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

1.1 The Standard Model

1.1.1 $U(1) \times SU(2) \times SU(3)$

1.1.2 Spontaneous Symmetry Breaking

1.1.3 The Higgs Mechanism

1.2 Supersymmetry

1.2.1 Why Supersymmetry?

The Hierarchy Problem

Gauge Coupling Unification

Dark Matter

1.2.2 Superfields

Covariant Derivatives

1.2.3 Supersymmetric Lagrangian

1.2.4 Superpartners

1.2.5 R-Parity

1.2.6 The Minimal Supersymmetric Standard Model (MSSM)

Lagrangian

MSSM Field Content

1.2.7 Phenomenology

1.2.8 Current Supersymmetric Limits

Chapter 2

Supersymmetry at the LHC

2.1 Hadronic Cross Sections

2.1.1 Partonic cross sections

2.1.2 Color Factors

2.1.3 Parton Distribution Functions

2.2 Squark-Squark Cross Section

2.2.1 Feynman Diagrams

2.2.2 Calculation to LO

2.2.3 NLO Corrections

Loop diagrams

Renormalization of Divergences

K-factor

Chapter 3

Gaussian Processes

3.1 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

$$P(X|I) + P(\bar{X}|I) = 1, \tag{3.1}$$

$$P(X, Y|I) = P(X|Y, I) \times P(Y|I), \tag{3.2}$$

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

$$P(X|Y, I) = \frac{P(Y|X, I) \times P(X|I)}{P(Y|I)}, \tag{3.3}$$

$$P(X|I) = \int_{-\infty}^{\infty} P(X, Y|I) dY. \tag{3.4}$$

This may seem like an obvious statement: theorem 3.3 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

3.1.1 Priors and Likelihood

Likelihood: probability of the observations given the parameters.

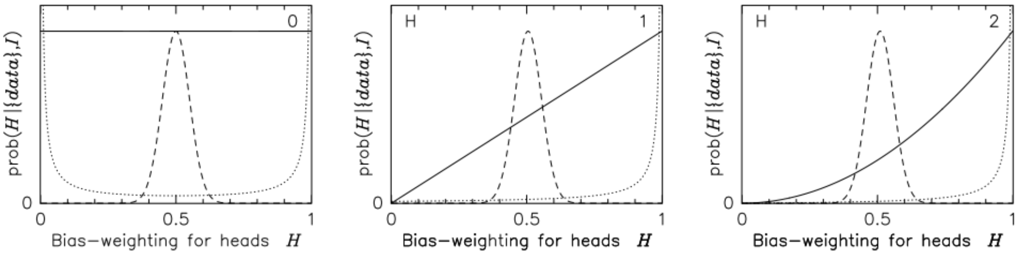


Figure 3.1: From [3].

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{marginal likelihood}}. \tag{3.5}$$

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.

3.1.2 Best Estimate and Reliability

Best estimate X_0 is at maximum of posterior

$$\frac{dP}{dX}\bigg|_{X_0} = 0, \qquad\qquad \frac{d^2P}{dX^2}\bigg|_{X_0} < 0. \tag{3.6}$$

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

$$L = L(X_0) + \frac{1}{2}\frac{d^2}{dx^2}L\bigg|_{X_0}(X - X_0)^2 + ..., \quad L = \log_e\Big[\text{prob}(x|\{data\}, I)\Big] \tag{3.7}$$

Proximate posterior with **Gaussian distribution**

$$\text{prob}(x|\mu,\sigma), \qquad\qquad \sigma = \Big(-\frac{d^2L}{dx^2}\Big)^{-1/2} \tag{3.8}$$

3.1.3 Covariance

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

$$\frac{dP}{dX_i}\bigg|_{X_{0j}} = 0, \tag{3.9}$$

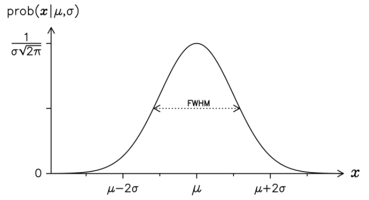


Fig. 2.3 The Gaussian, or normal, distribution. It is symmetric with respect to the maximum, at $x=\mu$, and has a full width at half maximum (FWHM) of about 2.35σ.

Figure 3.2: From [3].

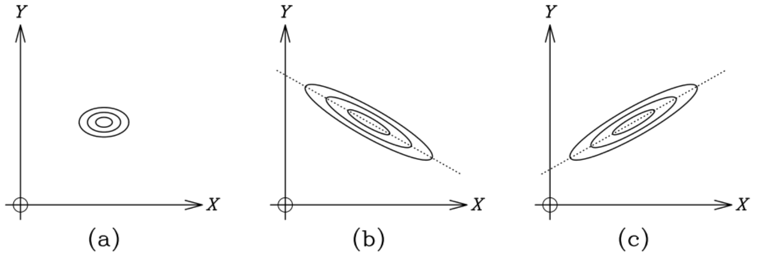


Fig. 3.7 A schematic illustration of covariance and correlation. (a) The contours of a posterior pdf with zero covariance, where the inferred values of X and Y are uncorrelated. (b) The corresponding plot when the covariance is large and negative; $Y+mX=$ constant along the dotted line (where $m>0$), 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 3.3: From [3]

In 2 dim

$$\begin{aligned} L =& L(X_0, Y_0) + \frac{1}{2}\Big[\frac{d^2L}{dX^2}\Big|_{X_0,Y_0}(X-X_0)^2 \qquad (3.10)\\ &+ \frac{d^2L}{dY^2}\Big|_{X_0,Y_0}(Y-Y_0)^2 + 2\frac{d^2L}{dXdY}\Big|_{X_0,Y_0}(X-X_0)(Y-Y_0)\Big] + ... \qquad (3.11)\end{aligned}$$

$$Q = \begin{pmatrix} X-X_0 & Y-Y_0 \end{pmatrix}\begin{pmatrix} A & C \\ C & B \end{pmatrix}\begin{pmatrix} X-X_0 \\ Y-Y_0 \end{pmatrix} \tag{3.12}$$

Q is the **covariance matrix**.

3.2 Gaussian Process Regression

Define mean and covariance function as

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{3.13}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{3.14}$$

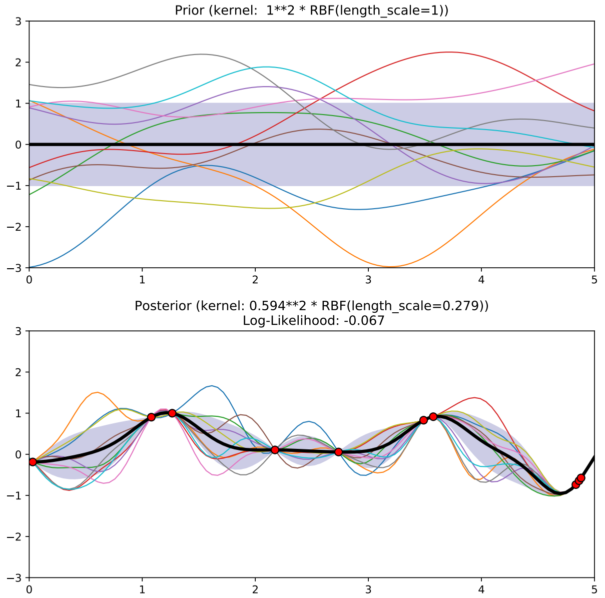


Figure 3.4: From scikitlearn

Write this as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3.15}$$

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

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X,X_*) & K(X_*,X_*) \end{bmatrix}\right) \tag{3.16}$$

Then **condition** distribution on observations

$$\mathbf{f}_* \big| X_*, X, \mathbf{f} \sim \mathcal{N}(K(X_*,X)K(X,X)^{-1}\mathbf{f}, K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)) \tag{3.17}$$

Can draw samples from this ditribution.

3.2.1 Gaussian Noise Model

Assume

$$y = f(\mathbf{x}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{3.18}$$

$$\operatorname{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \qquad \operatorname{cov}(\mathbf{y}) = K(X,X) + \sigma_n^2 \mathbb{I} \tag{3.19}$$

Distribution now becomes

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 \mathbb{I} & K(X,X_*) \\ K(X,X_*) & K(X_*,X_*) \end{bmatrix}\right) \tag{3.20}$$

Conditioned

$$\begin{aligned} \mathbf{f}_* \big| X_*, X, \mathbf{f} \sim \mathcal{N}(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)), \quad & & (3.21) \\ \bar{\mathbf{f}}_* &= K(X_*,X)[K(X,X) + \sigma_n^2 \mathbb{I}]^{-1}\mathbf{y}, & (3.22) \\ \operatorname{cov}(\mathbf{f}_*) &= K(X_*,X_*) - K(X_*,X)[K(X,X) + \sigma_n^2 \mathbb{I}]^{-1} K(X,X_*) & (3.23) \end{aligned}$$

GP prediction

$$\begin{aligned} \bar{f}_* &= \mathbf{k}_*^T (K + \sigma_n^2 \mathbb{I})^{-1}\mathbf{y}, & (3.24) \\ \mathbb{V}[f_*] &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (K + \sigma_n^2 \mathbb{I})^{-1}\mathbf{k}_*. & (3.25) \end{aligned}$$

3.3 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].

3.4 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

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j). \tag{3.26}$$

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

$$\int k(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mu(\mathbf{x}) d\mu(\mathbf{x}') \geq 0, \tag{3.27}$$

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

3.4.1 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

$$\mathbb{E}[|f(\mathbf{x}_k) - f(\mathbf{x}^*)|^2] \to 0 \text{ as } k \to \infty. \tag{3.28}$$

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

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \text{l.i.m.}_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}, \tag{3.29}$$

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

The Radial Basis Function (RBF)

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

$$k_{SE}(r) = \exp\Big(- \frac{r^2}{2\ell^2} \Big), \tag{3.30}$$

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))
```

The Matérn Kernel

The *Matérn class of covariance functions* is given by

$$k_{Matérn}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big)^{\nu} K_{\nu}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big), \tag{3.31}$$

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

$$k_{\nu=p+\frac{1}{2}} = \exp\Big(-\frac{\sqrt{2\nu}r}{\ell}\Big)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\Big(\frac{\sqrt{8\nu}r}{\ell}\Big)^{p-i}. \tag{3.32}$$

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

$$k_{\nu=3/2}(r) = \Big(1 + \frac{\sqrt{3}r}{\ell}\Big)\exp\Big(-\frac{\sqrt{3}r}{\ell}\Big), \tag{3.33}$$

$$k_{\nu=5/2}(r) = \Big(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\Big)\exp\Big(-\frac{\sqrt{5}r}{\ell}\Big). \tag{3.34}$$

```
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].

Chapter 4

Method

4.1 Distributed Gaussian Processes

4.1.1 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.**

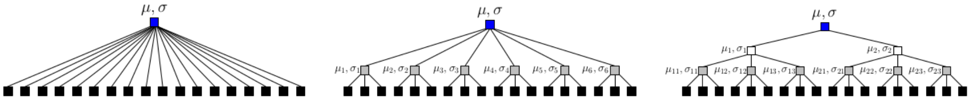
4.1.2 Product-of-Experts

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

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \approx \prod_{k=1}^{M} p_k(\mathbf{y}^{(k)}|\mathbf{X}^{(k)}, \boldsymbol{\theta}) \tag{4.1}$$

$$\log p(\mathbf{y}^{(k)}|\mathbf{X}^{(k)}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{(k)}(\mathbf{K}_{\psi}^{(k)} + \sigma_{\varepsilon}^2\mathbf{I})^{-1}\mathbf{y}^{(k)} - \frac{1}{2}\log|\mathbf{K}_{\psi}^{(k)} + \sigma_{\varepsilon}^2\mathbf{I}| \tag{4.2}$$



(a) 1-layer model.

(b) 2-layer model.

(c) 3-layer model.

Figure 4.1: From [1].

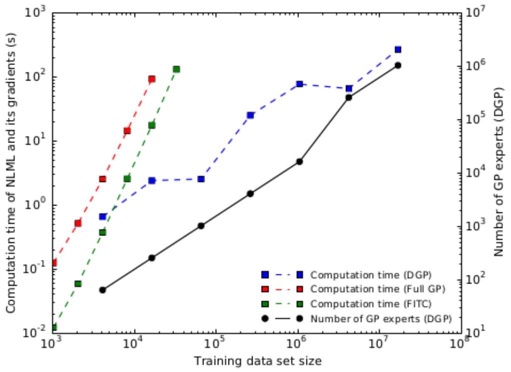


Figure 4.2: From [1].

4.1.3 Algorithm

$$\mu_*^{rbcm} = (\sigma_*^{rbcm})^2 \sum_k \beta_k \sigma_k^{-2}(\mathbf{x}_*)\mu_k(\mathbf{x}_*), \tag{4.3}$$

$$(\sigma_*^{rbcm})^{-2} = \sum_{k=1}^{M} \beta_k \sigma_k^{-2}(\mathbf{x}_*) + \big(1 - \sum_{k=1}^{M} \beta_k\big)\sigma_{**}^{-2}. \tag{4.4}$$

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

$$\mu(\mathbf{x}_*) = \mathbf{k}_*^T(\mathbf{K} + \sigma_\epsilon^2 \mathbb{I})^{-1}\mathbf{y}, \tag{4.5}$$

$$\sigma^2(\mathbf{x}_*) = k_{**} - \mathbf{k}_*^T(\mathbf{K} + \sigma_\epsilon^2 \mathbb{I})^{-1}\mathbf{k}_*. \tag{4.6}$$

Plot of time from the article

4.1.4 Implementing the Algorithm

Parallelizing the Algorithm

4.2 Model Selection

4.2.1 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)

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q))^T M(\mathbf{x}_p - \mathbf{x}_q) + \sigma_n^2 \delta_{pq}. \tag{4.7}$$

The matrix M can have several forms, for example

$$M_1 = \ell^{-2}\mathbb{I}, M_2 = \operatorname{diag}(\boldsymbol{\ell})^{-2}. \tag{4.8}$$

```
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) ;
        $\boldsymbol{\mu}[\text{expert}][x^*] = \mu_*^2$ (mean value from this expert);
        $\boldsymbol{\sigma}^2[\text{expert}][x^*] = \sigma_*^2$ (variance from this expert);
        $\boldsymbol{\sigma}_{**}^2[\text{expert}][x^*] = \sigma_{**}^2$ (variance from initial kernel)
    end
end
for each expert do
    for each $y_*$ in $\boldsymbol{y}_*$ do
        $\mu_* = \boldsymbol{\mu}[\text{expert}][x_*]$ (retrieve relevant values);
        $\sigma_*^2 = \boldsymbol{\sigma}^2[\text{expert}][x^*]$ ;
        $\sigma_{**}^2 = \boldsymbol{\sigma}_{**}^2[\text{expert}][x^*]$ ;
        $\beta = \frac{1}{2}(\log(\sigma_{**}^2) - \log(\sigma_*^2))$ ;
        $(\sigma_*^{rbcm})^{-2}[y_*] += \beta\sigma^{-2} + \big(\frac{1}{n_{experts}} - \beta\big)\sigma_{**}^{-2}$
    end
end
for each expert do
    for each $y_*$ in $\boldsymbol{y}_*$ do
        $\mu_* = \boldsymbol{\mu}[\text{expert}][x_*]$ (retrieve relevant values);
        $\sigma_*^2 = \boldsymbol{\sigma}^2[\text{expert}][x^*]$ ;
        $\sigma_{**}^2 = \boldsymbol{\sigma}_{**}^2[\text{expert}][x^*]$ ;
        $\beta = \frac{1}{2}(\log(\sigma_{**}^2) - \log(\sigma_*^2))$ ;
        $\mu_*^{rbcm}[y_*] += (\sigma_*^{rbcm})^2\beta\sigma_*^{-2}\mu_*$
    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).

4.2.2 Bayesian Model Selection

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

$$p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)}{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)} \tag{4.9}$$

$$p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i) = \int p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)d\mathbf{w} \quad \text{(marginal likelihood)} \tag{4.10}$$

$$p(\boldsymbol{\theta}|\mathbf{y}, X, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)}{p(\mathbf{y}|X, \mathcal{H}_i)} \tag{4.11}$$

$$p(\mathcal{H}_i|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|X)} \tag{4.12}$$

4.2.3 Cross Validation

Divide into k-subsets and use validation and test set. Requires a loss function, e.g. R^2.

4.2.4 Log Marginal Likelihood

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

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi. \tag{4.13}$$

The optimal parameters are found by maximizing the marginal likelihood

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|X, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^T K^{-1}\frac{\partial K}{\partial\theta_j}K^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(K^{-1}\frac{\partial K}{\partial\theta_j}). \tag{4.14}$$

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

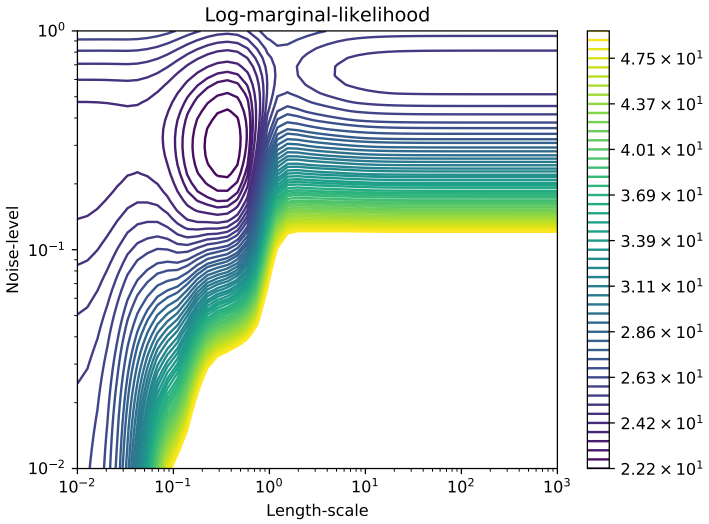


Figure 4.3: Used SciKitLearn. Several local maxima.

4.2.5 Loss functions

Mean Relative Deviance

R-Factor

4.3 Datasets

4.3.1 Data Generation

Prospino

- Possible issues

NLL-Fast

SOFTSUSY

4.4 Feature Distributions

4.4.1 $m_{\tilde{g}} - m_{\tilde{q}}$ Visualizations

4.5 Removing Outliers

4.5.1 Prospino Sets Cross Sections to 0 for $\bar{m}_{\tilde{q}}$

- 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 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. (4.4). 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. (4.5). The points seem to make the prediction for small cross sections worse, as this aerea appears to have a lot of noise.

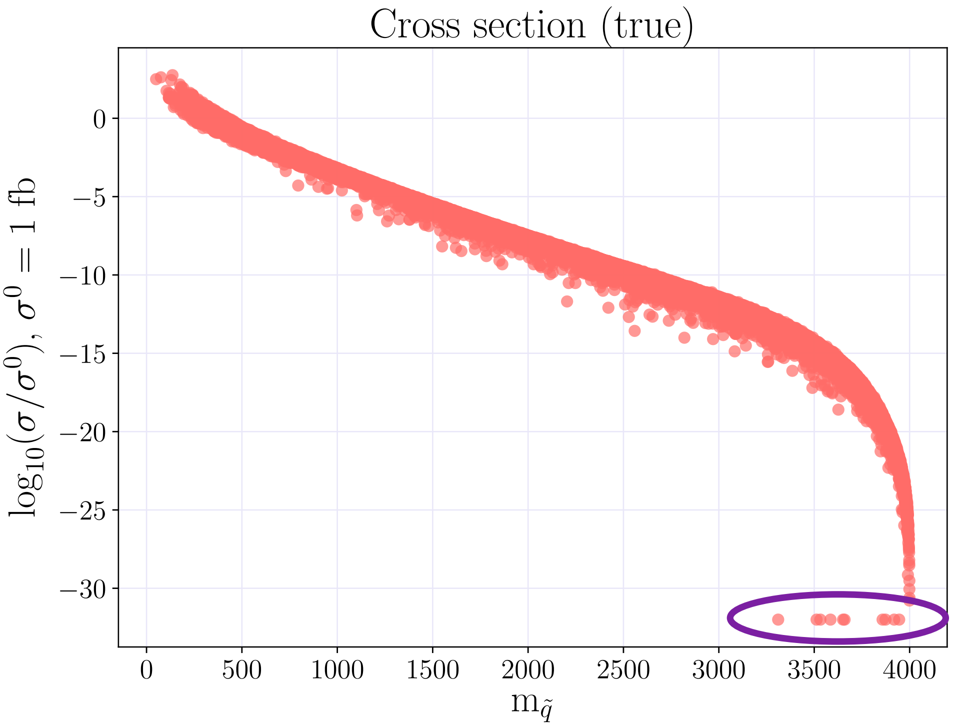


Figure 4.4: True values for the logarithm of NLO cross section as a function of $m_{\tilde{c}_L}$, for 20 000 points from the lin set. Outliers are circled in purple.

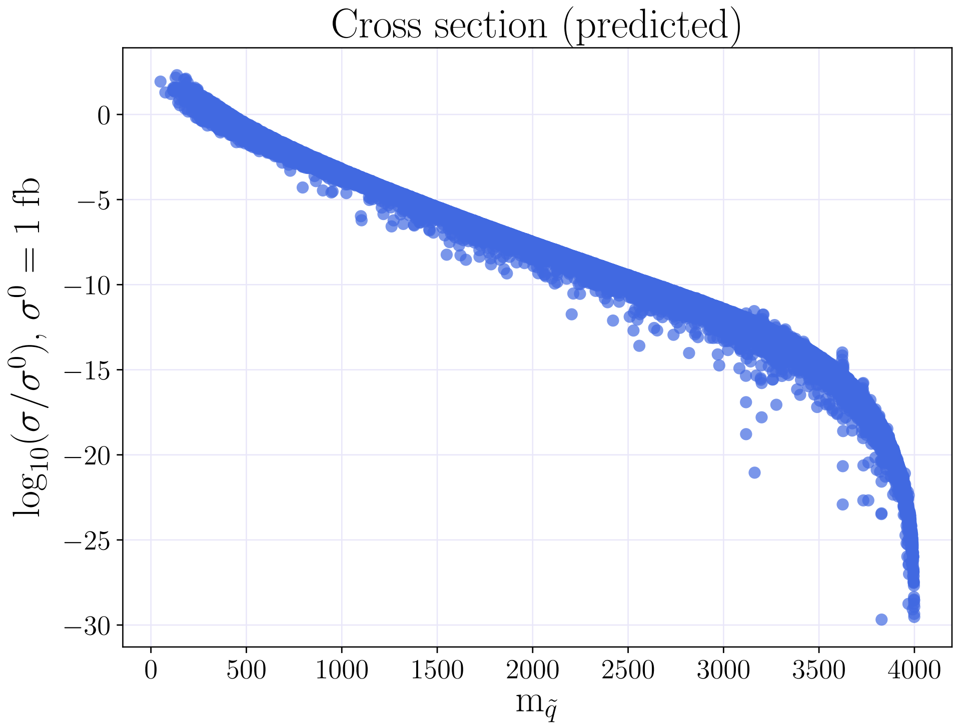


Figure 4.5: Values predicted by Distributed Gaussian processes on the lin set, using 4 experts with 11 000 points each, and the Matérn kernel. There are 20 000 test points, and the outliers from Fig. (4.4) are missing.

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. (4.6).

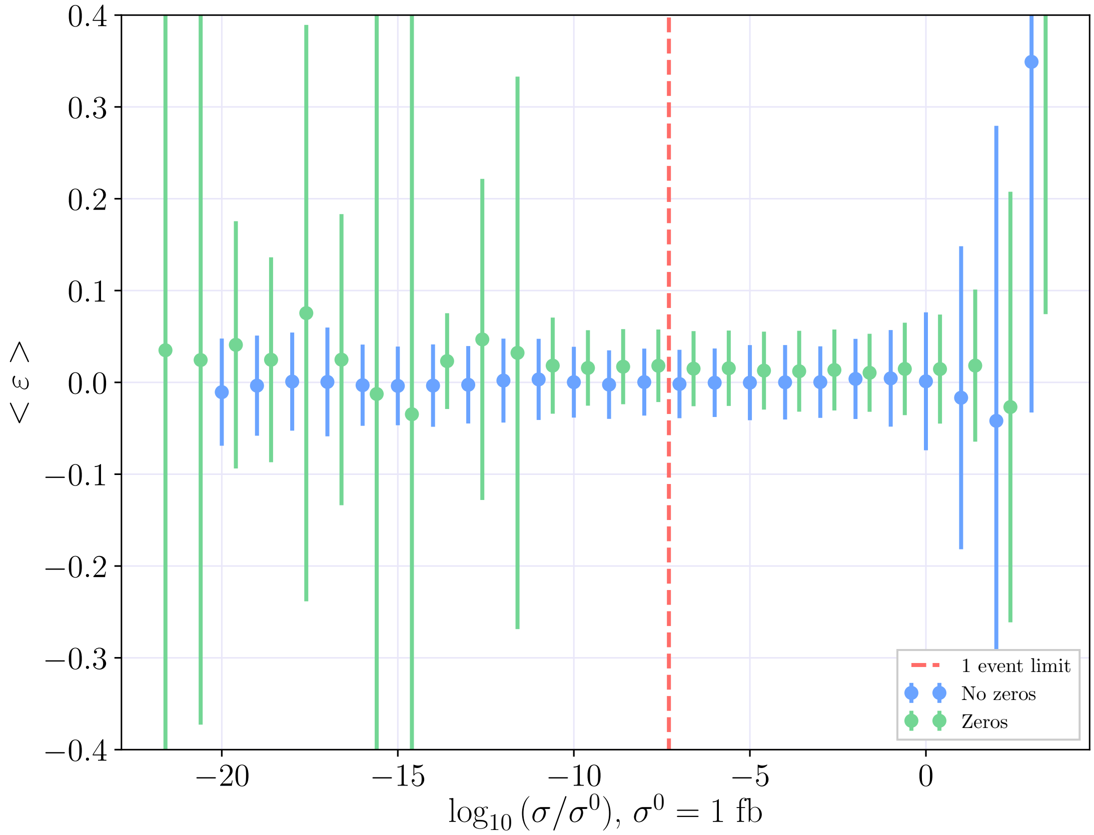


Figure 4.6: Mean standard deviation $\varepsilon = \frac{y_{true}-y_{dgp}}{y_{true}}$ for 4 experts with 11 000 points each from the lin set, with 20 000 test points. The Matérn kernel with $\nu = 1$ was used. The green lines are for before the outliers were removed, and the blue lines are from after.

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

Chapter 5

Results

5.1 Removing Outliers

5.2 When the Error is Known

5.2.1 α in Gaussian Processes

5.2.2 Benchmark

5.3 Feature choice

5.3.1 Largangian Masses

5.3.2 Physical Masses

5.3.3 Cross Section Parameters $\beta - \tilde{q}, m_-^2$

Changing to Vectorized $\boldsymbol{\ell}$

Mean Squark Mass

How Prospino calculates NLO terms.

5.4 Target choice

5.4.1 $\sigma/m_{\tilde{g}}$

5.4.2 Cut at $\sigma \propto 10^{-16}$

Chapter 6

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)

Chapter 7

Appendix A: The Gaussian Distribution

Gaussian Processes The Linear Model Kernel draw plot Covariance Kernel Matern RBF Vectorized length scale Mean Gaussian Noise Prior Posterior Distribution over functions Predicted using weigths and training points Hyperparameters Log Marginal Likelihood

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.