluster.

This process is repeated until the χ^2 distance between two solutions is small enough. We then assume convergence.

2.2.2 Regularized Unfolding

Regularized Unfolding [3] is another iterative approach to deconvolution. In each iteration a new solution is estimated by the maximum likelihood method. This method selects the distribution $\boldsymbol{f}$ of the target quantity that has the highest probability for the given observation $\boldsymbol{g}$. To substitute a density function for the likelihood we need to assume a distribution. After clustering the input space, a Poisson distribution of the absolute

frequencies $N\boldsymbol{g}$ with expectations $\lambda_i = N\boldsymbol{R}_{i\bullet}\boldsymbol{f}$ is a reasonable assumption. This leads to the equivalent problem of minimizing the loss function

$$\begin{aligned}\arg\min_f L(\boldsymbol{f}|\boldsymbol{g}) &= \arg\min_f -\ln \mathbb{P}(\boldsymbol{g}|\boldsymbol{f}) \\ &= \arg\min_f -\sum_i \ln\exp(-N\boldsymbol{R}_{i\bullet}\boldsymbol{f})\frac{(N\boldsymbol{R}_{i\bullet}\boldsymbol{f})^{Ng_i}}{(Ng_i)!} \\ &= \arg\min_f \underbrace{\sum_i \left(N\boldsymbol{R}_{i\bullet}\boldsymbol{f} - Ng_i \ln\left(N\boldsymbol{R}_{i\bullet}\boldsymbol{f}\right)\right)}_{=:\ell(\boldsymbol{f})}.\end{aligned}$$

To prevent overfitting, caused by correlation between adjacent bins, a regularization term is added to the loss function:

$$\ell_r(\boldsymbol{f}) = \ell(\boldsymbol{f}) + \frac{\tau}{2}\sum_{i=2}^{I-1}(-f_{i-1} + 2f_i - f_{i+1})^2.$$

Using the square of the finite differences as approximation for the second derivative the non-smoothness of $\boldsymbol{f}$ is clearly penalized, favoring smoother solutions. The regularization parameter τ is not given by the user. Instead, he specifies a number of degrees of freedom for the solution, which is internally translated to a corresponding value of τ by the algorithm. This value might change in every iteration complying with the given degrees of freedom which gives an advantage over using a constant value defined by the user.

Using a Newton optimizer [30] a new solution for the target density is computed in each iteration. This procedure is repeated until convergence. Convergence is assumed, when the likelihoods of two successive solutions do not differ enough.

In the original RUN publication [2] the estimated density was interpolated using B-Splines. B-splines are smooth functions with optimal approximation properties [42] when connecting a set of pairs (y_i, p_i). A cubic B-splines of order k fits polynomials of degree $k-1$ for every interval $[y_i, y_{i+1}]$. At the interval boundaries the polynomials of the adjacent intervals match in functions values and the first $k-2$ derivatives. As a result, the whole function is $k-2$ times continuously differentiable. While the estimation is technically continuous it only depends on the function values at the interval boundaries. In between, the function is smoothly fit without knowledge about the underlying distribution in these areas. In contract, histograms do not misguide the user by presenting a smoother solution than the actual fit. The B-splines approach was dropped in later publications in favor of histograms [3].

2.3 Deconvolution as a Classification Task

2.3.1 DSEA

The deconvolution problem can be transformed into a multinomial classification problem. This allows many tools from the field of machine learning to be used. DSEA and DSEA$^+$ use a classifier to reconstruct the desired distribution. An advantage is that during the reconstruction of the distribution the contributions of each example are accessible. This enables the possibility of unfolding in a sliding window by aggregating only the contributions of individual examples, which is advantageous for analyses of IACT data where γ ray sources change their emission over time.

In order to use DSEA [36, 37] with a classifier, the continuous state space of the target variable is partitioned into half-open intervals. These correspond to the classes of the classification, such that the classifier learns to predict the associated subspace of an example. For partitioning, the same intervals can be selected as in the classical methods. It is common to select equidistant intervals so that the resulting histogram is also equidistant. For each observation, the classifier predicts a class that corresponds to an interval of the target variable. After the aggregation of the classifier's predictions, the target density is obtained in a discretized form. Instead of aggregating only the predicted classes, the probabilities of all classes are included. These so-called confidence values represent the conditional probabilities of all classes given an observation:

$$c_M(y_i|x) = \hat{\mathbb{P}}(Y = y_i|X = x).$$

By considering all conditional probabilities, the uncertainty of the respective estimate is also taken into account. Neglecting these uncertainties leads to worse deconvolution results [36]. According to the law of total probability the distribution of the target is estimated by averaging the confidence values:

$$\hat{f}_i = \frac{1}{N} \sum_n c_{\mathcal{M}}(i|\boldsymbol{x}_n).$$

After the training, the output of the classifier depends on the distribution of the training data. This prior distribution generally does not match the unknown distribution of the target quantity. To reduce the influence of the prior, the solution is iteratively improved by adjusting the distribution of the training data to the calculated intermediate solution. For this purpose, the individual examples are re-weighted. Thus, the training distribution approaches the (unknown) distribution of the target variable in each iteration. After each update, the classifier is trained on the re-weighted training data. DSEA allows for the specification of arbitrary prior distributions by weighting the training data ahead of the first iteration.

The termination criterion of a minimum distance between two consecutive estimations is parameterized using data from simulations and is therefore rarely used. Instead, the

number of iterations is predefined. However, this number is also estimated from experiments with simulated data. In [7] it was shown that the estimations can degrade when the algorithm is not stopped after a suitable solution has been found. This problem is solved in DSEA$^+$ by adjusting the re-weighting of the training data.

2.3.2 DSEA$^+$

DSEA$^+$ [8] improves DSEA by two features. It solves an issue with the re-weighting of the training data and introduces an adjustable step size between two estimations.

After the re-weighting step in DSEA, the distribution of the training data is intended to match the current estimation. However, the class sizes are not taken into account, which means that classes with many examples are weighted higher than they should be. DSEA$^+$ solves this problem by correctly dividing the weights by the number of class examples. Thus, the aggregated density at the location of the class matches the desired value:

$$w_n^{(k)+} = \frac{\hat{f}_{i(n)}^{(k-1)}}{f_{i(n)}^t}. \tag{2.3}$$

$\hat{\boldsymbol{f}}^{(k-1)}$ is the estimated distribution of the target from the previous iteration and $\boldsymbol{f}^t$ the distribution of the training data. The corresponding index i of the bin, in which example $\boldsymbol{x}_n$ is located, is indicated here with $i(n)$. Modifying the weight update resolves the divergence problem of DSEA.

In addition, a step size α from one solution to the next is proposed to regulate convergence. Alongside various decay strategies for the step size adjustment, an adaptive step size using the regularized likelihood from RUN (section 2.2.2) has proven to be particularly effective. This strategy chooses the step size that maximizes the regularized likelihood ℓ_r in the search direction $p^{(k)}$:

$$\alpha_{RUN}^{(k)} \leftarrow \arg\min_{\alpha \geq 0} \ell_r \left(\hat{\boldsymbol{f}}^{(k-1)} + \alpha \cdot p^{(k)} \right).$$

The search direction is determined by DSEA by the difference between the current and the previous estimation:

$$p_i^{(k)} = \frac{1}{N} \sum_n c_{\mathcal{M}}(i|\boldsymbol{x}_n) - \hat{f}_i^{(k-1)}.$$

The algorithm stops if the new estimation is too close to the previous estimation. The χ^2 distance suggested for DSEA is used for this purpose. Since adaptive step sizes larger than 1 are also possible, DSEA$^+$ can converge much faster. This allows the stopping criterion to be reached earlier, saving time and resources. DSEA$^+$ is specified in algorithm 1.

Algorithm 1 DSEA$^+$: Improved Dortmund Spectrum Estimation Algorithm

Input:

Observed data set $\mathcal{D}_{\text{obs}} = \{\boldsymbol{x}_n \in \mathcal{X} \,|\, 1 \leq n \leq N\}$

Training data set $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_n, y_n) \in \mathcal{X} \times \mathbb{R} \,|\, 1 \leq n \leq N'\}$

Minimal χ^2 distance $\epsilon > 0$ between iterations (default: 10^{-6})

Prior density $\hat{\boldsymbol{f}}^{(0)}$ (default: $\hat{f}_i^{(0)} = \frac{1}{I} \,\forall 1 \leq i \leq I$)

Output:

Estimated target density $\hat{\boldsymbol{f}} \in \mathbb{R}^I$

$k \leftarrow 0$

repeat

  $k \leftarrow k + 1$

  $\forall 1 \leq n \leq N' : w_n^{(k)+} = \frac{\hat{f}_{i(n)}^{(k-1)}}{f_{i(n)}^t}$

  Infer $\mathcal{M}$ from $\mathcal{D}_{\text{train}}$ weighted by $\boldsymbol{w}^{(k)+}$

  $\forall 1 \leq i \leq I : p_i^{(k)} \leftarrow \frac{1}{N} \sum_n c_{\mathcal{M}}(i|\boldsymbol{x}_n) - \hat{f}_i^{(k-1)}$

  $\alpha_{\text{RUN}}^{(k)} \leftarrow \arg\min_{\alpha \geq 0} \ell_r \left(\hat{\boldsymbol{f}}^{(k-1)} + \alpha \cdot p^{(k)} \right)$

  $\hat{\boldsymbol{f}}^{(k)} \leftarrow \hat{\boldsymbol{f}}^{(k-1)} + \alpha_{\text{RUN}}^{(k)} \cdot p^{(k)}$

until $\chi^2_{\text{Sym}} \left(\hat{\boldsymbol{f}}^{(k)}, \hat{\boldsymbol{f}}^{(k-1)} \right) \leq \epsilon$

return $\hat{\boldsymbol{f}} \leftarrow \hat{\boldsymbol{f}}^{(k)}$

2.4 Deconvolution as a Regression Task

In [8] deconvolution was formulated as a classification problem. Since we are interested in the density of a continuous latent variable (the particle energy in the IACT domain), it may be more suitable to use regression techniques instead of classifiers. Direct prediction of target variables instead of representative bins will allow us to estimate a continuous density. In doing so, the estimated target density depends on the prediction of every observation instead of a fixed number of bins, as in the B-spline approach (see chapter 2.2.2). This leads to more granular estimations and hopefully to an increased accuracy of the deconvolution. Also, binning the data loses information about how far apart different bins are to each other as each bin is considered an orthogonal direction. In reality some bins are nearer and this information might be useful for the model. The current specification of the algorithm does not allow the use of regression methods. To overcome this, we must explore appropriate regression techniques that

- Support weighted training to re-weight the training samples in each iteration
- Return the uncertainty of each prediction as standard deviation or prediction interval

In the following chapter 3 we explore different regression methods that satisfy these conditions. To fully adapt DSEA to regression tasks and estimate continuous densities, larger adjustments are required:

- Remove the dependency on binning the data
- Explore options for the estimation of a continuous target density from the predictions

Density estimators are reviewed in chapter 4.

Chapter 3

Regression Analysis

Classification and regression analysis are two types of supervised learning. The goal of supervised learning is to infer the relation between a set of input variables $\boldsymbol{X_1}, \ldots, \boldsymbol{X_n}$ and the output variable $\boldsymbol{Y}$. The output can also be multidimensional, but is the one-dimensional energy γ in our use case. Usually we have a training set $(\mathbf{X}_{train}, \boldsymbol{y}_{train})$ of measured values and target values and another set $\mathbf{X}_{obs}$ where values for the target variables are missing. The supervised learning model learns about the relation between $\boldsymbol{X}$ and $\boldsymbol{Y}$ using the training set. Then the model is used to make predictions about $\boldsymbol{Y}$ for $\mathbf{X}_{obs}$. Depending on the image of $\boldsymbol{Y}$ we call the method either classification (when we predict quantitative outputs) or regression (when we predict qualitative outputs).

When employing a regression model the focus is usually on the predictions. The model seeks a function for predicting output values given some values as input. The output can be interpreted as expectation of the estimated conditional density:

$$f(x) = \mathbb{E}(\boldsymbol{Y}|\boldsymbol{X} = x). \tag{3.1}$$

But when additional properties of the random variable $\boldsymbol{Y}$ are of interest, regression models can be used as conditional density estimators. This is the case in DSEA where the uncertainty, a quantification of the accuracy of the predictions, is used to improve the estimation.

3.1 Uncertainty

For many applications returning the confidence of each estimation can be beneficial. In DSEA, the conditional probabilities of all classes form the confidence values. These conditional distributions are aggregated instead of just the predicted classes to improve the estimation. When used with a regression method, at least a second value, the uncertainty $\hat{\sigma}$, must be returned in addition to the estimator $\hat{y}$ from the regression. This allows any confidence interval of the estimator to be calculated as $[\hat{y} \pm q\hat{\sigma}]$. Here q is a suitable factor

depending on the distribution of the estimator. Instead of a symmetric interval, there are also approaches that return a lower and an upper uncertainty. Then the confidence intervals can be calculated as $[\hat{y} - q_l\hat{\sigma}_l, \hat{y} + q_r\hat{\sigma}_r]$.

If the distribution of the estimator is not known and no distribution assumption can be made, confidence intervals can be determined by cross-validation. The model is trained on the subset of the training data and evaluated using the remaining observation. To reduce the variance, multiple rounds of cross-validations are performed, interchanging the different partitions of the data. From the validation results the predictive performance of the model can be estimated, but we also get a non-parametric estimation of the uncertainty. Since this method is much more computationally demanding, we concentrate on regression methods that allow direct access to the uncertainty of estimations.

Traditionally, statistics offer various approaches to calculating uncertainties. These primarily include intervals based on the linear model:

- Point-wise confidence intervals
- Simultaneous confidence intervals and
- Prediction intervals.

In addition, there are methods from the field of machine learning that provide information on uncertainty. For DSEA it is also necessary that these support a weighting of the training data. The following regression methods have shown to be suitable for this purpose:

- Kernel ridge regression,
- Gaussian process regression,
- Mondrian tree and Mondrian forest regression.

These methods are now described in detail with an emphasis on the output of uncertainties.

3.2 Linear Regression

Linear models [16] are used for the statistical analysis of relationships between input variables and the target variable. The linear regression model is limited to a linear relationship:

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{e}.$$

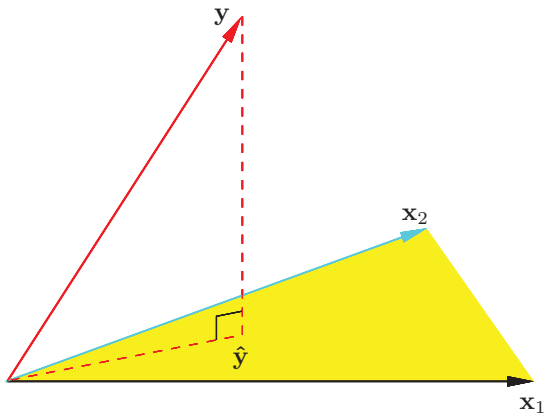


Figure 3.1: Least squares linear regression with two predictors. The outcome vector $\boldsymbol{y}$ is orthogonally projected onto $im(\mathbf{X})$, the hyperplane spanned by the vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The projection $\hat{\boldsymbol{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ represents the vector of predictions by the model [16].

In this section $\mathbf{X}$ refers to the design matrix

$$\mathbf{X} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n,1} & x_{n,2} & \dots & x_{n,d} \end{pmatrix},$$

$\boldsymbol{\beta}$ to the regression coefficients and $\boldsymbol{e}$ with $\mathbb{E}(\boldsymbol{e}) = \mathbf{0}$ and $\mathbb{C}\text{ov}(\boldsymbol{e}) = \sigma^2\mathbf{I}$ to the unobservable measuring errors, such that the expected value of the target variable $\mathbb{E}(\boldsymbol{y}) = \mathbf{X}\boldsymbol{\beta}$ is linear in the regression coefficients $\boldsymbol{\beta}$. The optimization problem is

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}\| .$$

A solution is obtained by the projection theorem [35].

3.2.1 Theorem (Projection theorem). *For the minimization problem* $\min_{\boldsymbol{\eta}\in\mathbf{U}} \|\boldsymbol{y} - \boldsymbol{\eta}\|$ *there exists a unique solution* $\hat{\boldsymbol{\eta}}$ *with*

$$\boldsymbol{y} - \hat{\boldsymbol{\eta}} \perp \mathbf{U}, \text{ that is } (\boldsymbol{y} - \hat{\boldsymbol{\eta}})^T\boldsymbol{\eta} = \mathbf{0} \quad \forall \boldsymbol{\eta} \in \mathbf{U} \subseteq \mathbb{R}^d.$$

The optimal solution $\hat{\boldsymbol{\eta}}$ *is obtained by orthogonal projection onto the vector subspace* $\mathbf{U}$.

The theorem is visualized in figure 3.1. It follows from the projection theorem with $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$

and $\mathbf{U} = im(\mathbf{X})$ that a unique minimum norm solution $\hat{\boldsymbol{\beta}}$ for the minimization problem $\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}\|$ exists. It is obtained from the normal equations

$$\begin{aligned} & \left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right) \in im(\mathbf{X})^{\perp} = ker\left(\mathbf{X}^T\right) \\ \Leftrightarrow \quad & \mathbf{X}^T\left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right)^T = \mathbf{0} \\ \Leftrightarrow \quad & \mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^T\boldsymbol{y}. \end{aligned}$$

$im(\mathbf{A})$, $ker(\mathbf{A})$ and $\mathbf{A}^{\perp}$ denote the column space, the null space and the orthogonal complement of $\mathbf{A}$ respectively. If $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ exists, we obtain the least squares estimator

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\boldsymbol{y}. \tag{3.2}$$

By the Gauss-Markov theorem we know that this is the best linear unbiased estimator (BLUE), as it has the smallest variance of all linear estimators. If $\mathbf{X}^T\mathbf{X}$ is singular the least squares estimator can be calculated using the pseudo inverse [27] $\mathbf{X}^+$:

$$\hat{\boldsymbol{\beta}} = \mathbf{X}^+\boldsymbol{y}.$$

For each matrix $\mathbf{A}$ there is a unique pseudo inverse $\mathbf{A}^+$. If $\mathbf{A}$ is invertible, then $\mathbf{A}^+ = \mathbf{A}^{-1}$ holds. This means that using the pseudo inverse also yields the Gauss-Markov estimator if it exists.

The prediction from a linear regression model is obtained by $\hat{\boldsymbol{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$. An unbiased estimator for the variance is the mean squares error (MSE)

$$\hat{\sigma}^2 = \frac{1}{n_{df}}\left\|\boldsymbol{y} - \hat{\boldsymbol{y}}\right\|^2. \tag{3.3}$$

The degrees of freedom n_{df} are determined by $n - rank(\mathbf{X})$ for the number of training samples n.

3.2.1 Confidence and Prediction Intervals

The MSE is used for the construction of various confidence intervals [38]. Confidence intervals specify a range for the parameter of interest. Each interval is associated with a certain probability $(1-\alpha)$ that the true parameter is captured by the interval. α is referred to as the significance level of the interval. Due to

$$\boldsymbol{x}_{n+1}^T\hat{\boldsymbol{\beta}} \sim \mathcal{N}\left(\boldsymbol{x}_{n+1}^T\boldsymbol{\beta}, \sigma^2\left(\boldsymbol{x_{n+1}}^T\left(\mathbf{X}^T\mathbf{X}\right)^+\boldsymbol{x}_{n+1}\right)\right)$$

we can use the variance of the estimation to construct point-wise $(1-\alpha)$ confidence intervals for the expectation $\mathbb{E}(\boldsymbol{y}|\boldsymbol{x}_{n+1})$ at the location $\boldsymbol{x}_{n+1}$:

$$\left[\boldsymbol{x}_{n+1}^T\hat{\boldsymbol{\beta}} \pm \hat{\sigma}t_{1-\frac{\alpha}{2}}\sqrt{\boldsymbol{x}_{n+1}^T\left(\mathbf{X}^T\mathbf{X}\right)^+\boldsymbol{x}_{n+1}}\right]. \tag{3.4}$$

Here $t_{1-\frac{\alpha}{2}}$ denotes the $(1-\frac{\alpha}{2})$ quantile of the t-distribution. It is used instead of the quantile of the normal distribution because the variance is estimated from the data. In contrast to the confidence intervals for the expected value, the prediction interval specifies an interval for a prediction itself. Because of $y(\boldsymbol{x}_{n+1}) \sim \mathcal{N}\left(\boldsymbol{x}_{n+1}^T\boldsymbol{\beta}, \sigma^2\right)$ and $y(\boldsymbol{x}_{n+1})$ is stochastically independent of $y_1, \ldots, y_n$ we obtain the distribution of the residuals:

$$\boldsymbol{x}_{n+1}^T\hat{\boldsymbol{\beta}} - y \sim \mathcal{N}\left(\boldsymbol{x}_{n+1}^T\boldsymbol{\beta}, \sigma^2\left(1 + \boldsymbol{x_{n+1}}^T\left(\mathbf{X}^T\mathbf{X}\right)^+\boldsymbol{x}_{n+1}\right)\right).$$

The consequential $(1-\alpha)$ prediction interval

$$\left[\boldsymbol{x}_{n+1}^T\hat{\boldsymbol{\beta}} \pm \hat{\sigma}t_{1-\frac{\alpha}{2}}\sqrt{1 + \boldsymbol{x}_{n+1}^T\left(\mathbf{X}^T\mathbf{X}\right)^+\boldsymbol{x}_{n+1}}\right] \tag{3.5}$$

contains on average $(1-\alpha)\cdot 100\%$ of all predictions (if test and training data come from the same distribution). Note the additional 1 within the square root compared to the point-wise confidence interval (3.4). In addition to the uncertainty of the parameter estimation, the random dispersion of the individual values is also included. Thus, the prediction interval is always greater than the confidence interval of the same significance level. Prediction intervals are of more practical use than confidence intervals. This is because they are not only concerned with the accuracy of predicting the true regression, but also with the accuracy of predicting the target values.

3.3 Kernel Ridge Regression

Regularization techniques try to reduce the variance of the estimate to avoid overfitting to the training data. This can be done, for example, by punishing high parameter values. Ridge regression [20] regularizes the optimization problem of the linear model with the ℓ_2 norm of the regression coefficients. The solution extends to

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2 \\
&= \arg\min_{\boldsymbol{\beta}} \left\| \begin{pmatrix} \boldsymbol{y} \\ \mathbf{0}_{n\times 1} \end{pmatrix} - \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{I}_{n\times n} \end{pmatrix}\boldsymbol{\beta} \right\|^2 \\
&\overset{(3.2)}{=} \left(\begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{I}_{n\times n} \end{pmatrix}^T \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{I}_{n\times n} \end{pmatrix}\right)^{-1} \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{I}_{n\times n} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{y} \\ \mathbf{0}_{n\times 1} \end{pmatrix} \\
&= \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\boldsymbol{y} \\
&= \mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}\right)^{-1}\boldsymbol{y}.
\end{aligned}$$

The last step follows from the push-through identity [17]. The regularization parameter λ controls the amount of penalty. The higher the value for λ is selected, the smaller the calculated regression coefficients become.

Kernel ridge regression [28] additionally uses the kernel trick [26], which allows for non-linear regression planes. The data is transformed into a high-dimensional space by a feature map ϕ. A linear function is fitted in the feature space, which corresponds to a non-linear function in the original space. The feature mapping does not have to be calculated explicitly and does not even have to be known. It is sufficient to specify the similarity of any pair $\boldsymbol{x}_i, \boldsymbol{x}_j$ as scalar product in the feature space by a using a kernel $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$. By defining the kernel matrices $\boldsymbol{\Sigma}$ to contain the kernel of each pair of elements the prediction of a kernel ridge model can be written as

$$\begin{aligned}\hat{y}_{n+1} &= \phi\left(\boldsymbol{x}_{n+1}\right)^T \hat{\boldsymbol{\beta}}_{\phi(X)} \\ &= \boldsymbol{\Sigma}_{x_{n+1}^T, X^T} \left(\boldsymbol{\Sigma}_{X,X^T} + \lambda \mathbf{I}\right)^{-1} \boldsymbol{y}.\end{aligned}$$

3.3.1 Asymptotic Prediction Interval

Härdle [14] gives an asymptotic distributional approximation relating to the Nadaraya-Watson estimator [29]:

$$\sqrt{n} \frac{\hat{y}(\boldsymbol{x}_{n+1}) - y(\boldsymbol{x}_{n+1})}{\sqrt{\frac{\sigma^2(\boldsymbol{x}_{n+1}) \|\mathcal{K}\|^2}{f(\boldsymbol{x}_{n+1})}}} \to \mathcal{N}\left(b, 1\right). \tag{3.6}$$

where the bias b is a complicated function of $f(\boldsymbol{x}_{n+1})$ and $y(\boldsymbol{x}_{n+1})$. If it would be possible to compute the bias, we should have done that in the first place to arrive at a better estimator. Instead, he assumes the bias is of negligible size compared to the variance. By deploying natural estimators for the variance σ^2 (3.3) and the density f (see section 4.2) we use this convergence to compute asymptotic $(1-\alpha)$ prediction intervals analogue to (3.5):

$$\left[\hat{y}(\boldsymbol{x}_{n+1}) \pm \hat{\sigma}(\boldsymbol{x}_{n+1}) t_{1-\frac{\alpha}{2}} \sqrt{1 + \frac{\|\mathcal{K}\|^2}{\hat{f}(\boldsymbol{x}_{n+1})}}\right].$$

3.4 Gaussian Processes

A Gaussian process [47] is a stochastic process in which each function value is modeled by a normal distribution. The normal distributions are stochastically dependent on each other. The dependencies are determined by their proximity using a kernel $\mathcal{K}$:

$$\boldsymbol{\Sigma}_{X,X^T} = \begin{pmatrix} \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_1) & \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_2) & \ldots & \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \mathcal{K}(\boldsymbol{x}_2, \boldsymbol{x}_1) & \mathcal{K}(\boldsymbol{x}_2, \boldsymbol{x}_2) & \ldots & \mathcal{K}(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \ldots & \ldots & \ldots & \ldots \\ \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_1) & \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_2) & \ldots & \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}.$$

Gaussian processes can be used for regression to make probabilistic predictions. Given an expectation and a learned covariance matrix, a conditional distribution is calculated at any new location. This procedure is made possible by the following property of the multivariate normal distribution.

3.4.1 Theorem (Conditional normal distribution). *Let* $\boldsymbol{X}_1, \boldsymbol{X}_2$ *be jointly normally distributed:* $\begin{pmatrix}\boldsymbol{X}_1\\ \boldsymbol{X}_2\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Sigma}_{1,1} & \boldsymbol{\Sigma}_{1,2}\\ \boldsymbol{\Sigma}_{2,1} & \boldsymbol{\Sigma}_{2,2}\end{pmatrix}\right)$. *Then* $\boldsymbol{X}_2$ *is conditionally normally distributed given* $\boldsymbol{X}_1 = \boldsymbol{x}_1$ *with*

$$\boldsymbol{X}_2|\boldsymbol{X}_1 = \boldsymbol{x}_1 \sim \mathcal{N}\left(\boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{2,1}\boldsymbol{\Sigma}_{1,1}^{-1}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{2,2} - \boldsymbol{\Sigma}_{2,1}\boldsymbol{\Sigma}_{1,1}^{-1}\boldsymbol{\Sigma}_{1,2}\right).$$

To use the theorem, the joint distribution of the training data $\boldsymbol{X} = \boldsymbol{x}$, which are assumed to make up a multivariate normal distribution, and a new observation $\boldsymbol{x}_{n+1}$ is formed. The joint covariance matrix is modeled using the kernel $\mathcal{K}$ again:

$$\begin{aligned}\boldsymbol{\Sigma}_{X\cup x_{n+1},(X\cup x_{n+1})^T} &= \begin{pmatrix}\boldsymbol{\Sigma}_{X,X^T} & \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_{n+1})\\ & \ldots\\ \mathcal{K}(\boldsymbol{x}_{n+1}, \boldsymbol{x}_1) & \ldots \quad \mathcal{K}(\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1})\end{pmatrix}\\ &= \begin{pmatrix}\boldsymbol{\Sigma}_{X,X^T} & \boldsymbol{\Sigma}_{X,x_{n+1}}\\ \boldsymbol{\Sigma}_{x_{n+1}^T,X^T} & \mathcal{K}(\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1})\end{pmatrix}.\end{aligned}$$

A prediction is then made as the expected value of the distribution at the desired location conditional on the distribution of the training data. From theorem 3.4.1 follows:

$$\hat{f}(\boldsymbol{x}_{n+1}) = \boldsymbol{\mu}_{x_{n+1}} + \boldsymbol{\Sigma}_{x_{n+1}^T,X^T}\boldsymbol{\Sigma}_{X,X^T}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_X). \tag{3.7}$$

The expected values $\boldsymbol{\mu}_{x_{n+1}}, \boldsymbol{\mu}_X$ are usually set to zero or to the mean value of the training data. Alternatively, a trend can be modeled by using estimated regression coefficients to specify a mean vector.

In addition, an estimated uncertainty of the prediction can be obtained from the variance of the conditional distribution:

$$\hat{\sigma}^2(\boldsymbol{x}_{n+1}) = \mathcal{K}(\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1}) - \boldsymbol{\Sigma}_{x_{n+1}^T,X^T}\boldsymbol{\Sigma}_{X,X^T}^{-1}\boldsymbol{\Sigma}_{X,x_{n+1}}.$$

3.4.1 Regularization

In its basic form the Gaussian process interpolates the training data: $f(x) = \hat{f}(x)$. The prediction uncertainty at these points is $\hat{\sigma}^2(x) = 0$. If this is not desirable because the data itself is stochastic or even necessary when identical observations have different target values, then noise is added to the diagonal of the learned covariance matrix (i.e. the modeled variance of the data). This regularizes the model. Another interpretation motivates the prevention of the singularity of the matrix by adding a scaled unit matrix to the covariance matrix. In addition, high variances of the calculated solution that occur with multicollinearity can be counteracted as it reduces the condition number of the matrix, making the inversion numerically more stable. Consequently, the prediction becomes as follows:

$$\hat{f}(\boldsymbol{x}_{n+1}) = \boldsymbol{\mu}_{x_{n+1}} + \boldsymbol{\Sigma}_{x_{n+1}^T,X^T}\left(\boldsymbol{\Sigma}_{X,X^T} + \lambda\mathbf{I}\right)^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_X). \tag{3.8}$$

Note the equivalence of (3.8) to kernel ridge regression (section 3.3) when using zero mean. Despite this, the solution can still differ since the hyperparameters of the kernel function are optimized differently. In addition, Gaussian processes are capable of estimating the variance due to their stronger theoretical foundation. For the variance estimator follows analogously:

$$\hat{\sigma}^2(\boldsymbol{x}_{n+1}) = \mathcal{K}(\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1}) - \boldsymbol{\Sigma}_{x_{n+1}^T, X^T} \left(\boldsymbol{\Sigma}_{X,X^T} + \lambda \mathbf{I}\right)^{-1} \boldsymbol{\Sigma}_{X, x_{n+1}}.$$

In this term $\mathcal{K}(\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+1})$ is simply the prior covariance. Subtracted is a term representing the information the data gives us about the distribution at $\boldsymbol{x}_{n+1}$. To get the predictive distribution we simply add λ to the variance as in (3.5). By using the $\left(1 - \frac{\alpha}{2}\right)$-quantile of the standard normal distribution we obtain the prediction intervals:

$$\left[\hat{f}(\boldsymbol{x}_{n+1}) \pm z_{1-\frac{\alpha}{2}} \sqrt{\lambda + \hat{\sigma}^2(\boldsymbol{x}_{n+1})}\right].$$

3.4.2 Weighted Training

The implementation of Gaussian processes in *scikit-learn* [32] does not support weighted training. Re-weighting the training set in every iteration is mandatory for good deconvolution results with DSEA. To enhance Gaussian processes by weighted training we take a closer look at the prediction function (3.8). The influence of input variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ takes place in the kernel matrices $\boldsymbol{\Sigma}_{x_{n+1}, X}$ and $\boldsymbol{\Sigma}_X + \lambda \mathbf{I}$. To have some observations have higher influence than others we could scale the kernel function depending on the observations. Alternatively, we could vary the noise level λ for each observation. We compare both approaches.

We begin with the varying noise level approach. For every observation we compute an individual noise level be factoring in their weight w_i:

$$\lambda_i' = \frac{\lambda}{w_i}. \tag{3.9}$$

This causes the impact of observations to be proportional to their weight. When the weight is 1 this reduces to λ which matches the unweighted training. When the weight decreases, the noise approaches infinity.

On the other hand, the weighted Gaussian process regression scales each kernel by the root of the corresponding weights so that the prediction function accumulates to

$$\begin{aligned}\hat{f}(\boldsymbol{x}_{n+1}) &= \boldsymbol{\mu}_{x_{n+1}} + \boldsymbol{w}_r^T \odot \boldsymbol{\Sigma}_{x_{n+1}^T, X^T} \left(\boldsymbol{w}_r \boldsymbol{w}_r^T \odot \boldsymbol{\Sigma}_{X,X^T} + \lambda \mathbf{I}\right)^{-1} \boldsymbol{w}_r \odot (\boldsymbol{y} - \boldsymbol{\mu}_X) \\ &= \boldsymbol{\mu}_{x_{n+1}} + \boldsymbol{\Sigma}_{x_{n+1}^T, X^T} \mathit{diag}(\boldsymbol{w}_r) \left(\mathit{diag}(\boldsymbol{w}_r)\, \boldsymbol{\Sigma}_{X,X^T}\, \mathit{diag}(\boldsymbol{w}_r) + \lambda \mathbf{I}\right)^{-1} \mathit{diag}(\boldsymbol{w}_r)\, (\boldsymbol{y} - \boldsymbol{\mu}_X)\end{aligned}$$

where $\boldsymbol{w}_r$ is the element-wise root of $\boldsymbol{w}$ and $\odot$ denotes the Hadamard product (element-wise product). We assume that the weights sum up to n: $\sum_i w_i = n$. Note, that the labels

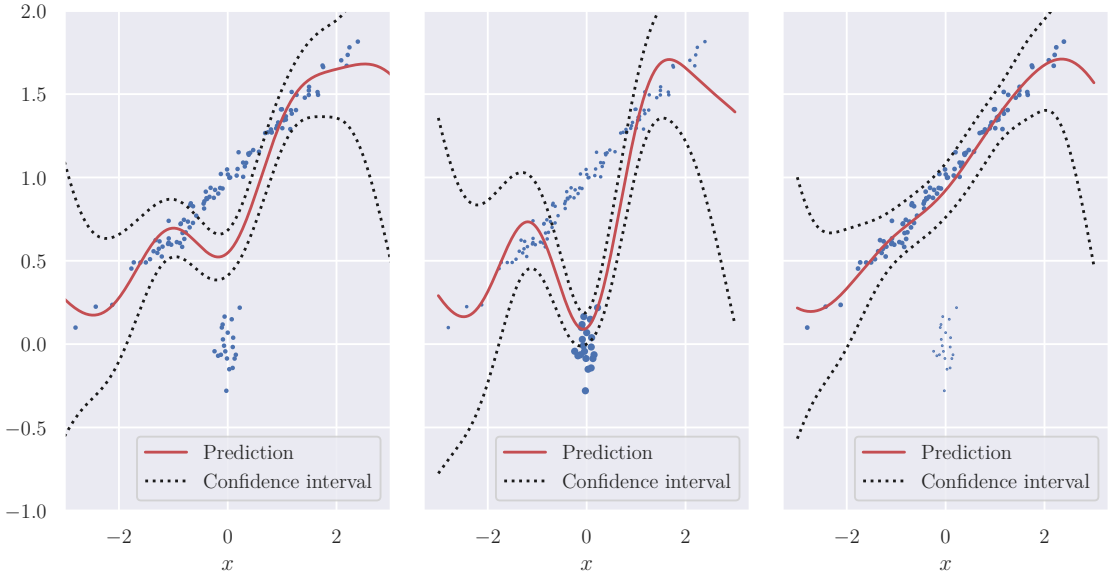


Figure 3.2: Weighted Gaussian process regression on a small synthetic data set. The data was sampled from two distributions, one of which represents a noise cluster. The observations were weighted according to the distribution from which they were obtained. The larger the point in the plot, the higher the weight. The left panel shows the fit on unweighted data. When assuming that the bottom cluster is of greater importance by assigning them higher weights the model adjusts the fit in the region of the cluster (center). Provided that noise points can be detected to adjust the weights accordingly, the Gaussian process can find a much better fit for the remaining observations (right).

$\boldsymbol{y}$ must be weighted, too. Let $\mathbf{D}$ be the diagonal matrix $\mathbf{D} = diag(\boldsymbol{w_r})$, therefore $\mathbf{D}^{-1}$ exist:

$$\begin{aligned}
\hat{f}(\boldsymbol{x}_{n+1}) &= \boldsymbol{\mu}_{x_{n+1}} + \boldsymbol{\Sigma}_{x_{n+1}^T,X^T}\mathbf{D}\left(\mathbf{D}\boldsymbol{\Sigma}_{X,X^T}\mathbf{D} + \lambda\mathbf{I}\right)^{-1}\mathbf{D}(\boldsymbol{y} - \boldsymbol{\mu}_X) \\
&= \boldsymbol{\mu}_{x_{n+1}} + \boldsymbol{\Sigma}_{x_{n+1}^T,X^T}\left(\mathbf{D}^{-1}\left(\mathbf{D}\boldsymbol{\Sigma}_{X,X^T}\mathbf{D} + \lambda\mathbf{I}\right)\mathbf{D}^{-1}\right)^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_X) \\
&= \boldsymbol{\mu}_{x_{n+1}} + \boldsymbol{\Sigma}_{x_{n+1}^T,X^T}\left(\boldsymbol{\Sigma}_{X,X^T} + \lambda\mathbf{D}^{-1}\mathbf{D}^{-1}\right)^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_X) \\
&= \boldsymbol{\mu}_{x_{n+1}} + \boldsymbol{\Sigma}_{x_{n+1}^T,X^T}\left(\boldsymbol{\Sigma}_{X,X^T} + \lambda diag\left(\frac{1}{w_1},\ldots,\frac{1}{w_n}\right)\right)^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_X).
\end{aligned}$$

Comparing this result to the variable noise Gaussian process regression we observe that the prediction functions match when using a noise level as in (3.9). We proved that both approaches are equivalent.

As seen in figure 3.2 using weighted observations can improve the estimation significantly when assigning reasonable weights. The possibility of a weighted training makes Gaussian processes usable for our application.

3.5 Decision Trees

A decision tree [6] divides the feature space recursively. The splits are determined during training by greedily maximizing a purity measure. In this way, an attempt is made to separate the data into groups that are as pure as possible. Regression trees often minimize the variance of the target variable in the nodes induced by the split. For inference, the tree is tracked from the root to a leaf through the usually binary decisions and then the representative class of the leaf is returned. In regression analysis, the returned value is the arithmetic mean of all target values of the training data that ended up in the relevant leaf.

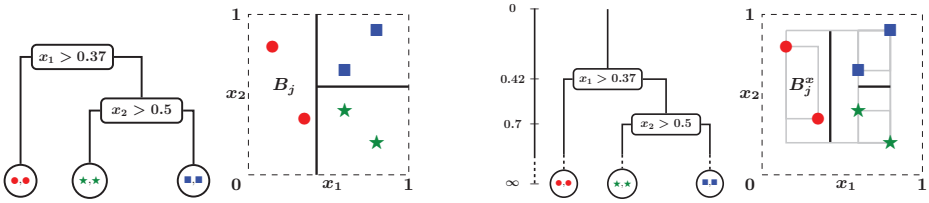
A random forest combines several decision trees to form an ensemble. Decorrelation of the individual models is a key factor in minimizing generalization errors. If no action is taken, all models may learn the same and the ensemble may no longer offer benefits. To support diversity, each tree is trained with its own bootstrap (subsample of the training data). In addition, only a random subset of all features are available in each split (typically $\sqrt{d}$ many for the number of features d). In order to infer from a forest, the outputs of the individual trees are aggregated to form a single output.

3.6 Mondrian Trees

Since the subspaces in the binary splits of the decision trees extend infinitely, data in unknown areas is also located in a subspace and is assigned the value of the corresponding leaf, although there is great uncertainty. Generally, there is no specification about the uncertainty of an output. Mondrian Trees [22, 23] try to solve these problems by taking into account the boundaries of the training data when passing through each node and by considering not only the leaf but all nodes on the path for the prediction. As with decision trees, multiple Mondrian trees can be combined to form an ensemble. The ensemble is called Mondrian forest.

Mondrian Trees select the feature for the next split randomly proportional to its range of values. The threshold of the split is selected randomly from a uniform distribution. The minima and maxima are determined by the training data set. Thus, features with a wide range of values are drawn more frequently. The impurity does not matter for the split.

Each new node j stores a bounding box $(\boldsymbol{l}_j, \boldsymbol{u}_j)$ that surrounds all training data within the subspace defined by the node. In addition, for each node the mean value and the standard deviation of the respective observations are stored and a value τ is assigned. τ is called the time of the split and is drawn from the exponential distribution with $\lambda = \|\boldsymbol{u}_j - \boldsymbol{l}_j\|_1$. Large bounding boxes result in large values for τ. The differences to ordinary decision trees are illustrated in figure 3.3.



(a) Decision Tree (b) Mondrian Tree

Figure 3.3: Panel (a) shows tree structure and partition of a decision tree, while panel (b) shows the corresponding Mondrian tree. Each node in the Mondrian tree is associated with a time $\tau \in (0, \infty)$ of the split. Also, the splits are committed only within the range of the training data in each block (denoted by gray rectangles). For example, consider the rectangle B_j associated with the red circles: $B_j = [0, 0.37] \times [0, 1]$ and $B_j^* \subseteq B_j$ is the smallest rectangle enclosing the two data points. [22]

Inferring from the model aggregates the distribution of all nodes on the path to the predicted leaf:

$$\mathbb{P}(y|\boldsymbol{X} = \boldsymbol{x}) = \sum_i w_i \mathbb{P}_{\mu_i, \sigma_i}.$$

For this mixture distribution, the expectation

$$\mu = \sum_i w_i \mu_i \tag{3.10}$$

and the variance

$$\begin{aligned} \sigma^2 &= \sum_i w_i \mu_i^{(2)} - \left(\sum_i w_i \mu_i \right)^2 \\ &= \sum_i w_i \left(\mu_i^2 + \sigma_i^2 \right) - \mu^2 \end{aligned} \tag{3.11}$$

can be calculated from its components. $\mu_i^{(2)}$ denotes the second moment of the i-th component. The weighting of the distributions is described in the following section.

3.6.1 Calculating the Node Weights

Two factors are used to determine the weights w_j of all nodes j when predicting the target value for a new observation $\boldsymbol{x}_{n+1}$. Δ_j indicates how much a bounding box decreases in comparison to the parent node, while η_j indicates the distance to the bounding box if the point is located outside:

$$\begin{aligned} \Delta_j &= \tau_j - \tau_{parent(j)} \\ \eta_j &= \sum_i \left(\max(x_{n+1,i} - u_{j,i}, 0) + \max(l_{j,i} - x_{n+1,i}, 0) \right). \end{aligned}$$

p_j is defined as the probability of separation: the probability that $\boldsymbol{x}_{n+1}$ does not fit into the subspaces of the training data. The higher Δ_j and η_j, the higher the probability of separation:

$$p_j(\boldsymbol{x}_{n+1}) = 1 - \exp\left(-\Delta_j \eta_j(\boldsymbol{x}_{n+1})\right).$$

The weight of a node indicates the probability that $\boldsymbol{x}_{n+1}$ will not fit into the subspaces of the training data starting from this node. For this, p_j is multiplied by the complementary probabilities that the point already did not fit at the parent nodes. The remaining weight is assigned to the leaf nodes causing the weights of each path to add up to 1:

$$w_j(\boldsymbol{x}_{n+1}) = \begin{cases} 1 - \sum_k \left(p_k(\boldsymbol{x}_{n+1}) \prod_{i \in anc(k)} (1 - p_i(\boldsymbol{x}_{n+1})) \right) & \text{if j is leaf} \\ p_j(\boldsymbol{x}_{n+1}) \prod_{i \in anc(j)} (1 - p_i(\boldsymbol{x}_{n+1})) & \text{else.} \end{cases} \tag{3.12}$$

By using a probability model, the bounding boxes do not impose hard thresholds. Instead, the more the bounding boxes are exceeded, the smaller are the weights of the child nodes. If the distance of a new observation to the training data is very large, the weights approach zero and the distribution of the root will be output. This is the prior distribution determined by the entire training data. On the other hand, an observation from the training set lies within all bounding boxes on its path, so all probabilities of separation are zero. The leaf is assigned weight 1 and all other nodes are assigned zero weight. This is ordinary decision tree inference.

3.6.2 Prediction Interval

In order to determine a prediction interval from the calculated variance (3.11), the distribution of the estimator must be known. We consider two methods based on different assumptions. Assuming that the target variable is distributed normally in each node, the aggregated distribution is a mixture of Gaussians

$$\begin{aligned} F_Y(y) &= \int_{-\infty}^{x} \sum_i w_i \mathbb{P}_{\mu_i,\sigma_i}(x)\,\mathrm{d}x \\ &= \sum_i w_i \int_{-\infty}^{x} \mathbb{P}_{\mu_i,\sigma_i}(x)\,\mathrm{d}x \\ &= \sum_i w_i \Phi\left(\frac{x-\mu_i}{\sigma_i}\right) \end{aligned} \tag{3.13}$$

with the cumulative distribution function Φ of the standard normal distribution. Unfortunately, there is no closed form for the quantile function F_Y^{-1}. We can still get any q-quantile numerically by computing the root of $F_Y - q$ (see figure 3.4).

Because any quantile falls between the minimum and the maximum of the quantiles of the components, the search interval can be restricted to

$$\left[\min_i \left(\Phi^{-1}\left(\frac{x-\mu_i}{\sigma_i}\right)\right), \max_i \left(\Phi^{-1}\left(\frac{x-\mu_i}{\sigma_i}\right)\right)\right] = \left[\min_i(\mu_i + \sigma_i z_q), \max_i(\mu_i + \sigma_i z_q)\right].$$

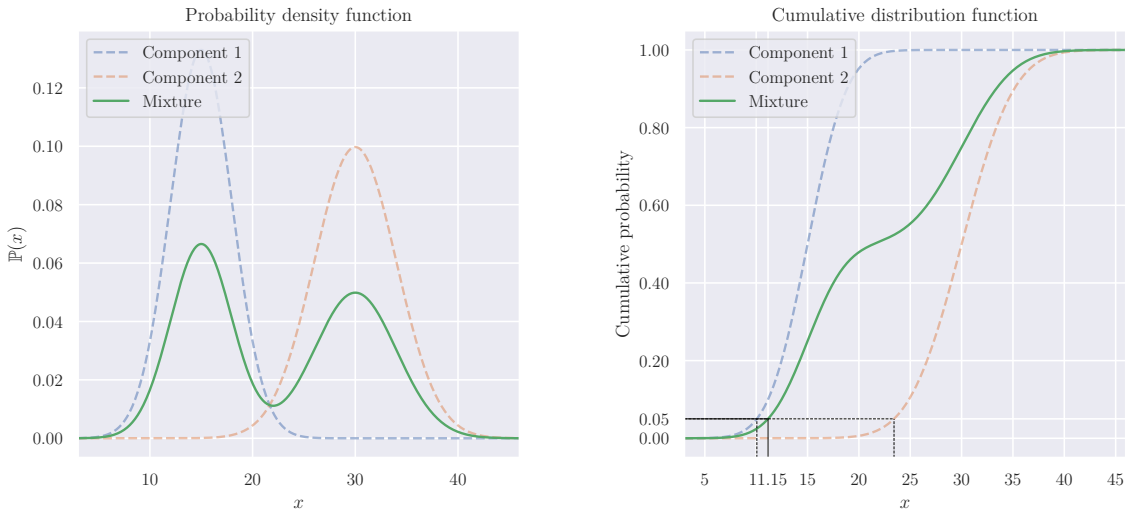


Figure 3.4: Quantile computation for the mixture of two equally weighted Gaussian distributions $\mathcal{N}(10, 1)$ and $\mathcal{N}(13, 1)$. The cumulative distribution function $F_Y(y)$ of the mixture is given by (3.13). The 0.05-quantile is the root of $F_Y(y) - 0.05 \approx 11.15$. Note that the $F_Y^{-1}(q)$ cannot be computed as a linear combination of the quantiles of the components, since the cumulative distribution functions are aggregated vertically, not horizontally.

The quantiles allow us to compute the prediction interval of arbitrary significance levels α:

$$\left[F_Y^{-1}\left(\frac{\alpha}{2}\right), F_Y^{-1}\left(1-\frac{\alpha}{2}\right)\right].$$

Note, that the prediction intervals are likely to be asymmetric about the mean. Since the quantile computation using the proposed method above is unusually expensive we consider the simpler interval

$$\left[\mu \pm t_{1-\frac{\alpha}{2}}\sigma\right]$$

using equations (3.10) and (3.11). This assumes a normal distribution of the mixture. The components of the mixture correspond to a decision path from the root of the tree to the relevant leaf. It follows that each component after the first is estimated from a subsample of the observations that are used to estimate the previous component. The mixture aggregates normal distribution that are estimated using highly similar data. Considering this, the mixture distribution should be sufficiently normal in most cases. Since the variance is estimated we use the quantiles of the t-distribution again.

3.7 Other Regression Methods

A variety of other regression methods were considered. These include another tree-based method, the gradient boosting regression, which allows for optimization of arbitrary differentiable loss functions. The Huber loss function

$$\ell_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta, \\ \delta\,|y - \hat{y}| - \frac{1}{2}\delta^2 & \text{else} \end{cases}$$

is a piece-wise composition of squared and absolute loss functions. The transition point δ defines points further away to be outliers, where the absolute loss is used. By choosing the values of δ to be the α-quantile of the distribution $y - \hat{y}$, gradient boosting can be used to predict arbitrary quantiles of the output variable. In this way prediction intervals of any significance level can be estimated. As with Mondrian trees, the prediction intervals are asymmetric about the mean.

Another method uses two separate models. One predicts the output, the other predicts the confidence of the first model. This composite approach is very flexible, since both models can be built by any regression method.

Improvements in neural networks have undoubtedly led to major advances in supervised learning. We can use a neural network with an output layer of two neurons to predict the target variable as well as the confidence of the prediction. An extension for asymmetric intervals can easily be achieved by selecting three output neurons.

All methods from this chapter are evaluated in the next section with regard to their output of uncertainties.

3.8 Evaluation

Appropriate regression methods return the local uncertainty of a data point. When the data point has many neighbors the uncertainty of the prediction should be low, it should be high with very few or no neighbors. Similarly, the uncertainty should increase with the variance of the target variable at the location of the data point. These limitations sound simple and intuitive, yet a short experiment on the Iris flower data set showed that several regression methods do not satisfy them.

The Iris flower data set is a multivariate data set that was introduced as an example of linear discriminant analysis. It became a typical test case for many statistical techniques in machine learning. The data set consists of 50 samples from each of three species of the Iris flower. Four features were measured from each sample: the length and the width of the sepals and petals. We use the Petal length to predict the petal width of the samples. The species, sepal length and sepal width are disregarded. The evaluated methods are:

- Ordinary least squares linear regression with point-wise confidence intervals and prediction intervals.

- Kernel ridge regression with asymptotic prediction intervals. For the kernel the Gaussian kernel was chosen.

- Gaussian process regression with 95% prediction intervals. The target values are normalized to fall back to the mean of the target values in regions where data is sparse. The regularization parameters α is set to 0.1.

- Mondrian tree and Mondrian forest regression. We use the simple and the numerically computed 95% prediction intervals.

- Gradient boosting regression with 2.5% and 97.5% quantiles.

- Random forest regression where kernel ridge regression is used to estimate the absolute residuals $\hat{\sigma}$ of the random forest. As a local regression method, the kernel ridge regression predicts zero in regions with no data points. This behavior is not desired in our case. Instead, the model should predict high values in these regions. To fall back to another prior $\hat{\sigma}_{prior}$ the standard deviations are shifted before training: $\hat{\sigma}' = \hat{\sigma} - \hat{\sigma}_{prior}$. The shift is reversed after prediction. $\hat{\sigma}_{prior}$ was chosen as $2 \cdot \max(\hat{\sigma})$. We use the intervals $[\hat{y}_i \pm t_{0.975}\hat{\sigma}_i]$ for comparison with the other methods. The kernel was chosen as Gaussian kernel.

- A Multi-layer perceptron with prediction and standard deviation outputs. The true standard deviation of the predictions is not known before training. To estimate them, the same $\hat{\sigma}_{prior}$ as above is used as the target standard deviation in a first training session. The network is then re-trained while updating the target standard deviations to the absolute residuals $|y - \hat{y}|$. We use the intervals $[\hat{y}_i \pm t_{0.975}\hat{\sigma}_i]$ for comparison with the other methods. The network consists of three hidden layers of size 10 and is trained with batch size 10. The inputs are chosen as all polynomial combinations of the features with degree less than or equal 3.

We use the implementation of Mondrian forests in *scikit-garden*[1] and the implementations in *scikit-learn*[2] for all other methods. Unless otherwise stated default values are used for all parameters.

As seen in the figure 3.5, linear regression and gradient boosting do not represent uncertainty particularly well in locations with unknown data. The neural network approach leads to very different results after each run, none of them significantly better than the one shown. This is probably due to the small data set. However, since these methods are

[1] `https://github.com/scikit-garden/scikit-garden`
[2] `https://github.com/scikit-learn/scikit-learn`

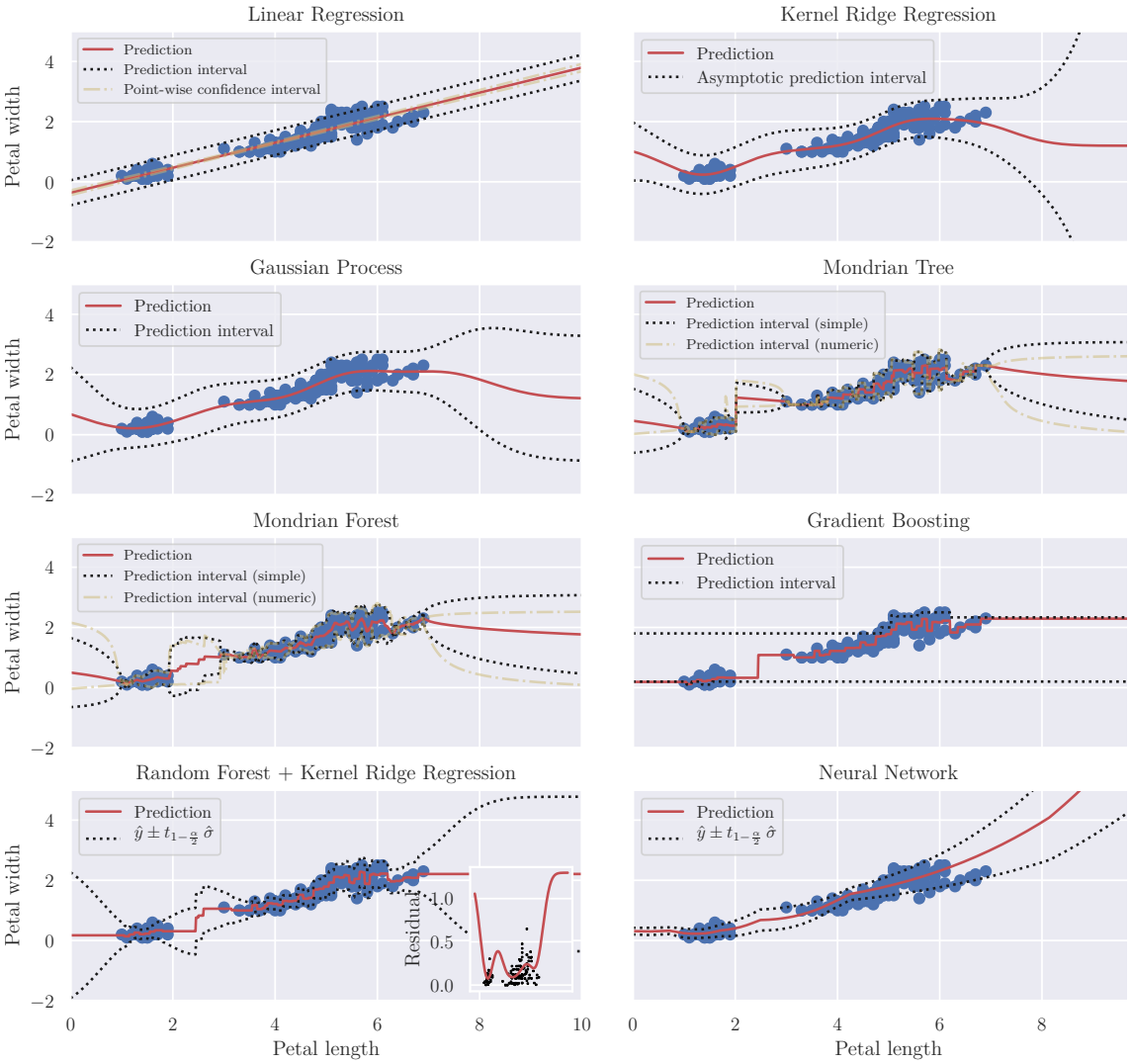


Figure 3.5: Comparison of the uncertainty estimations of the presented regression models. In a simple experiment on the Iris flower data set several regression techniques were compared regarding their modeling of uncertainty. Mondrian forests, Gaussian processes and kernel ridge regression showed promising results as they increase the uncertainty in locations exempt from data. The composite model is another option worth exploring but has less theoretical foundation than the aforementioned models. Linear regression, gradient boosting and neural networks are discarded because their output of uncertainties do not meet our requirements.

not convincing in this simple example, they are not considered further. The uncertainty was best modeled by Mondrian forests, Gaussian processes and Kernel regression. The composite model, where the second model predicts the uncertainty of the first, is another

promising method worth investigating. By using a kernel ridge regression as the second model any regression model can be amplified to return uncertainties.

We have now found useful regression methods to replace the classifier in the DSEA algorithm. In the next chapter we explore how we can use the uncertainties returned by the regression method to improve the estimation of continuous densities.

Chapter 4

Density Estimation

Statistics deals with tools that allow conclusions to be drawn about the total population from an observed sample. The density estimation assumes that the sample comes from an unknown distribution and tries to estimate it using the sample. On the one hand, there are parametric approaches in which a known distribution family is assumed. Then only the parameters of the distribution have to be estimated. On the other hand, there are non-parametric approaches that estimate a density without making any distributional assumptions. These are more flexible and therefore more relevant for estimating the distribution of γ particle energy in Cherenkov astronomy. For this reason, the present chapter is limited to non-parametric methods. Before we move onto continuous density estimation we review the current method for estimation: the histogram.

4.1 Histogram

The oldest and most popular density estimator is the histogram [40]. For histograms, the value range is divided into usually equidistant intervals and the numbers of sample elements in the individual ranges are counted. The relative number gives the value of the density at the respective position. The equidistant intervals are calculated as

$$[x_0 + mh, x_0 + (m+1)h), m \in \mathbb{Z}.$$

with x_0 the starting point and h the bin width. The density is then defined as

$$\hat{f}(x) = \frac{\text{number of } X_i \text{ in the same bin as } x}{nh}.$$

This histogram estimates a discrete density that has steps at the interval boundaries. It thus assigns the same estimate to each location within a bin. This makes histograms an easy to use and to interpret density estimator and an excellent choice for data representation. This advantage diminishes in an iterative application of density estimation, where the individual densities are not examined. As we see later in this chapter the histogram is, in fact, an inefficient use of the data.

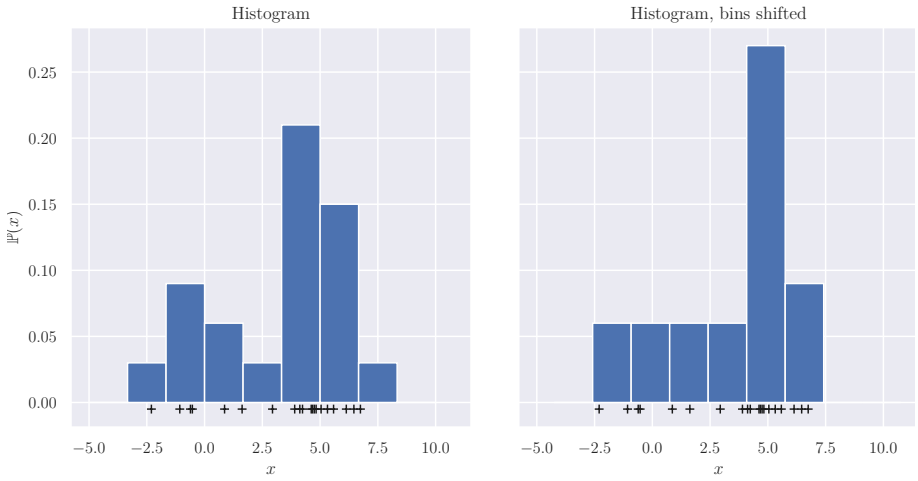


Figure 4.1: Impact of starting point selection for histograms. A major problem with histograms is that the choice of binning can have an excessive effect on the resulting visualization. Not only the choice of the bin width and the number of bins, but also the choice of the starting point of the bins are non-trivial decisions. Both histograms are over the same data, but the histogram in the right panel has its bins shifted (by shifting the starting point x_0). The results of the two visualizations look very different. This might lead to contrasting interpretations of the data.

4.1.1 Bin Width and Starting Point

To construct a histogram, it is necessary to specify a bin width. It controls the trade-off between bias and variance. A low bin width results in a high variance estimator that may fit too well to the random noise in the observations. The higher the bin width, the higher the bias to the uniform distribution. The impact of the bin width is obvious, but also the chosen starting point of the bins can have major impact on the histogram estimation as shown in figure 4.1.

4.1.2 Weighted Estimation

Histograms can be adapted to estimate densities of weighted data sets. The weighted estimator computes a weighted sum of the observations in the same bin instead of just counting them:

$$\hat{f}(x) = \frac{\text{sum of weights of } X_i \text{ in the same bin as } x}{nh}.$$

It is assumed that the weights add up to 1. We investigate how the weighted histograms can improve the estimation when incorporating uncertainties to weight the observations.

4.2 Kernel Density Estimation

A popular approach in non-parametric density estimation is the kernel density estimator (KDE) [14, 46]. Intuitively, a kernel $\mathcal{K}$ is placed on each observation. Then all kernels are summed up to obtain the estimation:

$$\hat{f}(x) = \frac{1}{n} \sum_i \mathcal{K}_h (x - x_i) = \frac{1}{nh} \sum_i \mathcal{K} \left(\frac{x - x_i}{h} \right). \tag{4.1}$$

In dense locations many kernels are added on top of each other, resulting in a high value of the estimated density. The estimator can also be interpreted as a weighted sum of individually scaled kernels. The kernel determines the shape and the range of the estimated density. When the kernel function is continuous, the estimation will be too. When the kernel function has finite support, the estimation will too. A popular kernel is the radial basis function (RBF) or Gaussian kernel

$$\begin{aligned} \mathcal{K}_{\text{Gauss}}(x) &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2} \right) \\ &= \phi(x), \end{aligned} \tag{4.2}$$

which results in relative smooth estimates. However, the unbounded support makes it an expensive choice when estimating the density of large data sets. The Epanechnikov kernel

$$\mathcal{K}_{\text{Epanechnikov}}(x) = \begin{cases} \frac{3}{4}\left(1 - x^2\right) & \text{if } -1 \leq x \leq 1 \\ 0 & \text{else} \end{cases} \tag{4.3}$$

places a similar bell curve on each observation but has finite support. This can be utilized to efficiently compute the target density at queried locations by partitioning the data beforehand.

The bandwidth parameter h is required for the calculation of (4.1). Similar to the bin widths of histograms, this controls the smoothness of the estimator and thus the trade-off between bias and variance. In the following chapter we discuss a number of options for selecting a suitable bandwidth. Most options refer to the optimal bandwidth.

4.2.1 Optimal Bandwidth

The choice of bandwidth is of crucial importance when estimating a density and should always depend on the application. If the density estimation is only intended for visualization purposes, a manual adjustment of the bandwidth may be sufficient until a desired result is achieved. However, this is not possible for all applications, including the iterative deconvolution. First of all, a density is estimated in each iteration and should not necessarily always use the same bandwidth. Second, deconvolution should work independently of the training data for new observations, so manually setting a bandwidth is problematic.

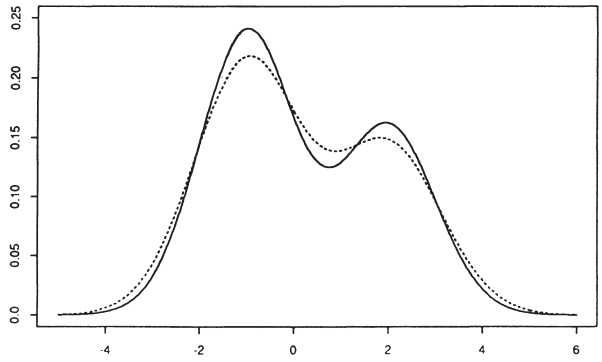


Figure 4.2: Visualization of the kernel density bias effect with a true density (solid line) and the expectation of a Gaussian kernel density estimation (dashed line). This explains why the estimation tends to overestimate local minima and underestimate local maxima. The model has a tendency (as almost any other model) to oversmooth sharp peaks in the data [14].

A solution is provided by methods that automatically calculate a suitable bandwidth. To derive the optimal bandwidth, we take a look at the bias and variance at each location x:

$$\mathbb{B}\text{ias}\left(\hat{f}(x)\right) = \frac{h^2}{2}\mu_2(\mathcal{K})f''(x) + o\left(h^2\right).$$

$\mu_2(\mathcal{K})$ denotes the second moment of the kernel. The bias is proportional to h^2. Choosing a small bandwidth reduces the bias. It also depends on the curvature of the density function at x. As a result, the bias is positive at local minima and negative at local maxima. This effect is displayed in figure 4.2. For the variance holds:

$$\mathbb{V}\text{ar}\left(\hat{f}(x)\right) = \frac{1}{nh}f(x)\left\|\mathcal{K}\right\|^2 + o\left(\frac{1}{nh}\right).$$

The variance is proportional to f and $\left\|\mathcal{K}\right\|^2 = \int_{-\infty}^{\infty}\mathcal{K}(x)^2\,\mathrm{d}x$ and decreases as nh increases. Thus, bias and variance vary with the choice of h in opposite directions. This illustrates the bias-variance dilemma. If one chooses a small bandwidth, the variance becomes large, if one chooses a large bandwidth, the bias becomes large.

The Mean Squared Error (MSE) combines both properties in one term. Minimizing the MSE will result in a reasonable trade-off between bias and variance:

$$MSE\left(\hat{f}(x)\right) = \mathbb{V}\text{ar}\left(\hat{f}(x)\right) + \mathbb{B}\text{ias}\left(\hat{f}(x)\right)^2.$$

The term still depends on the location x. The Mean Integrated Squared Error (MISE) [33] is a global measure by integrating the MSE:

$$\begin{aligned} MISE\left(\hat{f}\right) &= \int_{-\infty}^{\infty} MSE(\hat{f}(x))\,\mathrm{d}x \\ &\to \frac{1}{nh}\left\|\mathcal{K}\right\|^2 + \frac{h^4}{4}\mu_2(\mathcal{K})^2\left\|f''\right\|^2. \end{aligned} \tag{4.4}$$

The complete proof of convergence can be found in appendix A. Incidentally, minimizing expression (4.4) for all non-negative kernels yields the Epanechnikov kernel which means it is asymptotically MISE optimal [15]. As a compromise of bias and variance, we choose h so that the asymptotic MISE (AMISE) is minimal:

$$\frac{\partial AMISE\left(\hat{f}\right)}{\partial h} = -\frac{1}{nh^2}\left\|\mathcal{K}\right\|^2 + h^3\mu_2(\mathcal{K})^2\left\|f''\right\|^2 \stackrel{!}{=} 0 \tag{4.5}$$

$$\Rightarrow h_{\text{opt}} = \left(\frac{\left\|\mathcal{K}\right\|^2}{n\mu_2(\mathcal{K})^2\left\|f''\right\|^2}\right)^{\frac{1}{5}}. \tag{4.6}$$

Inserting this bandwidth into the AMISE yields $n^{-\frac{4}{5}}$ as convergence order for kernel densities. A convergence order of $n^{-\frac{2}{3}}$ is obtained for histograms by a similar procedure. This is another reason to prefer kernel densities over histograms.

The optimal bandwidth h_{opt} decreases with an increasing sample size n. Furthermore the unknown $\left\|f''\right\|^2$ is a prerequisite for the calculation. Therefore, the optimal h can only be determined by assuming the distribution of the model. $\left\|f''\right\|^2$ indicates the non-smoothness of f, so smaller bandwidths are optimal for strongly fluctuating densities. For many distributions $\left\|f''\right\|^2$ can be calculated, such as for $\mathcal{N}\left(\mu,\sigma^2\right)$:

$$\begin{aligned}
\left\|f''\right\|^2 &= \int_{-\infty}^{\infty} f''(x)^2\,\mathrm{d}x \\
&\stackrel{\text{o.B.d.A. } \mu=0}{=} \int_{-\infty}^{\infty}\left(\left(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2}\right)\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right)\right)^2\mathrm{d}x \\
&\stackrel{u=\frac{x}{\sigma}}{=} \int_{-\infty}^{\infty}\left(\left(\frac{u^2-1}{\sigma^2}\right)\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{u^2}{2}\right)\right)^2\sigma\,\mathrm{d}u \\
&= \sigma^{-5}\int_{-\infty}^{\infty}\left(\left(u^2-1\right)\phi(u)\right)^2\mathrm{d}u \\
&= \sigma^{-5}\left\|\phi''\right\|^2 \\
&= \sigma^{-5}\frac{3}{8\sqrt{\pi}}.
\end{aligned}$$

The constants $\mu_2(\phi)^2$, $\left\|\phi\right\|^2$, $\left\|\phi''\right\|^2$ are tabulated for the standard normal distribution [14]. Together with the Gaussian kernel (4.2) this leads to Silverman's rule-of-thumb [40]:

$$\begin{aligned}
h_{\text{rot}} &= \left(\frac{\left\|\phi\right\|^2}{n\mu_2(\phi)^2\left\|f''\right\|^2}\right)^{\frac{1}{5}} \\
&= \left(\frac{\frac{1}{2\sqrt{\pi}}}{n\cdot 1\cdot\sigma^{-5}\frac{3}{8\sqrt{\pi}}}\right)^{\frac{1}{5}} \\
&\approx 1.06\cdot\hat{\sigma}n^{-\frac{1}{5}}.
\end{aligned}$$

The standard deviation is estimated as usual:

$$\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_i (x_i-\hat{\mu})^2}.$$

In practice, h_{rot} provides an appropriate bandwidth if the distribution does not deviate too much from the normal distribution. This is true for unimodal, approximately symmetrical distributions. If this assumption is inappropriate, the following robust estimator [40] can be used instead. This is known as the better rule-of-thumb:

$$\begin{aligned} h_{\text{brot}} &= 1.06 \cdot \min\left\{\hat{\sigma}, \frac{IQR}{2\Phi^{-1}(\frac{3}{4})}\right\} \cdot n^{-\frac{1}{5}} \\ &= 1.06 \cdot \min\left\{\hat{\sigma}, \frac{IQR}{1.34}\right\} \cdot n^{-\frac{1}{5}}. \end{aligned}$$

The interquartile range (IQR) is used to bound the estimation of the standard deviation. Using this bandwidth reduces the risk of over-smoothing if the target density is not unimodal. Still, the bandwidth is a good deal wider than the optimum for other distributions than the normal distribution. Silverman suggests to reduce the bandwidth even further to find a good estimate for a broader range of distributions. He chooses a reduced value for the factor 1.06 that loses no more than 10% efficiency on IMSE:

$$h'_{\text{brot}} = 0.9 \cdot \min\left\{\hat{\sigma}, \frac{IQR}{1.34}\right\} \cdot n^{-\frac{1}{5}}.$$

If relying on a parametric form is not acceptable, k-fold cross-validation provides a non-parametric option for bandwidth selection [13]. The bandwidth with the best average performance on the evaluation sets is selected from a number of multiple candidates $h_1, \ldots, h_m$. This method is more expensive by a factor in $\mathcal{O}(k \cdot m)$. Another approach is the plug-in method [39]. Instead of using $||f''||$ from a known distribution, it is estimated from the data to calculate the MISE-optimal bandwidth. The plug-in approach was proven to be computationally similarly complex as cross-validation [24]. Therefore, both methods are not practical for our application. Also, the estimated density of the particle energy in the IACT domain is expected to be unimodal, so the rule-of-thumb should work well enough.

4.2.2 Variable Bandwidth

It is possible to assign an individual bandwidth to each observation [44]. This can be used in cases where the location of some observations is more certain than that of others. The most popular use of variable bandwidth is to make the bandwidth dependent on the density at the location of training samples. Data points in regions where data is sparse will have flatter kernels. KDE with constant bandwidth results in random noise in regions of low density. According to the curse of dimensionality regions of low density are almost everywhere in high-dimensional space. This makes an approach with varying bandwidth particularly useful for estimation of high-dimensional densities. Using a flatter kernel results in a lower variance of the estimation in these regions.

A key property of DSEA is the aggregation of uncertainties, so the continuous density estimator should do this as well. KDE can easily be extended to take the uncertainties σ_u from the regression into account. The uncertainties are used to compute the bandwidths of the kernels at the corresponding positions. This results in higher peaks (lower bias), when uncertainty is low and less overfitting (lower variance), when uncertainty is high. We suggest an additive and a multiplicative approach to calculate the bandwidth:

$$h_{\text{u-add}}(x_i) = \rho \cdot h_{\text{brot}} + \lambda \dot{\sigma}_u(x_i) \tag{4.7}$$

$$h_{\text{u-mul}}(x_i) = \rho \cdot h_{\text{brot}} \cdot (1 + \lambda \cdot \sigma_u(x_i)) \,. \tag{4.8}$$

These should have a different impact on the estimation, but both approaches reduce to the rule-of-thumb when uncertainties are zero (if $\rho = 1$). The parameters ρ and λ can be used for fine-tuning the optimal bandwidth. Alternatively, one can use the range of the prediction interval of choice to replace the uncertainty of a prediction for further fine-tuning.

4.2.3 Weighted Estimation

Weighted Kernel Density Estimation incorporated weighted data to estimate the target density. This is another way to vary the influence of different observations on the result. To estimate a weighted kernel density, each kernel is multiplied with the weight w_i of the corresponding observation:

$$\hat{f}(x) = \sum_i w_i \mathcal{K}_h \left(x - x_i\right)$$

$$\sum w_i = 1.$$

The weights must sum up to 1 to ensure that the estimation is a probability density function. Note that using constant weights $w_i = \frac{1}{n}$ is equivalent to the standard unweighted kernel density estimation (4.1).

To embed the uncertainties of the regression methods we choose a weight inversely proportional to the uncertainty σ_i of an observation:

$$w_i = \frac{\sum_j \exp\left(-\sigma_j\right)}{\exp\left(-\sigma_i\right)}. \tag{4.9}$$

The normalization ensures that the weights sum up to 1. The exponential function helps making the weights more robust when dealing with a high range of uncertainties. Omitting the exponential function in (4.9) could lead to a very high weight that exceeds the sum of all other weights.

The altering of the influence of each observation is similar to variable bandwidth KDE. To see their difference, consider the case where an uncertainty is near zero. Choosing $w_i = 0$ in a weighted KDE is the same as removing the observation from the sample.

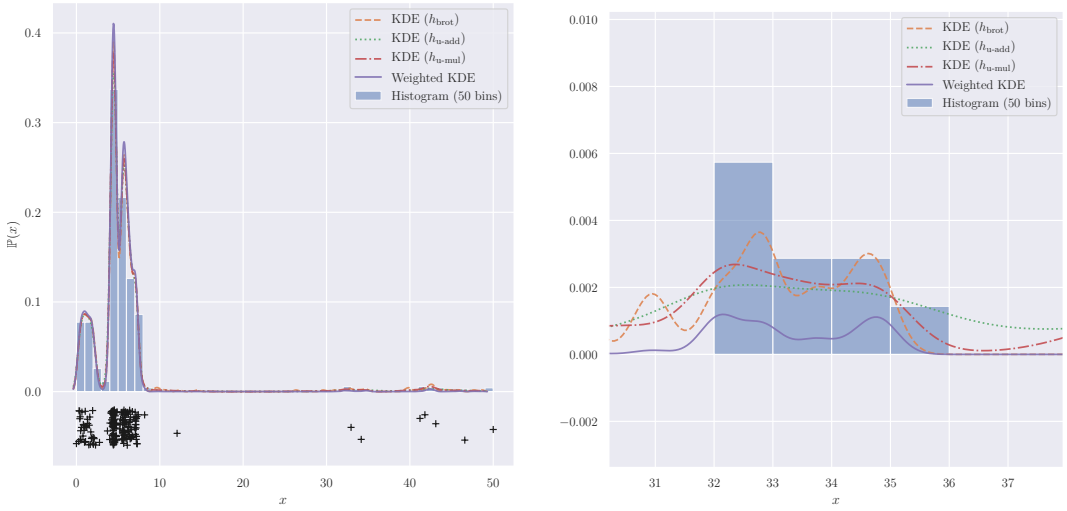


Figure 4.3: Comparison of kernel density estimation with different bandwidth strategies and weighted kernel density estimation. The task is to estimate a density from predictions (denoted by black crosses). All methods estimate similar densities when the uncertainty is low. After zooming to the interval $x \in [31, 37]$ where the uncertainty is high, the subtle differences between the strategies are evident. The better rule-of-thumb disregards the uncertainties and overfits to the stochastic predictions. The uncertainty bandwidths smooth out the area, while weighted KDE assigns this area a lower weight. The latter might not be desirable in the pursuit of a representative density. Also note that the histogram does not represent the predictions very well because it does not distinguish between the modes $x = 4.4$ and $x = 5.7$ despite choosing a large number of bins. For long tailed distributions, equidistant binning tends to lose information about values near zero.

Choosing h_i near infinity in a variable bandwidth KDE pushes the estimation towards the uniform distribution.

Figure 4.3 compares the different bandwidth strategies, namely the better rule-of-thumb, additive uncertainty and multiplicative uncertainty, where the latter two incorporate the uncertainties of the prediction. Also compared is the weighted KDE, where the weights are chosen inversely proportional to the uncertainties. It can be concluded that taking into account the uncertainties can improve the estimation of the target density. It can even be reasonable to use them both simultaneously.

Chapter 5

Continuous Spectrum Estimation Algorithm

The *Continuous Spectrum Estimation Algorithm* (CSEA) extends DSEA by the possibility of a continuous deconvolution. To achieve this, the classifier employed in DSEA is replaced by a regression method. This leads to the prediction of a real-valued target variable. To allow for iterative improvements of the estimation the regression method needs to support weighted training and the output of uncertainties of the predictions. In chapter 3 we found suitable regression methods, that satisfy these conditions.

To assemble a continuous density from the predictions CSEA also replaces the histogram used in DSEA by a continuous density estimator. Widespread adoption and theoretical superiority of kernel density estimation (see section 4.2) makes it the best candidate for the task. An important characteristic of DSEA is the aggregation of all conditional probabilities estimated by the classifier. To apply this idea in CSEA we explored ways to vary the contributions of each prediction in a kernel density estimation. We suggested to vary them depending on the uncertainty of the respective prediction.

In every iteration of CSEA the regression model is trained and the density estimator is used to estimate a continuous target density by aggregation of the regression output. As in DSEA, the updated estimation is then used to re-weight the training data for the next iteration. The modifications that have been made to optimally support the combination of regression and continuous density estimation are discussed in the following.

5.1 Re-weighting the Training Data

The training data is re-weighted in every iteration, so that its distribution matches the current estimate. This reduces the influence of the distribution of the training data which does not necessarily represent the distribution of the observation. To compute the new weight for each observation $(\boldsymbol{x}_i, y_i)$, DSEA$^+$ divides the estimated probability of y_i by the

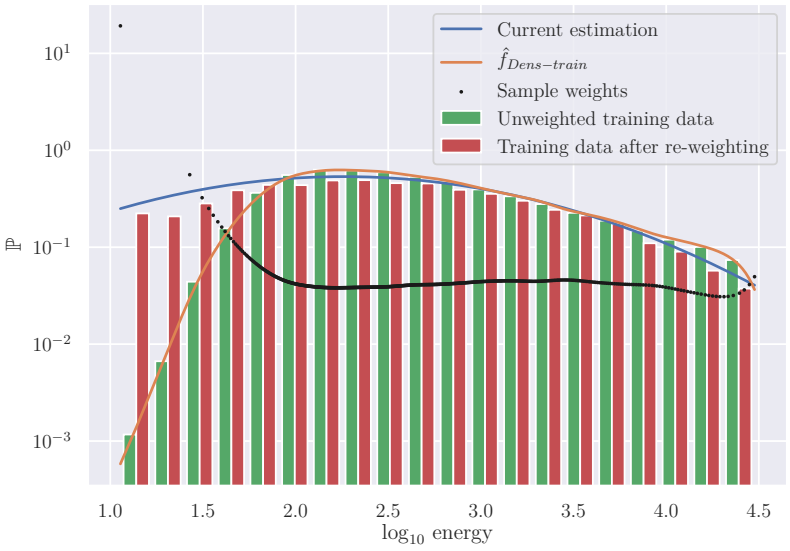


Figure 5.1: Re-weighting the training data in CSEA. In every iteration the training data is re-weighted so that its distribution matches the current estimation. This is achieved by dividing the estimate of the target density by the density estimation of the unweighted training data. After re-weighting, the weighted histogram of the training data matches the current estimation.

number of observations in the same bin. Afterwards, the total area of the bin represents the probability.

Specifying the number of elements in the same bin to be 1 in the continuous case would result in the same divergence problem as in the original DSEA formulation. To apply the idea above for the continuous estimate we need to make sure to divide by the weighted sum of elements at the location of y_i. This is exactly what kernel density estimators compute by weighting the samples depending on the distance to y_i using a kernel function. We simply employ a second instance of the density estimator to estimate the density of the (unweighted) training set. This is only done once at the beginning of the algorithm since the training set does not change.

To update the weights, the estimated probability for each sample, which itself is estimated using the density estimator on the predictions, is divided by the probability of the training data at the sample's location:

$$\boldsymbol{w}^{(k)} \leftarrow \frac{\hat{f}^{(k-1)}(\boldsymbol{y})}{\hat{f}_{\text{Dens-train}}(\boldsymbol{y})}.$$

This way, the distribution of the weighted training data matches the current estimate (see figure 5.1).

5.2 Adaptive Step Size

DSEA$^+$ adopts the adaptive step size from the RUN algorithm for DSEA. This step requires a discretization of the input space to compute the optimal step size via likelihood maximization. CSEA uses the same decision tree clustering as DSEA$^+$ to obtain the discretized training samples into J bins. In contrast to DSEA, CSEA needs to discretize the estimate in each iteration to pass it to the optimizer.

In each iteration the current estimate is discretized by querying the density estimator trained on the predictions at the locations of the bin centers. This and the previous discrete estimation are passed to the optimizer to compute the optimal step size:

$$\alpha_{\text{RUN}}^{(k)} = \underset{\alpha \geq 0}{\arg\min}\, \ell_r \left(\hat{\boldsymbol{g}}^{(k-1)} + \alpha \cdot \left(\hat{f}_{\text{Dens-obs}}^{(k)}(\text{bins}) - \hat{\boldsymbol{g}}^{(k-1)} \right) \right).$$

The discretized and the continuous estimations are both updated using $\alpha_{\text{RUN}}^{(k)}$:

$$\begin{aligned} \hat{\boldsymbol{g}}^{(k)} &= \hat{\boldsymbol{g}}^{(k-1)} + \alpha_{\text{RUN}}^{(k)} \cdot \left(\hat{f}_{\text{Dens-obs}}^{(k)}(\text{bins}) - \hat{\boldsymbol{g}}^{(k-1)} \right) \\ \hat{f}^{(k)} &= \hat{f}^{(k-1)} + \alpha_{\text{RUN}}^{(k)} \cdot \left(\hat{f}_{\text{Dens-obs}}^{(k)} - \hat{f}^{(k-1)} \right). \end{aligned}$$

Figure 5.2 visualized the update on both the discrete and the continuous estimates.

Because the step size was computed maximizing the discretized density, applying it to the continuous density can lead to negative probabilities, when the optimal step size is greater than 1. This can be avoided by bounding the search space to an interval of allowed step sizes for both density updates.

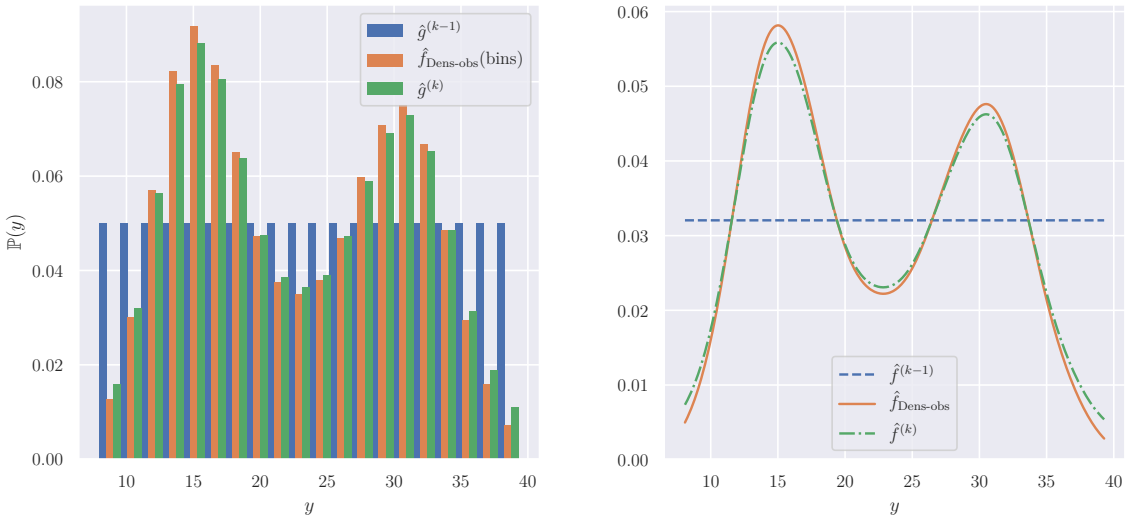


Figure 5.2: The adaptive step size in CSEA. The regularized likelihood is maximized on the discretized estimations. The optimal step size was computed as $\alpha_{\text{RUN}}^{(k)} = 0.91$ and is used to update both the discrete and the continuous estimates.

5.3 Stopping Criterion

DSEA uses the probabilistic symmetric χ^2-distance to quantify the progress in each iteration. When two consecutive solution do not differ enough, i.e. the distance does not exceed a predetermined ϵ, the algorithm is stopped. The χ^2-distance is derived from the Pearson's χ^2-test [31] that compares the distribution of two samples for equality. The distance is computed as the squared difference of the frequencies weighted by the reciprocal of their sum:

$$\chi^2(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \frac{(x_i - y_i)^2}{x_i + y_i}.$$

Summands in the χ^2-distance are usually defined zero, when the denominator is zero. Since DSEA uses histograms to represent the estimation, the χ^2 is an adequate measure to use in the stopping criterion. We explain in the following why it can still be used as a meaningful measure in CSEA despite dealing with continuous densities.

Let $\hat{\boldsymbol{g}}$ and $\boldsymbol{v}$ be the discrete estimation and the weights of the training set estimated by DSEA. The χ^2-distance in iteration k is

$$\begin{aligned}
\chi^2_{\text{Sym}}\left(\hat{\boldsymbol{g}}^{(k)}, \hat{\boldsymbol{g}}^{(k-1)}\right) &= \sum_i \frac{\left(\hat{g}_i^{(k)} - \hat{g}_i^{(k-1)}\right)^2}{\hat{g}_i^{(k)} + \hat{g}_i^{(k-1)}} \\
&= \sum_i \frac{\left(\sum_{j \in n(i)} \left(v_j^{(k+1)} - v_j^{(k)}\right)\right)^2}{\sum_{j \in n(i)} \left(v_j^{(k+1)} - v_j^{(k)}\right)} \\
&= \sum_i \frac{\left(g_i^t \left(v_{n(i)_1}^{(k+1)} - v_{n(i)_1}^{(k)}\right)\right)^2}{g_i^t \left(v_{n(i)_1}^{(k+1)} + v_{n(i)_1}^{(k)}\right)},
\end{aligned}$$

where $\boldsymbol{g}^t$ is the distribution of the unweighted training set and $n(i)$ yields the samples in bin i. The weights of these are all identical, i.e. $w_{n(i)_i} = w_{n(i)_1} \forall i$ and defined zero, when the bin is empty. Note, that this is consistent with the weight update in DSEA$^+$ (2.3). The χ^2-distance between consecutive iterations is therefore a weighted sum of the change in training weights $\boldsymbol{v}$ between the two iterations. This makes sense since the estimation does not update when the weights do not change (neglecting the randomness of the learned model). When the difference in weights is infinitesimally small, there is no need for more iterations.

Now we apply the above equation to the continuous estimation $\hat{f}$ and the weights $\boldsymbol{w}$ as estimated by CSEA. An example in bin i is simply the example $\boldsymbol{x}_i$ itself. Remember

that in CSEA the weights are computed as $\frac{\hat{f}^{(k-1)}(\boldsymbol{y})}{\hat{f}_{\text{Dens-train}}(\boldsymbol{y})}$ where $\hat{f}_{\text{Dens-train}}$, as estimated by the density estimator, is queried at the target values from the training set. This results in:

$$\begin{aligned}
&\sum_i \frac{\left(\hat{f}_{\text{Dens-train}}(y_i)\left(w_i^{(k+1)} - w_i^{(k)}\right)\right)^2}{\hat{f}_{\text{Dens-train}}(y_i)\left(w_i^{(k+1)} + w_i^{(k)}\right)} \\
=&\sum_i \frac{\left(\hat{f}_i^{(k)} - \hat{f}_i^{(k-1)}\right)^2}{\hat{f}_i^{(k)} + \hat{f}_i^{(k-1)}} \\
=&\chi^2_{\text{Sym}}\left(\hat{f}^{(k)}(\boldsymbol{y}), \hat{f}^{(k-1)}(\boldsymbol{y})\right).
\end{aligned}$$

It implies that we get an analogue stopping criterion when computing the distance between consecutive estimations evaluated at the training set's target values $\boldsymbol{y}$.

5.4 Distribution Calibration

Unfortunately, some regression methods underestimate the target value's range. This is because regression methods tend to avoid predicting extreme values. It is particularly visible in ensembles. When the prediction is the average of multiple outputs some outputs will pull the output towards the mean of the value range. Another factor is the missing extrapolation of some models. Most tree-based models return the mean of training targets associated with the predicted leaf. This will not allow for any predictions outside the range of the training data. The minimum and maximum values are only predicted when the relevant leaf contains only these values, otherwise the averaging will prevent the prediction of extreme values.

Since they provide a robust performance, random forests are heavily used in Cherenkov astronomy [43]. When used in regression analysis the averaging over multiple trees and the averaging in the leaves can both contribute to the problem of reducing the range of the estimation. The effect can amplify by repeating the estimation based on the previous iteration. Depending on the configuration a calibration of the estimate may be indispensable in order not to diverge from a good solution.

The problem of distribution calibration is well-known in classification, where the outputs of a classifier does not always give appropriate posterior probability estimations. There exist several correction techniques such as isotonic regression [48] and beta calibration [21]. In regression analysis, however, the problem is far less researched, since the focus is usually on the expectation of the predicted conditional distribution. Traditionally this problem has been covered by quantile regression, e.g. gradient boosting (see section 3.7). Recently there have been advances in calibrating the conditional distribution by applying beta calibration maps to its cumulative distribution function similar to beta calibration for classifiers. The existing techniques concentrate on the calibration of each prediction's

distribution where we are content with a calibration of the aggregated predictions, i.e. the distribution of predictions.

We suggest two methods for distribution calibration. The first is based on learning the moments, the second is more flexible by learning the quantiles.

5.4.1 Moment Calibration

The calibration based on moments is a very simple technique intended to provide a baseline. The model is trained and the predictions are accumulated. We use a 3-fold cross-validation on the data available for training. From the distribution of predictions, the moments are estimated. We use the first (the mean) and the centralized second (the variance). An optimal model should have a prediction distribution similar to the distribution of the training set. When it differs and the deviation has a pattern, it might be sufficient to transform the output distribution to bring its moments into line with the training distribution's moments. Let μ_{train}, σ^2_{train} and μ_{predict}, $\sigma^2_{\text{predict}}$ be the sample mean and the sample variance of the distribution of the training set and the predictions respectively and $\bar{\hat{y}}$ the sample mean of the predictions. Then the predictions can be calibrated as follows:

$$\hat{\boldsymbol{y}}_{\text{calibrated}} = \bar{\hat{y}} + \left(\hat{\boldsymbol{y}} - \bar{\hat{y}}\right) \cdot \underbrace{\sqrt{\frac{\sigma^2_{\text{train}}}{\sigma^2_{\text{predict}}}}}_{scale} + \underbrace{\mu_{\text{train}} - \mu_{\text{predict}}}_{shift} \cdot$$

On average the distribution of the calibrated predictions will share the mean and the variance with the training set (if the samples originate from the same population). Once *shift* and *scale* are learned, the technique can also be applied to calibrate the predictions on data of unknown distributions, assuming that the model distorts the moments similarly.

The method was evaluated on a skewed data set using a Gaussian process. Although Gaussian processes do not share the problems of a reduced range mentioned above it still falls back to the prior mean when queried at sparse locations, like many other regression techniques. The quantiles of the predictions before and after calibration are compared with the quantiles of the true target values y in a Q-Q plot (figure 5.3). The distribution of predictions before and after calibration are compared with the true distribution in figure 5.4. It shows that calibration can be valuable when the prediction distribution is supposed to represent the distribution of the observed data, as in CSEA.

The proposed method provides an adequate calibration for many configurations. But it did not find a good calibration of the skewed distribution's left tail, because only the first two moments were considered here. The technique can be extended by learning higher moments to allow for a more delicate calibration of skewed distributions. Instead, we explore a more flexible approach, that remaps the quantiles.

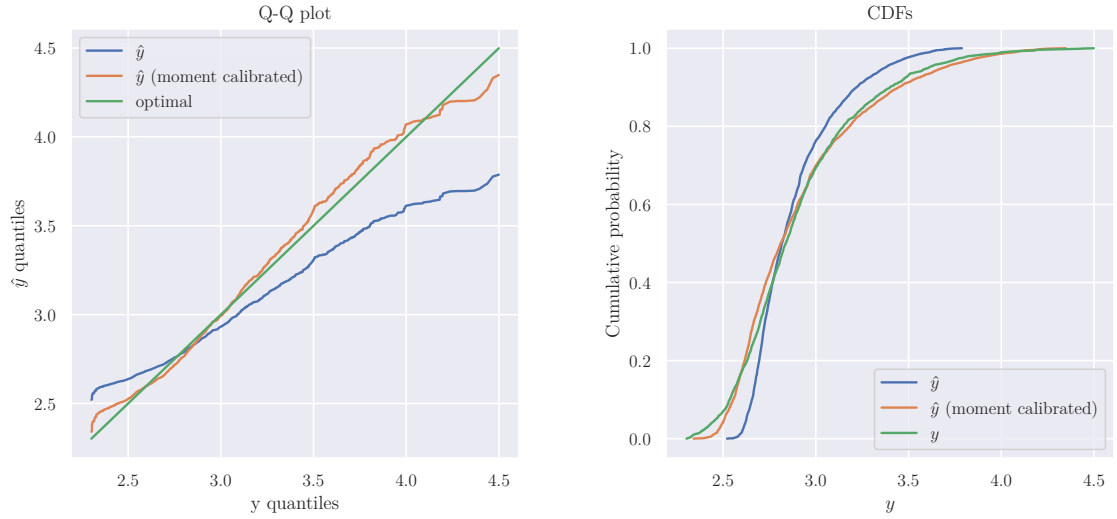


Figure 5.3: Q-Q plot of the predictions before and after moment calibration (left). The moment calibration transforms the predictions by learning the intrinsic moment distortion of a model. This linear transformation moves the quantiles towards the diagonal by multiplying each prediction's distance to the mean with $scale = 1.6$ and adding $shift = 0.03$. In the right panel the cumulative distribution function (CDF) of the moment calibrated predictions matches the empirical CDF of the training set much better.

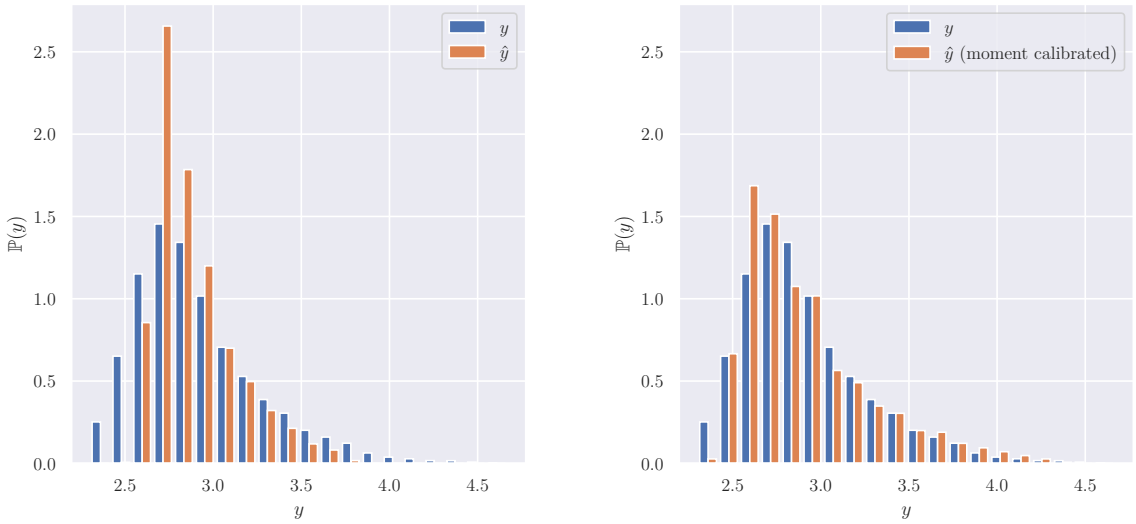


Figure 5.4: Histograms showing the effect of moment calibration. The histogram on the left shows the distribution of predictions before calibration. They clearly differ because the regression method employed has an intrinsic bias towards the center of the distribution. The histogram on the right shows the distribution after calibration. While the technique improves the estimation, the calibration does not differ between the tails of skewed distributions.

5.4.2 Quantile Calibration

The quantile calibration is very similar to [41], but instead of calibrating the conditional distributions the focus is on the calibration of the prediction distribution. First, we compare the CDFs of the training set and the predictions in a P-P plot (figure 5.5). By having a closed form of this map, the predictions can be calibrated by remapping its quantiles. To get such a map, fitting the CDF of the beta distribution is a popular choice, since both are confined to the interval $[0, 1]$ on each axis with fixed points $(0, 0)$ and $(1, 1)$. The beta distribution was not flexible enough for our use case, especially in the tails, so we suggest to fit a Gaussian process instead.

The Gaussian process can now transform any quantile to the corresponding quantile of the training set if it were distributed according to the prediction distribution. This is important for predicting distributions that differ from the training distribution. If the Gaussian process was instead fitted to the Q-Q plot, it would map to absolute values. We apply the quantile calibration to the same setup as in the experiment above (figure 5.6). Note, that the quantile calibration cannot calibrate the range of the prediction distribution since the Gaussian process maps only to existing quantiles in the predictions. A combination of both the moment and the quantile calibration may be beneficial in some cases.

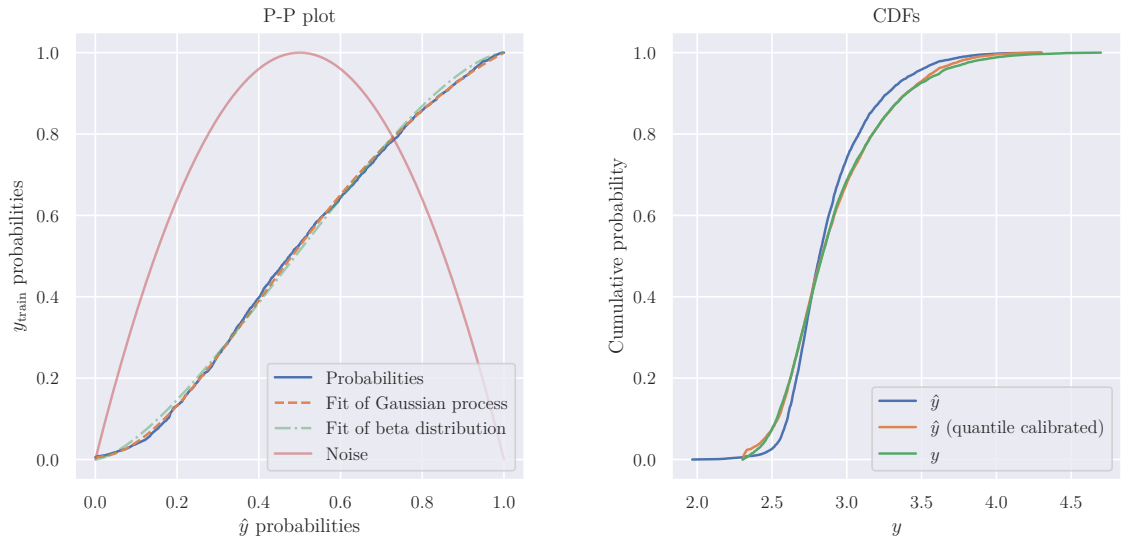


Figure 5.5: P-P plot of the prediction probabilities against the training set probabilities. A Gaussian process is fit to the curve and provides a better estimate than the beta distribution. A varying noise level, modeled with the PDF of a $Beta(2, 2)$ distribution, was used in the fitting of the Gaussian process to favor a good fit in the tails, since the tails of the distribution are the main reason the calibration is necessary. The right panel shows that the CDFs align very well.

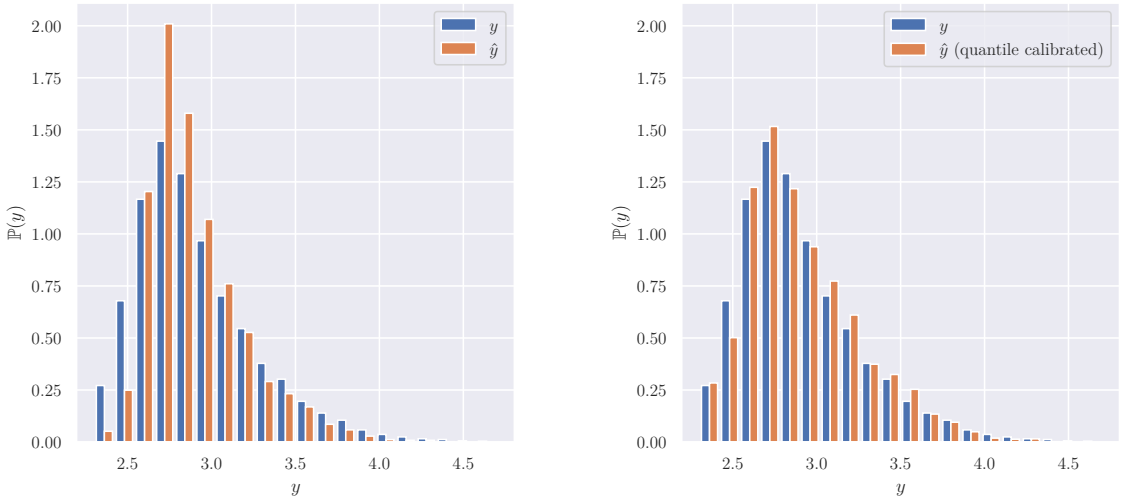


Figure 5.6: Histograms showing the effect of quantile calibration. The histogram on the left shows the distribution of predictions before calibration. The histogram on the right shows that the distribution after calibration matches the true distribution very well.

5.5 The Algorithm

Putting everything together, CSEA (algorithm 2) is obtained: an algorithm for continuous deconvolution of probability density functions. The implementation is publicly available as a Python package[1]. The algorithm is described briefly.

First, the density $\hat{f}_{\text{Dens-train}}$ of the training data is estimated using a density estimator of choice. The prior distribution $\hat{f}^{(0)}$, which defaults to the continuous uniform distribution, is discretized by querying it at the bin centers to get the initial $\hat{\boldsymbol{g}}^{(0)}$. Then, in every iteration k the weights $\boldsymbol{w}^{(k)}$ are updated such that the weighted training data resembles the current estimation. The regression model of choice is trained on the training data and predicts target values $\hat{\boldsymbol{y}}^{(k)}$ and associated uncertainties $\hat{\boldsymbol{\sigma}}^{(k)}$ for all samples in the observed data. These predictions are calibrated as needed. The density estimator aggregated the predictions and uncertainties to provide an estimation for the unfolded distribution $\hat{f}^{(k)}_{\text{Dens-obs}}$. It is discretized again by querying it at the bin centers. This and the previous discretization are passed to the optimizer to compute the optimal step size $\alpha^{(k)}_{\text{RUN}}$ by minimizing the regularized likelihood. The discretized and the continuous estimations are updated in search direction by the optimal step size. This procedure is repeated until the χ^2-distance between two consecutive estimations falls below the predefined ϵ. The last estimation is returned.

[1] `https://bitbucket.org/maik-schmidt/csea`

Algorithm 2 CSEA: Continuous Spectrum Estimation Algorithm

Input:

Observed data set $\mathcal{D}_{\text{obs}} = \{\boldsymbol{x}_n \in \mathcal{X} \,|\, 1 \leq n \leq N\}$

Training data set $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_n, y_n) \in \mathcal{X} \times \mathbb{R} \,|\, 1 \leq n \leq N'\}$

Discretization bins $(b_1, \ldots, b_J)^T \in \mathbb{R}^J$

Minimal χ^2 distance $\epsilon > 0$ between iterations (default: 10^{-6})

Prior density $\hat{f}^{(0)} : \mathcal{X} \to \mathbb{R}$ (default: $\hat{f}^{(0)}(y_i) = \frac{1}{\max_i y_i - \min_i y_i} \, \forall 1 \leq i \leq N$)

Output:

Estimated target density $\hat{f} : \mathcal{X} \to \mathbb{R}$

$\hat{f}_{\text{Dens-train}} \leftarrow \text{DensityEstimator}(\mathcal{D}_{\text{train}})$

$\hat{\boldsymbol{g}}^{(0)} \leftarrow \hat{f}^{(0)}(\text{bins})$

$k \leftarrow 0$

repeat

  $k \leftarrow k + 1$

  $\boldsymbol{w}^{(k)} \leftarrow \frac{\hat{f}^{(k-1)}(\boldsymbol{y})}{\hat{f}_{\text{Dens-train}}(\boldsymbol{y})}$

  Infer $\mathcal{M}$ from $\mathcal{D}_{\text{train}}$ weighted by $\boldsymbol{w}^{(k)}$

  $\hat{\boldsymbol{y}}^{(k)}, \hat{\boldsymbol{\sigma}}^{(k)} \leftarrow \text{predict}_{\mathcal{M}}(\mathcal{D}_{\text{obs}})$

  Calibrate $\hat{\boldsymbol{y}}^{(k)}$ (optional)

  $\hat{f}^{(k)}_{\text{Dens-obs}} \leftarrow \text{DensityEstimator}(\hat{\boldsymbol{y}}^{(k)}, \hat{\boldsymbol{\sigma}}^{(k)})$

  $\alpha^{(k)}_{\text{RUN}} \leftarrow \arg\min_{\alpha \geq 0} \ell_r \left(\hat{\boldsymbol{g}}^{(k-1)} + \alpha \cdot \left(\hat{f}^{(k)}_{\text{Dens-obs}}(\text{bins}) - \hat{\boldsymbol{g}}^{(k-1)} \right) \right)$

  $\hat{\boldsymbol{g}}^{(k)} \leftarrow \hat{\boldsymbol{g}}^{(k-1)} + \alpha^{(k)}_{\text{RUN}} \cdot \left(\hat{f}^{(k)}_{\text{Dens-obs}}(\text{bins}) - \hat{\boldsymbol{g}}^{(k-1)} \right)$

  $\hat{f}^{(k)} \leftarrow \hat{f}^{(k-1)} + \alpha^{(k)}_{\text{RUN}} \cdot \left(\hat{f}^{(k)}_{\text{Dens-obs}} - \hat{f}^{(k-1)} \right)$

until $\chi^2_{\text{Sym}} \left(\hat{f}^{(k)}(\boldsymbol{y}), \hat{f}^{(k-1)}(\boldsymbol{y}) \right) \leq \epsilon$

return $\hat{f} \leftarrow \hat{f}^{(k)}$

Chapter 6

Evaluation

In addition to the parameters inherited from DSEA, such as step size and number of bins, additional parameters emerge for CSEA, such as the bandwidth for the kernel density estimator and the selection of a regression method and its parameters. Since the old parameters have already been studied in detail in [7], the evaluation in this work concentrates on the new parameters. The results of the numerous configurations are publicly available in a separate repository[1] for the experiments. All experiments can be conveniently navigated via a graphical user interface. The repository can also be used to evaluate user-defined configurations.

The experiments are rerun 20 times to get rough estimations of the lower and upper 5%-percentiles. Ordinary 20-fold cross-validation would partition the data into 20 sets and then use 19 sets for training and evaluate on only one set at a time. This severely lowers the size of $\mathcal{D}_{\text{obs}}$. Instead, we use a repeated 3-fold cross-validation that shuffles the data after each repetition. The preliminary experiments to optimize the hyperparameters are only repeated three times resulting in rough estimates for the quartiles. The training and observed data are sampled randomly as state-of-the-art performance when using inappropriately distributed training data was already shown for DSEA in [7].

All diagrams use a linear probability scale due to the low probabilities at the ends of most continuous densities. This also makes the comparison between the distributions more intuitive. First, we discuss the data sets and distance measurements used to evaluate the deconvolution result of each configuration.

6.1 Data Sets

For the evaluation three different data sets are used. The first is synthetic and the other two are IACT data sets associated with the FACT and the MAGIC telescope systems.

[1] `https://bitbucket.org/maik-schmidt/csea-exp`

These are the same data sets that are used in [7] which will make a comparison to already existing results easier.

The toy data set consists of ten input variables that correlate with the target variable which was generated from a mixture of the two equally weighted Gaussians $\mathcal{N}(15, 3)$ and $\mathcal{N}(30, 4)$. Synthetic data has the advantage that the true distribution is known, which is useful for the evaluation.

The IACT data sets contain simulated data for the telescope systems. MAGIC consists of two telescopes yielding a much higher surface than the relatively small FACT telescope. This translates to a much wider range of measured γ particle energy. Extracting the Hillas parameters [19] of the observations is often used in Cherenkov astronomy. These describe characteristics such as the orientation and size of a shower and form the input variables from which the particle energy is predicted. The energy ranges over several orders of magnitude and is $\log_{10}$ transformed to make its highly skewed distribution less skewed. The small energy range of the FACT telescope still results in a distribution with a very high slope on the lower side of the spectrum. This makes the estimation of the distribution especially difficult.

6.2 Distance Measures

To evaluate the methods, the distance of the deconvolution results to the true distributions is measured. We avoid binning the distribution for comparison as this would lose information. Because the PDFs of the true distributions are unavailable (except for the toy data set), the empirical CDFs of the observed data and the estimation are used. This limits the choice of a distance measure to measures that operate on CDFs.

A popular dissimilarity measure on distributions is the Kolmogorov Smirnov (KS) [25] distance. It is easily computed as the maximum vertical distance between two CDFs:

$$D(F_1, F_2) = \sup_x |F_1(x) - F_2(x)|.$$

The KS distance only takes into account the maximal deviation. One advantage of the KS distance is that it is always bounded by 1, regardless of the value range. We will use the KS distance to compare DSEA and CSEA as the computation is the same for discrete and continuous densities. This helps DSEA as the KS distance is only affected by the discretization error of a single bin instead of summing the discretization errors of all bins.

If we consider two distributions as piles of earth, then the Earth Mover's Distance (EMD) [34] computes the minimum total distance the earth must be moves to transform one distribution into the other. Given two continuous probability distributions, the EMD is

most easily understood as the total area between their CDFs F_1 and F_2. This is equivalent to the Wasserstein metric [12]

$$\begin{aligned} W(\boldsymbol{X}, \boldsymbol{Y}) &= \inf \mathbb{E}(|\boldsymbol{X} - \boldsymbol{Y}|) \\ &= \int_{-\infty}^{\infty} |F_1(x) - F_2(x)| \, \mathrm{d}x, \end{aligned}$$

which is more commonly referred to when $\boldsymbol{X}$ and $\boldsymbol{Y}$ are continuous random variables. Numerically it can be computed by evaluating the quantile functions [18]:

$$EMD(F_1, F_2) = \int_0^1 |F_1^{-1}(x) - F_2^{-1}(x)| \, \mathrm{d}x.$$

While more expensive to compute, the EMD catches even small changes in the distributions. For all cases where different CSEA results are compared to each other, the EMD is used. But keep in mind that it will be naturally higher on the toy data set because the range of values is at least ten times higher.

6.3 Baseline Model

For the experiments a baseline model is deployed that provides acceptable estimations on the different data sets without tuning any parameters. Its configuration is specified in the following:

- **Regression model:** A Mondrian forest assembled from 30 Mondrian trees. This model was chosen due to its similarity to random forests, which are heavily used in Cherenkov astronomy, but it also provides uncertainties about the predictions. In addition, tree-based models scale well with large amounts of data.
- **Density estimator:** A kernel density estimator with Epanechnikov kernel and bandwidth calculated using the better rule-of-thumb.
- **Step size:** The adaptive step size. A discretized representation of the estimation is maintained to minimize the regularized likelihood from the RUN algorithm.
- **Calibration:** None. The baseline model uses no calibration as it is only needed in some configurations.
- $\boldsymbol{K}$**:** 5. The maximum number of iterations.
- $\boldsymbol{\epsilon}$**:** 0. The convergence criterion is disabled for the experiments to analyze the behavior of CSEA when the criterion is not chosen adequately.
- **Number of bins:** 20. The number of bins is used in the discretization of the input space, the histogram estimator and for the comparison with DSEA.

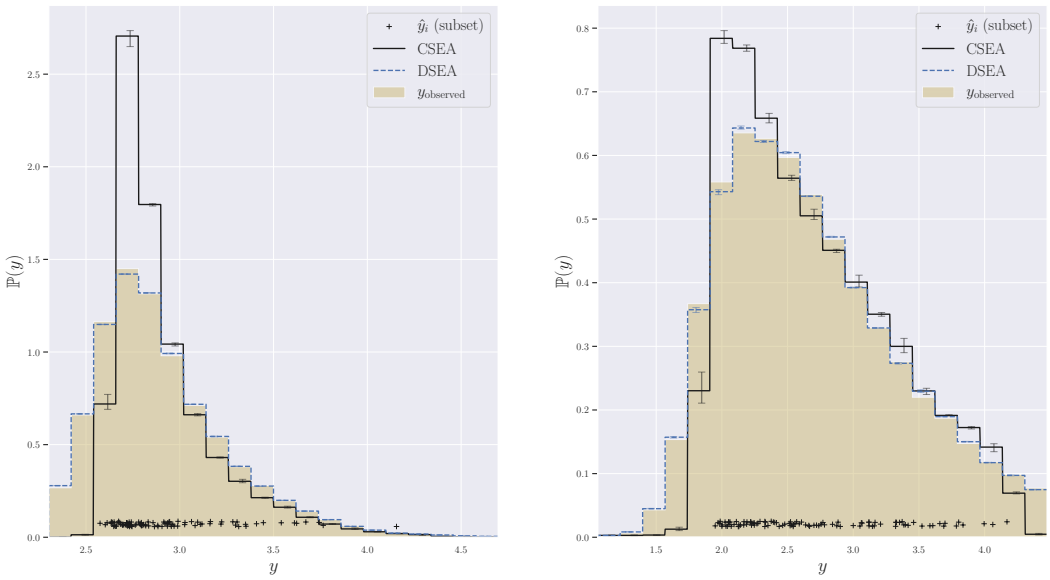
- **Maximum data set size:** 10^5. This limitation is solely for allowing multiple runs of different configurations in a reasonable time.

Unless otherwise stated, the values of the base model are used. For comparison, DSEA is called with the same parameters except a random forest instead of the Mondrian forest and histogram estimation instead of KDE. The improved version of DSEA, DSEA$^+$ (see chapter 2.3.2), is used in all experiments.

6.4 Evaluation of Density Estimators

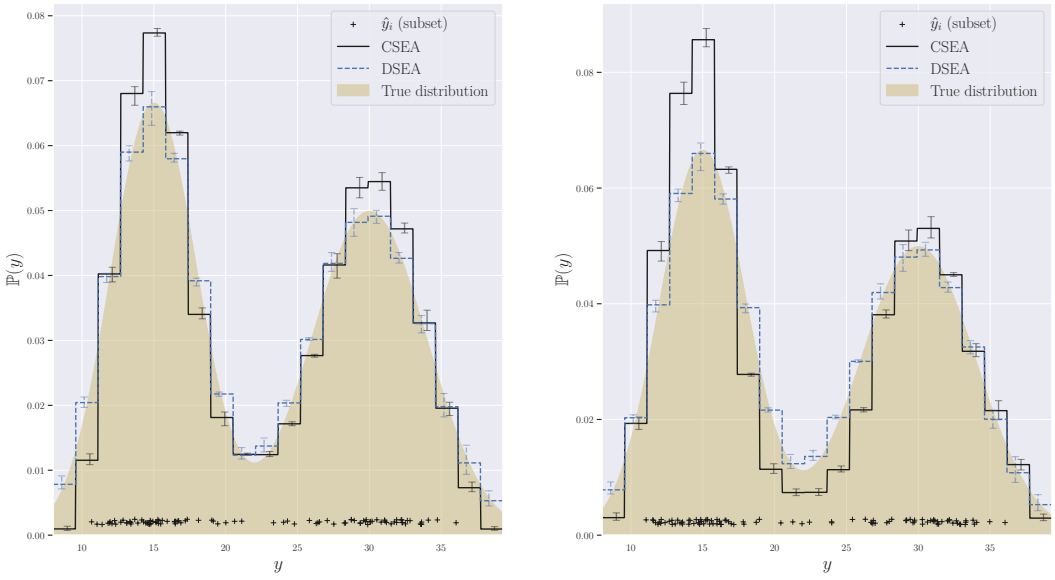
6.4.1 Histogram

A histogram estimator is employed to compare the deconvolution results of CSEA with those of DSEA. As shown in figures 6.1 and 5.4 the solutions found by DSEA fit much better to the observed data. Nevertheless, these results were sort of expected since DSEA is optimized for the estimation of histograms. Still, this shows that the classifier embedded in DSEA cannot simply replaced by a regression method without degrading the deconvolution



(a) FACT **(b)** MAGIC

Figure 6.1: Histogram estimation on FACT (left) and MAGIC (right) data sets. The predictions by the regression method employed in CSEA are denoted by black crosses. It is evident that DSEA finds a much better solution on both data sets when using a histogram estimator. Also note that the solutions found by DSEA are significantly more consistent by hardly deviating from each other. CSEA with a histogram estimator is not able to produce acceptable fits. It turns out that embedding a regression method alone is not sufficient to achieve results on a par with DSEA.



(a) Unweighted histogram **(b)** Weighted histogram

Figure 6.2: Unweighted (left) and weighted (right) histogram estimation on the toy data set. The weighting of the predictions leads to a lower probability estimation in sparse regions. This, in turn, lets the estimation overshoot the peaks of the distribution since the density must sum up to 1. We conclude that weighting the predictions based on uncertainties does not have the effect we intended.

results. In order to maintain the quality of the estimations, the histogram needs to be replaced as well.

The estimation of weighted histograms based on the uncertainty of the predictions does not differ from the unweighted case, except on the toy data set. The weighting based on uncertainty leads to reduced probability estimation in regions of high uncertainty. Since uncertainty scales with the data available for training, sparse regions have naturally higher uncertainty than dense regions. This of course means that the weights in dense regions increase. The weighed density estimation therefore overshoots the peaks of the distribution. This is evident in figure 6.2. As already presumed for the weighted kernel density estimation that this behavior might not be desirable in the pursuit of a representative density (see section 4.2.3), we conclude that weighted density estimation is not the right approach for our scenario. The Kolmogorov distances to the empirical CDFs of the observed data are presented in table 6.1.

It was shown that CSEA does not match the results of DSEA when employing a histogram estimator. As a result, the kernel density estimator is used. Since KDE has a highly influential parameter in the kernel bandwidth, we evaluate a number of configurations first. At the end of the chapter the best performing configuration is compared to DSEA again.

Algorithm	Toy data	FACT	MAGIC
DSEA	**0.051**	**0.086**	**0.055**
CSEA (unweighted)	0.067	0.249	0.099
CSEA (weighted)	0.114	-	-

Table 6.1: Kolmogorov distances to the true CDFs for DSEA and CSEA with a histogram estimator. DSEA performs much better on the IACT data sets. Also shown is that estimating a weighted histogram leads to considerably worse deconvolution results.

6.4.2 Kernel Density Estimation

In order to maximize the use of the kernel density estimator, an appropriate bandwidth must be selected. Silverman's rule-of-thumb aims to solve this problem by automatically selecting a bandwidth based on the data. It turns out that this is not the universal remedy we had hoped for. Instead, the quality of the deconvolution results varies greatly not only between the different data sets, but also between the regression methods used. We opt for a grid-search over a number of bandwidths and the available regression methods to find the top performer for each configuration. The candidates for the bandwidth are still based on the rule-of-thumb due to its theoretical foundation. The preceding factor in Silverman's rule-of-thumb has high impact on the deconvolution result and should be chosen carefully. The optimal factor for normal distributions is 1.06, while Silverman suggests to use 0.9 to find a good estimate for a broader range of distributions. Some even reduce the factor to as low as 0.5. These form the candidates for the bandwidth.

We evaluate the baseline model with different regression methods to find the best factor for each data set and method. The quality of each configuration is assessed via the Earth Mover's distance between the estimation in the final iteration and the true distribution. The results are presented in table 6.2. It is shown that in some configurations the bandwidth factor has an enormous impact on the estimate, whereas in others it has almost no impact. We come to the conclusion that the bandwidth should always be chosen depending on the data set and the regression method used. For the remaining chapter, we imply that the best bandwidth from table 6.2 is used when referring to a configuration.

Note that a high dependency on a properly selected bandwidth is problematic since the bandwidth can only be optimized for simulated data in the IACT domain. Unfortunately, the optimality of a bandwidth is not necessarily transferable to real measurements of the telescopes.

Model	Factor	Toy data	FACT	MAGIC
Mondrian forest	0.5	0.383	**0.019**	**0.018**
	0.9	0.367	0.089	0.040
	1.06	**0.363**	0.125	0.058
Random forest	0.5	0.255	0.063	0.038
	0.9	0.247	0.0629	0.0378
	1.06	**0.242**	**0.0627**	**0.0376**
Kernel ridge	0.5	0.339	**0.037**	**0.014**
	0.9	0.306	0.166	0.074
	1.06	**0.289**	0.221	0.103
Gaussian process	0.5	**0.312**	**0.208**	**0.067**
	0.9	0.321	0.285	0.104
	1.06	0.327	0.320	0.126

Table 6.2: EMD after the last iteration using KDE with different bandwidths. The IACT data sets generally demand a lower bandwidth than the toy data set. Remember that the EMD is naturally higher on the toy data set because of the higher range of values.

6.5 Evaluation of Regression Methods

We evaluate the influence of the chosen regression methods on the deconvolution result. The evaluated methods are Mondrian forests, random forests, kernel ridge regression and Gaussian processes. Due to the necessary matrix inversion we limit the maximum data set size to 20000 for kernel ridge and Gaussian process regression.

The efficient implementations of kernel ridge regressions, Gaussian processes and random forests are used from *scikit-learn*, while *scikit-garden* is used for the implementation of Mondrian forests. All default parameters are used, with the exception of the following changes:

- The number of estimators in random forests and Mondrian forests is set to 30.
- The kernel ridge regularization parameter is set to 0.1, the kernel is specified as Gaussian.
- The Gaussian process normalizes the target values to fall back to the mean instead of zero in regions where data is sparse.

We use our own implementations of Histograms and KDE that allow for a weighted estimation and a variable bandwidth.

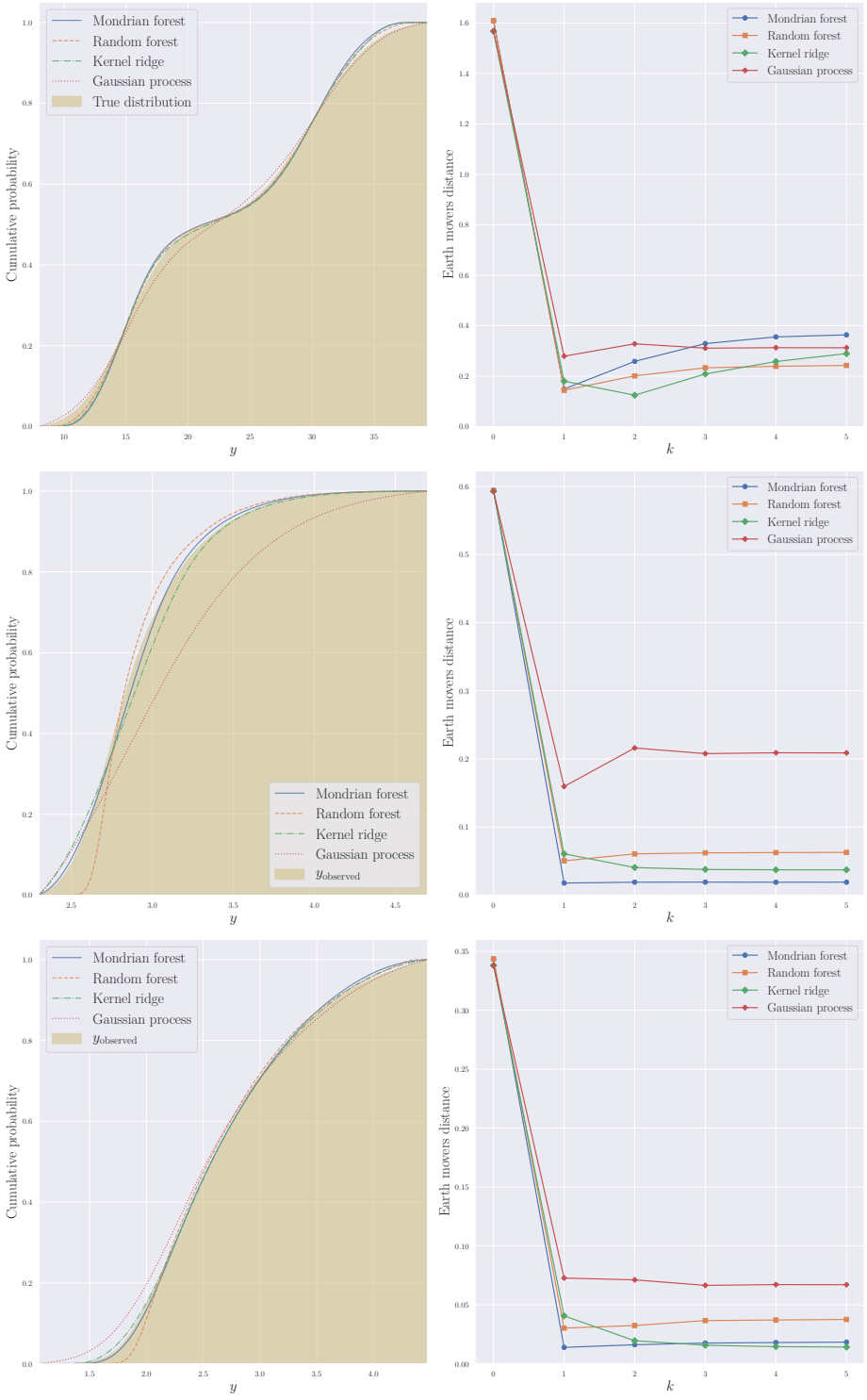


Figure 6.3: Evaluation of CSEA on the toy data, FACT and MAGIC. The estimations are averaged across all repetitions. The CDFs are shown on the left, while the EMDs are shown on the right. Error bars are omitted for clarity. Random forests, Mondrian forests and kernel ridge regression achieve the best results on the toy data, FACT and MAGIC respectively. The divergence on the toy data set is addressed in the following section.

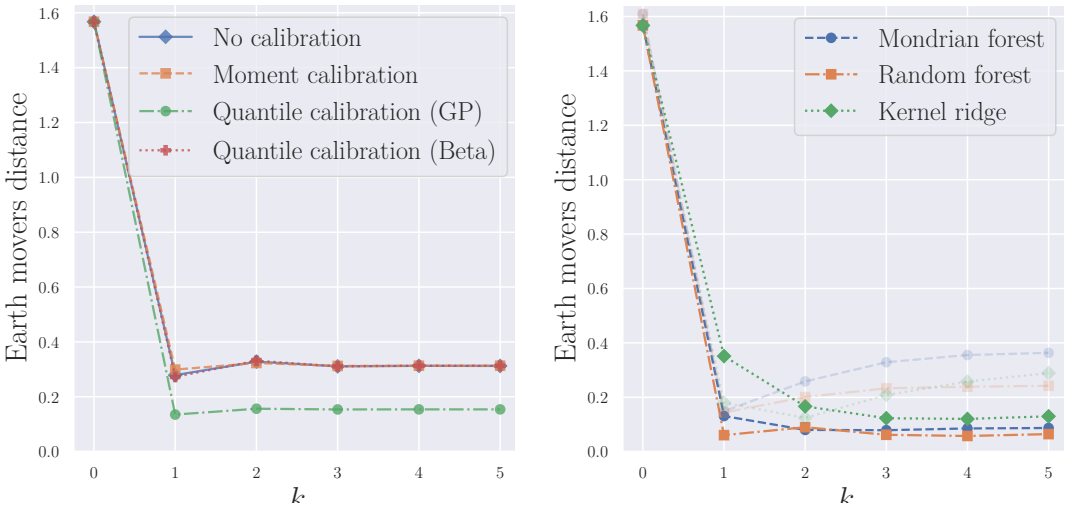
For every regression methods and data set, the CDF of the fit is showcased in figure 6.3. We use the CDFs for comparison because the true continuous PDFs of the IACT data sets are unknown, while the CDFs can be easily estimated by the empirical CDFs. On the toy data neither regression method finds a satisfying estimation. The estimations even degrade for all methods but the Gaussian process after having found an appropriate solution. This effect is concerning and will be addressed in the following section. On the IACT data sets kernel ridge regression and the Mondrian forest deliver appealing solutions. The Mondrian forest dominates the FACT data set, while the kernel ridge regression marginally outperforms the Mondrian forest model on the MAGIC data set. The good performance of the kernel ridge model is surprising since it is only trained on 20% of the data available for the tree-based models.

6.6 Evaluation of Distribution Calibrations

It has been observed that the estimates can degrade after having found an appropriate solution. We explaine this with the intrinsic prediction bias of some regression methods. When uncertainty is high, they favor predictions near the mean. As a consequence, the prediction distribution has a lower variance than desired. In the experiments above, this effect was observed on the toy data set on almost all models. This may be explained by the bimodality of the toy data set. Falling back to the mean of bimodal distributions imposes a heavier penalty on the quality of the estimations than for unimodal distributions. The use of a distribution calibration was suggested to restore the undistorted shape of the distribution. We compare the performance of CSEA when using no calibration, the moment calibration (see section 5.4.1) and the quantile calibration fitting either a Beta distribution or a Gaussian process (see section 5.4.2).

The suggested calibration techniques are evaluated on the toy data set for the Gaussian process as it was already shown that it can benefit from a calibration. The results are shown in figure 6.4 (a). The quantile calibration that fits a Gaussian process to the P-P plot obtains the best deconvolution result. The other methods do not improve the estimation significantly. We evaluate the quantile calibration when using the other models in figure 6.4 (b). On the toy data set, quantile calibration prevents divergence on all models. This is despite training the calibrated tree-based models on only 20% of the data available for the uncalibrated models. This limitation confines the size of the kernel matrix as the Gaussian process is invoked for every sample in the observed data set. This aspect could be improved in the future by choosing a more efficient model to compute the quantile map without sacrificing the expressiveness of a Gaussian process.

Calibration did not reliably improve the deconvolution result of the best configurations on the IACT data sets as these are already well calibrated. It even impaires the estimates in some cases. Since the quantile calibration can lead to non-smoothness of the estimation



(a) Gaussian process **(b)** Quantile calibration (GP)

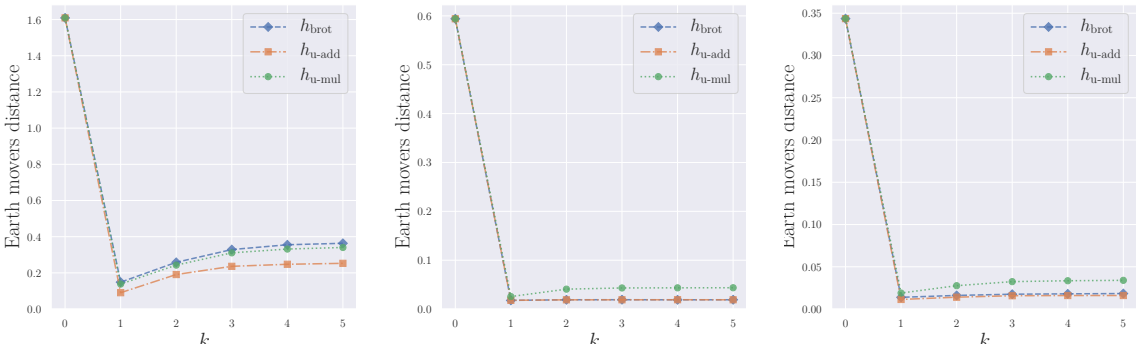
Figure 6.4: EMD of uncalibrated and calibrated deconvolutions on the toy data set. The distribution estimated by a Gaussian process is calibrated using the moment calibration and each of the quantile calibrations (left). Only the quantile calibration, that uses itself a Gaussian process to compute the quantile map, is able to improve the estimate. The quantile calibration is applied to the remaining models and prevents the divergence of all models (right). The EMDs of the uncalibrated models are illustrated partially transparent.

and involves additional computational cost we discourage its use when a calibration is not necessary.

6.7 Evaluation of Variable Bandwidths

We evaluate the variable bandwidth KDE by embedding the uncertainties of the predictions. In theory, this should improve the result by flattening the contribution of uncertain predictions. This assumes that the regression method returns uncertainties that are appropriate for the use in the bandwidth. The best models from the previous sections, namely Mondrian forests and kernel ridge regression, are evaluated using the bandwidths $h_{\text{u-add}}$ and $h_{\text{u-mul}}$ as defined in (4.7) with $\rho = 0.5$ and $\lambda = 1$. Remember that the optimal bandwidth factor for each model is chosen according to table 6.2. Random forests are omitted because they do not return uncertainties.

The results for Mondrian forests are shown in figure 6.5. The additive bandwidth improves the estimation on all data sets. It helps them to surpass the kernel ridge regression on the MAGIC data set. This makes Mondrian forests with the additive bandwidth the best model on both IACT data sets. The multiplicative bandwidth does not work well on the IACT data sets.



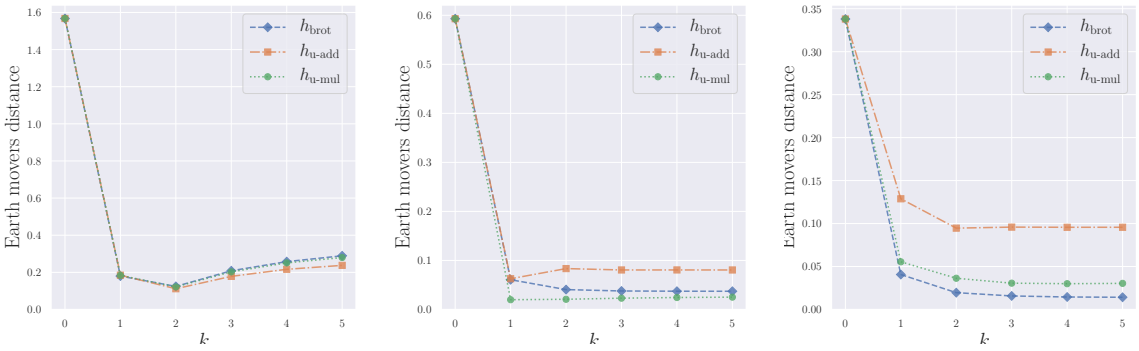
(a) Toy data **(b)** FACT **(c)** MAGIC

Figure 6.5: EMD of Mondrian forest regression with constant and variable bandwidths over the iterations. The additive bandwidth embedding the uncertainties slightly improves the deconvolution results on all data sets. This produces the best results on both IACT data sets so far.

Kernel ridge regression itself uses a density estimator to estimate the uncertainties (see 3.3.1), though this one is multi-dimensional as it estimates the density of the input domain. After the curse of dimensionality, the input domain is assigned near zero probability almost everywhere. Choosing the Epanechnikov kernel amplifies this problem as it has finite support. For this reason, we choose the Gaussian kernel for the kernel ridge density estimator. As bandwidth the multi-dimensional generalization of the rule-of-thumb

$$h_{\text{d-rot}} = \frac{4}{2 \cdot d + 1}^{\frac{1}{d+4}} \cdot \min\left\{\hat{\sigma}, \frac{IQR}{1.34}\right\} \cdot n^{-\frac{1}{d+4}}$$

is employed where d denotes the dimensionality and $\hat{\sigma}$ and IQR are averaged over all dimensions. This density estimator still estimates arbitrary low probabilities which result



(a) Toy data **(b)** FACT **(c)** MAGIC

Figure 6.6: EMD of kernel ridge regression with constant and variable bandwidths over the iterations. Neither of the variable bandwidths reliably improves the deconvolution results. This might be due to the approximate nature of the uncertainties.

in uncertainties with a range over several orders of magnitude. To solve this problem, the uncertainties are transformed as follows:

$$\sigma'_u(x_i) = \frac{\log_{10}(\sigma_u(x_i) + 1)}{\frac{1}{N}\sum_j \log_{10}(\sigma_u(x_j) + 1)}.$$

This induces some symmetry and scales the values to have a mean of 1. The transformed uncertainties are inserted in $h_{\text{u-add}}$ and $h_{\text{u-mul}}$ as usual. The kernel of the density estimator on the predictions is still the Epanechnikov kernel as in all other configurations in order to make fair comparisons. The results of kernel ridge regression with variable bandwidths are shown in figure 6.6. The variable bandwidths do not reliably improve the estimations. This might be because they originate from an asymptotic distributional approximation (3.6), while the uncertainties of the Mondrian forest are based on the sample variances in its relevant nodes.

6.8 Discussion

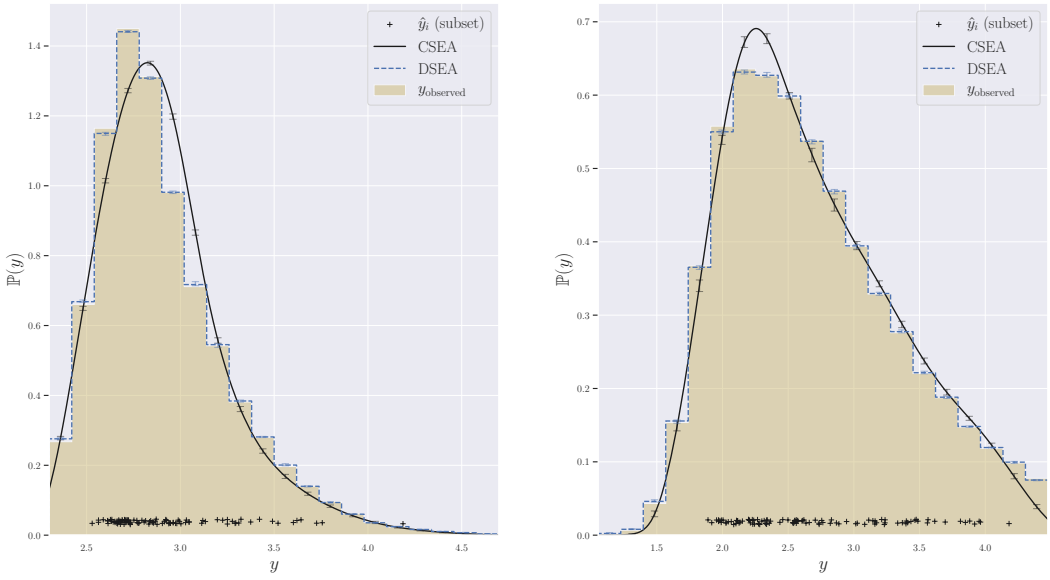
A relatively good performance of Mondrian forests was more or less expected. Since they scale well with large data sets, they were trained on five times more data than the Gaussian process and the kernel ridge model. In most of the experiments they also outperform random forests and their results have even been improved by their output of uncertainties. Using the additive approach to a variable bandwidth by incorporating the uncertainties lead to improvements on all data sets.

Surprisingly well were the estimations by the kernel ridge model. When uncertainties are disregarded, it performed best on the MAGIC data set. The estimation could not be improved reliably by the use of a variable bandwidth. This may be because the uncertainties are provided by an asymptotic distributional approximation.

While all models find adequate fits to the toy data set, the divergence of all models, with the exception of Gaussian processes, is worrying. The divergence was addressed by the distribution calibration. The quantile calibration fitting a Gaussian process showed the most promising results. It succeeded in preventing the divergence of all models.

The best estimations on the FACT and the MAGIC data set delivers the Mondrian forest with the additive uncertainty bandwidth $h_{\text{u-add}}$. The estimated probability density functions with CSEA and DSEA are shown in figure 6.7. The Kolmogorov distances of the deconvolution results to the empirical target CDFs are compared in table 6.3. We come to the conclusion that CSEA can outperform DSEA in quality of the estimations according to the KS distance when choosing adequate parameters.

Nonetheless, the dependence on a well-chosen KDE bandwidth of the algorithm is concerning. Even when using the rule-of-thumb, one still has to optimize the preceding factor. In addition, the factor is strongly dependent on the data set and the regression model used. This is especially problematic when one wants to estimate densities on data



(a) FACT **(b)** MAGIC

Figure 6.7: The best deconvolution results on the FACT and the MAGIC data set respectively. For both a Mondrian forest with bandwidth $h_{\text{u-add}}$ was used. The estimation of the FACT distribution still looks like CSEA underestimations the slope, while the estimation of the MAGIC distribution appears to be well fitted. The larger error bars of CSEA compared to DSEA are not concerning as this is a direct consequence of the real-valued predictions. Note that the histogram representation of the true target values introduces discretization errors and was conducted for visual comparison only. The estimations from CSEA are likely closer to the true density than the shown y_{observed}.

where the target variable is unknown. It is therefore not possible to assess the quality of the models. This is an advantage of the discrete algorithms that give reliable estimations without parameter optimizations. The Mondrian forest using $h_{\text{u-add}}$ performed well on both data sets with the same parametrization. Further research is needed on whether this makes them a candidate for a robust deconvolution on various IACT data sets.

Algorithm	MAGIC	FACT
DSEA	0.086	0.054
CSEA	**0.041**	**0.012**

Table 6.3: Kolmogorov distances for DSEA and CSEA using a Mondrian forest with bandwidth $h_{\text{u-add}}$.

Chapter 7

Conclusion

The main contribution of this work was the development of CSEA: an algorithm for continuous deconvolution of probability density functions. CSEA extends DSEA by allowing the use of regression methods and a kernel density estimator.

To find suitable candidates, various regression methods were studied with an emphasis on their output of uncertainties. Among Gaussian processes we found candidates in kernel ridge regression and Mondrian forests by specifying proper prediction intervals. In the process the equivalence of weighted Gaussian processes and variable noise Gaussian processes was proven. Furthermore, we developed a composite model that allows arbitrary regression methods to output uncertainties.

For the estimation of a continuous density, kernel density estimation was surveyed. We explored options to incorporate the uncertainties from the regression methods. This resulted in an additive and a multiplicative bandwidth based on Silverman's rule-of-thumb. The advantage of a variable bandwidth over weighted kernel density estimation has been addressed when choosing them based on uncertainties.

Last but not least we developed two techniques for distribution calibration. These are necessary when using a regression method that has an intrinsic bias towards the mean of the predictions as this leads to an estimated density with a lower variance than desired. This is a problem when the prediction distribution is supposed to represent the distribution of the observed data, as in CSEA. The moment calibration scales the predictions to restore the desired moments, while the quantile calibration maps all quantiles to the corresponding quantiles of the training data if it were distributed according to the predictions.

Finally, we evaluated all developed methods on three data sets: A synthetic Gaussian mixture and two simulated data sets for the telescope systems FACT and MAGIC. It was shown that CSEA outperforms DSEA on the IACT data sets according to the Kolmogorov Smirnov distance when an adequate parametrization is chosen. The best performing model for both data sets was a Mondrian forest using the additive bandwidth approach that embeds the uncertainties of the predictions.

7.1 Outlook

7.1.1 Distribution Calibration

While the calibration techniques developed have already provided better estimates for the synthetic data set, there is still room for improvement. The moment calibration does not particularly accommodate skewed distributions and transforms the predictions at both tails using the same formula. This was attempted to solve with the quantile calibration. Unfortunately, the quantile calibration is computationally more expensive. Embedding a Gaussian process for computing the quantile map results in CSEA losing its ability to process large amounts of data regardless of the regression method chosen. Further research could lead to the replacement of the Gaussian process by a simpler model that retains its expressiveness. Although the quantile calibration is more versatile than the moment calibration, it is incapable to extend the range of predictions. A combination of both approaches could lead to a technique that can transform predictions beyond the extrema, while at the same time tailoring well to skewed distributions.

7.1.2 Detecting Concept Drifts

An advantage of DSEA and CSEA over other deconvolution methods is that they return the contribution of every sample to the resulting density. It was already suggested to take advantage of this for time series analyses [7]. Instead of reconstructing the density of the entire observed data, only the contributions of a subset are aggregated. This makes deconvolution in a sliding window over a time frame possible for analyzing time-dependent data sets (see figure 7.1)

One use case of a time-dependent deconvolution in Cherenkov astronomy is the detection of flares. These are events in which the energy distribution changes severely by an increased frequency of high-energy particles. Flares are of particular interest and lead to different theories. The detection of concept drifts in the unfolded distributions may lead

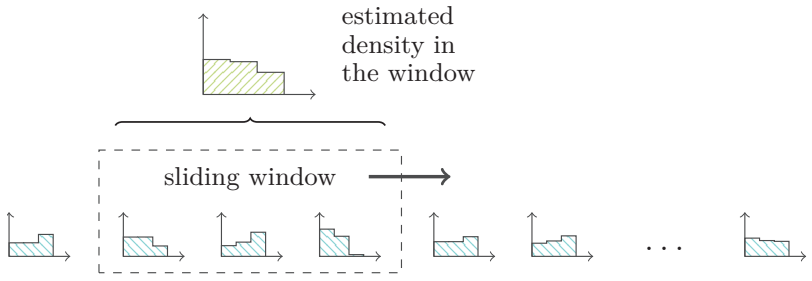


Figure 7.1: Deconvolution in a sliding window over a time frame. Instead of reconstructing the density of the entire observed data, only the contributions of a subset are aggregated [7].

to an indication of these events. Since the shape of the particle distribution during flares is more precisely specified in a continuous form, CSEA may have an advantage over DSEA for the detection. Furthermore, the increased granularity of its estimate makes it more sensitive to changes in the target distribution.

Appendix A

Proof of the Asymptotic Mean Integrated Squared Error

We prove that the Mean Integrated Squared Error has the following property:

$$MISE\left(\hat{f}(x)\right) \to \frac{1}{nh}\left\|\mathcal{K}\right\|^2 + \frac{h^4}{4}\mu_2(\mathcal{K})^2\left\|f''\right\|^2.$$

This property is used in chapter 4.2 to derive the optimal bandwidth in a kernel density estimation. We assume that the kernel $\mathcal{K}$ has the following properties:

$$\begin{aligned} \mathcal{K}(x) &\geq 0 && \text{(Positivity)} \\ \int_{-\infty}^{\infty} \mathcal{K}(x)\,\mathrm{d}x &= 1 && \text{(Unit measure)} \\ \mathcal{K}(x) &= \mathcal{K}(-x) && \text{(Symmetry).} \end{aligned}$$

The MISE uses both bias und variance to measure the quality of fit. We first derive them for arbitrary x:

$$\begin{aligned} \mathbb{B}\mathrm{ias}\left(\hat{f}(x)\right) &= \mathbb{E}\left(\hat{f}(x) - f(x)\right) \\ &= \frac{1}{n}\sum_i \mathbb{E}(\mathcal{K}(x - X_i)) - f(x) \\ &= \mathbb{E}(\mathcal{K}_h(x - X_1)) - f(x) \\ &= \int_{-\infty}^{\infty} \mathcal{K}\left(x - u\right) f(u)\,\mathrm{d}u - f(x) \\ &= \int_{-\infty}^{\infty} \frac{1}{h}\mathcal{K}\left(\frac{x-u}{h}\right) f(u)\,\mathrm{d}u - f(x) \\ &\overset{t=\frac{x-u}{h}}{=} \int_{-\infty}^{\infty} \mathcal{K}\left(t\right) f(x - ht)\,\mathrm{d}t - f(x) \\ &\overset{\int\mathcal{K}=1}{=} \int_{-\infty}^{\infty} \mathcal{K}(t)\left(f(x - ht) - f(x)\right)\,\mathrm{d}t. \end{aligned}$$

Using the Taylor expansion at $x - ht$

$$f(x-ht) = f(x) - htf'(x) + \frac{1}{2}h^2t^2f''(x) + \dots \tag{A.1}$$

the bias can be approximated as follows:

$$\begin{aligned}\mathbb{B}\text{ias}\left(\hat{f}(x)\right) &\approx hf'(x)\underbrace{\int_{-\infty}^{\infty} t\mathcal{K}(t)\,\mathrm{d}t}_{=\mathbb{E}(\mathcal{K})=0} + \frac{1}{2}h^2f''(x)\int_{-\infty}^{\infty} t^2\mathcal{K}(t)\,\mathrm{d}t + o\left(h^2\right)\\ &= \frac{h^2}{2}\mu_2(\mathcal{K})f''(x) + o\left(h^2\right).\end{aligned}$$

The variance is approximated taking the same approach:

$$\begin{aligned}\mathbb{V}\text{ar}\left(\hat{f}(x)\right) &= \mathbb{V}\text{ar}\left(\frac{1}{n}\sum_i \mathcal{K}_h(x-X_i)\right)\\ &= \frac{1}{n}\mathbb{V}\text{ar}\left(\mathcal{K}_h(x-X_1)\right)\\ &= \frac{1}{n}\left(\mathbb{E}\left(\mathcal{K}_h\left((x-X_1)^2\right)\right) - \mathbb{E}\left(\mathcal{K}_h(x-X_1)\right)^2\right)\\ &= \frac{1}{n}\int_{-\infty}^{\infty}\mathcal{K}\left(x-u\right)^2 f(u)\,\mathrm{d}u - \frac{1}{n}\left(\int_{-\infty}^{\infty}\mathcal{K}\left(x-u\right)f(u)\,\mathrm{d}u\right)^2\\ &= \frac{1}{n}\int_{-\infty}^{\infty}\mathcal{K}\left(x-u\right)^2 f(u)\,\mathrm{d}u - \frac{1}{n}\left(\mathbb{B}\text{ias}\left(\hat{f}(x)\right) + f(x)\right)^2\\ &\approx \frac{1}{n}\int_{-\infty}^{\infty}\frac{1}{h^2}\mathcal{K}\left(\frac{x-u}{h}\right)^2 f(u)\,\mathrm{d}u + o\left(\frac{1}{n}\right)\\ &\overset{t=\frac{x-u}{h}}{=} \frac{1}{nh}\int_{-\infty}^{\infty}\mathcal{K}\left(t\right)^2 f(x-ht)\,\mathrm{d}t + o\left(\frac{1}{n}\right)\\ &\overset{A.1}{\approx} \frac{1}{nh}f(x)\int_{-\infty}^{\infty}\mathcal{K}\left(t\right)^2\,\mathrm{d}t - \frac{1}{n}f'(x)\int_{-\infty}^{\infty}t\mathcal{K}\left(t\right)^2\,\mathrm{d}t + o\left(\frac{1}{nh}\right)\\ &\overset{\mathcal{K}(t)=\mathcal{K}(-t)}{=} \frac{1}{nh}f(x)\left\|\mathcal{K}\right\|^2 + o\left(\frac{1}{nh}\right).\end{aligned}$$

We integrate the Mean Squared Error (MSE) to get the Mean Integrated Squared Error (MISE) as a global measure:

$$\begin{aligned}MISE\left(\hat{f}\right) &= \int_{-\infty}^{\infty} MSE(\hat{f}(x))\,\mathrm{d}x\\ &= \int_{-\infty}^{\infty}\left(\mathbb{V}\text{ar}\left(\hat{f}(x)\right) + \mathbb{B}\text{ias}\left(\hat{f}(x)\right)^2\right)\,\mathrm{d}x\\ &\approx \int_{-\infty}^{\infty}\left(\frac{1}{nh}f(x)\left\|\mathcal{K}\right\|^2 + o\left(\frac{1}{nh}\right) + \left(\frac{h^2}{2}\mu_2(\mathcal{K})f''(x) + o\left(h^2\right)\right)^2\right)\,\mathrm{d}x\\ &= \frac{1}{nh}\left\|\mathcal{K}\right\|^2\underbrace{\int_{-\infty}^{\infty}f(x)\,\mathrm{d}x}_{=1} + \frac{h^4}{2}\mu_2(\mathcal{K})^2\int_{-\infty}^{\infty}f''(x)^2\,\mathrm{d}x + o\left(h^4\right) + o\left(\frac{1}{nh}\right)\\ &= \frac{1}{nh}\left\|\mathcal{K}\right\|^2 + \frac{h^4}{2}\mu_2(\mathcal{K})^2\left\|f''\right\|^2 + o\left(h^4\right) + o\left(\frac{1}{nh}\right).\end{aligned}$$

Assuming $n \to \infty$, $h \to 0$ and $\frac{1}{nh} \to 0$ this leads to the asymptotic MISE (AMISE):

$$AMISE\left(\hat{f}\right) = \frac{1}{nh}\left\|\mathcal{K}\right\|^2 + \frac{h^4}{4}\mu_2(\mathcal{K})^2\left\|f''\right\|^2.$$

List of Tables

List of Figures

List of Algorithms

Bibliography

[1] Anderhub, H, M Backes, A Biland, Vittorio Boccone, I Braun, T Bretz, J Buss, Franck Cadoux, V Commichau, L Djambazov et al.: *Design and operation of FACT–the first G-APD Cherenkov telescope.* Journal of Instrumentation, 8(06):P06008, 2013.

[2] Blobel, Volker: *Unfolding methods in high-energy physics experiments.* Technical Report, CERN, 1984.

[3] Blobel, Volker: *An unfolding method for high energy physics experiments.* arXiv preprint hep-ex/0208022, 2002.

[4] Bockermann, Christian, Kai Brügge, Jens Buss, Alexey Egorov, Katharina Morik, Wolfgang Rhode and Tim Ruhe: *Online analysis of high-volume data streams in astroparticle physics.* In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 100–115. Springer, 2015.

[5] Börner, Mathis, Tobias Hoinka, Maximilian Meier, Thorben Menne, Wolfgang Rhode and Katharina Morik: *Measurement/simulation mismatches and multivariate data discretization in the machine learning era.* In *Proc. of the ADASS XXVII*, 2017.

[6] Breiman, Leo: *Classification and regression trees.* Routledge, 2017.

[7] Bunse, Mirko: *DSEA Rock-Solid – Regularization and Comparison with other Deconvolution Algorithms.* Master's thesis, TU Dortmund, 44221 Dortmund, Germany, July 2018.

[8] Bunse, Mirko, Nico Piatkowski, Katharina Morik, Tim Ruhe and Wolfgang Rhode: *Unification of Deconvolution Algorithms for Cherenkov Astronomy.* In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 21–30. IEEE, 2018.

[9] Colin, P, D Borla Tridon, E Carmona, F De Sabata, M Gaug, S Lombardi, P Majumdar, A Moralejo, V Scalzotto and J Sitarek: *Performance of the MAGIC telescopes in stereoscopic mode.* arXiv preprint arXiv:0907.0960, 2009.

[10] D'Agostini, Giulio: *A multidimensional unfolding method based on Bayes' theorem.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 362(2-3):487–498, 1995.

[11] D'Agostini, Giulio: *Improved iterative Bayesian unfolding.* arXiv preprint arXiv:1010.0632, 2010.

[12] Deza, Michel Marie and Elena Deza: *Encyclopedia of distances.* In *Encyclopedia of distances*, pages 1–583. Springer, 2009.

[13] Hall, Peter et al.: *Large sample optimality of least squares cross-validation in density estimation.* The Annals of Statistics, 11(4):1156–1174, 1983.

[14] Härdle, Wolfgang: *Smoothing techniques: with implementation in S.* Springer Science & Business Media, 2012.

[15] Hart, Jeffrey D: *Kernel regression estimation with time series errors.* Journal of the Royal Statistical Society: Series B (Methodological), 53(1):173–187, 1991.

[16] Hastie, Trevor, Robert Tibshirani, Jerome Friedman and James Franklin: *The elements of statistical learning: data mining, inference and prediction.* The Mathematical Intelligencer, 27(2):83–85, 2005.

[17] Henderson, Harold V and Shayle R Searle: *On deriving the inverse of a sum of matrices.* Siam Review, 23(1):53–60, 1981.

[18] Henderson, Keith, Brian Gallagher and Tina Eliassi-Rad: *EP-MEANS: An efficient nonparametric clustering of empirical probability distributions.* In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 893–900. ACM, 2015.

[19] Hillas, A Michael: *Cerenkov light images of EAS produced by primary gamma.* 1985.

[20] Hoerl, Arthur E and Robert W Kennard: *Ridge regression: Biased estimation for nonorthogonal problems.* Technometrics, 12(1):55–67, 1970.

[21] Kull, Meelis, Telmo Silva Filho and Peter Flach: *Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers.* In *Artificial Intelligence and Statistics*, pages 623–631, 2017.

[22] Lakshminarayanan, Balaji, Daniel M Roy and Yee Whye Teh: *Mondrian forests: Efficient online random forests.* In *Advances in neural information processing systems*, pages 3140–3148, 2014.

[23] Lakshminarayanan, Balaji, Daniel M Roy and Yee Whye Teh: *Mondrian forests for large-scale regression when uncertainty matters.* In *Artificial Intelligence and Statistics*, pages 1478–1487, 2016.

[24] Liu, Bin, Ying Yang, Geoffrey I Webb and Janice Boughton: *A comparative study of bandwidth choice in kernel density estimation for naive Bayesian classification.* In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 302–313. Springer, 2009.

[25] Massey Jr, Frank J: *The Kolmogorov-Smirnov test for goodness of fit.* Journal of the American statistical Association, 46(253):68–78, 1951.

[26] Mercer, James: *Xvi. functions of positive and negative type, and their connection the theory of integral equations.* Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character, 209(441-458):415–446, 1909.

[27] Moors, Eliakim: *On the reciprocal of the general algebraic matrix, abstract.* Bull. Amer. Math. Soc, 26:394–395, 1920.

[28] Murphy, Kevin P: *Machine learning: a probabilistic perspective.* MIT press, 2012.

[29] Nadaraya, Elizbar A: *On estimating regression.* Theory of Probability & Its Applications, 9(1):141–142, 1964.

[30] Nocedal, Jorge and Stephen Wright: *Numerical optimization.* Springer Science & Business Media, 2006.

[31] Pearson, Karl: *X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling.* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 50(302):157–175, 1900.

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

[33] Rosenblatt, Murray: *Remarks on some nonparametric estimates of a density function.* The Annals of Mathematical Statistics, pages 832–837, 1956.

[34] Rubner, Yossi, Carlo Tomasi and Leonidas J Guibas: *A metric for distributions with applications to image databases.* In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 59–66. IEEE, 1998.

[35] Rudin, Walter: *Real and complex analysis*. Tata McGraw-hill education, 2006.

[36] Ruhe, Tim: *D-SEA: A Data Mining Approach to Unfolding*. 2013.

[37] Ruhe, Tim, M Börner, Max Wornowizki, T Voigt, Wolfgang Rhode and Katharina Morik: *Mining for spectra-the Dortmund spectrum estimation algorithm*. In *Proc. of the ADASS XXVI*, 2016.

[38] Scheffe, Henry: *The analysis of variance*, volume 72. John Wiley & Sons, 1999.

[39] Sheather, Simon J and Michael C Jones: *A reliable data-based bandwidth selection method for kernel density estimation*. Journal of the Royal Statistical Society: Series B (Methodological), 53(3):683–690, 1991.

[40] Silverman, Bernard W: *Density estimation for statistics and data analysis*. Routledge, 2018.

[41] Song, Hao, Tom Diethe, Meelis Kull and Peter Flach: *Distribution calibration for regression*. arXiv preprint arXiv:1905.06023, 2019.

[42] Stoer, Josef and Roland Bulirsch: *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.

[43] Temme, Fabian, Max Ludwig Ahnen, Matteo Balbo, Matthias Bergmann, Adrian Biland, Christian Bockermann, Thomas Bretz, Kai Arno Brügge, Jens Buss, Daniela Dorner et al.: *FACT-First Energy Spectrum from a SiPM Cherenkov Telescope*. In *The 34th International Cosmic Ray Conference*, volume 236, page 707. SISSA Medialab, 2016.

[44] Terrell, George R, David W Scott et al.: *Variable kernel density estimation*. The Annals of Statistics, 20(3):1236–1265, 1992.

[45] Tridon, D Borla, T Schweizer, F Goebel, R Mirzoyan, M Teshima, MAGIC Collaboration et al.: *The MAGIC-II gamma-ray stereoscopic telescope system*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 623(1):437–439, 2010.

[46] Wand, Matt P and M Chris Jones: *Kernel smoothing*. Chapman and Hall/CRC, 1994.

[47] Williams, Christopher KI and Carl Edward Rasmussen: *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[48] Zadrozny, Bianca: *Reducing multiclass to binary by coupling probability estimates*. In *Advances in neural information processing systems*, pages 1041–1048, 2002.