The dings that will become a thesis at some point

by

Ingrid Holm

Thesis

for the degree of

Master of Science

Faculty of Mathematics and Natural Sciences
University of Oslo

February 2018

Abstract

This is an abstract text.

To someone

This is a dedication to my cat.

Acknowledgements

I acknowledge my acknowledgements.

Contents

Chapter 1

Introduction

- why nlo?
    - why gp?
    - why dgp?

Chapter 2

Physics Background

2.1 The Standard Model

We assume in this thesis that the reader is familiar with the Standard Model of particle physics. Even so, a small introduction is in order. The Standard Model is the best-working framework we have for calculating the behaviour of very small particles at very high energies. At these small levels an important distinction we make is between *fermions* and *bosons*: particles with half-integer and integer spin values, respectively. Fermions interact through the exchange of bosons, and the Standard Model bosons are the *photon* (electromagnetic interaction), the *gluon* (strong interaction that holds atoms together), the W and Z bosons (the weak interaction) and the famously elusive *Higgs boson.* The rules of motion and interaction can all be found using the *Lagrangian* of the Standard Model. A Lagrangian is a measure that is invariant to transformations under the Lorentz group – or a change of reference frame as it is called in special relativity. The Lagrangian of the SM is given by

$$\mathcal{L} = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu} + i\bar{\psi}\not{D}\psi + h.c. + x_i y_{ij} x_j \phi + h.c. + |D\phi|^2 - V(\phi), \tag{2.1}$$

where $F_{\mu\nu}$ is the electromagnetic field strength, ψ is the fermion field, $\not{D}$ is the covariant derivative, $x_i y_{ij} x_j \phi$ are the mass terms, and $|D\phi|^2 - V(\phi)$ is the kinetic term and potential for the Higgs.

2.2 Supersymmetry

Supersymmetry is essentially the extension of the SM, which assumes the SM groups $SU(3) \times SU(2) \times U(1)$ is a subgroup of the superalgebra. This algebra is a coset space of the super-Poincaré algebra, and the Lorentz algebra. The coordinates in this space are $z^\pi = (x^\mu, \theta^A, \bar{\theta}_{\dot{A}})$, where x^μ are the well known Minkowski coordinates, and $\theta^A, \bar{\theta}_{\dot{A}}$ are Grassmann numbers that act as parameters of the operators Q. A general element of the superalgebra can be written as

$$g = \exp[-ix^\mu P_\mu + i\theta^A Q_A + i\bar{\theta}_{\dot{A}}\bar{Q}^{\dot{A}} - i\omega_{\rho\nu}M^{\rho\nu}], \tag{2.2}$$

where x^μ, θ^A, $\bar{\theta}_{\dot{A}}$ and $\omega_{\rho\nu}$ are the parameters of the group, and P_μ, Q_A, $\bar{Q}^{\dot{A}}$ and $M^{\rho\nu}$ are the group generators. Because of commutation relations and parameterisations (REFERER TIL NOE HER - ARE?) we can write a supersymmetric transformation simply as

$$\delta_S = \alpha^A Q_A + \bar{\alpha}_{\dot{A}}\bar{Q}^{\dot{A}}. \tag{2.3}$$

This algebra introduces a symmetry between fermions and bosons, which is brought in by the group generators Q which turn a fermion into a boson and vice versa. These generators can be written as

$$P_\mu = i\partial_\mu, \tag{2.4}$$
$$iQ_A = -i(\sigma^\mu\bar{\theta})_A\partial_\mu + \partial_A, \tag{2.5}$$
$$i\bar{Q}^{\dot{A}} = -i(\bar{\sigma}^\mu\theta)^{\dot{A}}\partial_\mu + \partial^{\dot{A}}. \tag{2.6}$$

On our way to describing a supersymmetric Lagrangian we need a derivative that is invariant under supersymmetry transformations. The general covariant derivatives are defined as

$$D_A \equiv \partial_A + i(\sigma^\mu\bar{\theta})_A\partial_\mu, \tag{2.7}$$
$$\bar{D}^{\dot{A}} \equiv -\partial^{\dot{A}} - i(\sigma^\mu\theta)^{\dot{A}}\partial_\mu. \tag{2.8}$$

The next step is to find something for these derivatives to work *on*. For this we define the *superfields* $\Phi(x,\theta,\bar{\theta})$. These can be expanded in power series of θ, $\bar{\theta}$ and x, but we won't do that here. We are more concerned with the following relations

$$\bar{D}_{\dot{A}}\Phi(x,\theta,\bar{\theta}) = 0 \qquad \text{(left-handed scalar superfield),} \tag{2.9}$$
$$D^A\Phi^\dagger(x,\theta,\bar{\theta}) = 0 \qquad \text{(right-handed scalar superfield)} \tag{2.10}$$
$$\Phi(x,\theta,\bar{\theta}) = \Phi^\dagger(x,\theta,\bar{\theta}) \qquad \text{(vector superfield).} \tag{2.11}$$

It can be shown that the left- and right-handed scalar fields can be written as, respectively,

$$\Phi(x,\theta,\bar{\theta}) = A(x) + i(\theta\sigma^\mu\bar{\theta})\partial_\mu A(x) - \frac{1}{4}\theta\theta\bar{\theta}\bar{\theta}\Box A(x) + \sqrt{2}\theta\psi(x) \tag{2.12}$$
$$- \frac{i}{\sqrt{2}}\theta\theta\partial_\mu\psi(x)\sigma^\mu\bar{\theta} + \theta\theta F(x), \tag{2.13}$$
$$\Phi^\dagger(x,\theta,\bar{\theta}) = A^*(x) - i(\theta\sigma^\mu\bar{\theta})\partial_\mu A^*(x) - \frac{1}{4}\theta\theta\bar{\theta}\bar{\theta}\Box A^*(x) + \sqrt{2}\bar{\theta}\bar{\Psi}(x) \tag{2.14}$$
$$- \frac{i}{\sqrt{2}}\bar{\theta}\bar{\theta}\theta\sigma^\mu\partial_\mu\bar{\Psi}(x) + \bar{\theta}\bar{\theta}F^*(x). \tag{2.15}$$

From Eq. (2.11) we see that the structure of a general vector field should be

$$\begin{aligned}\Psi(x,\theta,\bar\theta) =& f(x) + \theta\varphi(x) + \bar\theta\bar\varphi(x) + \theta\theta m(x) + \bar\theta\bar\theta m^*(x) \quad (2.16)\\ &+ \theta\sigma^\mu\bar\theta V_\mu(x) + \theta\theta\bar\theta\bar\lambda(x) + \bar\theta\bar\theta\theta\lambda(x) + \theta\theta\bar\theta\bar\theta d(x), \quad (2.17)\end{aligned}$$

where $f(x)$, $d(x)$ are real scalar fields, $\varphi(x)$, $\lambda(x)$ are Weyl spinors, $m(x)$ is a complex scalar field and $V_\mu(x)$ is a real Lorentz four-vector.

Because of various restrictions on the supersymmetric Lagrangian (such as renormalizability), the most general Lagrangian we may write as a function of the scalar superfields is

$$\mathcal{L} = \Phi_i^\dagger\Phi_i + \bar\theta\bar\theta W[\Phi] + \theta\theta W[\Phi^\dagger], \quad (2.18)$$

where $\Phi_i^\dagger\Phi_i$ is the kinetic term, and $W[\Phi]$ is the superpotential

$$W[\Phi] = g_i\Phi_i + m_{ij}\Phi_i\Phi_j + \lambda_{ijk}\Phi_i\Phi_j\Phi_k, \quad (2.19)$$

Supersymmetry algebra requires that there is an equal number of fermions and bosons, and that these appear in pairs: the 'old' SM particles, and their sparticles. An unbroken supersymmetry would mean that the particle-sparticle pairs should have the same mass. Since these particles have not been observed, Supersymmetry is assumed to be a broken symmetry.

The simplest way of breaking SUSY is the *soft breaking*. This entails adding terms to the Lagrangian which cause spontaneous symmetry breaking. Initially, this introduces 104 new parameters (in addition to the existing SM ones). Luckily, various bounds, like the ones from gauge symmetry, reduce the number of parameters significantly.

The Supersymmetric Lagrangian results in couplings that violate both lepton and baryon numbers. Therefore, we introduce a new, multiplicative conserved quantity, named R-parity

$$R = (-1)^{2s+3B+L}. \quad (2.20)$$

This quantity is $+1$ for SM particles, and -1 for the sparticles. Since R-parity must be conserved, this means that sparticles always appear in pairs. A further consequence of this is that there must exist a stable, lightest supersymmetric particle, to which all other supersymmetric particles decay eventually. This is a good candidate for dark matter [4].

2.2.1 Why Supersymmetry?

The Hierarchy Problem

Gauge Coupling Unification

Dark Matter

2.2.2 Superfields

Covariant Derivatives

2.2.3 Supersymmetric Lagrangian

2.2.4 Superpartners

2.2.5 R-Parity

2.2.6 The Minimal Supersymmetric Standard Model (MSSM)

Lagrangian

MSSM Field Content

Chapter 3

Supersymmetry at Hadron Colliders

3.0.1 Phenomenology

3.0.2 Current Supersymmetric Limits

3.1 Hadronic Cross Sections

3.1.1 Partonic cross sections

Parton Distribution Functions

3.2 Squark-Squark Cross Section

3.2.1 Feynman Diagrams

3.2.2 Calculation to LO

3.2.3 NLO Corrections

Loop diagrams

Renormalization of Divergences

K-factor

3.2.4 State-of-the-art Tools

Prospino
NLL-fast

Chapter 4

Gaussian Processes

4.1 Introduction to Bayesian Statistics

In statistics we distinguish between Bayesian and frequentist statistics, in this thesis we will focus on the former. Bayesian statistics may be called the science of qualified guesses. It's basic principles can be derived from the familiar rules of probability

$$P(X|I) + P(\bar{X}|I) = 1, \tag{4.1}$$

$$P(X,Y|I) = P(X|Y,I) \times P(Y|I), \tag{4.2}$$

commonly known as the *sum rule* and *product rule*, respectively. From these simple expressions we can derive the most central theorem's of Bayesian statistics: namely *Bayes theorem* and *marginalization*, given by

$$P(X|Y,I) = \frac{P(Y|X,I) \times P(X|I)}{P(Y|I)}, \tag{4.3}$$

$$P(X|I) = \int_{-\infty}^{\infty} P(X,Y|I) dY. \tag{4.4}$$

This may seem like an obvious statement: theorem 4.3 is just a way of rephrasing that the probability of X and Y must be the same as the probability of Y and X. However, if we choose X and Y more carefully, magic happens. Assume that X is some prediction we have about a , and Y is data

4.1.1 Priors and Likelihood

Likelihood: probability of the observations given the parameters.

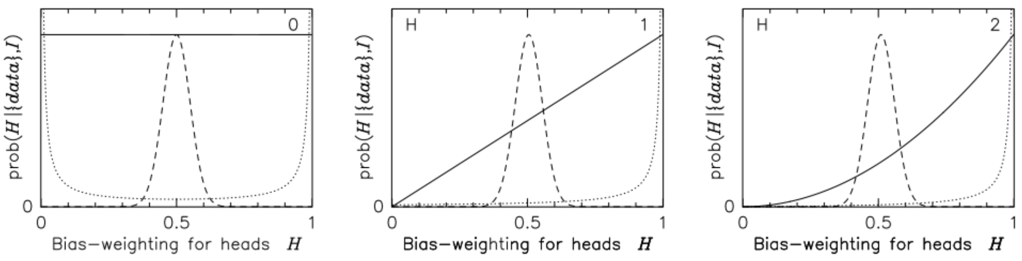


Figure 4.1: From [3].

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{marginal likelihood}}. \tag{4.5}$$

Prior: prior belief or assumption about data. Is modified through likelihood function. Example of coin toss from Sivia.

Posterior: probability of value of a parameter given data and relevant background information.

Likelihood: Probability of parameter given observation.

4.1.2 Best Estimate and Reliability

Best estimate X_0 is at maximum of posterior

$$\frac{dP}{dX}\Big|_{X_0} = 0, \qquad \frac{d^2P}{dX^2}\Big|_{X_0} < 0. \tag{4.6}$$

How reliable is this best estimate? Find width using Taylor, and take log

$$L = L(X_0) + \frac{1}{2}\frac{d^2}{dx^2}L\Big|_{X_0}(X - X_0)^2 + ..., \quad L = \log_e\Big[\text{prob}(x|\{data\}, I)\Big] \tag{4.7}$$

Proximate posterior with **Gaussian distribution**

$$\text{prob}(x|\mu, \sigma), \qquad \sigma = \Big(-\frac{d^2L}{dx^2}\Big)^{-1/2} \tag{4.8}$$

4.1.3 Covariance

Is the reliability for several parameters $\{X_i\}$.

$$\frac{dP}{dX_i}\Big|_{X_{0j}} = 0, \tag{4.9}$$

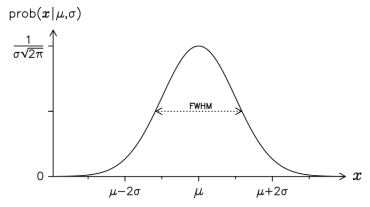


Fig. 2.3 The Gaussian, or normal, distribution. It is symmetric with respect to the maximum, at $x=\mu$, and has a full width at half maximum (FWHM) of about 2.35σ.

Figure 4.2: From [3].

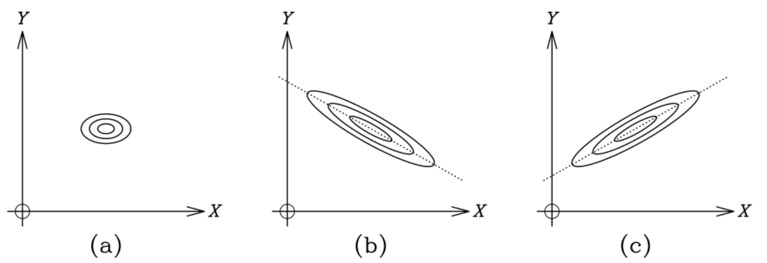


Fig. 3.7 A schematic illustration of covariance and correlation. (a) The contours of a posterior pdf with zero covariance, where the inferred values of X and Y are uncorrelated. (b) The corresponding plot when the covariance is large and negative; $Y+mX=$ constant along the dotted line (where $m>0$), emphasizing that only this sum of the two parameters can be inferred reliably. (c) The case of positive correlation, where we learn most about the difference $Y-mX$; this is constant along the dotted line.

Figure 4.3: From [3]

In 2 dim

$$L = L(X_0, Y_0) + \frac{1}{2}\left[\frac{d^2L}{dX^2}\bigg|_{X_0,Y_0}(X-X_0)^2 \tag{4.10}$$

$$+\frac{d^2L}{dY^2}\bigg|_{X_0,Y_0}(Y-Y_0)^2 + 2\frac{d^2L}{dXdY}\bigg|_{X_0,Y_0}(X-X_0)(Y-Y_0)\right] + \ldots \tag{4.11}$$

$$Q = \begin{pmatrix} X-X_0 & Y-Y_0 \end{pmatrix}\begin{pmatrix} A & C \\ C & B \end{pmatrix}\begin{pmatrix} X-X_0 \\ Y-Y_0 \end{pmatrix} \tag{4.12}$$

Q is the **covariance matrix**.

4.2 Gaussian Process Regression

Define mean and covariance function as

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{4.13}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{4.14}$$

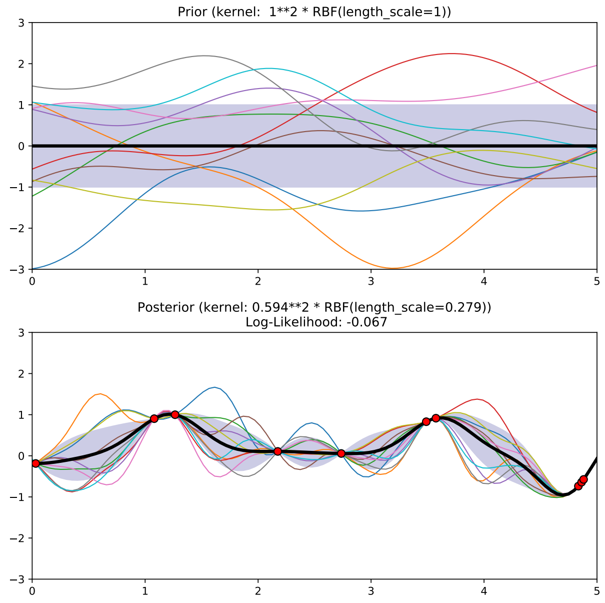


Figure 4.4: From scikitlearn

Write this as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{4.15}$$

Joint distribution for NOISE FREE, train $\mathbf{f}$, test $\mathbf{f}_*$

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X,X_*) & K(X_*,X_*) \end{bmatrix}\right) \tag{4.16}$$

Then **condition** distribution on observations

$$\mathbf{f}_* \big| X_*, X, \mathbf{f} \sim \mathcal{N}(K(X_*,X)K(X,X)^{-1}\mathbf{f}, K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)) \tag{4.17}$$

Can draw samples from this ditribution.

4.2.1 Gaussian Noise Model

Assume

$$y = f(\mathbf{x}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{4.18}$$

$$\mathrm{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \qquad \mathrm{cov}(\mathbf{y}) = K(X,X) + \sigma_n^2 \mathbb{I} \tag{4.19}$$

Distribution now becomes

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}K(X,X)+\sigma_n^2\mathbb{I} & K(X,X_*)\\ K(X,X_*) & K(X_*,X_*)\end{bmatrix}\right) \tag{4.20}$$

Conditioned

$$\mathbf{f}_*\big|X_*, X, \mathbf{f} \sim \mathcal{N}(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)), \tag{4.21}$$

$$\bar{\mathbf{f}}_* = K(X_*,X)[K(X,X)+\sigma_n^2\mathbb{I}]^{-1}\mathbf{y}, \tag{4.22}$$

$$\operatorname{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)[K(X,X)+\sigma_n^2\mathbb{I}]^{-1}K(X,X_*) \tag{4.23}$$

GP prediction

$$\bar{f}_* = \mathbf{k}_*^T(K+\sigma_n^2\mathbb{I})^{-1}\mathbf{y}, \tag{4.24}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^T(K+\sigma_n^2\mathbb{I})^{-1}\mathbf{k}_*. \tag{4.25}$$

4.2.2 Algorithm

```
Data: $X$ (inputs), $\mathbf{y}$ (targets), $k$ (covariance function/kernel), $\sigma_n^2$ (noise
      level), $\mathbf{x}_*$ (test input).
L = Cholesky decomposition $(K + \sigma_n^2 I)$ ;
$\boldsymbol{\alpha} = (L^T)^{-1}(L^{-1}\mathbf{y})$ ;
$\bar{f}_* = \mathbf{k}_*^T\boldsymbol{\alpha}$ ;
$\mathbf{v} = L^{-1}\mathbf{k}_*$ ;
$\mathbb{V}[f_*] = k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{v}^T\mathbf{v}$ ;
$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T\boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2}\log 2\pi$ ;
Result: $f_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood).
```

Algorithm 1: Algorithm 2.1 from [2].

4.3 Covariance Functions

A function that only depends on the difference between two points, $\mathbf{x}-\mathbf{x}'$, is called *stationary*. This implies that the function is invariant to translations in input space. If, in addition, it only depends on the length $r = |\mathbf{x}-\mathbf{x}'|$, the function is *isotropic* (invariant to rigid rotations in input space). Isotropic functions are commonly referred to as *radial basis functions* (RBFs). The covariance function can also depend on the dot product, $\mathbf{x}\cdot\mathbf{x}'$, and is then called a *dot product* covariance function.

A function which maps two arguments $\mathbf{x} \in \mathcal{X}$, $\mathbf{x}' \in \mathcal{X}$ into $\mathbb{R}$ is generally called a *kernel* k. Covariance functions are symmetric kernels, meaning that

$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$. The matrix containing all the covariance elements is called the *covariance matrix*, or the Gram matrix K, whose elements are given by

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j). \tag{4.26}$$

There are some restrictions on the covariance matrix, namely that is has to be *positive semidefinite* (PSD). This means that the $n \times n$ matrix K satisfies $Q(\mathbf{v}) = \mathbf{v}^T K \mathbf{v} \geq 0$ for all $\mathbf{v} \in \mathbb{R}^n$. A kernel k is PSD if

$$\int k(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mu(\mathbf{x}) d\mu(\mathbf{x}') \geq 0, \tag{4.27}$$

for all $f \in L_2(\mathcal{X}, \mu)$.

Mean Square Continuity and Differentiability

Let $\mathbf{x}_1, \mathbf{x}_2, ...$ be a series of points, and $\mathbf{x}^*$ be a point in $\mathbb{R}^D$ such that $|\mathbf{x}_k - \mathbf{x}^*| \to 0$ as $k \to \infty$. The condition for a process $f(\mathbf{x})$ to be mean square continuous at $\mathbf{x}^*$ is then

$$\mathbb{E}[|f(\mathbf{x}_k) - f(\mathbf{x}^*)|^2] \to 0 \text{ as } k \to \infty. \tag{4.28}$$

A random field is continuous in mean square if and only if its covariance function $k(\mathbf{x}, \mathbf{x}')$ is continuous at the point $\mathbf{x} = \mathbf{x}' = \mathbf{x}^*$. This reduces to $k(\mathbf{0})$ for stationary covariance functions.

The mean square derivative of $f(\mathbf{x})$ in the ith direction is given by

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \text{l.i.m.}_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}, \tag{4.29}$$

where l.i.m. denotes the limit in mean square and $\mathbf{e}_i$ is the unit vector in the ith direction.

4.3.1 The Radial Basis Function (RBF)

The *squared exponential covariance function* (SE) has the form

$$k_{SE}(r) = \exp\Big(-\frac{r^2}{2\ell^2}\Big), \tag{4.30}$$

where ℓ is the *characteristic length scale*. The SE is infinitely differentiable, and so is very smooth.

Called

```
from sklearn.gaussian_process.kernels import RBF
rbf = RBF(length_scale=10, length_scale_bounds=(1e-2, 1e2))
```

4.3.2 The Matérn Kernel

The *Matérn class of covariance functions* is given by

$$k_{Matérn}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big)^{\nu} K_{\nu}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big), \tag{4.31}$$

where $\nu, \ell > 0$, and K_ν is a modified Bessel function. For $\nu \to \infty$ this becomes the SE. In the case of ν being half integer, $\nu = p + \frac{1}{2}$, the covariance function is simply the product of an exponential and a polynomial

$$k_{\nu=p+\frac{1}{2}} = \exp\Big(-\frac{\sqrt{2\nu}r}{\ell}\Big)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\Big(\frac{\sqrt{8\nu}r}{\ell}\Big)^{p-i}. \tag{4.32}$$

In machine learning the two most common cases are for $\nu = 3/2$ and $\nu = 5/2$

$$k_{\nu=3/2}(r) = \Big(1 + \frac{\sqrt{3}r}{\ell}\Big)\exp\Big(-\frac{\sqrt{3}r}{\ell}\Big), \tag{4.33}$$

$$k_{\nu=5/2}(r) = \Big(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\Big)\exp\Big(-\frac{\sqrt{5}r}{\ell}\Big). \tag{4.34}$$

```
from sklearn.gaussian_process.kernels import Matern
matern = Matern(length_scale=10, length_scale_bounds=(1e-2,
    1e2), nu=1.5)
```

Other Kernels

There are other kernels, but they are not used here. Can me multiplied and summed. For more info check chapter 4 in [2].

4.3.3 Hyperparameters

Each kernel has a vector of hyperparameters, e.g. $\boldsymbol{\theta} = (\{M\}, \sigma_f^2, \sigma_n^2)$ for the radial basis function (RBF)

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q))^T M(\mathbf{x}_p - \mathbf{x}_q) + \sigma_n^2\delta_{pq}. \tag{4.35}$$

The matrix M can have several forms, for example

$$M_1 = \ell^{-2}\mathbb{I}, M_2 = \mathrm{diag}(\boldsymbol{\ell})^{-2}. \tag{4.36}$$

4.4 Model Selection

4.4.1 Bayesian Model Selection

Feature selection at several levels: posterior over *parameters*, posterior over *hyperparameters* and posterior for the *model*,

$$p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)}{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)} \tag{4.37}$$

$$p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i) = \int p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)d\mathbf{w} \quad \text{(marginal likelihood)} \tag{4.38}$$

$$p(\boldsymbol{\theta}|\mathbf{y}, X, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)}{p(\mathbf{y}|X, \mathcal{H}_i)} \tag{4.39}$$

$$p(\mathcal{H}_i|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|X)} \tag{4.40}$$

4.4.2 Log Marginal Likelihood

For Gaussian Processes with Gaussian we can find the exact expression for the marginal likelihood,

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi. \tag{4.41}$$

The optimal parameters are found by maximizing the marginal likelihood

$$\frac{\partial}{\partial \theta_j}\log p(\mathbf{y}|X, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^T K^{-1}\frac{\partial K}{\partial \theta_j}K^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(K^{-1}\frac{\partial K}{\partial \theta_j}). \tag{4.42}$$

This is what SciKitLearn uses, but can have **multiple local maxima**. Plug in Fig.

4.4.3 Cross Validation

Divide into k-subsets and use validation and test set. Requires a loss function, e.g. R^2. Cross validate using Scikit-Learn, get a validation plot. Over training, over fitting, under-fitting.

4.4.4 Loss functions

Mean Relative Deviance

The mean relative deviance is used to quantify the quality of predictions

$$\varepsilon = \frac{y_{true} - y_{GP}}{y_{true}} \tag{4.43}$$

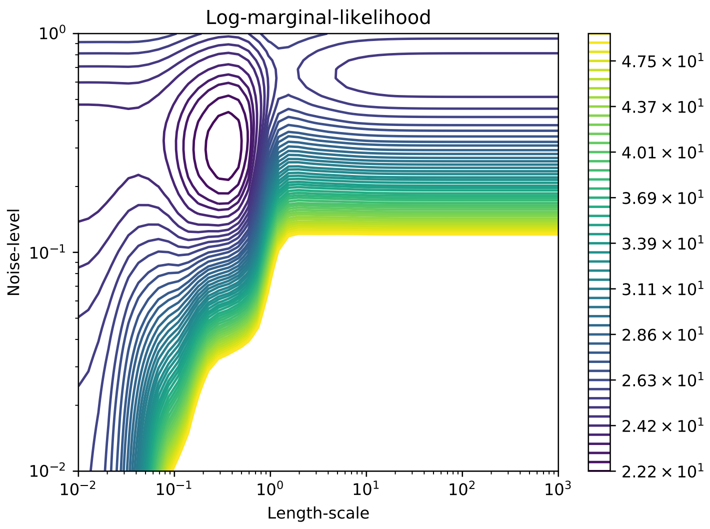


Figure 4.5: Used SciKitLearn. Several local maxima.

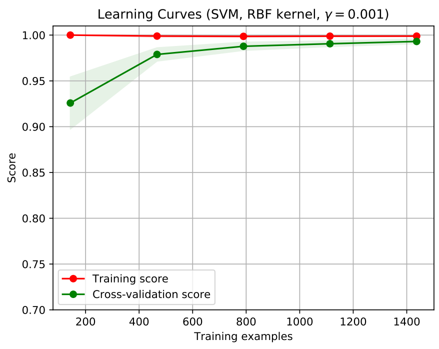


Figure 4.6: From scikit-learn.

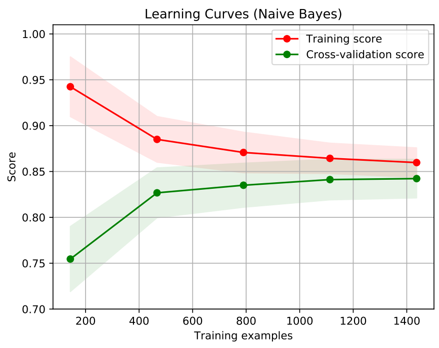
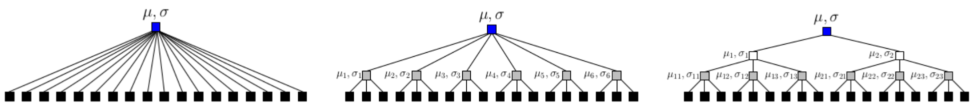


Figure 4.7: From scikit-learn.



(a) 1-layer model. (b) 2-layer model. (c) 3-layer model.

Figure 4.8: From [1].

R-Factor

4.5 Distributed Gaussian Processes

4.5.1 Limitations of Gaussian Processes

Problem because of $(K + \sigma_n^2 \mathbb{I})^{-1}$, means inverting an $n \times n$-matrix. Training and predicting limits of $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$. Limit = $\mathcal{O}(10^4)$. Some solutions exist, but **no prediction of variance is given with p-o-e.**

4.5.2 Product-of-Experts

Divide data between experts. "The assumption of independent GP experts leads to a block-diagonal approximation of the kernel matrix, which (i) allows for efficient training and predicting (ii) can be computed efficiently (time and memory) by parallelisation" [1].

Independence assumption

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \approx \prod_{k=1}^{M} p_k(\mathbf{y}^{(k)}|\mathbf{X}^{(k)}, \boldsymbol{\theta}) \tag{4.44}$$

$$\log p(\mathbf{y}^{(k)}|\mathbf{X}^{(k)}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{(k)}(\mathbf{K}_\psi^{(k)} + \sigma_\varepsilon^2 \mathbf{I})^{-1}\mathbf{y}^{(k)} - \frac{1}{2}\log|\mathbf{K}_\psi^{(k)} + \sigma_\varepsilon^2 \mathbf{I}| \tag{4.45}$$

4.5.3 Algorithm

$$\mu_*^{rbcm} = (\sigma_*^{rbcm})^2 \sum_k \beta_k \sigma_k^{-2}(\mathbf{x}_*)\mu_k(\mathbf{x}_*), \tag{4.46}$$

$$(\sigma_*^{rbcm})^{-2} = \sum_{k=1}^{M} \beta_k \sigma_k^{-2}(\mathbf{x}_*) + \big(1 - \sum_{k=1}^{M} \beta_k\big)\sigma_{**}^{-2}. \tag{4.47}$$

The posterior distribution for the test point $\mathbf{x}_*$ is given by a Gaussian with mean and variance

$$\mu(\mathbf{x}_*) = \mathbf{k}_*^T(\mathbf{K} + \sigma_\epsilon^2 \mathbb{I})^{-1}\mathbf{y}, \tag{4.48}$$

$$\sigma^2(\mathbf{x}_*) = k_{**} - \mathbf{k}_*^T(\mathbf{K} + \sigma_\epsilon^2 \mathbb{I})^{-1}\mathbf{k}_*. \tag{4.49}$$

Plot of time from the article

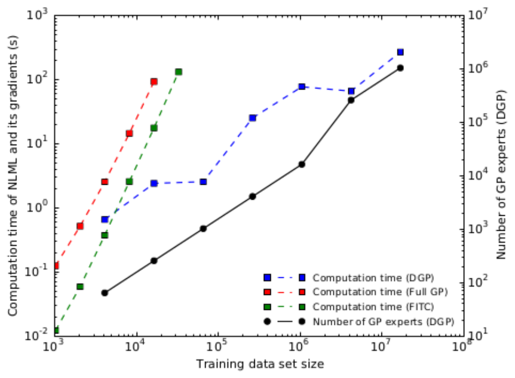


Figure 4.9: From [1].

4.5.4 Implementing the Algorithm

Parallelizing

4.5.5 Benchmark

```
Data: $N_{experts}$ (number of experts), $X$ (inputs), $\mathbf{y}$ (targets), $k$ (covariance
      function/kernel), $\sigma_n^2$ (noise level), $\mathbf{x}^*$ (test input), $\mathbf{y}^*$ (test target)
$X_{train}$, $X_{test}$, $y_{train}$, $y_{test}$ = train-test-split $(X, y)$ (scikit-learn) ;
$y = \log_{10}(y)$ ;
$n = \frac{\text{Number of data points}}{N_{experts}}$ ;
$subsets = array\_split(X_{train}, n)$ ;
$\mu_{rbcm} = []$, $\sigma_{rbcm} = []$ (empty lists to be filled later);
for each expert do
    $gp_{temporary} = GaussianProcessRegressor.fit(X_{expert}, y_{expert})$ ;
    for each $y^*$ in $\boldsymbol{y}^*$ do
        $\mu_*, \sigma_*^2 = gp_{temporary}.predict(x^*)$ ;
        $\sigma_{**}^2 = k(x^*, x^*)$ ;
        (fill inn the values) ;
        $\boldsymbol{\mu}$[expert][$x^*$] = $\mu_*^2$ (mean value from this expert);
        $\boldsymbol{\sigma}^2$[expert][$x^*$] = $\sigma_*^2$ (variance from this expert);
        $\boldsymbol{\sigma}_{**}^2$[expert][$x^*$] = $\sigma_{**}^2$ (variance from initial kernel)
    end
end
for each expert do
    for each $y_*$ in $\boldsymbol{y}_*$ do
        $\mu_* = \boldsymbol{\mu}$[expert][$x_*$] (retrieve relevant values);
        $\sigma_*^2 = \boldsymbol{\sigma}^2$[expert][$x^*$] ;
        $\sigma_{**}^2 = \boldsymbol{\sigma}_{**}^2$[expert][$x^*$] ;
        $\beta = \frac{1}{2}(\log(\sigma_{**}^2) - \log(\sigma_*^2))$ ;
        $(\sigma_*^{rbcm})^{-2}[y_*] += \beta\sigma^{-2} + \big(\frac{1}{n_{experts}} - \beta\big)\sigma_{**}^{-2}$
    end
end
for each expert do
    for each $y_*$ in $\boldsymbol{y}_*$ do
        $\mu_* = \boldsymbol{\mu}$[expert][$x_*$] (retrieve relevant values);
        $\sigma_*^2 = \boldsymbol{\sigma}^2$[expert][$x^*$] ;
        $\sigma_{**}^2 = \boldsymbol{\sigma}_{**}^2$[expert][$x^*$] ;
        $\beta = \frac{1}{2}(\log(\sigma_{**}^2) - \log(\sigma_*^2))$ ;
        $\mu_*^{rbcm}[y_*] += (\sigma_*^{rbcm})^2\beta\sigma_*^{-2}\mu_*$
    end
end
$\epsilon = \frac{10^{\mu_{rbcm}} - 10^{y_{test}}}{10^{y_{test}}}$ (relative error);
Result: Approximative distribution of $f_* = f(\mathbf{x}_*)$ with mean $\mu_*^{rbcm}$ and
        variance $(\sigma_*^{rbcm})^2$.
```

Algorithm 2: Algorithm for using rBCM on a single test point $\mathbf{x}_*$. The $GaussianProcessRegressor.fit()$-function is a function in scikit-learn, that uses Algorithm (1).

Chapter 5

Calculating Cross Sections with Distributed Gaussian Processes

5.1 Data Generation

How was the data generated?

Prospino

- Possible issues
    - NLL-FAST
    - SOFTSUSY

Feature Distributions

Lin and log distributions.

Data Quality

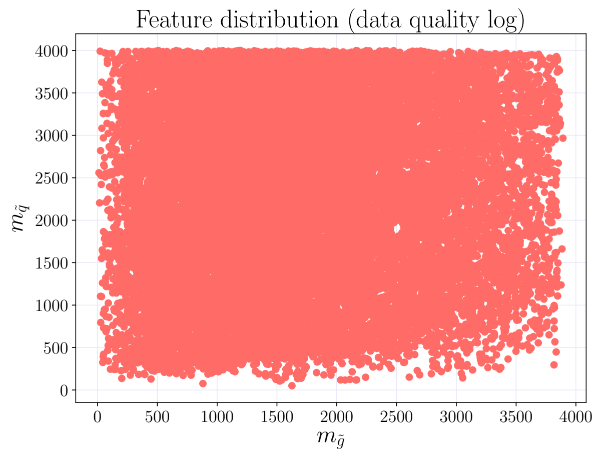
5.2 Dataset Transformations

5.2.1 Removing Outliers

Prospino Sets NLO Cross Sections to 0

- When all squark masses are large, and larger than $m_{\tilde{c}_L}$, the K-factor is zero and $LO \neq 0$ but $NLO = 0$.

The new running of 4 experts with 11 000 points each did not give a very good result. Looking at the plots of σ_{m_g} versus $m_{\tilde{q}}$ and $m_{\tilde{g}}$ made us notice a few outliers which where all of the order $\log_{10} \sigma = -32$. This we attributed to the program running the Gaussian processes setting all zero-cross sections to



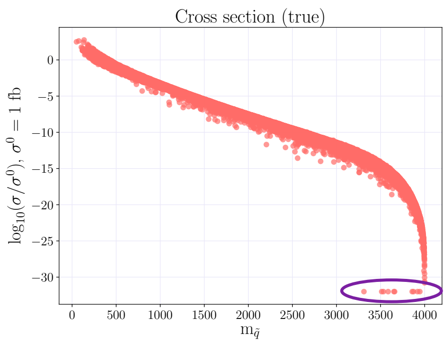


Figure 5.1: True values for the logarithm of NLO cross section as a function of $m_{\tilde{c}_L}$, for 20 000 points from the lin set. Outliers are circled in purple.

equal 10^{-32}. Looking further into the slha-files we noticed that for these points, all squark masses were very high, but the relevant squark mass m_{c_L} was a little lower than the others. This probably caused the K-factor to be zero, and therefore all NLO cross sections in this file were zero, while all LO cross section were not. We removed these points and saw a large improvement in the prediction for $\sigma_{m_{\tilde{g}}}$, making it much more stable in the low cross sections. The varying quality of the previous runs was attributed to the large dependence of number of outliers of the prediction.

We noticed the outliers when plotting the cross section as a function og the squark-mass, as seen in Fig. (5.1). The distributed Gaussian processes are unable to predict these outliers, which come from setting zero-cross sections to 10^{-32}, as can be seen from Fig. (5.2). The points seem to make the prediction for small cross sections worse, as this aerea appears to have a lot of noise.

The problematic points were therefore removed in a new run with 4 experts with 11 000 points each, still using the Matérn kernel. The resulting mean relative deviations can be seen in Fig. (5.3).

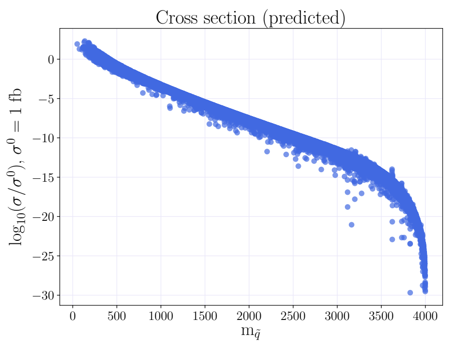


Figure 5.2: Values predicted by Distributed Gaussian processes on the lin set, using 4 experts with 11 000 points each, and the Matérn kernel. There are 20 000 test points, and the outliers from Fig. (5.1) are missing.

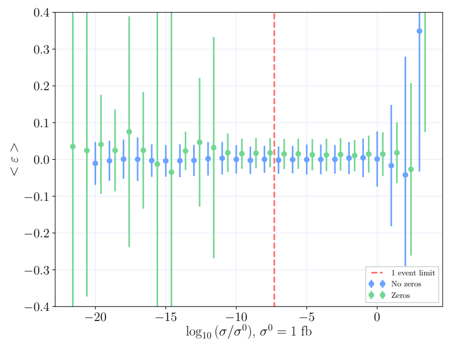


Figure 5.3: Mean standard deviation $\varepsilon = \frac{y_{true} - y_{dgp}}{y_{true}}$ for 4 experts with 11 000 points each from the lin set, with 20 000 test points. The Matérn kernel with $\nu = 1$ was used. The green lines are for before the outliers were removed, and the blue lines are from after.

5.2.2 Lower Cuts on Cross sections ($\propto 10^{-16}$)

5.3 Choosing the Kernel

Matérn

Hyperparameter ν

Vectorized ℓ

5.4 When the Error is Known

Proof for Relative Errors

The error in the observations comes from the numerical error (?) in the Prospino calculation. The relative error has a standard deviation of $\varepsilon = 0.002$ multiplied by the cross section. The question is *whether we can use this information* when doing the GP fit. Denote the cross section provided by Prospino as Y_i and the real cross section as y_i^{true}, we then have

$$Y_i = y_i^{true} + \epsilon_i = y_i^{true}(1 + \varepsilon_i). \tag{5.1}$$

However, we do not calculate the prediction of the cross section, but the *logarithm* of the cross section. The distributions are then

$$Y_i = \mathcal{N}(y_i^{true}, \varepsilon y_i^{true}), \tag{5.2}$$

where the only random variable is ε. Changing variables to $\log_{10}$ gives

$$X_i = \log_{10} Y_i \to Y_i = 10^{X_i} \tag{5.3}$$

$$P_{X_i}(X_i) = P_{Y_i}(Y_i)\Big|\frac{\partial Y_i}{\partial X_i}\Big| \tag{5.4}$$

$$= P_{Y_i}(y_i) 10^{X^i} \log 10 \tag{5.5}$$

$$= \mathcal{N}(10^{x_i^{true}}, \varepsilon 10^{x_i^{true}}) \cdot 10^{X_i} \cdot \log 10. \tag{5.6}$$

This means that the relevant distribution is in fact

$$X_i = \log_{10} Y_i = \log_{10} y_i^{true} + \log_{10}(1 + \mathcal{N}(0, \varepsilon)),$$

where we can make the expansion

$$\log_{10}(1 + \mathcal{N}(0, \varepsilon)) \simeq \frac{\mathcal{N}(0, \varepsilon)}{\log 10} - \frac{\mathcal{N}(0, \varepsilon)^2}{\log 100} + ... \tag{5.7}$$

Since the leading order term is clearly the dominant term, the logarithm of the cross section may be written as

$$X_i \simeq \log_{10} y_i^{true} + \frac{1}{\log 10}\mathcal{N}(0, \varepsilon) \tag{5.8}$$

Since the distribution has a standard deviation $\varepsilon = 2 \cdot 10^{-3}$, the Gaussian noise covariance should be

$$\left(\frac{\varepsilon}{\log 10}\right)^2 = \frac{(2 \cdot 10^{-3})^2}{(\log 10)^2} = \frac{4 \cdot 10^{-6}}{5.301} \simeq 7.544 \cdot 10^{-7}.$$

5.5 Features and Target

5.5.1 Lagrangian Masses

Not good.

5.5.2 Physical Masses

Adding the Mean Mass

How Prospino calculates NLO terms.

Chapter 6

Results

6.1 Final Kernel

Matérn

Hyperparameters

- LML plot

Gaussian Noise α

6.1.1 Posterior

Drawing Samples

6.2 Mean Relative Deviance

6.3 Learning Curve

$d_L d_L$

$d_L u_L$

6.4 Distributed Gaussian Processes

Adding Experts Improves the Prediction

- Plot of loss function as function of number of experts.
    - Time table

6.5 Comparison with Existing Methods

6.5.1 NLL-fast

- Non-degenerate masses and cross sections

6.5.2 Prospino

- Faster

Chapter 7

Conclusions

- Distributed Gaussian Processes work: more experts give better results
    - Adding the mean was very important
    - The method is sensitive to outliers
    - Works well with relatively few data points (4×8000 points)

Appendix A: The Gaussian Distribution

Bibliography

[1] Marc Peter Deisenroth and Jun Wei Ng. Distributed gaussian processes. *arXiv preprint arXiv:1502.02843*, 2015.

[2] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[3] Devinderjit Sivia and John Skilling. *Data analysis: a Bayesian tutorial*. OUP Oxford, 2006.

[4] Steven Weinberg. *The Quantum Theory of Fields*, volume 1. Cambridge University Press, 1995.

Appendix B: Benchmarks

7.1 When the Error is Known

Most observations are noisy versions of the true function values. In [2] there is an example using additive independent identically distributed Gaussian noise ε with variance σ_n^2 (and standard deviation σ_n). The benchmark function attempts to convert the noisy function at hand to the form

$$y = f(x) + \varepsilon \ , \ \varepsilon \sim \mathcal{N}(0, \sigma_n). \tag{7.1}$$

This may, according to 5.8, be done. Assume a relatively simple function which spans over several orders of magnitude, with a shape that can easily be modelled by the RBF kernel. The function used here is

$$f(x) = 1000 \exp\Big(-\frac{x^2}{10}\Big) \ , \ x \in [0, 30], \tag{7.2}$$

with a corresponding noisy function

$$y = f(x) + \varepsilon f(x), \tag{7.3}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$.

A plot of the true function and noisy function, along with their logarithms, are shown in Fig. (7.1 -7.2). A large standard deviation for the error is used for illustrational purposes. The error is relatively much smaller for the log of functions, and more evenly distributed, as can be seen from the zoom in in Fig (7.3).

For the actual training and prediction, a standard deviation of $\sigma_{std} = 0.001$ was used (to get a similar number to the real case of the cross sections). A single Gaussian process was performed with 500 training points and 4000 test points, using the RBF kernel. The Gaussian process was first tested on the true function, and the error measure was the relative error given by

$$\epsilon_{GP} = \frac{y_{true} - y_{predicted}}{y_{true}}, \tag{7.4}$$

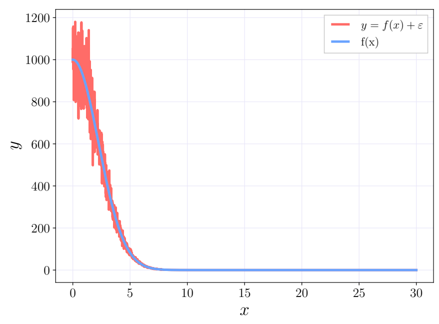


Figure 7.1: The pure and noisy function with $\varepsilon \sim \mathcal{N}(0, \sigma_{std} = 0.1)$.

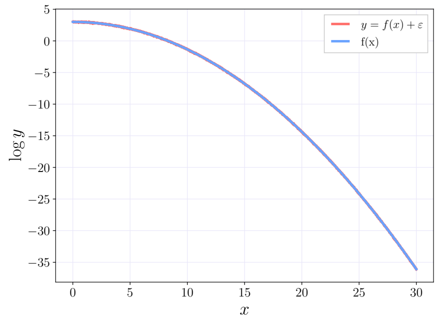


Figure 7.2: The logarithm of the pure and noisy function with $\varepsilon \sim \mathcal{N}(0, \sigma_{std} = 0.1)$.

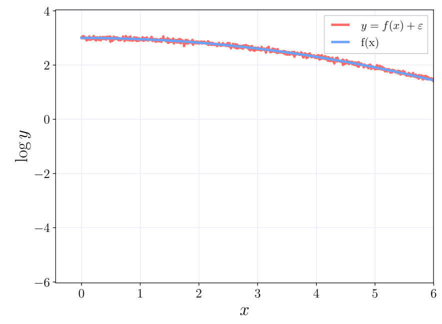


Figure 7.3: The logartihm of the pure and noisy function with $\varepsilon \sim \mathcal{N}(0, \sigma_{std} = 0.1)$.

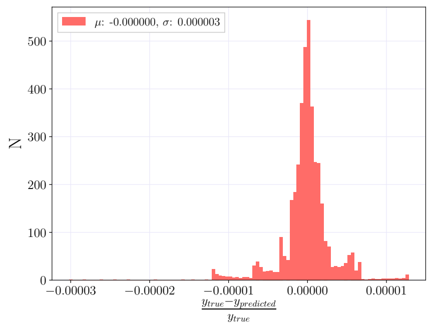


Figure 7.4: Error histogram of the Gaussian process prediction for the true function $f(x)$ using a noiseless RBF-kernel.

where $y_{predicted} = 10^{\log_{10} f_{predicted}}$, and $f_{predicted}$ is what the GP predicts. The optimal hyperparameters for the RBF kernel on the true function were

$$\text{kernel}_{true} = 3.16^2 \exp\left(-\frac{x^2}{5.6^2}\right),$$

with no noise. A histogram of the errors is shown in Fig. (7.4).

The prediction for the error is

$$\left(\frac{\varepsilon}{\log 10}\right)^2 = \frac{(1 \cdot 10^{-3})^2}{(\log 10)^2} = 1.886 \cdot 10^{-7}.$$

Setting $\alpha = 1.886 \cdot 10^{-7}$ gives the best prediction, with a kernel with very similar hyperparameters to the noise-free case

$$\text{kernel}_{noisy} = 3.16^2 \exp\left(-\frac{x^2}{5.82^2}\right). \tag{7.5}$$

It also improves the prediction significantly from the case where no noise is fed to the estimator. A plot of the case where $\alpha = 10^{-10}$ and $\alpha = 1.886 \cdot 10^{-7}$ are shown in Fig. (7.5) and Fig. (7.6), respectively.

The best value for α is in fact very close to our estimate, as can be seen from Fig. (). Note that $\alpha \propto 10^{-8}$ gives a smaller mean than $\alpha \propto 10^{-7}$, but it also has a much larger variance.

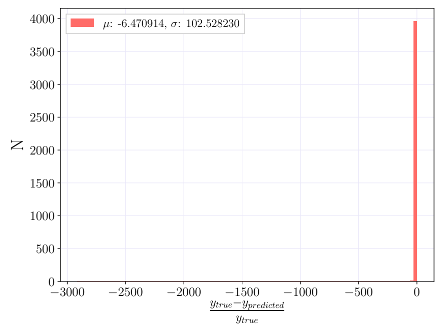


Figure 7.5: Error histogram of the Gaussian process prediction for the noisy function $y(x)$ using a noiseless RBF-kernel ($\alpha = 10^{-10}$).

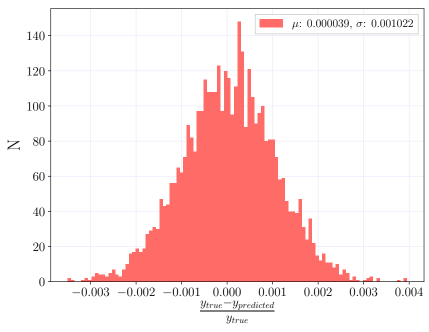


Figure 7.6: Error histogram of the Gaussian process prediction for the noisy function $y(x)$ using an RBF-kernel with $\alpha = 1.886 \cdot 10^{-6}$.

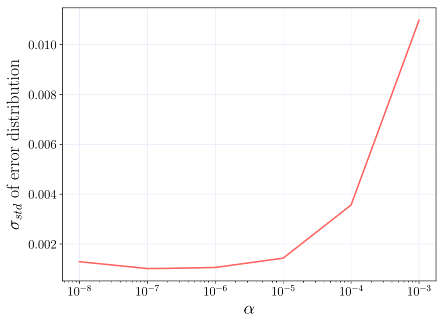


Figure 7.7: Standard deviation of error distribution of the Gaussian process prediction for a varying noise level variance α.