Gaussian Processes for Cross Section Evaluation

by

Ingrid Holm

Thesis

for the degree of

Master of Science

Faculty of Mathematics and Natural Sciences
University of Oslo

May 2018

Abstract

This thesis explores fast evaluation of supersymmetric cross sections using Gaussian processes — a statistical learning method for regression. The time consuming nature of accurate evaluation of higher order cross sections has been a limiting factor in searches for supersymmetry at the LHC and elsewhere. The recent proposition of distributing Gaussian processes between individual estimators, in the form of a robust Bayesian Comittee Machine, allows for the use of larger datasets than before, thus putting the desired accuracy for regression within reach. The distributed Gaussian processes also allow for parallelisation to make evaluation even faster. A distributed Gaussian model for squark pair production in proton-proton collisions was built and tested, predicting cross sections within 10% of next-to-leading order cross sections calculated by `Prospino 2.1` in a fraction of the time.

In loving memory of

This is a dedication to my cat.

Takk

Først og fremst tusen takk til Are Raklev, som har vært en fantastisk veileder og som ga meg en oppgave jeg virkelig har kost meg med. Jeg har satt veldig stor pris på at døren din alltid har vrt åpen, og at du har forklart meg ting gang p gang, uten at jeg har flt meg dum. En takk til Anders og Jeriek også, som svarte på spørsmål om uforståelig statistikk til alle mulige tider, og Jon Vegard som banet veien for min oppgave.

Takk til hele teorigruppa for at det har vært en så hyggelig gruppe å være en del av. Enten det har vrt til lunsj eller kake, eller eldgammel Bailey's funnet under ryddig av kontorer, har jeg alltid kost meg og flt meg hjemme i deres selskap. Særlig takk til August, min gode venn og kontorkamerat, som jeg nok ikke hadde kommet meg gjennom masteren uten (det hadde i hvertfall vært mye kjedeligere).

Tusen takk til mamma, pappa, Georg, farmor og Aba, for at dere er min evige heiagjeng og glede her i livet, og minner meg på at jeg nok overlever masteren også.

De som fortjener mest av æren for min fine tid på Blindern er faministene. Tusen takk Vilde, Pernille, Mari, Elisabeth og Helene. Takk for at dere alltid får meg til å le, for alle merkelige og morsomme ting vi har funnet på, for de sårt trengte Spania-turene våre, og for at dere alltid er der når jeg trenger dere.

Contents

Introduction

The recent discovery of the Higgs boson [1,2] and its precisely measured mass has validated the Standard Model of particle physics at the electroweak energy scale. Between the electroweak scale and the Planck scale, there is a large hierarchy which cannot be explained in the Standard Model without enormous fine-tuning of its parameters. Fine-tuning parameters in order to explain experimental results makes a model theoretically unsatisfactory. If the parameters of the Standard Model are *not* fine tuned, however, large discrepancies arise between the measured and predicted Higgs boson mass. This is called the *hierarchy problem*. The hierarchy problem can be solved by a theoretical extension of the Standard Model called *supersymmetry*. In supersymmetry every Standard Model particle has a supersymmetric partner, whose spin differs by 1/2 from the spin of the Standard Model particle.

Supersymmetry is yet to be observed. One of the experiments searching for supersymmetry is the *Large Hadron Collider* at CERN. Collision data from the Large Hadron Collider is available for 7 TeV, 8 TeV and 13 TeV center of mass energy collisions, and is used to search for supersymmetric signals. In the absence of a signal, global fits are used to find exclusion limits on supersymmetric models, *e.g.* which masses the supersymmetric particles *cannot* have. If a signal is found in the future, global fits will be used to find the parameters of the model. To this end, supersymmetric cross sections need to be calculated with the highest possible accuracy.

Cross sections in quantum field theory become more accurate the more terms are included in the perturbative coupling expansion, such as the next-to-leading and next-to-next-to-leading order terms. In particular, proton-proton cross sections in the *Minimal Supersymmetric Standard Model* — which is the model investigated in this thesis — are highly sensitive to higher order QCD terms.

In [3] the next-to-leading order cross sections for supersymmetric strong processes are calculated. The next-to-leading terms, although important, are time-consuming to evalulate due to the large number of mechanisms. Today, supersymmetric next-to-leading order cross sections can be calculated numerically using tools such as `Prospino 2.1` [4] and `NLL-fast 2.1` [5]. While the former requires long computation times, the latter has a more limited parameter space.

In this thesis a statistical learning method, called *Gaussian processes*, is pro-

posed as a new tool to estimate next-to-leading order supersymmetric cross sections. Gaussian process regression is an algorithm that predicts a Gaussian distributions over function value. Gaussian processes use Bayesian principles to supplement training data with prior knowledge. Seeing as Gaussian processes scale poorly with the size of training data, the *distributed Gaussian processes* proposed in [6] are here used for the larger datasets needed for the desired accuracy of regression. The cross sections investigated in this thesis are for the pair production of squarks in proton-proton collisions.

A basic review of the hierarchy problem and supersymmetry is given in Chapter 1. In Chapter 2 the possibility of supersymmetric production at hadron colliders is discussed, and the cross section for squark pair production is calculated to leading order and next-to-leading order terms are discussed. Basics of Bayesian statistics and Gaussian processes are introduced in Chapter 3, along with the model selection techniques of Bayesian model selection and cross validation. The performance of Gaussian processes for evaluating squark pair production cross sections are investigated in Chapter 4, and the results for the best Gaussian process estimators are shown in Chapter 5, before the conclusion.

1

Physics Background

In this chapter supersymmetry and some of the motivations for an extension of the Standard Model are introduced, assuming familiarity with quantum field theory, the Standard Model of particle physics and some group theory. The Higgs mechanism and the hierarchy problem are reviewed, before supersymmetry is outlined. The Minimal Supersymmetric Standard Model is introduced, with its corresponding field content. Finally, the versions of the Minimal Supersymmetric Standard Model used in this thesis are briefly outlined.

1.1 The Standard Model

The *Standard Model of particle physics* (SM) has successfully explained almost all experimental results in particle physics to date and predicted several phenomena before they were observed. One of the main attributes of the Standard Model is that particles with different values of the *spin* quantum number behave differently. Particles with half-integer and integer spin values are called *fermions* and *bosons*, respectively. Fermions are particles such as *quarks* and *leptons*, which interact through the exchange of force carrying bosons. The Standard Model bosons are the *photon* (electromagnetic interaction), the *gluon* (strong interaction that holds nuclei and atoms together), the W and Z bosons (the weak interaction) and the famously elusive *Higgs boson*, that provides masses for the Standard Model particles. The equations of motion and allowed interactions can all be derived from the *Lagrangian* of the Standard Model. The Lagrangian is invariant to external symmetry transformations under the Lorentz group — the group that represents the symmetries of Special Relativity — which are changes of reference frame and rotations.

The Higgs Mechanism

The Standard Model is a gauge theory based on the internal symmetry group $SU(3)_C \times SU(2)_L \times U(1)_Y$. The $SU(3)_C$ group is the symmetry group for strong interactions, or quantum chromodynamics, and $SU(2)_L \times U(1)_Y$ is the electroweak symmetry group. In order for the particles to obtain masses the electroweak symmetry must be spontaneously broken. To keep charge conservation it is broken down to $U(1)_{\text{em}}$. The symmetry is broken when the Higgs field obtains a non-zero *vacuum expectation value* (vev) — meaning that it has some field value when the governing potential is at its minimum. The Higgs field Φ is a self-interacting complex $SU(2)_L$ doublet whose Lagrangian is given by

$$\mathcal{L}_\Phi = \partial_\mu \Phi^\dagger \partial^\mu \Phi + V(\Phi), \tag{1.1}$$

where the first term is the kinetic term, and the scalar potential describing the Higgs, $V(\Phi)$, is the famous Mexican hat potential

$$V(\Phi) = \mu^2 \Phi^\dagger \Phi + \lambda (\Phi^\dagger \Phi)^2. \tag{1.2}$$

For $\mu^2 < 0$ and $\lambda > 0$ this potential aquires a non-trivial minimum given by

$$|\Phi_0| = \sqrt{\frac{-\mu^2}{2\lambda}} \equiv \frac{v}{\sqrt{2}}, \tag{1.3}$$

where v is the vacuum expectation value. This leads to the Lagrangian developing mass terms for fermions and the gauge bosons Z and $W^\pm$. The mass terms are proportional to v, *e.g.*

$$M_W = \frac{1}{2} v g,$$

where M_W is the mass of the W boson and g is the $SU(2)_L$-coupling. The Higgs' own mass provides one of the strongest arguments for introducing supersymmetry, namely the *hierarchy problem*, which is discussed in the following section.

Another argument for the introduction of supersymmetry is gauge coupling unification. Gauge coupling unification is the assumption that the Standard Model symmetry group is a unified gauge group, *e.g.* $SU(5)$ or $SO(10)$, broken down to $SU(3)_C \times SU(2)_L \times U(1)_Y$ at some high energy scale. This cannot be realised in the SM, but is possible in supersymmetric extensions. However, gauge unification is not discussed in this thesis.

1.2 The Hierarchy Problem

The Higgs boson was discovered at the Large Hadron Collider (LHC) in 2012, and its mass was measured to be around $m_H \sim 125$ GeV [1]. The Higgs mass

in the Standard Model receives fermionic and bosonic loop-contributions to its mass, such as those shown in Fig. 1.1. The expression for the mass can then be written in terms of the bare parameter m_{H0} and the corrections Δm_H

$$m_H^2 = m_{H0}^2 + \Delta m_H^2.$$

Loop diagrams contain divergences, because of integrals over all possible momenta for the virtual particles in the loops. A way to get rid of these divergences is to regularise the expressions. One way to do regularisation is dimensional regularisation. There is also a neat trick that introduces a *cut-off scale*, which sets an upper limit for the integral over momentum. A common choice for the cut-off scale Λ is the Planck scale, as this is where new physics is needed to explain gravity in the Standard Model. The Planck scale is of the order of $\Lambda_P \sim 10^{18}$ GeV. After regularization, the generic mass correction terms are

$$\Delta m_H^2 = -\frac{|\lambda_f|^2}{8\pi^2}\Lambda_P^2 + \frac{\lambda_s}{16\pi^2}\Lambda_P^2 + ..., \tag{1.4}$$

where λ_s is the coupling of the Higgs to a scalar, and λ_f is the Higgs coupling to a fermion. There is one such term per fermion and scalar, with different couplings. The problem now becomes apparent: the correction to the mass squared is proportional to the Planck scale squared, placing the Higgs mass at the order of 10^{18} GeV, yet the mass has been experimentally measured around 125 GeV. There must be some colossal cancellation of terms with a tremendous tuning of the SM parameters in λ_s and λ_f^2 to fit experimental data. Tuning of parameters is undesirable — the model should be as natural as possible.

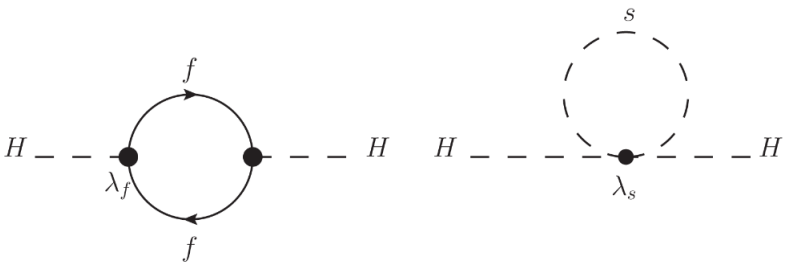


Figure 1.1: Fermion and scalar one-loop corrections to the Higgs mass. Figure from [7].

Supersymmetry provides an elegant solution to the hierarchy problem. In simple terms, supersymmetry introduces fermionic superpartners for the bosons, and vice versa. These are called sparticles, and in unbroken supersymmetry the particles and their corresponding sparticles have identical mass, and their couplings to the Higgs are the same $\lambda_s = |\lambda_f|^2$. In addition, there are twice as

many scalars as fermions, which gives a perfect cancellation of these enormous corrections. Unbroken supersymmetry therefore solves the hierarchy problem. As will be discussed, supersymmetry must in fact be a broken symmetry, and broken supersymmetry is revisited in Sec. 1.3.3.

1.3 Supersymmetry

Supersymmetry is an extension of the Lorentz symmetry, mentioned earlier, in relativistic quantum field theory. Relativistic field theories are invariant under boosts, rotations and translations in spacetime, called Poincaré transformations. A Poincaré transformation of the position four-vector of a particle, x^μ, is given as

$$x^\mu \to x'^\mu = \Lambda^\mu{}_\nu x^\nu + a^\mu, \tag{1.5}$$

where $\Lambda^\mu{}_\nu$ is a Lorentz transformation and a^μ is a translation. The assumption behind supersymmetry is that Nature obeys a non-trivial extension of the related Poincaré algebra, namely the *superalgebra*.

1.3.1 Basic Formalism

Superalgebra

The Poincaré algebra is given by the following commutation relations

$$[P^\mu, P^\nu] = 0, \tag{1.6}$$
$$[M^{\mu\nu}, P^\rho] = i(g^{\nu\rho}P^\mu - g^{\mu\rho}P^\nu), \tag{1.7}$$
$$[M^{\mu\nu}, M^{\rho\sigma}] = i(g^{\nu\rho}M^{\mu\sigma} + g^{\mu\sigma}M^{\nu\rho} - g^{\nu\sigma}M^{\mu\rho} - g^{\mu\rho}M^{\nu\sigma}), \tag{1.8}$$

where P^μ are the generators of translation, and $M^{\mu\nu}$ are the generators of the Lorentz group (boosts and rotations). The Poincaré *superalgebra* is given by the commutation relations in Eqs. (1.6) – (1.8), and the following commutation and anticommutation relations [8]

$$\{Q_A, Q_B\} = \{\bar{Q}_A, \bar{Q}_B\} = 0, \tag{1.9}$$
$$\{Q_A, \bar{Q}_{\dot{A}}\} = 2(\sigma^\mu)_{A\dot{A}}P_\mu, \tag{1.10}$$
$$[Q_A, P^\mu] = [\bar{Q}_A, P^\mu] = 0, \tag{1.11}$$
$$[Q_A, M^{\mu\nu}] = \frac{1}{2}(\sigma^{\mu\nu})^B_A Q_B, \tag{1.12}$$
$$[\bar{Q}_{\dot{A}}, M^{\mu\nu}] = \frac{1}{2}(\bar{\sigma}^{\mu\nu})^{\dot{B}}_{\dot{A}} Q^\dagger_{\dot{B}}, \tag{1.13}$$

where Q_A and $\bar{Q}_{\dot{A}}$ are the superalgebra generators and $A = 1, 2$ and $\dot{A} = 1, 2$ are the indices of two Weyl spinors.

The supersymmetry generators, Q, turn fermions into bosons and vice versa. More specifically, these operators have the following commutation relations with the rotation generator $J^3 = M^{12}$

$$[Q_A, J^3] = \frac{1}{2}(\sigma^3)^B_A Q_B, \tag{1.14}$$

which for the Q_1 generator becomes

$$[Q_1, J^3] = \frac{1}{2}Q_1. \tag{1.15}$$

Using this operator on a state in an irreducible representation of the Poincaré algebra with mass m and spin j_3 gives

$$J^3 Q_1 \left|m, j_3\right\rangle = (j_3 - \frac{1}{2})Q_1 \left|m, j_3\right\rangle, \tag{1.16}$$

thus lowering the spin of the state by 1/2. Similarly, Q_2 would increase the spin. They do not, however, change the mass. This can be seen from Eq. (1.11)

$$P^\mu P_\mu Q_A \left|m, j_3\right\rangle = Q_A P^\mu P_\mu \left|m, j_3\right\rangle = m^2 Q_A \left|m, j_3\right\rangle. \tag{1.17}$$

Superpartners

States that transform into each other via Q_A and $\bar{Q}_{\dot{A}}$ are called *superpartners*. From Eq. (1.17) it can be seen that, in unbroken supersymmetry, the partnering fermions and bosons have the same mass. Superpartners fall into irreducible representations of the superalgebra called *supermultiplets*. The superpartners are the fermionic and bosonic states of the supermultiplets. The supersymmetry generators commute with the gauge generators as well, so superpartners that are part of the same supermultiplet must also have the same electric charges, weak isospin and colour degrees of freedom [9]. The number of bosonic and fermionic degrees of freedom in each supermultiplet is equal, since $\{Q, \bar{Q}\} \sim P$ and P is a one-to-one mapping

$$n_B = n_F. \tag{1.18}$$

The simplest possibility (smallest irreducible representation) for a supermultiplet that obeys Eq. (1.18) has a single Weyl fermion, with $n_F = 2$, and two real scalars, each with $n_B = 1$, assembled to a complex scalar field. This is called a *chiral* multiplet. Chiral multiplets are the only supermultiplets that can contain fermions whose left-handed partners transform differently under the gauge-group than their right-handed partners [9]. Since this is the case for the SM fermions, these must be members of chiral supermultiplets. The superpartners of the quarks and leptons must therefore be spin-0 bosons. The scalar partners

of the fermions are denoted by the prefix 's' for scalar, such as the *squarks* and the *sleptons*. The left- and right-handed pieces of the quarks and leptons are separate two-component Weyl spinors since the SM fermions are Dirac-fermions, so each has its own complex scalar partner. For example, the up-type quark u has two scalar partners, $\widetilde{u}_R$ and $\widetilde{u}_L$, where superpartners are denoted by a tilde $\sim$. The squarks are not right- or lefthanded, the names simply indicate which fermion they are partners of.

The next-to-simplest supermultiplet has a spin-1 vector boson, with $n_B = 2$, and a massless spin-1/2 Weyl spinor, also with $n_F = 2$. The vector bosons are the gauge bosons, and their fermionic superpartners are called *gauginos*. These multiplets are called *vector*, or *gauge* supermultiplets.

Superspace

The elements of the superalgebra and their representations can be described using *superspace*. Coordinates in superspace are given by $z^\pi = (x^\mu, \theta^A, \bar{\theta}_{\dot{A}})$, where x^μ are the well-known Minkowski coordinates, and $\theta^A, \bar{\theta}_{\dot{A}}$ are four *Grassmann numbers* contained in Weyl spinors with indices A and $\dot{A}$. Grassmann numbers are numbers that *anti-commute*, and for θ_A, $\bar{\theta}^{\dot{B}}$ the following is therefore true

$$\{\theta^A, \theta^B\} = \{\theta^A, \bar{\theta}^{\dot{B}}\} = \{\bar{\theta}^{\dot{A}}, \theta^B\} = \{\bar{\theta}^{\dot{A}}, \bar{\theta}^{\dot{B}}\} = 0, \tag{1.19}$$

which gives the relationships

$$\theta_A^2 \equiv \theta_A \theta_A = -\theta_A \theta_A = 0, \tag{1.20}$$

$$\theta^2 \equiv \theta\theta \equiv \theta^A \theta_A = -2\theta_1 \theta_2, \tag{1.21}$$

$$\bar{\theta}^2 \equiv \bar{\theta}\bar{\theta} \equiv \bar{\theta}_{\dot{A}} \bar{\theta}^{\dot{A}} = -2\bar{\theta}^{\dot{1}} \bar{\theta}^{\dot{2}}. \tag{1.22}$$

Any power series of functions of a Grassmann number θ_A therefore terminates as a function of θ_A (or $\bar{\theta}^{\dot{A}}$)

$$f(\theta_A) = a + b\theta_A, \qquad \frac{df}{d\theta_A} = a, \tag{1.23}$$

and the integrals are defined as

$$\int \mathrm{d}\theta_A \equiv 0, \qquad \int \mathrm{d}\theta_A \theta_A \equiv 1. \tag{1.24}$$

Integrals over the Grassmann number Weyl-spinors are

$$\int \theta\theta \ \mathrm{d}^2\theta = 1, \qquad \int \bar{\theta}\bar{\theta} \ \mathrm{d}^2\bar{\theta} = 1, \qquad \int (\theta\theta)(\bar{\theta}\bar{\theta}) \ \mathrm{d}^4\theta = 1, \tag{1.25}$$

where $\mathrm{d}^2\theta = -\frac{1}{4}\mathrm{d}\theta^A \mathrm{d}\theta_A$, $\mathrm{d}^2\bar{\theta} = -\frac{1}{4}\mathrm{d}\bar{\theta}^A \mathrm{d}\bar{\theta}_A$ and $\mathrm{d}^4\theta = \mathrm{d}^2\theta \mathrm{d}^2\bar{\theta}$.

In superspace the component fields of a supermultiplet, as described in Sec. 1.3.1, are united into a single *superfield*.

Superfields

In superspace, any *superfield* Φ — a function on superspace $\Phi(x^\mu, \theta^A, \bar{\theta}_{\dot{A}})$ — can be expanded as a power series in the anti-commuting variables, with components that are functions of the four-vector x^μ. A general superfield can then be written as

$$\begin{aligned}\Phi(x,\theta,\bar{\theta}) =& a(x) + \theta\xi(x) + \bar{\theta}\bar{\chi}(x) + \theta\theta b(x) + \bar{\theta}\bar{\theta}c(x) \\ &+ \bar{\theta}\bar{\sigma}^\mu\theta v_\mu(x) + \bar{\theta}\bar{\theta}\theta\eta(x) + \theta\theta\bar{\theta}\bar{\zeta}(x) + \theta\theta\bar{\theta}\bar{\theta}d(x),\end{aligned} \tag{1.26}$$

where $a(x)$, $b(x)$, $c(x)$, $v_\mu(x)$ and $d(x)$ are bosonic fields, and $\xi_A(x)$, $\bar{\chi}^{\dot{A}}(x)$, $\eta_A(x)$ and $\bar{\zeta}^{\dot{A}}(x)$ are Weyl-spinors, and the matrices $\bar{\sigma}^\mu$ are given by

$$\sigma^0 = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}, \quad \sigma^1 = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, \quad \sigma^2 = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}, \quad \sigma^3 = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}, \tag{1.27}$$

and $\bar{\sigma}^0 = \sigma^0$ and $\bar{\sigma}^i = -\sigma^i,\ i = 1,2,3$.

The general covariant derivatives that are invariant under supersymmetry transformations are defined as

$$D_A \equiv \partial_A + i(\sigma^\mu\bar{\theta})_A\partial_\mu, \tag{1.28}$$

$$\bar{D}^{\dot{A}} \equiv -\partial^{\dot{A}} - i(\sigma^\mu\theta)^{\dot{A}}\partial_\mu. \tag{1.29}$$

These covariant derivatives work on the aforementioned superfields, Φ. A chiral supermutliplet is described by a *chiral superfield*, which must obey the following constraints

$$\bar{D}_{\dot{A}}\Phi(x,\theta,\bar{\theta}) = 0 \qquad \text{(left-handed chiral superfield)}, \tag{1.30}$$

$$D^A\Phi^\dagger(x,\theta,\bar{\theta}) = 0 \qquad \text{(right-handed chiral superfield)}. \tag{1.31}$$

The fields Φ are required to be Lorentz scalars or pseudoscalars, which restricts the properties of their component fields. Using the general form of a superfield, Eq. (1.26), and the constraints from the covariant derivatives, it can be shown that the left- and right-handed chiral fields can be written in terms of their component fields as [7]

$$\begin{aligned}\Phi(x,\theta,\bar{\theta}) =& A(x) + i(\theta\sigma^\mu\bar{\theta})\partial_\mu A(x) - \frac{1}{4}\theta\theta\bar{\theta}\bar{\theta}\Box A(x) + \sqrt{2}\theta\psi(x) \\ &- \frac{i}{\sqrt{2}}\theta\theta\partial_\mu\psi(x)\sigma^\mu\bar{\theta} + \theta\theta F(x),\end{aligned} \tag{1.32}$$

$$\begin{aligned}\Phi^\dagger(x,\theta,\bar{\theta}) =& A^*(x) - i(\theta\sigma^\mu\bar{\theta})\partial_\mu A^*(x) - \frac{1}{4}\theta\theta\bar{\theta}\bar{\theta}\Box A^*(x) + \sqrt{2}\bar{\theta}\bar{\psi}(x) \\ &- \frac{i}{\sqrt{2}}\bar{\theta}\bar{\theta}\theta\sigma^\mu\partial_\mu\bar{\psi}(x) + \bar{\theta}\bar{\theta}F^*(x),\end{aligned} \tag{1.33}$$

where $A(x)$ and $F(x)$ are complex scalars and $\psi_A(x)$ and $\bar{\psi}^{\dot{A}}(x)$ are left-handed and right-handed Weyl spinors, respectively.

Similarly, a *vector superfield* is used to represent the vector supermultiplet. A vector superfield $V(x, \theta, \bar{\theta})$, is obtained by imposing the constraint

$$V(x, \theta, \bar{\theta}) = V^\dagger(x, \theta, \bar{\theta}). \tag{1.34}$$

From Eq. (1.34) the structure of a general vector field in terms of component fields is [9]

$$\begin{aligned} V(x, \theta, \bar{\theta}) =& f(x) + \theta\varphi(x) + \bar{\theta}\bar{\varphi}(x) + \theta\theta m(x) + \bar{\theta}\bar{\theta}m^*(x) \\ &+ \theta\sigma^\mu\bar{\theta}V_\mu(x) + \theta\theta\bar{\theta}\bar{\lambda}(x) + \bar{\theta}\bar{\theta}\theta\lambda(x) + \theta\theta\bar{\theta}\bar{\theta}d(x), \end{aligned} \tag{1.35}$$

where $f(x)$, $d(x)$ are real scalar fields, $\varphi_A(x)$, $\lambda_A(x)$ are Weyl spinors, $m(x)$ is a complex scalar field and $V_\mu(x)$ is a real Lorentz four-vector. Given a chiral superfield Φ, the combinations $\Phi + \Phi^\dagger$, $i(\Phi - \Phi^\dagger)$ and $\Phi^\dagger\Phi$ are all real, vector superfields. In the gauge supermultiplet representation of the superalgebra, the superfield V does not correspond to the promised number of degrees of freedom. This problem is fixed by the *super-gauge* introduced in Sec. 1.3.2.

1.3.2 Supersymmetric Lagrangian

Symmetry transformations of the Lagrangian should leave the action

$$S \equiv \int \mathrm{d}^4x\, \mathcal{L}, \tag{1.36}$$

invariant. This is automatically fulfilled if the Lagrangian only changes by a total derivative. It can be shown that the highest order component fields in θ and $\bar{\theta}$ of the scalar and vector superfields have this property under supersymmetry transformations. To ensure that the action is invariant under supersymmetry transformations, the Lagrangian is redefined such that

$$S = \int d^4x \int \mathrm{d}^4\theta\, \mathcal{L}, \tag{1.37}$$

where there is now an integral over Grassmann numbers. The integral over $\int \mathrm{d}^4\theta$ projects out the highest order component fields, because of the calculus of Grassmann numbers defined earlier.

Restrictions on the supersymmetric Lagrangian, such as invariance under supersymmetric transformations and renormalizability, mean that the most general Lagrangian as a function of the scalar superfields Φ_i is

$$\mathcal{L} = \Phi_i^\dagger\Phi_i + \bar{\theta}\bar{\theta}W[\Phi] + \theta\theta W[\Phi^\dagger], \tag{1.38}$$

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

$$W[\Phi] = g_i \Phi_i + m_{ij} \Phi_i \Phi_j + \lambda_{ijk} \Phi_i \Phi_j \Phi_k, \tag{1.39}$$

where m_{ij} and λ_{ijk} are symmetric. Specifying a supersymmetric Lagrangian with a given superfield content then only requires specifying the superpotential.

Supergauge

The Lagrangian should also be gauge invariant. Consider a general, Abelian or non-Abelian, group G with the Lie algebra of group generators t_a that fulfill

$$[t_a, t_b] = i f_{ab}{}^c t_c, \tag{1.40}$$

where f_{ab}^c are the structure constants. An element g in the group G can be written down in the unitary representation

$$U(g) = e^{-iq\Lambda_a t^a}, \tag{1.41}$$

where Λ_a is a parameter of the representation. The supergauge transformation (global or local) on left-handed chiral superfields Φ_i is thus defined as [7]

$$\Phi \to \Phi' = e^{-iq\Lambda_a T^a} \Phi, \tag{1.42}$$

where q is the charge of the superfield Φ under G, Λ_a are the parameters of the transformation, and T^a are the generators of the gauge group in a chosen representation. For a left-handed superfield Φ_i the Λ^a must also be left-handed superfields, and correspondingly a right-handed superfield $\Phi^\dagger$ must have right-handed superfields $\Lambda^{\dagger a}$.

For the Lagrangian to be gauge invariant the superpotential W must be gauge invariant as well. From the requirement that $W[\Phi] = W[\Phi']$, some restrictions on the superpotential follow

$$g_i = 0 \quad \text{if} \quad g_i U_{ir} \neq g_r, \tag{1.43}$$

$$m_{ij} = 0 \quad \text{if} \quad m_{ij} U_{ir} U_{js} \neq m_{rs}, \tag{1.44}$$

$$\lambda_{ijk} = 0 \quad \text{if} \quad \lambda_{ijk} U_{ir} U_{js} U_{kt} \neq \lambda_{rst}, \tag{1.45}$$

where the indices on U are matrix indices.

The kinetic term must also be invariant under gauge transformations. For this term to be invariant, a gauge compensating vector superfield V^a for each Lie algebra generator T_a with the appropriate gauge transformation is introduced. The kinetic term can then be written as $\Phi^\dagger e^{qV^a T_a} \Phi$, and it transforms as

$$\Phi^\dagger e^{qV^a T_a} \Phi \to \Phi'^\dagger e^{qV'^a T_a} \Phi' = \Phi^\dagger e^{iq\Lambda^{a\dagger} T_a} e^{qV'^a T_a} e^{-iq\Lambda^a T_a} \Phi, \tag{1.46}$$

meaning that the vector superfield V^a must transform as

$$e^{qV'^a T_a} = e^{-iq\Lambda^{a\dagger} T_a} e^{qV^a T_a} e^{iq\Lambda^a T_a}. \tag{1.47}$$

Supersymmetric Field Strength

With the introduction of a vector superfield, V^a, the supersymmetric Lagrangian also requires field strengths, analogous to the electromagnetic field strength $F_{\mu\nu}$. The supersymmetric field strengths are

$$W_A \equiv -\frac{1}{4}\bar{D}\bar{D}e^{-V}D_A e^V, \tag{1.48}$$

$$\bar{W}_{\dot{A}} \equiv -\frac{1}{4}DDe^{-V}\bar{D}_{\dot{A}}e^V, \tag{1.49}$$

where $V = V^a T_a$. W_A ($\bar{W}_{\dot{A}}$) is a left-handed (right-handed) superfield, and it can be shown that the trace $\mathrm{Tr}[W_A W^A]$ is supergauge invariant [7].

The Lagrangian for a supersymmetric theory with (possibly) non-Abelian gauge groups is then

$$\mathcal{L} = \Phi^\dagger e^{qV}\Phi + \delta^2(\bar{\theta})W[\Phi] + \delta^2(\theta)W[\Phi^\dagger] + \frac{1}{2T(R)}\delta^2(\bar{\theta})\mathrm{Tr}[W_A W^A], \tag{1.50}$$

where $T(R)$ is the Dynkin index for correct normalization of the field strength, $\delta^2(\bar{\theta}) = \bar{\theta}\bar{\theta}$ and $\delta^2(\theta) = \theta\theta$. The Dynkin index of the representation R in terms of matrices T_a is given by $\mathrm{Tr}[T_a, T_b] = T(R)\delta_{ab}$.

1.3.3 Soft Supersymmetry Breaking

As previously mentioned, particles and their corresponding sparticles have identical masses in unbroken supersymmetry. Since sparticles have not yet been observed, supersymmetry must be a broken symmetry. In this section soft supersymmetry breaking is considered as a way of providing extra masses to sparticles, without compromising the solution to the hierarchy problem.

The Standard Model particles obtain mass through spontaneous symmetry breaking of the electroweak symmetry, as described in Sec. 1.1. Supersymmetry is also assumed to be spontaneously broken, but at a high, inaccessible scale. This is often called a *hidden sector*, where supersymmetry is broken, and mediated down to the visible sector through some mechanism. Supersymmetry can be spontaneously broken via *soft breaking*. Soft breaking entails adding terms to the Lagrangian that break supersymmetry explicitly, while preserving the cancellations of divergences that fixes the hierarchy problem. These are called *soft terms*, and there are several restrictions on them. That a term is *soft* means that the coupling has mass dimension one or higher, to avoid divergences from loop contributions to scalar masses. The possible soft terms can be written in terms of their component fields

$$\begin{aligned}\mathcal{L}_{soft} = &-\frac{1}{2}M\lambda^A\lambda_A - \left(\frac{1}{6}a_{ijk}A_iA_jA_k + \frac{1}{2}b_{ij}A_iA_j + t_iA_i + \text{ c.c.}\right)\\ &- m_{ij}^2 A_i^* A_j,\end{aligned} \tag{1.51}$$

where λ_A are Weyl spinor fields and A_i are scalar fields. The couplings consist of gaugino masses M for each gauge group, scalar squared-mass terms m_{ij}^2 and b_{ij}, scalar trilinear couplings a_{ijk} and tadpole terms t_i. The soft breaking terms thereby give masses to both the scalar and fermionic superpartners of the SM particles.

The restrictions on the new parameters are necessary to avoid reintroducing the hierarchy problem. If the breaking terms are soft, the correcting mass terms are at most

$$\Delta m_h^2 = -\frac{\lambda_s}{16\pi^2} m_s^2 \ln \frac{\Lambda_{UV}}{m_s^2} + ...,$$

at leading order in the breaking scale Λ_{UV}, where m_s is the soft breaking mass parameter in Eq. (1.51). In this scheme m_s is restricted to $\mathcal{O}(1$ TeV) in order to make the cancellations small.

1.4 The Minimal Supersymmetric Standard Model

The *Minimal Supersymmetric Standard Model* (MSSM) is minimal in the sense that it requires the least amount of superfields introduced in order to have all the SM fields and supersymmetry. The MSSM is based on the minimal extension of the Poincaré algebra in Eqs. (1.6) – (1.8). In this section the field content of the MSSM and the introduction of R-parity is discussed, before the MSSM-24 and CMSSM and their parameters are introduced.

1.4.1 Field Content

As discussed in Sec. 1.3.1, the SM fermions and the Higgs boson are contained in chiral supermultiplets, and the SM gauge bosons are contained in vector supermultiplets. A chiral supermultiplet contains one Weyl spinor with two fermionic degrees of freedom, and two real scalars, with one bosonic degree of freedom each. To form a Dirac fermion, both a left-handed and a right-handed Weyl spinor are needed. These are obtained from a left-handed chiral supermultiplet, and a *different* right-handed chiral supermultiplet. The four fermionic degrees of freedom become two fermions — a particle and an antiparticle — and the four scalar degrees of freedom become four scalar particles, a pair of left- and right-handed scalars, and their antiparticles.

Leptons

For leptons the left-handed chiral superfields are contained in the $SU(2)_L$ doublets L_i, and the right-handed chiral superfields are in $SU(2)_L$ singlets $\bar{E}_i$, given

by

$$L_i = \begin{pmatrix} \nu_i \\ l_i \end{pmatrix} \quad \text{and} \quad \bar{E}_i, \tag{1.52}$$

where $i = 1, 2, 3$ is the generation index. The superfields l_i and $\bar{E}_i$ combine to give charged leptons and sleptons, and ν_i give the (left-handed) neutrinos and sneutrinos. Note that there is no right-handed $\bar{N}_i$. This is a convention, as MSSM is older than the discovery of massive neutrinos.

Quarks

Similarly, for up-type and down-type quarks the left-handed chiral superfields contained in the $SU(2)_L$ doublet Q_i, and the right-handed chiral superfields $\bar{U}_i$ and $\bar{D}_i$ are given by

$$Q_i = \begin{pmatrix} u_i \\ d_i \end{pmatrix}, \ \bar{U}_i \quad \text{and} \quad \bar{D}_i. \tag{1.53}$$

The superfields u_i and $\bar{U}_i$ give the up-type SM quarks and squarks, and the superfields d_i and $\bar{D}_i$ give the down-type SM quarks and squarks. Colour indices are omitted for simplicity.

Gauge Bosons

To represent the gauge bosons, vector supermultiplets are used, as discussed in Sec. 1.3.1. A vector supermultiplet contains two fermionic and two bosonic degrees of freedom, in the form of a massless vector boson and one Weyl spinor of each handedness. As noted in Sec. 1.3.2, one superfield V^a is needed per generator of the algebra T_a for each of the gauge groups in $SU(3)_C$, $SU(2)_L$ and $U(1)_Y$. These vector supermultiplets are represented by the vector superfields

$$C^a, \ W^a, \ \text{and} \ B^0. \tag{1.54}$$

The spin-1 bosons constructed from these superfields are the SM gauge bosons g, W^0, W^+, W^- and B^0. The spin-1/2 superpartners of the gluons g are the *gluinos* $\widetilde{g}$. For the $SU(2)_L \times U(1)_Y$ gauge bosons the superpartners are W^0, $\widetilde{W}^+$, $\widetilde{W}^-$ and $\widetilde{B}^0$, named the *winos* and the *bino*. After electroweak symmetry breaking, B^0 and W^0 mix to give the mass eigenstates Z^0 and γ, and the corresponding mixtures of $\widetilde{B}^0$ and $\widetilde{W}^0$ are called *zino* $\widetilde{Z}^0$ and *photino* $\widetilde{\gamma}$.

Higgs Boson

Finally, supermultiplets are needed for the Higgs. The Higgs is a scalar particle, so it must reside in a chiral supermultiplet, as discussed in Sec. 1.3.1. The

Supermultiplet	Scalars	Fermions	Vectors	$SU(3)_c$	$SU(2)_L$	$U(1)_Y$
Q_i	$(\widetilde{u}_{iL}, \widetilde{d}_{iL})$	(u_{iL}, d_{iL})		3	2	$\frac{1}{6}$
$\bar{u}_i$	$\widetilde{u}^*_{iR}$	$u^\dagger_{iR}$		$\bar{3}$	1	$-\frac{2}{3}$
$\bar{d}_i$	$\widetilde{d}^*_{iR}$	$d^\dagger_{iR}$		$\bar{3}$	1	$\frac{1}{3}$
L_i	$(\widetilde{\nu}_{iL}, \widetilde{e}_{iL})$	(ν_{iL}, e_{iL})		1	2	$-\frac{1}{2}$
$\bar{e}_i$	$\widetilde{e}^*_{iR}$	$e^\dagger_{iR}$		1	1	1
H_u	(H^+_u, H^0_u)	$(\widetilde{H}^+_u, \widetilde{H}^0_u)$		1	2	$\frac{1}{2}$
H_d	(H^0_d, H^-_d)	$(\widetilde{H}^0_d, \widetilde{H}^-_d)$		1	2	$-\frac{1}{2}$
g		$\widetilde{g}$	g	8	1	0
W		$\widetilde{W}^{1,2,3}$	$W^{1,2,3}$	1	3	0
B		$\widetilde{B}$	B	1	1	0

Table 1.1: Gauge and chiral supermultiplets in the Minimal Supersymmetric Standard Model with SM gauge group representations. The index $i = 1, 2, 3$ runs over the three generations of quarks and lepton. Table from [8].

supersymmetry version of the SM Higgs $SU(2)_L$ doublet would mix left- and right-handed superfields in order to give masses to all the fermions, and therefore cannot appear in the superpotential. The minimal allowed Higgs content are two $SU(2)_L$ Higgs doublets H_u and H_d, indexed according to the quarks they give mass to. The superfield doublets are

$$H_u = \begin{pmatrix} H^+_u \\ H^0_u \end{pmatrix}, \qquad H_d = \begin{pmatrix} H^0_d \\ H^-_d \end{pmatrix}, \tag{1.55}$$

where signs indicate eletric charge. These left-handed chiral supermultiplets contain in total four Weyl spinors and eight bosonic degrees of freedom. Three degrees of freedom are used to give masses to the $W^\pm$ and Z^0 bosons through the Higgs mechanism. The remaining five are manifest through the scalar mass eigenstates h^0, H^0, A^0 and $H^\pm$. The Weyl spinors combine into the *higgsinos*. The entire field content of the MSSM is listed in Table 1.1.

1.4.2 MSSM Lagrangian

The Lagrangian for the MSSM may now be constructed from the supermultiplets, and consists of kinetic terms $\mathcal{L}_{\text{kin}}$, supersymmetric field strength terms $\mathcal{L}_V$, the superpotential terms $\mathcal{L}_W$ and the soft breaking terms $\mathcal{L}_{\text{soft}}$,

$$\mathcal{L}_{\text{MSSM}} = \mathcal{L}_{\text{kin}} + \mathcal{L}_V + \mathcal{L}_W + \mathcal{L}_{\text{soft}}. \tag{1.56}$$

The kinetic terms are constructed from the supermultiplets introduced above

$$\begin{aligned}\mathcal{L}_{\text{kin}} =& L_i^\dagger e^{\frac{1}{2}g\sigma W-\frac{1}{2}g'B}L_i + Q_i^\dagger e^{\frac{1}{2}g_s\lambda C+\frac{1}{2}g\sigma W+\frac{1}{3}\cdot\frac{1}{2}g'B}Q_i \\ &+ \bar{U}_i^\dagger e^{\frac{1}{2}g_s\lambda C-\frac{4}{3}\cdot\frac{1}{2}g'B}\bar{U}_i + \bar{D}_i^\dagger e^{\frac{1}{2}g_s\lambda C-\frac{2}{3}\cdot\frac{1}{2}g'B}\bar{D}_i \\ &+ \bar{E}_i^\dagger e^{2\frac{1}{2}g'B}\bar{E}_i + H_u^\dagger e^{\frac{1}{2}g\sigma W+\frac{1}{2}g'B}H_u + H_d^\dagger e^{\frac{1}{2}g\sigma W-\frac{1}{2}g'B}H_d, \end{aligned} \tag{1.57}$$

where g', g and g_s are the couplings of the $U(1)_Y$, $SU(2)_L$ and the $SU(3)_C$.

The supersymmetric field strength contributions with pure gauge terms are

$$\mathcal{L}_V = \frac{1}{2}\text{Tr}\{W^A W_A\}\bar{\theta}\bar{\theta} + \frac{1}{2}\text{Tr}\{C^A C_A\}\bar{\theta}\bar{\theta} + \frac{1}{4}B^A B_A\bar{\theta}\bar{\theta} + \text{c.c.}, \tag{1.58}$$

with the field strengths W_A, C_A and B_A given by

$$W_A = -\frac{1}{4}\bar{D}\bar{D}e^{-W}D_A e^W, \qquad W = \frac{1}{2}g\sigma^a W^a, \tag{1.59}$$

$$C_A = -\frac{1}{4}\bar{D}\bar{D}e^{-C}D_A e^C, \qquad C = \frac{1}{2}g_s\lambda^a C^a, \tag{1.60}$$

$$B_A = -\frac{1}{4}\bar{D}\bar{D}D_A B, \qquad B = \frac{1}{2}g'B^0. \tag{1.61}$$

The possible gauge invariant terms in the superpotential are

$$\begin{aligned} W =& \mu H_u H_d + \mu_i' L_i H_u + y_{ij}^e L_i H_d E_j + y_{ij}^u Q_i H_u \bar{U}_j + y_{ij}^d Q_i H_d \bar{D}_j \\ &+ \lambda_{ijk} L_i L_j \bar{E}_k + \lambda'_{ijk} L_i Q_j \bar{D}_k + \lambda''_{ijk}\bar{U}_i\bar{D}_j\bar{D}_k, \end{aligned} \tag{1.62}$$

where $H_u H_d$ is shorthand for $H_u^T i\sigma_2 H_d$ — and similarly for the other doublet pairs — which is a construction invariant under $SU(2)_L$. The parameters μ and μ_i' are new Lagrangian mass parameters, the y's are the SM Yukawa couplings, and the λ's are new trilinear couplings.

1.4.3 R-parity

The most general supersymmetric Lagrangian with the fields in Sec. 1.4.2 results in couplings that violate lepton and baryon numbers, such as $\mu_i' L_i H_u$ and $\lambda_{ijk}\bar{U}_i\bar{D}_j\bar{D}_k$. However, these violations are under strict restrictions from experiment, such as the search for proton decay, $p \to e^+\pi^0$. This decay would violate both baryon and lepton number by 1 unit, but has not been observed. A new, multiplicative conserved quantity was therefore introduced [10], namely *R-parity*

$$P_R \equiv (-1)^{3(B-L)+2s}, \tag{1.63}$$

where s is spin, B is baryon number and L is lepton number. This quantity is $+1$ for SM particles and scalar Higgs fields, and -1 for sparticles. If R-parity is to be conserved sparticles must therefore always be produced and annihilated in pairs. A further consequence is that there must exist a stable, *lightest supersymmetric particle* (LSP), to which all other supersymmetric particles eventually decay. For this particle to have gone undetected it should have zero eletric and colour charge. These properties make the LSP a good candidate for dark matter [11].

1.4.4 Soft Breaking terms

The MSSM must also have soft breaking terms. The allowed soft breaking terms that conserve R-parity and gauge invariance are, in component fields, as follows

$$\begin{aligned}\mathcal{L}_{\text{soft}} =& -\frac{1}{2}M_1\widetilde{B}\widetilde{B} - \frac{1}{2}M_2\widetilde{W}^a\widetilde{W}^a - \frac{1}{2}M_3\widetilde{g}^a\widetilde{g}^a + c.c. \\ & - a^u_{ij}\widetilde{Q}_i H_u \widetilde{u}^*_{jR} - a^d_{ij}\widetilde{Q}_i H_d \widetilde{d}^*_{iR} - a^e_{ij}\widetilde{L}_i H_d \widetilde{e}^*_{jR} + c.c. \\ & - (m^2_u)_{ij}\widetilde{u}^*_{iR}\widetilde{u}_{jR} - (m^2_d)_{ij}\widetilde{d}^*_{iR}\widetilde{d}_{jR} - (m^2_e)_{ij}\widetilde{e}^*_{iR}\widetilde{e}_{jR} \\ & - (m^2_Q)_{ij}\widetilde{Q}^\dagger_i\widetilde{Q}_j - (m^2_L)_{ij}\widetilde{L}^\dagger_i\widetilde{L}_j \\ & - m^2_{H_u}H^*_u H_u - m^2_{H_d}H^*_d H_d - (bH_u H_d + c.c.),\end{aligned} \tag{1.64}$$

where the M_i are potentially complex valued, introucing six new parameters; the a_{ij} are potentially complex valued, introducing 54 new parameters, b is potentially complex valued, introducing two new parameters; the m^2_{ij} are complex valued and hermitian, introducing 47 new parameters. After removing excessive degrees of freedom found through field redefinitions, the MSSM Lagrangian has introduced a total of 105 new parameters, where 104 come from the soft terms and μ comes from the superpotential.

1.4.5 Radiative Electroweak Symmetry Breaking

As discussed in Sec. 1.1, the SM particles obtain their mass when the Higgs has a field value at the minimum of its governing potential. In supersymmetry, the scalar potential for the Higgs component fields is

$$\begin{aligned}V(H_u, H_d) =& |\mu|^2(|H^0_u|^2 + |H^+_u|^2 + |H^0_d|^2 + |H^-_d|^2) \\ & + \frac{1}{8}(g^2 + g'^2)(|H^0_u|^2 + |H^+_u|^2 - |H^0_d|^2 - |H^-_d|^2)^2 \\ & + \frac{1}{2}g^2|H^+_u H^{0*}_d + H^0_u H^{-*}_d|^2 \\ & + m^2_{H_u}(|H^0_u|^2 + |H^+_u|^2) + m^2_{H_d}(|H^0_d|^2 + |H^-_d|^2) \\ & + [b(H^+_u H^-_d - H^0_u H^0_d) + \text{c.c.}].\end{aligned} \tag{1.65}$$

Using $SU(2)_L$ gauge freedom, this potential can be simplified to

$$\begin{aligned}V(H^0_u, H^0_d) =& (|\mu|^2 + m^2_{H_u})|H^0_u|^2 + (|\mu|^2 + m^2_{H_d})|H^0_d|^2 \\ & + \frac{1}{8}(g^2 + g'^2)(|H^0_u|^2 - |H^0_d|^2)^2 - (bH^0_u H^0_d + \text{c.c.}),\end{aligned} \tag{1.66}$$

at the minimum. Analogous to the SM, $SU(2)_L \times U(1)_Y$ should be broken down to $U(1)_{\text{em}}$ in order to give masses to gauge bosons and SM fermions. It can be shown that Eq. (1.66) has a minimum for finite field values, that this minimum

has a remaining $U(1)_{\text{em}}$ symmetry, and that the potential is bounded from below. For the potential to have a negative squared mass term the following is required

$$b^2 > (|\mu|^2 + m_{H_u}^2)(|\mu|^2 + m_{H_d}^2), \tag{1.67}$$

and the potential is bounded from below if

$$2b < 2|\mu|^2 + m_{H_u}^2 + m_{H_d}^2. \tag{1.68}$$

If $m_{H_u} = m_{H_d}$ at some high scale, the requirements of Eq. (1.67) and Eq. (1.68) cannot be simultaneously satisfied at that scale. However, to one-loop the Renormalization Group Equations (RGE)[1] for $m_{H_u}^2$ and $m_{H_d}^2$ are

$$16\pi^2 \beta_{m_{H_u}^2} \equiv 16\pi^2 \frac{dm_{H_u}^2}{dt} = 6|y_t|^2(m_{H_u}^2 + m_{Q_3}^2 + m_{u_3}^2) + ... \tag{1.69}$$

$$16\pi^2 \beta_{m_{H_d}^2} \equiv 16\pi^2 \frac{dm_{H_d}^2}{dt} = 6|y_b|^2(m_{H_d}^2 + m_{Q_3}^2 + m_{d_3}^2) + ..., \tag{1.70}$$

where y_t and y_b are the top and bottom quark Yukawa couplings, respectively, and $m_{Q_3} = m_{33}^Q$, $m_{u_3} = m_{33}^u$ and $m_{d_3} = m_{33}^d$. Since $y_t \gg y_b$, $m_{H_u}^2$ runs much faster than $m_{H_d}^2$ as they approach the electroweak scale, and $m_{H_u}^2$ can become negative, helping to satisfy Eq. (1.67) and Eq. (1.68). This is called *radiative electroweak symmetry breaking*.

The vector boson masses are known from experiment, and provide constraints on Higgs vacuum expectation values $v_u = \langle H_u^0 \rangle$ and $v_d = \langle H_d^0 \rangle$

$$v_u^2 + v_d^2 \equiv v^2 = \frac{2m_Z^2}{g^2 + g'^2} \approx (174 \text{ GeV})^2. \tag{1.71}$$

The vevs therefore provide a single free parameter, that can be expressed as

$$\tan\beta \equiv \frac{v_u}{v_d}. \tag{1.72}$$

Using the condition for the existence of an extremal point for the potential V

$$\frac{\partial V}{\partial H_u^0} = \frac{\partial V}{\partial H_d^0} = 0, \tag{1.73}$$

parameters can be eliminated. Either the Higgs masses $m_{H_d}^2$ and $m_{H_u}^2$ can be eliminated, or the parameters $|\mu|$ and b. The sign of μ, however, cannot be eliminated, because only the magnitude of mu, $|\mu|$, appears in the potential. The Higgs sector parameters can then either be

$$\tan\beta, \quad \mu, \quad b, \tag{1.74}$$

which are the parameters used in the MSSM-24 discussed in Sec. 1.4.7, or

$$\tan\beta, \quad m_{H_u}, \quad m_{H_d}, \quad \text{sgn } \mu, \tag{1.75}$$

which are the parameters used in the CMSSM discussed in Sec. 1.4.8.

[1] The RGE describes how parameters change as a function of energy scale.

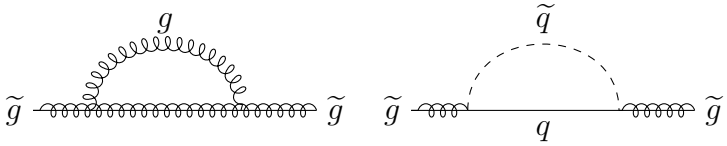


Figure 1.2: One loop contributions to the gluino mass.

1.4.6 Sparticles

The most important sparticles in this thesis are the gluinos and the squarks. In this section the masses of the gluinos and the squarks, as well as the neutralinos, are quickly rewieved.

Gluinos

The gluino is the superpartner of the gluon, which is the boson responsible for the strong interaction. At tree level the gluino does not mix with anything in the MSSM and the mass is the soft term M_3, but with loop contributions such as those in Fig. 1.2 the mass runs quickly with energy μ. The gluino mass with one loop contributions in the $\overline{DR}$ scheme is

$$m_{\widetilde{g}} = M_3(\mu)\left[1 + \frac{\alpha_s}{4\pi}\left(15 + 6\ln\frac{\mu}{M_3} + \sum_{\text{all } \widetilde{q}} A_{\widetilde{q}}\right)\right], \tag{1.76}$$

where the squark-quark loop contributions are given by

$$A_{\widetilde{q}} = \int_0^1 dx\ x\ln\left(x\frac{m_{\widetilde{q}}^2}{M_3^2} + (1-x)\frac{m_q^2}{M_3^2}x(1-x) - i\epsilon\right). \tag{1.77}$$

Squarks

In supersymmetry there are two squarks $\widetilde{q}_L$, $\widetilde{q}_R$ per quark q, and in the MSSM several terms contribute to their masses. For the first two generations, which are the ones relevant to this thesis, the main contributions come from the soft terms and the scalar potential. In the contributions from soft terms one usually assumes that the soft masses are close to diagonal to prevent experimentally challenged flavour changing neutral currents, and provide contributions $-m_Q^2\widetilde{Q}_i^\dagger\widetilde{Q}_i$ and $-m_q^2\widetilde{q}_{iR}^*\widetilde{q}_{iR}$ for an $SU(2)_L$ doublet $\widetilde{Q}_i$ and singlet $\widetilde{q}_i$ with index i, respectively. Note that same generation left-handed squarks share the same soft mass parameter, $m_{Q_i}^2$, which makes the splitting between $m_{\widetilde{d}_L}^2$ and $m_{\widetilde{u}_L}^2$ much smaller than the splitting for $m_{\widetilde{d}_R}^2$ and $m_{\widetilde{u}_R}^2$ in most of the parameter space.

The scalar potential contributes with hyperfine terms of the form $\frac{1}{2}\sum g_a^2(A^*T^aA)^2$. This becomes of the form $(\text{sfermion})^2(\text{Higgs})^2$ when one of the scalar fields A is

a Higgs field and the other is a sfermion. When the Higgs develope vacuum expectation values v, these terms become mass terms for the squarks

$$\Delta_Q = (T_{3F} g^2 - Y_F g'^2)(v_d^2 - v_u^2) = (T_{3F} - Q_F \sin^2\theta_W)\cos 2\beta \; m_Z^2, \tag{1.78}$$

where the isospin T_3, hypercharge Y, and electric charge Q are the charges of the left-handed supermultiplet, F, to which the squark belongs. These terms are the same for other sfermions.

The mass terms of the first generation of squarks are then

$$m^2_{\widetilde{u}_L} = m^2_{Q_1} + \Delta\widetilde{u}_L, \tag{1.79}$$
$$m^2_{\widetilde{d}_L} = m^2_{Q_1} + \Delta\widetilde{d}_L, \tag{1.80}$$
$$m^2_{\widetilde{u}_R} = m^2_{u_1} + \Delta\widetilde{u}_R, \tag{1.81}$$
$$m^2_{\widetilde{d}_R} = m^2_{d_1} + \Delta\widetilde{d}_R, \tag{1.82}$$

with the mass splittings between the same generation left-handed squarks given by

$$m^2_{\widetilde{d}_L} - m^2_{\widetilde{u}_L} = -\frac{1}{2} g^2 (v_d - v_u) = -\cos 2\beta \; m_W^2. \tag{1.83}$$

The second generation has a corresponding structure.

Neutralinos and Charginos

Electroweak symmetry breaking allows the gauge fields to mix. The only requirement is that fields of the same $U(1)_{\text{em}}$ charge mix. This gives fields like the photino and zino, which are supersymmetric partners to the photon and Z-boson. The photino and zino are mixes of the neutral fields $\widetilde{B}^0$ and $\widetilde{W}^0$. However, the gauge fields are also free to mix with the higgsinos, the fermions in the Higgs superfields, giving particles known as *neutralinos*. There are four neutralinos,

$$\widetilde{\chi}_i^0 = N_{i1}\widetilde{B}^0 + N_{i2}\widetilde{W}^0 + N_{i3}\widetilde{H}_d^0 + N_{i4}\widetilde{H}_u^0, \tag{1.84}$$

for $i = 1, 2, 3, 4$, where N_{ij} indicates how much of each component field is mixed in the neutralino. There are also charged particles, known as *charginos*, which are similar to the neutralinos but mixes $\widetilde{W}^+$, $\widetilde{H}_u^+$, $\widetilde{W}^-$ and $\widetilde{H}_d^-$.

1.4.7 MSSM-24

As noted in Sec. **??**, the MSSM introduces 105 new parameters, where 104 come from soft terms and one comes from the scalar superpotential. However, experimental results put restrictions on many of these parameters. One model with a

restricted number of parameters is the MSSM-24, where the number of parameters has been reduced to 24. This is the model used to generate data in this thesis.

Off-diagonal terms in the slepton and squark mass matrices $(m_f^2)_{ij},\ i \neq j$ could induce flavour-changing neutral current processes, such as $\mu \to e\gamma$. Since these processes have strong experimental bounds, the squark and lepton mass matrices are assumed to be diagonal,

$$(m_f^2)_{ij} = \text{diag}(m_{\widetilde{f}1}^2, m_{\widetilde{f}2}^2, m_{\widetilde{f}3}^2),\ f = u, d, e, Q, L. \tag{1.85}$$

Another restriction comes from CP-violation. To avoid inducing large CP-violating phases, the gaugino masses and trilinear couplings are assumed to be real

$$\text{Im}(M_1) = \text{Im}(M_2) = \text{Im}(M_3) = \text{Im}(A_0^u) = \text{Im}(A_0^d) = \text{Im}(A_0^e) = 0. \tag{1.86}$$

Finally, as there is a one-to-one correspondence between the trilinear terms in the SM y_{ij}^f and in the MSSM a_{ij}^f, these are taken to be related through proportionality constants

$$a_{ij}^u = A_0^u y_{ij}^u,\ a_{ij}^d = A_0^d y_{ij}^d,\ a_{ij}^e = A_0^e y_{ij}^e. \tag{1.87}$$

The parameters of the MSSM-24 are then the following

$M_1, M_2, M_3,$	Gaugino mass parameters,
A_0^u, A_0^d, A_0^e	Trilinear couplings,
$\tan\beta, \mu, b,$	Higgs parameters,
$m_{\widetilde{Q}_1}^2, m_{\widetilde{Q}_2}^2, m_{\widetilde{Q}_3}^2$	Squark mass parameters,
$m_{\widetilde{u}_1}^2, m_{\widetilde{u}_2}^2, m_{\widetilde{u}_3}^2$	
$m_{\widetilde{d}_1}^2, m_{\widetilde{d}_2}^2, m_{\widetilde{d}_3}^2,$	
$m_{\widetilde{L}_1}^2, m_{\widetilde{L}_2}^2, m_{\widetilde{L}_3}^2,$	Slepton mass parameters
$m_{\widetilde{e}_1}^2, m_{\widetilde{e}_2}^2, m_{\widetilde{e}_3}^2.$	

In addition, a scale must be given where these parameters are defined, because they run with energy. In this thesis the scale is set to $Q = 1$ TeV.

1.4.8 Constrained MSSM

Another version of the MSSM with (much) fewer parameters is the *Constrained MSSM* (CMSSM), also called *minimal supergravity* (mSUGRA). As mentioned in Sec. 1.3.3 supersymmetry is assumed to be broken in a *hidden sector*, which is some non-accessible high energy scale where the fields of the sector have very small or no direct couplings to the fields in the visible sector. These fields aquire

a nonzero vacuum expectation value that is mediated down to the visible sector, via an interaction that is common for both sectors.

In the CMSSM the breaking mechanism used is the *Planck-Mediated Symmetry Breaking* (PMSB). In PMSB some gravity mechanism at the Planck energy scale $\Lambda_P \sim 10^{18}$ GeV mediates the breaking of supersymmetry, which explains the name minimal super*gravity*. In this scheme it is assumed that soft terms only provide the following four (and a half) parameters

$$m_{1/2}, \qquad m_0^2, \qquad A_0, \qquad \tan\beta, \qquad \text{sgn}\ \mu, \tag{1.88}$$

which give, at the grand unifying scale $Q = M_{\text{GUT}}$, for the soft breaking terms

$$M_3 = M_2 = M_1 = m_{1/2}, \tag{1.89}$$

$$(m_Q^2)_{ij} = (m_u^2)_{ij} = (m_d^2)_{ij} = (m_e^2)_{ij} = (m_L)_{ij}^2 = \text{diag}\ (m_0^2), \tag{1.90}$$

$$a_{ij}^u = A_0 y_{ij}^u,\ a_{ij}^d = A_0 y_{ij}^d,\ a_{ij}^e = A_0 y_{ij}^e, \tag{1.91}$$

$$m_{H_u}^2 = m_{H_d}^2 = m_0^2. \tag{1.92}$$

The mass parameters are then run down to the electroweak scale, *e.g.* including the hyperfine mass splitting for $m_{\widetilde{u}_L}$ and $m_{\widetilde{d}_L}$.

2

Supersymmetry at Hadron Colliders

The Large Hadron Collider (LHC) at CERN is currently the largest, and one of the most important particle physics experiments in the world. In this chapter, some of the advantages and challenges of using hadron colliders are discussed, along with techniques for moving from theory to observable signals. A short description of supersymmetric phenomenology at hadron colliders is followed by the discussion of some current bounds on the squark and gluino masses. Finally, the cross section for squark pair production is calculated to leading order, and next-to-leading order contributions are investigated.

2.1 Hadron Colliders

Hadron colliders are discovery machines. Because the quarks and gluons that constitute hadrons can have a wide range of energies — as will be discussed in Sec. 2.1.1 — and the maximum energies reached by circular accelerators are very high, hadron colliders are a potential source of discovery and surprises. Lepton colliders, on the other hand, have clean collision kinematics and are therefore precision machines. Lepton colliders can be used for precision measurements, *e.g.* of the electroweak interaction. Since supersymmetry has not been discovered yet, hadron colliders are used to look for the wide range of possible supersymmetry productions. This section commences by discussing synchrotron radiation, and why it makes hadrons more suitable than leptons for circular accelerators at high energies. Then the hadron momentum, more specifically how momentum is distributed between the constituent quarks and gluons, is given as a motivation to use parton distribution functions. Luminosity is briefly discussed, before the LHC is given as an example hadron collider.

Colliding hadrons makes it possible to reach very high center-of-mass energies, as they allow for the use of circular accelerators. While most linear colliders

collide leptons, such as e^-e^+, leptons are not well-suited for circular accelerators at high energies because of *synchrotron radiation*. Synchrotron radiation is the radiation of energy from a charged particle being accelerated. The power radiated by a relativistic charged particle forced to move in circular motion with radius R is given in natural units by the Larmor formula [12] multiplied by the 4th power of the Lorentz factor

$$P = \frac{q^2\beta^4\gamma^4}{6\pi R^2}, \tag{2.1}$$

where $\beta = v/c$ and $\gamma \approx E/mc^2$, and E, m and v are the energy, mass and velocity of the particle, respectively. The radiated power goes as $P \sim \frac{E^4}{m^4R^2}$. Light particles, such as leptons, therefore lose a lot of energy in circular accelerators unless the radius, R, of the accelerator is made very large. Because the proton is much heavier than the electron this effect becomes relatively small, allowing the LHC to operate at energies currently as high as 13 TeV. The circular form means the accelerating structures can be reused as many times as one desires, thus putting 'no limits' on the energies obtained.

There are, of course, limits. At energies of around $5 - 7$ TeV per particle, synchrotron radiation becomes an important effect also for protons. The photons emitted from synchrotron radiation hit the walls of the vacuum chamber walls, where they can interact with electrons and cause *electron clouds*. Electron clouds occur when electrons are ejected from the walls of the vacuum chamber and accelerated towards a passing beam bunch. When the electrons reach the center of the chamber, however, the beam bunch has passed, and the now-energetic electrons hit the other side of the chamber, producing more free electrons. This becomes a 'cloud' of electrons that can in turn affect the particle beam.

The very high energies at the LHC also require strong magnets to bend the beam trajectories. The *Large Hadron Collider* (LHC) is a circular particle accelerator that collides protons. It has a circumference of 27 kilometres, and uses magnets to bend the particle beams in circular motion. The two particle beams travel in opposite directions in separate beam pipes, before they are made to collide at the collision experiments, such as ATLAS and CMS. The beams are guided through the pipes by *superconducting magnets*, that are used to bend and focus the beams. To bend the beam, dipole magnets are used, while quadrupole magnets focus the beams in the transverse directions. For the magnets to be superconducting, which means that electricity flows almost without resistance or loss of energy, the magnets must be held at a very low temperature. A distribution system of liquid helium connected to most of the accelerator keeps a temperature of -271.3 °C.

2.1.1 Parton Distribution Functions

Unlinke synchrotron radiation, which is an issue for leptons and hadrons alike at differing energy scales, *uncertainty in longitudinal momentum* is a challenge specific to hadron collisions. Hadrons are made up of valence and sea quarks, which make the kinematics of collisions very difficult to determine. Valence quarks are the quarks used to classify a hadron, such as *uud* for the proton and *udd* for the neutron. In addition to these, hadrons contain a sea of virtual quarks and gluons from the strong interaction. The quarks and gluons, called *partons*, distribute the hadron momentum somewhat stochastically amongst themselves. Since the distribution of energy and momentum is *a priori* unknown, so is the momentum of the ingoing parton, and thus the center of mass energy of the partonic collision. There is, however, some experimental knowledge of the statistical distribution of longitudinal momentum contained in *parton distribution functions*.

Partonic cross sections are calculated for the collision of *partons*. The partonic cross section is a function of the squared center-of-mass energy of the partons, s, which can be written as a fraction of the total squared center-of-mass energy of the colliding hadrons

$$s = x_1 x_2 S, \tag{2.2}$$

where S is the center of mass energy of the colliding hadrons, and x_i is the momentum fraction of the parton p_i. To obtain the total cross section, the partonic cross section is integrated over all possible fractions x_1 and x_2 using *parton distribution functions* (PDFs) $f_i^{h_k}(x_i)$. The PDFs determine the distribution over the fraction of total momenta for a parton p_i in the hadron h_k. For example, the fraction for an up-flavour quark in a proton would be much larger than that for a top-flavour quark at LHC energies. Examples of parton distributions for u, $\bar{u}$, d, $\bar{d}$, $s = \bar{s}$ and g are shown in Fig. 2.1 for two different values of the factorisation scale Q. Integrating over parton distribution functions and summing over partonic cross sections yields the *total hadronic cross section*

$$\sigma(S, Q^2) = \sum_{i,j=g,q,\bar{q}} \int_\tau^1 dx_1 \int_{\tau/x_1}^1 dx_2 f_i^{h_1}(x_1, Q^2) f_j^{h_2}(x_2, Q^2) \hat{\sigma}_{ij}(x_1 x_2 S, Q^2)\Big|_{\tau = 4m^2/S}. \tag{2.3}$$

where $\hat{\sigma}_{ij}$ are the partonic cross sections, m is the mean mass of final state particles, Q is the factorisation scale, and initial state partons $i, j = g, q, \bar{q}$ are summed over. In this thesis the CTEQ6 parton distribution functions from the LHAPDF Fortran library [13] are used.

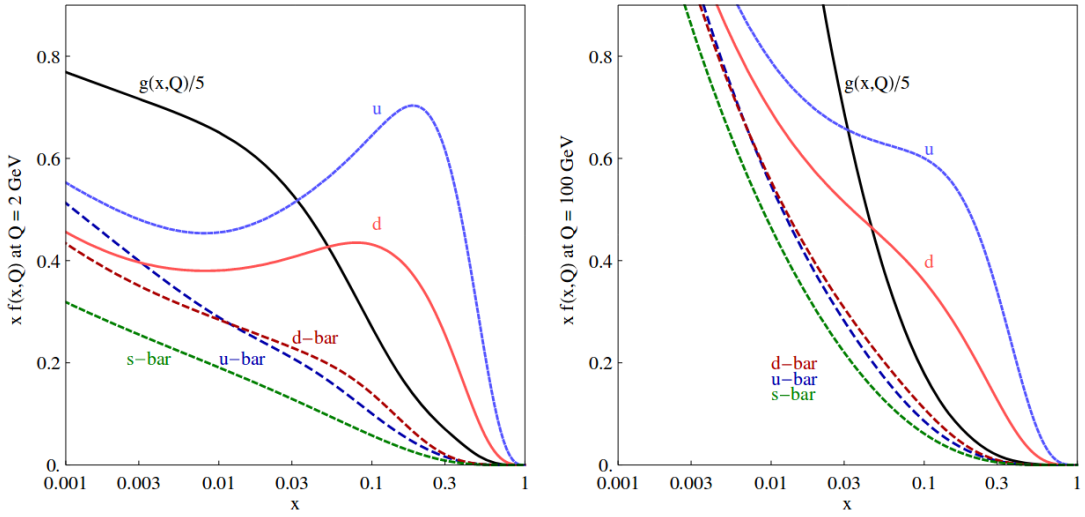


Figure 2.1: The CT14 parton distribution functions of the proton at factorisation scales $Q = 2$ TeV and $Q = 100$ GeV for u, $\bar{u}$, d, $\bar{d}$, $s = \bar{s}$ and g. Figure from [14].

2.1.2 Luminosity

The total number of expected events at the LHC, N_{exp}, is the product of the cross section σ_{exp} and the time integral over the *instantaneous luminosity*

$$N_{\text{exp}} = \sigma_{\text{exp}} \times \int \mathrm{d}t \; \mathscr{L}(t). \tag{2.4}$$

The *instantaneous luminosity* $\mathscr{L}$ is a measure of the number of collisions in a collider per unit time. For two bunches containing n_1 and n_2 particles colliding with a frequency f, the instantaneous luminosity can be written in simple terms as

$$\mathscr{L} = f \frac{n_1 n_2}{4\pi \sigma_x \sigma_y}, \tag{2.5}$$

where σ_x and σ_y characterise the transverse beam sizes in the horisontal and vertical directions. The *integrated luminosity*, $\mathscr{L}_{\text{int}}$, is the time integral over the instantaneous luminosity. The integrated luminosity is often given in units of inverse cross sections, *e.g.* inverse femtobarn $1\ \text{fb}^{-1} = 10^{43}\ \text{m}^{-2}$.

The integrated luminosity of a dataset can be used to set limits on the size of interesting cross sections, by using the relationship between number of particles and integrated luminosity in Eq. (2.4). Setting $N_{\text{exp}} = 1$ for a single produced particle, and using the integrated luminosity $\mathscr{L}_{\text{int}} = 20.3\ \text{fb}^{-1}$ for the 8 TeV LHC dataset considered in this thesis, the lower limit on interesting cross sections is

$$\sigma = \frac{1}{20.3\ \text{fb}^{-1}} \approx 0.05\ \text{fb}. \tag{2.6}$$

Cross sections below $\sigma \sim \mathcal{O}(10^{-3}\text{ fb})$ correspond to less than 0.02 produced particles in the 8 TeV dataset, and will therefore be considered less important in this thesis.

2.2 Supersymmetry Phenomenology at the LHC

Phenomenology is the application of theory to predict the properties of physics models at experiments such as the Large Hadron Collider discussed in Sec. 2.1. This section deals with the possibility of searches for supersymmetry at hadron colliders using missing transverse energy and quark jets. Some current bounds on sparticles are discussed as well.

2.2.1 Jets and Transverse Kinematics

The longitudinal momentum in hadron collisions is difficult to estimate — as discussed in Sec. 2.1.1 — so the kinematics are considered mainly in the *transverse plane*.[1] Since the particle beams are collided in the longitudinal direction the total momentum in the transverse directions is zero before the collision. Conservation of momentum then states that the total transverse momentum must remain zero *after* the collision. This balance in momentum can be combined with the signals left by particles in the detectors to determine the *missing transverse energy* of an interaction.

The *missing transverse energy* is used to search for weakly interacting particles that do not leave traces in the detectors, and is defined as

$$\not{E}_T = \left| \sum \mathbf{p}_T \right|, \tag{2.7}$$

where the sum runs over the transverse momenta of all visible final state particles. Any excess of momentum in one direction indicates that there is missing momentum in the opposite direction. This is the missing transverse energy, which is attributed to weakly interacting particles, such as neutrinos ν.

Particles that *do* leave traces in detectors are hadrons. Partons produced in proton collisions made up *hadronic jets*. Jets are collimated bunches of final-state partons that appear in hard interactions. The partons produced in a proton-proton collision cannot remain isolated because of *colour confinement*, which prevents coloured particles from being in unbound states at LHC energies. Quarks and gluons therefore bunch together and form hadrons — a process called *hadronisation*. The hadronisation leads to a collimated spray of hadrons, which are the jets. Jets can leave traces in both calorimeters, and the signals are combined to form reconstructed jets. These are in turned used to investigate the

[1] The plane that is transverse to the direction of the beam.

transverse kinematics of the collision, *e.g.* by determining the aforementioned missing energy.

2.2.2 Searches for Supersymmetry

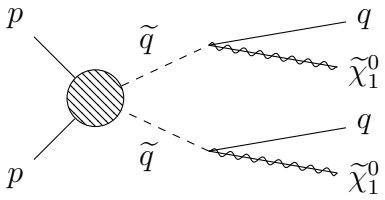
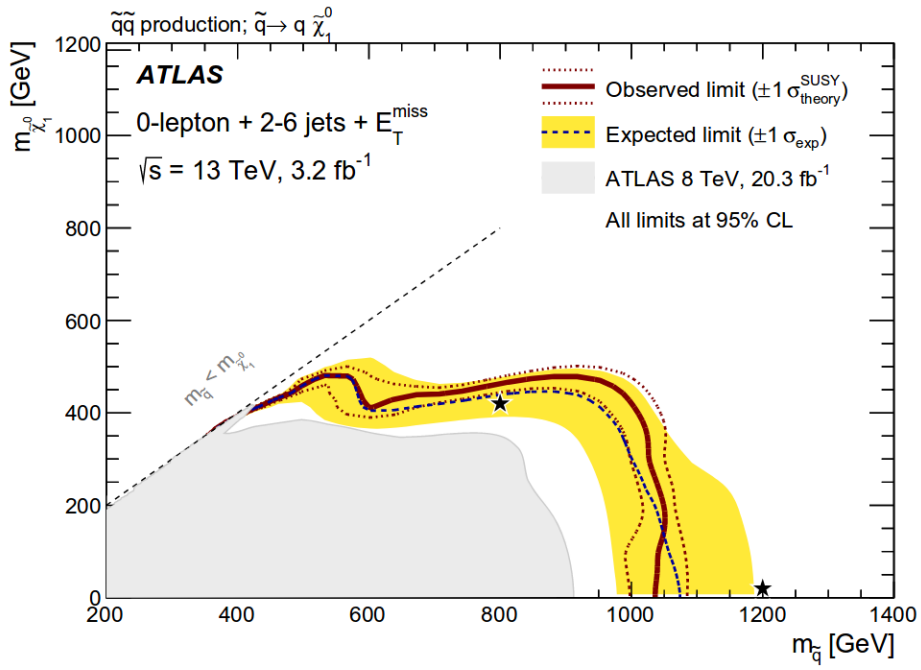


Figure 2.2: Possible signature of a supersymmetric QCD process, with two quark jets and large missing transverse energy in the final state.

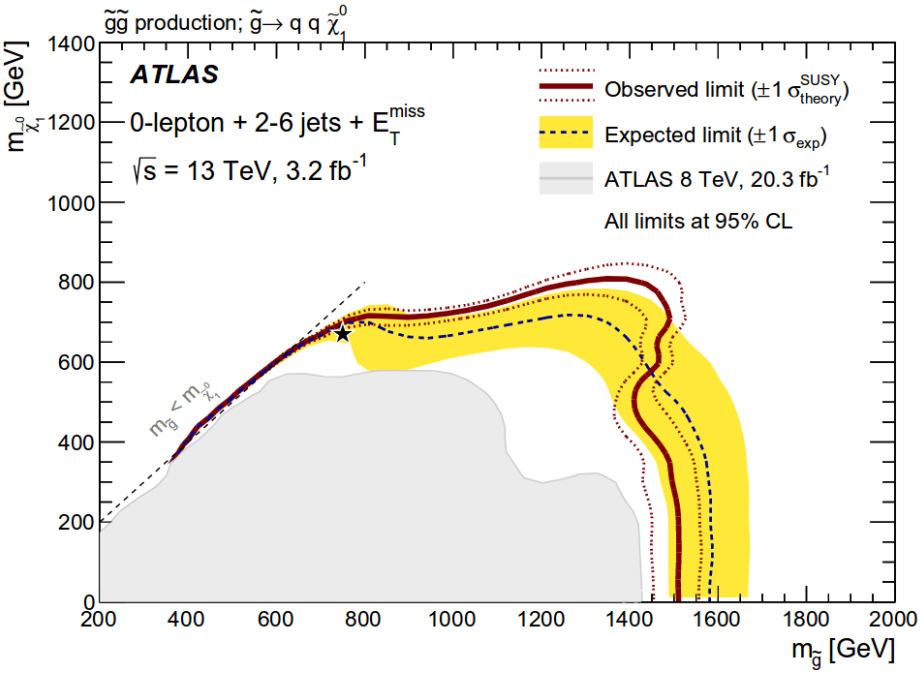
Supersymmetry production at hadron colliders will likely be in the form of QCD processes, as the colliding particles are quarks and gluons. The traces left in detectors by such processes will be closely tied to the conservation of R-parity, in that all sparticles in the MSSM are produced in pairs and eventually decay to the LSP, as mentioned in Sec. 1.4.3. If the LSP is indeed only weakly interacting it is difficult to detect, and therefore searched for at the LHC using missing transverse energy as described in Sec. 2.2.1. The momentum in Eq. (2.7) can be from jets, as these should be common in supersymmetry processes at the LHC. Since the LSP is assumed colour neutral, the colour charge of gluinos and squarks must end up in coloured SM particles in the form of jets. An example of squark pair production with subsequent jet initiators is shown in Fig. 2.2. The hadronic jets produced from supersymmetric processes have significant background from SM processes. Allowing for R-parity violation opens up for more final-state possibilities. The LSP can then decay, and sparticles can be produced one at a time.

2.2.3 Current Bounds on Sparticles

Searches and corresponding limits are available from the data recorded in 2015 by the ATLAS experiment in $\sqrt{s} = 13$ TeV proton-proton collisions at the LHC, with an integrated luminosity of 3.2 fb^{-1}. The analysis in [15] included searches for jets and missing transverse energy. Simplified models were assumed, considering only gluinos and the first two generations of squarks with R-parity conservation and the lighest neutralino as the lightest supersymmetric particle. At 95% confidence level, exclusion limits of the gluino and squark masses are set at 1.51 TeV and 1.03 TeV, respectively [15], assuming a *massless lightest neutralino*. A plot of exclusion limits in the squark – neutralino and gluino – neutralino mass planes is shown in Fig. 2.3. In this thesis squark and gluino masses in the interval $[0, 4000]$ GeV are explored, as the neutralino is assumed to be massive.



(a) Squark production



(b) Gluino production

Figure 2.3: Exclusion limits from the ATLAS experiment in $\sqrt{s} = 13$ TeV proton-proton collisions, for **(a)** light-flavour squark pairs with decoupled gluinos, and **(b)** gluino pairs with decoupled squarks. The blue dashed lines show the exclusion limits at 95% CL, with the yellow bands indicating the 1σ excursions due to experimental and background-only theoretical uncertainties. Obsverved limits are indicated by maroon lines, with the nominal limit (solid) and varying the renormalization scale and PDFs (dotted). The analysis assumes conservation of R-parity and a lightest neutralino LSP. The dashed straight lines indicate the limit where $m_{\tilde{q}/\tilde{g}} < m_{\tilde{\chi}_1^0}$, so the squarks and gluinos cannot decay to the lightest neutralino above this line. Figures from [15].

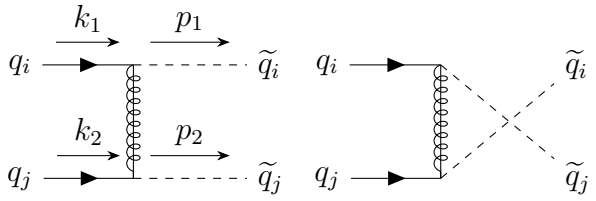


Figure 2.4: Feynman diagrams for squark pair production in quark-quark collisions, both t- and u-channel diagrams. Note that the u-channel (right diagram) is only possible for $i = j$.

2.3 Squark-Squark Cross Section

The process investigated in this thesis is the production of squark pairs in quark-quark collisions,

$$q_i q_j \to \widetilde{q}_i \widetilde{q}_j, \tag{2.8}$$

where i, j are the first and second generation quark flavours u, d, s, c. Feynman diagrams for tree-level contributions are found in Fig. 2.4. For same flavour quarks the t- and u-channel contribute, while only the t-channel contributes for different flavours.

2.3.1 Leading Order Cross Section

In this section the partonic cross section for squark pair-production is calculated to leading order. The exchanged gluino momentum is denoted p in the calculations, and defined as $p = k_2 - p_2$ for the t-channel and $p = k_2 - p_1$ for the u-channel. The following set of kinematical invariants are used

$$\begin{aligned}
s &= (k_1 + k_2)^2 = 2k_1 \cdot k_2, & t_1 &= (k_2 - p_2)^2 - m_{\widetilde{q}}^2, \quad t_g = (k_2 - p_2)^2 - m_{\widetilde{g}}^2, \\
t &= (k_2 - p_2)^2 = m_{\widetilde{q}}^2 - 2(k_2 \cdot p_2), & u_1 &= (k_1 - p_2)^2 - m_{\widetilde{q}}^2, \quad u_g = (k_1 - p_2)^2 - m_{\widetilde{g}}^2, \\
u &= (k_1 - p_2)^2 = m_{\widetilde{q}}^2 - 2(k_1 \cdot p_2),
\end{aligned}$$

where the Mandelstam variables are related by $t+u+s = p_1^2+p_2^2$. The expressions for the different chiralitites are considered later, until then the chiral projection operators in the matrix element are denoted as P and P'. The Feynman gauge is used, and the $n_f = 4$ light flavour quarks are treated as massless. The total matrix element is

$$i\mathcal{M} = i\mathcal{M}_t + i\mathcal{M}_u, \tag{2.9}$$

which gives four terms for the matrix element squared

$$|\mathcal{M}|^2 = |\mathcal{M}_t|^2 + |\mathcal{M}_t \mathcal{M}_u| + |\mathcal{M}_u \mathcal{M}_t| + |\mathcal{M}_u|^2. \tag{2.10}$$

Pure t-channel

The matrix element for the pure t-channel from Eq. (2.9), $i\mathcal{M}_t$, becomes (reading direction is from q_j to q_i)

$$\begin{aligned} i\mathcal{M}_t &= \bar{v}(k_1)\Big(-i\sqrt{2}gP(t_a)^{ij}\Big)\times\delta^{ab}\frac{i}{\not{p}-m_{\tilde{g}}}\times\Big(-i\sqrt{2}gP'(t_b)^{lk}\Big)\times u(k_2) \\ &= -(t_a)^{ij}(t^a)^{lk}\times\frac{i2g^2}{t_g^2}\times\bar{v}(k_1)P(\not{p}+m_{\tilde{g}})P'u(k_2), \end{aligned}$$

where the colour factor has been factored out. The matrix element squared is then

$$\begin{aligned} |\mathcal{M}_t|^2 = (t_a)^{ij}(t^a)^{lk}(t_b)_{ij}(t^b)_{lk}\times\frac{4g^4}{t_g^2}&\big(\bar{v}(k_1)P(\not{p}+m_{\tilde{g}})P'u(k_2)\big) \\ &\times\big(\bar{u}(k_2)P(\not{p}+m_{\tilde{g}})P'v(k_1)\big). \end{aligned}$$

To sum over colours, the following relation is used for an $SU(N)$ gauge group [16]

$$\sum_a (t^a)_{ij}(t_a)_{lk} = \frac{1}{2}(\delta_{ik}\delta_{lj}-\frac{1}{N}\delta_{ij}\delta_{lk}), \tag{2.11}$$

which gives for the colour factor

$$\begin{aligned} \sum_{a,b}(t^a)^{ij}(t_a)^{kl}(t^b)_{ij}(t_b)_{kl} &= \frac{1}{4}\big(\delta_{ik}\delta_{lj}-\frac{1}{N}\delta_{ij}\delta_{lk}\big)\big(\delta^{ik}\delta^{lj}-\frac{1}{N}\delta^{ij}\delta^{lk}\big) \\ &= \frac{1}{4}(N^2-1) \\ &= \frac{1}{2}NC_F, \end{aligned}$$

where $C_F = (N^2-1)/(2N)$. Averaging over spin gives

$$\sum|\mathcal{M}_t|^2 = \frac{1}{2}NC_F\times\frac{4g^4}{t_g^2}\mathrm{tr}\big[\not{k}_1P(\not{p}+m_{\tilde{g}})P'\not{k}_2P(\not{p}+m_{\tilde{g}})P'\big],$$

where, as previously mentioned, the quarks are considered massless. The final state squarks can right- og left-handed, and all four combinations contribute to the total cross section. Recall that the squarks are not, in fact, right- or left-handed, but are scalar partners of the right- and left-handed parts of the quarks.

Different chiralities $P = P_{R/L}$, $P' = P_{L/R}$

For squarks of different chiralities the trace becomes

$$\begin{aligned} \mathrm{tr}\big[\not{k}_1P_{R/L}(\not{p}+m_{\tilde{g}})P_{L/R}\not{k}_2P_{R/L}(\not{p}+m_{\tilde{g}})P_{L/R}\big] &= 2\big(2(p\cdot k_2)(k_1\cdot p)-p^2(k_1\cdot k_2)\big) \\ &= (t-m_{\tilde{q}}^2)s+(t-m_{\tilde{q}}^2)(u-m_{\tilde{q}}^2)-ts \\ &= t_1u_1-m_{\tilde{q}}^2s, \end{aligned}$$

where $P_{R/L}P_{R/L} = P_{R/L}$, $(\gamma^5)^2 = 1$ and $\mathrm{tr}[\gamma^5\gamma^\mu\gamma^\nu] = 0$. For same flavour ingoing quarks, qq, the only possibility for different chiralities is $\widetilde{q}_R\widetilde{q}_L$. For different flavour quarks, $q'q$, there are two possibilities, namely $\widetilde{q}_R\widetilde{q}'_L$ and $\widetilde{q}_L\widetilde{q}'_R$.

Equal chiralities $P = P_{R/L}$, $P' = P_{R/L}$

For squarks of equal chiralities the trace becomes

$$\mathrm{tr}\big[\not{k}_1 P_{R/L}(\not{p} + m_{\widetilde{g}})P_{R/L}\not{k}_2 P_{R/L}(\not{p} + m_{\widetilde{g}})P_{R/L}\big] = 2m_{\widetilde{g}}^2(k_1 \cdot k_2) = m_{\widetilde{g}}^2 s.$$

For equal flavour quarks qq there are now two possibilities for equal chirality squarks, namely $\widetilde{q}_R\widetilde{q}_R$ and $\widetilde{q}_L\widetilde{q}_L$. For different flavour quarks qq' the two possibilities for equal chiralities are $\widetilde{q}_L\widetilde{q}'_L$ and $\widetilde{q}_R\widetilde{q}'_R$.

Note that the contribution to $\widetilde{q}_R\widetilde{q}_R$ is identical to $\widetilde{q}_L\widetilde{q}_L$ in QCD processes. If no electroweak corrections are included in the higher order term, the NLO cross section should also be identical as a function of $m_{\widetilde{q}}$, because only the electroweak interaction couples to right- and left-handed states differently. This will become important later in Sec. 5.3.

Sum over Chiralities

For incoming quarks q_iq_j the sum over chiralities yields

$$\sum |\mathcal{M}_t|^2 = NC_F \times \frac{4g^4}{t_g^2}\Big[\delta_{ij}\big(\frac{1}{2}(t_1u_1 - sm_{\widetilde{q}}^2) + sm_{\widetilde{g}}^2\big) + (1-\delta_{ij})\big(t_1u_1 - s(m_{\widetilde{q}}^2 - m_{\widetilde{g}}^2)\big)\Big],$$

where the terms proportional to δ_{ij} only contribute for same flavour quarks, and the terms proportional to $1 - \delta_{ij}$ contribute for different flavour quarks.

Pure u-channel

The expression for the u-channel matrix element in Eq. (2.9) is identical to the t-channel matrix element, but for the exchange $t_g^2 \to u_g^2$. The u-channel only contributes for same flavour quarks, $i = j$, so the matrix element squared is

$$\sum |\mathcal{M}_u|^2 = NC_F\frac{4g^4}{u_g^2}\delta_{ij}\Big[\frac{1}{2}(t_1u_1 - sm_{\widetilde{q}}^2) + sm_{\widetilde{g}}^2\Big].$$

Cross term

The cross term between the t- and u-channel diagrams in Eq. (2.9) consists of two terms. The first term, $\mathcal{M}_t\mathcal{M}_u$, is given by

$$\mathcal{M}_t\mathcal{M}_u = (t^a)^{ij}(t_a)^{kl}(t^b)_{ik}(t_b)_{jl}\frac{-i2g^2}{t_g}\big(\bar{v}(k_1)P(\not{p}+m_{\widetilde{g}})P'u(k_2)\big) \times\frac{i2g^2}{u_g}\big(\bar{u}(k_2)P(\not{p}+m_{\widetilde{g}})P'v(k_1)\big)$$

The colour factor is again found using Eq. (2.11), which gives

$$\sum_{a,b}(t^a)^{ij}(t_a)^{kl}(t^b)_{ik}(t_b)_{jl} = -\frac{1}{2}C_F.$$

Averaging over spins gives

$$\sum \mathcal{M}_t\mathcal{M}_u = -\frac{1}{2}C_F \times \mathrm{tr}\left[\frac{4g^4}{u_g t_g}\big(\not{k}_1 P(\not{k}_2 - \not{p}_2 + m_{\widetilde{g}})P'\not{k}_2 P(\not{k}_2 - \not{p}_1 + m_{\widetilde{g}})P'\big)\right].$$

Summing over equal and different chiralities gives

$$\sum \mathcal{M}_t\mathcal{M}_u = -\frac{1}{2}C_F\frac{4g^4}{u_g t_g}\big(-t_1u_1 + m_{\widetilde{q}}^2 s + m_{\widetilde{g}}^2 s\big)$$

The other cross term in Eq. (2.9), $\mathcal{M}_u\mathcal{M}_t$, is similar, but yields a slightly different expression

$$\sum \mathcal{M}_u\mathcal{M}_t = -\frac{1}{2}C_F\frac{4g^4}{u_g t_g}\big(u_1t_1 - m_{\widetilde{q}}^2 s + m_{\widetilde{g}}^2 s\big).$$

Adding the cross terms gives the expression for the total cross term in Eq. (2.10)

$$\sum |\mathcal{M}_t\mathcal{M}_u + \mathcal{M}_u\mathcal{M}_t| = -\frac{1}{2}C_F\delta_{ij}\frac{4g^4}{u_g t_g}\big(2m_{\widetilde{g}}^2 s\big).$$

Matrix Elements

The matrix elements can be divided into contributions to $\widetilde{q}_{iR}\widetilde{q}_{iR}$, $\widetilde{q}_{iR}\widetilde{q}_{iL}$, $\widetilde{q}_{iR}\widetilde{q}_{jR}$ and $\widetilde{q}_{iR}\widetilde{q}_{jL}$ production, and equivalently for $R \leftrightarrow L$. The matrix elements of the

different processes are

$$\widetilde{q}_{iR}\widetilde{q}_{iR}, \widetilde{q}_{iL}\widetilde{q}_{iL}: \quad \sum |\mathcal{M}|_{iRiR} = 4g^4 s m_{\widetilde{g}}^2 \Big[\frac{1}{2} N C_F \Big(\frac{1}{t_g^2} + \frac{1}{u_g^2}\Big) - C_F \frac{1}{t_g u_g}\Big], \tag{2.12}$$

$$\widetilde{q}_{iR}\widetilde{q}_{iL}: \quad \sum |\mathcal{M}|_{iRiL} = 4g^4 (u_1 t_1 - s m_{\widetilde{q}}^2) \Big[\frac{1}{2} N C_F \Big(\frac{1}{t_g^2} + \frac{1}{u_g^2}\Big)\Big], \tag{2.13}$$

$$\widetilde{q}_{iR}\widetilde{q}_{jR}, \widetilde{q}_{iL}\widetilde{q}_{jL}: \quad \sum |\mathcal{M}|_{iRjR} = 4g^4 s m_{\widetilde{g}}^2 \Big[\frac{1}{2} N C_F \frac{1}{t_g^2}\Big], \tag{2.14}$$

$$\widetilde{q}_{iR}\widetilde{q}_{jL}, \widetilde{q}_{iL}\widetilde{q}_{jR}: \quad \sum |\mathcal{M}|_{iRjL} = 4g^4 (t_1 u_1 - s m_{\widetilde{q}}^2) \Big[\frac{1}{2} N C_F \frac{1}{t_g^2}\Big]. \tag{2.15}$$

Note that, as mentioned above, the contributions to $\widetilde{q}_R\widetilde{q}_R$ and $\widetilde{q}_L\widetilde{q}_L$ are identical as functions of $m_{\widetilde{q}}$.

Summing over the terms from different chirality combinations gives for the total sum over matrix element squared

$$\begin{aligned}\sum |\mathcal{M}|^2 = \delta_{ij} \Bigg[& 2g^4 N C_F (u_1 t_1 - s m_{\widetilde{q}}^2) \Big(\frac{1}{t_g^2} + \frac{1}{u_g^2}\Big) \\ & + 4g^4 s m_{\widetilde{g}}^2 \Big(N C_F \Big(\frac{1}{t_g^2} + \frac{1}{u_g^2}\Big) - 2 C_F \frac{1}{u_g t_g}\Big)\Bigg] \\ & + (1 - \delta_{ij}) \Bigg[4g^4 N C_F \frac{u_1 t_1 - s(m_{\widetilde{g}}^2 - m_{\widetilde{g}}^2)}{t_g^2}\Bigg],\end{aligned} \tag{2.16}$$

where, again, the terms proportional to δ_{ij} contribute for same flavour quarks, and terms proportional to $1 - \delta_{ij}$ contribute for different flavour quarks.

Partonic Cross Section

The $SU(3)$ colour factors are given by $N = 3$, which gives $C_F = 4/3$. The cross section for a two-body reaction may be written as

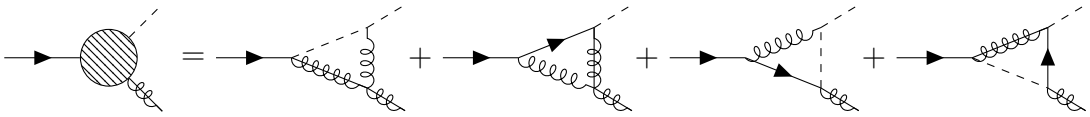
$$\frac{d\sigma}{dt} = \frac{1}{64\pi s} \frac{1}{|\mathbf{p}_{1\mathrm{cm}}|^2} |\mathcal{M}|^2, \tag{2.17}$$

where $\mathbf{p}_{1\mathrm{cm}}$ is the three-momenta of one ingoing particle in the center of mass frame. Integration over t gives the Born level[2] partonic cross section for squark pair production $q_i q_j \to \widetilde{q}_i \widetilde{q}_j$

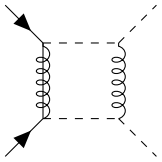
$$\begin{aligned}\hat{\sigma}^B = & \frac{\pi \alpha_s^2}{s} \Bigg[\beta_{\widetilde{q}} \Big(-\frac{4}{9} - \frac{4m_-^4}{9(m_{\widetilde{g}}^2 s + m_-^4)}\Big) + \Big(-\frac{4}{9} - \frac{8m_-^2}{9s}\Big) L_1 \Bigg] \\ & + \delta_{ij} \frac{\pi \alpha_s^2}{s} \Bigg[\frac{8 m_{\widetilde{g}}^2}{27(s + 2m_-^2)} L_1 \Bigg],\end{aligned} \tag{2.18}$$

[2]Tree-level.

(a) Gluino self-energy.



(b) Quark-squark-gluino coupling.



(c) Box diagram.

Figure 2.5: A selected set of Feynman diagrams for the virtual corrections to $q_i q_j \to \widetilde{q}_i \widetilde{q}_j$ [3].

where

$$L_1 = \ln\left(\frac{s + 2m_-^2 - s\beta_{\widetilde{q}}}{s + 2m_-^2 + s\beta_{\widetilde{q}}}\right), \quad \beta_{\widetilde{q}} = \sqrt{1 - \frac{4m_{\widetilde{q}}^2}{s}}, \quad m_-^2 = m_{\widetilde{g}}^2 - m_{\widetilde{q}}^2, \quad \alpha_s = \frac{g_s^2}{4\pi},$$

which is identical to the one found in [3].

2.4 Next-to-leading Order Corrections

The leading order cross section for squark pair production was calculated in Sec. 2.3.1. The *next-to-leading order* (NLO) terms contain virtual corrections to the Born level Feynman diagrams in Fig. 2.4. By allowing more complicated Feynman diagrams, virtual particles can appear in the process. These never make it to the final state and become real particles, but are instead absorbed by the process. Since they are never real, or on-shell, they can have any momentum and mass. Thus, it is necessary to integrate over all possible momenta, which leads to divergences and infinities.

There are three main categories of divergences, namely ultraviolet, infrared and collinear. Ultraviolet divergences appear when the integral is over an energy without upper bounds. When the opposite problem occurs, that low momentum gives infinities, this is called an infrared divergence. Collinear divergences are sometimes called mass singularities, and stem from masses going to zero or where infinities can appear from splitting at zero angle.

In order to include the corrections and still get physical cross sections, it is necessary to integrate out the divergences using dimensional regularization

and renormalise parameters. Renormalising a parameter means giving it a scale dependence, and baking the infinities into this expression. Physically, this can be interpreted as giving the coupling constant a length scale (energy scale^{-1}) dependence. For example, the electromagnetic coupling becomes stronger the closer to a charged particle you get. These calulations are often complicated and tedious. Some of the diagrams for virtual corrections to $q_i q_j \to \widetilde{q}_i \widetilde{q}_j$ are shown in Fig. 2.5.

There are several reasons for calculating these cumbersome NLO terms. Firstly, the leading order terms are highly dependent on the *a priori* unknown renormalization scale because of missing higher order contributions to the cross section, leading to a large uncertainty in theoretical predictions (up to a factor of two). Adding higher order terms reduces this scale dependency. An example is shown in Fig. 2.6, where the LO and NLO cross sections for $qq \to \widetilde{q}\widetilde{q}$ are plotted as a function of the renormalization scale at the LHC for $\sqrt{s} = 14$ TeV [3]. The NLO terms are clearly less dependent on the scale than the LO terms. Secondly, the NLO contributions are expected to be large and positive, thereby raising the cross sections significantly and allowing for stronger bounds on the gluino and squark masses [3].

The next-to-leading order cross sections are calculated in [3], assuming degenerate squark masses $m_{\widetilde{q}}$, with the five lightest quark masses set to zero, and the top quark mass to $m_t = 175$ GeV. The two free parameters left in the cross sections are then $m_{\widetilde{g}}$ and $m_{\widetilde{q}}$, as seen in Eq. (2.18). The renormalization scheme used is the $\overline{MS}$ scheme.

Next-to-leading Order Partonic Cross-Section

As discussed in Sec. 2.1, cross sections for hadron colliders require first finding the partonic cross sections. In order to analyze these, scaling functions are introduced [3] giving the general LO+NLO result for all strong production processes as

$$\hat{\sigma}_{ij} = \frac{\alpha_s^2(Q^2)}{m^2}\Big\{ f_{ij}^B(\eta, r) + 4\pi\alpha_s(Q^2)\Bigg[f_{ij}^{V+S}(\eta, r, r_t) + f_{ij}^H(\eta, r) + \bar{f}_{ij}(\eta, r)\log\left(\frac{Q^2}{m^2}\right)\Bigg]\Big\}, \tag{2.19}$$

where Q^2 is the renormalization scale discussed previously, often set to $Q^2 = m^2$, when $m = (\sqrt{p_1^2} + \sqrt{p_2^2})/2$ is the average mass of the produced particles. The partonic cross section is written as a function of the parameters

$$\eta = \frac{s}{4m^2} - 1, \qquad r = \frac{m_{\widetilde{g}}^2}{m_{\widetilde{q}}^2}, \qquad r_t = \frac{m_t^2}{m^2}. \tag{2.20}$$

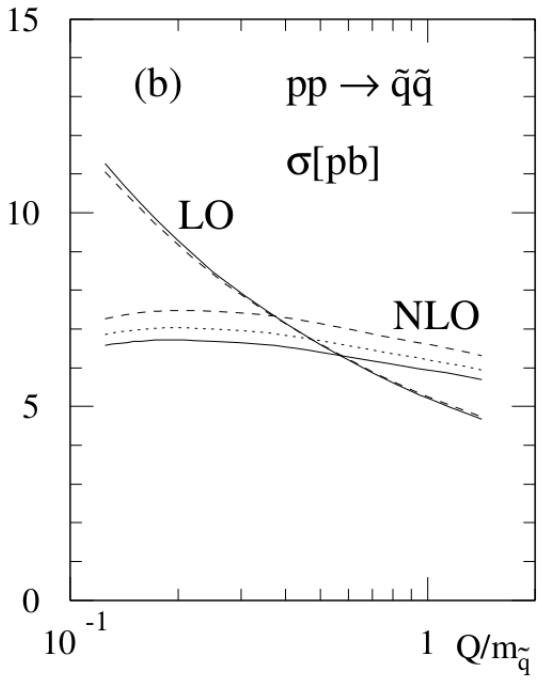


Figure 2.6: The dependence on the renormalization scale Q for the LO and NLO cross sections for squark-squark production at the LHC ($\sqrt{s} = 14$ TeV). Parton densities are GRV94 (solid), CTEQ3 (dashed) and MRS(A') (dotted). Mass parameters are $m_{\tilde{q}} = 600$ GeV, $m_{\tilde{g}} = 500$ GeV and $m_t = 175$ GeV. Figure from [3].

The scaling functions f are as follows: the Born term f^B from Eq. (2.18), the hard gluon corrections f^H, the scale-dependent contributions $\bar{f}$ and the sum of virtual and soft-gluon corrections f^{V+S}. The gluon corrections come from adding a final state gluon to the partonic reaction

$$q_i q_j \to \widetilde{q}_i \widetilde{q}_j g. \tag{2.21}$$

The energy near the threshold is the base for an important part of the contributions to the cross section [3]. In this region the scaling functions can be expanded in the low velocity of produced particles β, leading to the following expressions [3]

$$\begin{aligned}
f^B_{qq} &= \frac{8\pi\beta m_{\widetilde{q}}^2 m_{\widetilde{g}}^2}{27(m_{\widetilde{q}}^2 + m_{\widetilde{g}}^2)^2}, & f^B_{q'q} &= \frac{8\pi\beta m_{\widetilde{q}}^2 m_{\widetilde{g}}^2}{9(m_{\widetilde{q}}^2 + m_{\widetilde{g}}^2)^2} \\
f^{V+S}_{qq} &= f^B_{qq}\frac{1}{24\beta} & f^{V+S}_{q'q} &= f^B_{q'q}\frac{1}{24\beta} \\
f^H_{qq} &= f^B_{qq}\Big[\frac{2}{3\pi^2}\log^2(8\beta^2) - \frac{7}{2\pi^2}\log(8\beta^2)\Big] & f^H_{q'q} &= f^B_{q'q}\Big[\frac{2}{3\pi^2}\log^2(8\beta^2) - \frac{19}{6\pi^2}\log(8\beta^2)\Big] \\
\bar{f}_{qq} &= -f^B_{qq}\frac{2}{3\pi^2}\log(8\beta^2) & \bar{f}_{q'q} &= -f^B_{q'q}\frac{2}{3\pi^2}\log(8\beta^2).
\end{aligned} \tag{2.22}$$

To find the total cross section partonic cross section is then integrated over, as discussed in Sec. 2.1.1. In [3] the integrals are calculated numerically using the VEGAS integration routine [17].

K-Factor

To quantify the change in the cross section from adding NLO terms, the *K-factor* is introduced. The K-factor is the ratio between the cross sections

$$K = \sigma_{NLO}/\sigma_{LO}. \tag{2.23}$$

The K-factor for squark production for varying mass ratios $m_{\widetilde{q}}/m_{\widetilde{g}} = 2.0$, 1.6, 1.2, 0.8 at the LHC at $\sqrt{s} = 14$ TeV is shown in Fig. 2.7. As seen from the figure, the K-factor is larger than one and quite stable as a function of the squark mass.

2.4.1 State-of-the-art Tools

There are two current numerical tools for the calculation of NLO supersymmetric cross sections, namely `Prospino 2.1` [4] and `NLL-fast 2.1` [5].

Prospino

`Prospino 2.1` [4] is a numerical tool that calculates supersymmetric cross sections to next-to-leading order for non-degenerate squark masses. First the LO

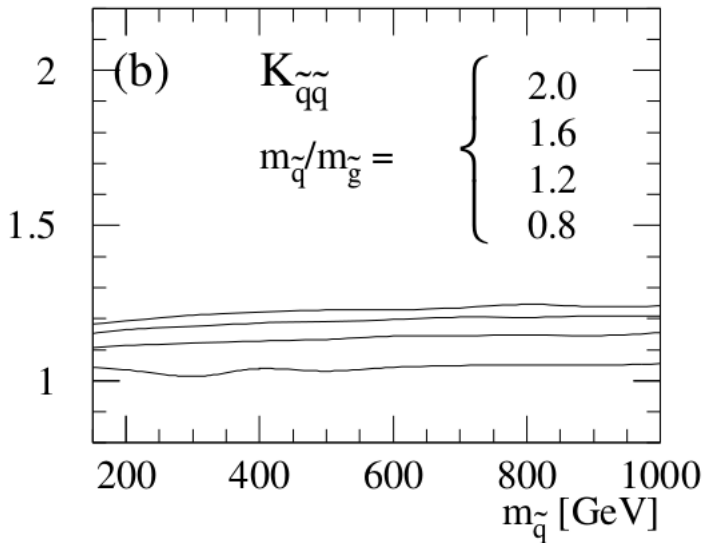


Figure 2.7: The K-factors for the LHC ($\sqrt{s} = 14$ TeV). Parton densities are GRV94, with scale $Q = m$, and the top squark mass is $m_t = 175$ GeV. Figure from [3].

cross sections are calculated for the five or four lightest squark flavours. The LO and NLO cross sections are then calculated for the mean value of the squark masses, and the corresponding K-factor is calculated. The LO cross sections for the individual squark processes are multiplied by the K-factor, giving an approximation to the NLO terms for non-degenerate masses. The calculation of NLO cross sections for squark production is outlined in the section above, and described in more detail in [4].

`Prospino 2.1` assumes a particular renormalization scale to calculate cross sections. The uncertainty from the scale dependence is estimated in `Prospino 2.1` by calculating cross sections for twice and half the chosen renormalization scale, and seeing how the cross sections change. As demonstrated in Fig. 2.6 the scale dependence of cross sections is reduced by adding higher order terms. As a consequence, the scale dependence is also a way of estimating the size of the contribution of higher order terms.

As was discovered during the project, `Prospino 2.1` has a technical weakness in the heavy squark mass space. `Prospino 2.1` sets the next-to-leading order cross sections to zero when the mean squark mass is above threshold ($m^2 > s$). For individual masses *below* the threshold, with nonzero leading order cross sections, this means inconsistently setting next-to-leading order cross sections to zero. This creates illogical points — or outliers — in the dataset.

Evaluating cross sections with `Prospino 2.1` is also highly time-consuming. Evaluating all the strong process cross sections for a single CMSSM benchmark point takes about 15 minutes of CPU time on a modern processor [18].

NLL-fast

`NLL-fast 2.1` [3, 19–22] computes the hadronic cross sections of gluino and squark pair production including NLO QCD corrections, and the resummation of soft gluon emission at next-to-leading-logarithmic (NLL) accuracy. It also provides an error estimation, based on the errors from renormalization scale dependence and the parton distribution functions. The calculation is done by reading in tables of pre-calculated LO and NLO+NLL results, the scale uncertainty and the PDF and α_s error, and uses fast grid interpolation to calculate cross sections for any squark and gluino mass in the range $[200, 2500]$ GeV for $\sqrt{s} = 8$ TeV.

`NLL-fast 2.1` is limited in that it only calculates cross sections for mass-degenerate squarks. Since the total cross section for squark pair production consists of 36 different cross sections for combinations of the four lightest quarks, the estimate from `NLL-fast 2.1` is very simplified and unfit, *e.g.* for the MSSM-24 as will be demonstrated in Sec. 5.2.2.

Accuracy

Including the NLO+NLL partonic contributions has reduced the theoretical error to below 10% for a wide range of processes and masses, see the discussion in [18]. There are other uncertainties, from the parton distribution functions (PDFs) and α_s. These must also be included in the total error estimate. PDFs are based on experimental data, so the uncertainty increases with the sparticle masses because the PDFs are most poorly constrained at large scales Q and at large parton x. However, with new data from the LHC errors from PDFs and α_s will be reduced over time.

On the basis of this, in order to keep the regression errors subdominant, the goal is that the relative regression errors obtained in this thesis should not exceed 10%.

3

Gaussian Processes

In this chapter Gaussian process regression is introduced. First, some concepts and terminology in Bayesian statistics are reviewed. Then, the concept of covariance, and how it can be determined by covariance functions is explored. Thereafter, the mathematical framework of Gaussian processes is introduced, and the Gaussian noise model is used as an example. Bayesian model selection and cross validation are introduced as tools to quantify and improve the quality of predictions.

3.1 Introduction to Bayesian Statistics

There are two general philosophies in statistics, namely *Bayesian* and *frequentist* statistics. Statisticians from both branches would probably consider the following statement to be true

> *Statisticians use probability to describe uncertainty.*

The difference between Bayesian and frequentist statistics lies in the definition of the *uncertain*. Since uncertainty is described by probability, the definition of probability must also differ. A distinction is made between *objective* and *subjective* probability. The former is used by frequentists, and Bayesians use a combination of the two. Consider an example in which a statistician throws a die. Before throwing, he is uncertain about the outcome of the toss. This uncertainty related to the outcome is *objective*: no one can know if he will throw a 1 or a 4. On the other hand, he might also be uncertain about the underlying probability distribution of the toss. Is the die loaded? Is one of the edges sharper than the others? This uncertainty is *subjective*, as it may vary depending on how much information is available about the system, and how that information is used.

One of the main critisisms of subjective probability posed by frequentists is that the final probability depends on who you ask.

3.1.1 Bayes' Theorem

The difference between frequentist and Bayesian statistics also lies in the application of *Bayes' theorem* [23]. Bayes' theorem can be derived from the familiar rules of probability for random variables X and Y given the information I,

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

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

commonly known as the *sum rule* and *product rule*, respectively. The probability $P(X|I)$ is the probability of outcome X given the information I, and $P(X|Y, I)$ is the probability of outcome X given the information I *and* outcome Y. The bar over $\bar{X}$ means that the outcome X does *not* happen. The sum rule states that the total probability of the outcomes X and $\bar{X}$ is equal to 1. This is rather intuitive, considering that an event either takes place or not. The product rule concerns the probability of both outcomes X and Y. This is equal to the probability of Y times the probability of X given that Y has already occurred.

The product rule can be used to derive Bayes' theorem, first formulated by Thomas Bayes,

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

Bayes' theorem states that the probability of Y given X is proportional to the probability of X given Y, times the probability of Y. The proportionality factor is one over the probability of X. Surprisingly, there is nothing Bayesian — in the modern statistical sense — about Bayes' theorem.

In Bayesian statistics the theorem is used to perform inference about probability distributions. Probability distributions are functions that describe how the probability of a random variable X is distributed, and are here denoted $P(X|\Theta, I)$. The *parameters*, Θ, of these functions determine the shape of the distribution, and are *not* random variables. In physics, measurements are often used to find out more about the model. For example, data from the LHC is used to find the top mass m_t. The LHC data are then the variables X, and m_t is the parameter Θ.

Bayes' theorem can be used to find the probability of the *parameters* Θ given a set of random variables X drawn from the distribution. The resulting expression is the *posterior probability distribution*

$$P(\Theta|X, I) = \frac{P(X|\Theta, I)P(\Theta|I)}{P(X|I)}, \tag{3.4}$$

where $P(\Theta|I)$ and $P(X|\Theta, I)$ are the *prior* and *likelihood*, respectively, and $P(X|I)$ is a normalization constant called the *marginal likelihood*.

The prior probability, $P(\Theta|I)$, represents the state of knowledge about the parameters given the background information I, before data has been analysed. In the example above, the prior would be the assumed top mass based on theoretical calculations in the SM. The prior is modified by the data through the likelihood, $P(X|\Theta, I)$, which is the probability of the data given the parameters. The marginal likelihood is the probability of the data independently of the parameters, and can by Eq. (3.2) be written

$$P(X|I) = \int P(X|\Theta, I)P(\Theta|I)\ \mathrm{d}\Theta. \tag{3.5}$$

The marginal likelihood is revisited in Sec. 3.4.1.

In other words, Eq. (3.4) states the probability of parameters Θ given the knowledge of outcomes X. The parameters Θ are sometimes called the *hypothesis*, and the variables X are called the *evidence*. Bayes' theorem is then used to modify the hypothesis using the evidence.

3.1.2 Update of Belief

The crucial parting of Bayesian statistics from frequentist statistics is the introduction of the *prior*, which expresses a probability distribution on the *parameters* of the probability distribution before data, and the ability to update the probability distribution using data. The prior expresses a *prior* belief or assumption of the parameters given the background information, and has to be decided on beforehand. Conversely, the likelihood is the probability of getting the observations X given the particular values of the parameters Θ.

As mentioned previously, the measure $P(\Theta|X, I)$ from Eq. (3.4) is called the posterior distribution. This can be thought of as the prior belief, modified by how well this belief fits the data,

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

Consider an example. The statistician mentioned before now sets about tossing a coin. Before tossing he assumes the probability of heads as uniformly distributed, because is of the suspicious kind of statisticians. He therefore adopts a flat, or uniform, prior probability distribution. The uniform distribution is illustrated in the first panel in Fig. 3.1. After one toss he gets heads, and the posterior changes to a function with high probability for heads, and low for tails, illustrated in the second panel. After four tosses, of which two gave heads and two gave tails, the posterior in the third panel shows an equal probabilty for heads and tails, with a wide distribution centered at 0.5. After several tosses the distribution converges to a narrow peak around 0.25, illustrated in the fourth panel. This indicates an unfair coin that is biased towards tails.

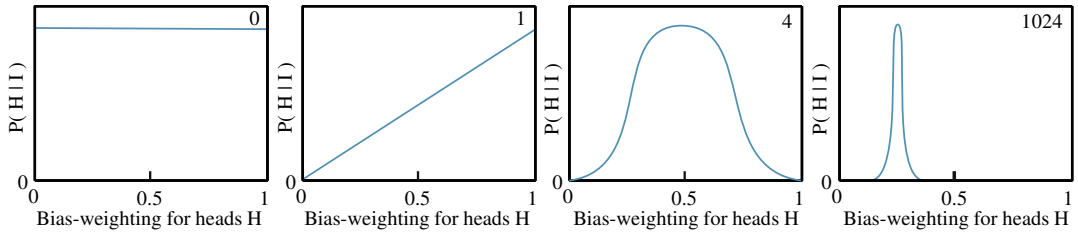


Figure 3.1: The posterior probability distribution of the bias-weighting of a coin for the uniform prior, $P(H|I)$. The first panel from the left is before the coin is tossed, the second panel is after 1 toss, the third is after 4 tosses, and the fourth is after 1024 tosses. The posterior converges towards a narrow peak at 0.25, so the coin is biased.

As seen from the example, Bayes' theorem can be used to iteratively update the posterior probability. After one coin toss, the distribution in the second panel of in Fig. 3.1 can be thought of as the new prior, which is in turn modified by more tosses.

3.1.3 Best Estimate and Reliability

The quantity of interest, such as the parameters of a distribution, will from now on be denoted X. Given a posterior distribution $P(X|\mathcal{D}, I)$ over X, where the prior given some information I has been modified by some data $\mathcal{D}$, the posterior can be approximated by a Gaussian distribution with a *mean* and *variance*.

The formal definition of the *mean value* of X, denoted $m[X]$, is given by the *expectation value* $\mathbb{E}[X]$. The mean value of a variable X following a probability distribution $P(X|I)$ is then defined as

$$m[X] = \mathbb{E}[X] = \int XP(X|I)\ \mathrm{d}X. \tag{3.6}$$

The variance $\mathbb{V}[X]$ is defined as the expectation value of the square of deviations from the mean. For X following a probability distribution $P(X|I)$, the variance is then given by

$$\mathbb{V}[X] = \mathbb{E}\big[(X - X_0)^2\big] = \int (X - X_0)^2 P(X|I)\ \mathrm{d}X. \tag{3.7}$$

The variance in X is often denoted $\sigma_X^2 = \mathbb{V}[X]$, and its square root is the *standard deviation* $\sqrt{\sigma_X^2} = \sigma_X$. The variance can be generalised to distributions over several variables. The variance of one quantity can then affect the variance of another. This correlation is called *covariance*, and is discussed in Sec. 3.1.3.

Returning to the posterior distribution $P(X|\mathcal{D}, I)$, the mean and variance of the approximated Gaussian distribution are given by the *best estimate* and

reliability, respectively. The best estimate X_0 is the outcome with the highest probability. In other words, it maximises the posterior distribution

$$\frac{dP}{dX}\Big|_{X_0} = 0, \qquad \frac{d^2P}{dX^2}\Big|_{X_0} < 0, \tag{3.8}$$

where P is the posterior $P(X|\mathcal{D}, I)$. The second derivative must be negative to ensure that X_0 is, in fact, a maximum.

Once a best estimate is found, it is important to know how reliable it is. Reliability, or uncertainty, is related to the width of the distribution. The width of the distribution indicates how much the posterior distribution is smeared out around the best estimate X_0. A narrow distribution has low uncertainty, while a wide distribution has large uncertainty. As an example, the third panel in Fig. 3.1 shows a distribution with a mean value of 0.5 with large uncertainty, while the fourth panel shows a distribution with mean 0.25 with small uncertainty.

The width is found by taking the logarithm[1] and Taylor expanding the posterior distribution $P(X|\mathcal{D}, I)$ around X_0. From Eq. (3.8) the condition of the best estimate is that $dL/dX|_{X_0} = 0$, so the expansion is given by

$$L = L(X_0) + \frac{1}{2}\frac{d^2L}{dX^2}\Big|_{X_0}(X - X_0)^2 + ..., \qquad L = \log_e\Big[P(X|\mathcal{D}, I)\Big] \tag{3.9}$$

The first term, $L(X_0)$, is just a constant and can be absorbed in the normalisation. The dominant term in determining the width is therefore the quadratic term in Eq. (3.9).

Taking the exponential of Eq. (3.9) and ignoring higher order terms, the posterior can be approximated as

$$P(X|\mathcal{D}, I) \approx A \exp\left[\frac{1}{2}\frac{d^2L}{dX^2}\Big|_{X_0}(X - X_0)^2\right], \tag{3.10}$$

where $A = \exp\big[L(X_0)\big]$ is a constant. Equation (3.10) is now in the shape of a *Gaussian distribution.*

The Gaussian Distribution

The general *Gaussian distribution* is given by

$$P(X|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(X - \mu)^2}{2\sigma^2}\right], \tag{3.11}$$

where μ and σ^2 are the two parameters of the Gaussian distribution mentioned above. An example of a Gaussian distribution is shown in Fig. 3.2. The parameter

[1] L is a monotonic function of P, so the maximum of L is at the maximum of P.

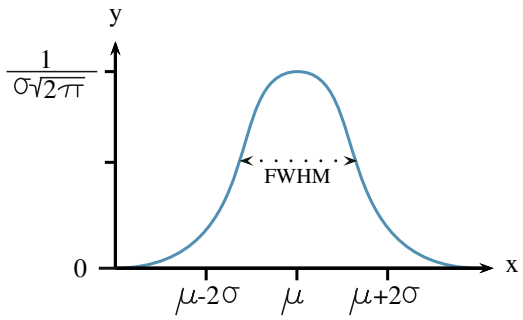


Figure 3.2: A Gaussian probability distribution. The maximum is at the mean value μ, with a full width at half maximum (FWHM) at around 2.35σ.

μ is the mean value of X defined in Eq. (3.6), $m[X]$, and σ^2 is the variance of the distribution around the mean defined in Eq. (3.7), $\mathbb{V}[X]$.

The mean and variance for the approximated Gaussian distribution of the posterior $P(X|\mathcal{D}, I)$ are then given by

$$m[X] = X_0, \quad \mathbb{V}[X] = \Big(- \frac{d^2 L}{dX^2} \Big)^{-1/2}. \tag{3.12}$$

The Gaussian distribution is also referred to as the *normal distribution*, and a Gaussian distribution with mean μ and variance σ^2 is therefore denoted $\mathcal{N}(\mu, \sigma^2)$. The notation $Y \sim \mathcal{N}(\mu, \sigma^2)$ means a *random variable Y drawn from a Gaussian distribution with mean μ and variance σ^2*. The Gaussian distribution is symmetric with respect to the maximum at the mean μ, and has a full width at half maximum (FWHM) at around 2.35σ, as seen in Fig. 3.2.

Covariance

In distributions over several quantities of interest X_i, $i = 1, ..., N$, called *multivariate distributions* and denoted $P(X_i|\mathcal{D}, I)$, the *covariance* denotes how much the variance of one quantity X_p affects the variance of another quantity X_q. The covariance between X_p and X_q is denoted $\mathrm{cov}(X_p, X_q)$, and given by the expectation value of the product of deviations from the means for all quantities of interest X_i

$$\begin{aligned} \mathrm{cov}(X_p, X_q) &= \mathbb{E}\big[(X_p - X_{p_0})(X_q - X_{q_0})\big] \\ &= \int (X_p - X_{p_0})(X_q - X_{q_0}) P(X_i|\mathcal{D}, I)\ \mathrm{d}X_i. \end{aligned} \tag{3.13}$$

Note that the covariance of X_p with itself is the variance, $\mathrm{cov}(X_p, X_p) = \sigma_p^2$, so σ_p^2 is on the diagonal.

The covariance indicates how the variance of one quantity affects another quantity. If, for example, a high value for the variable X_p leads to a high value

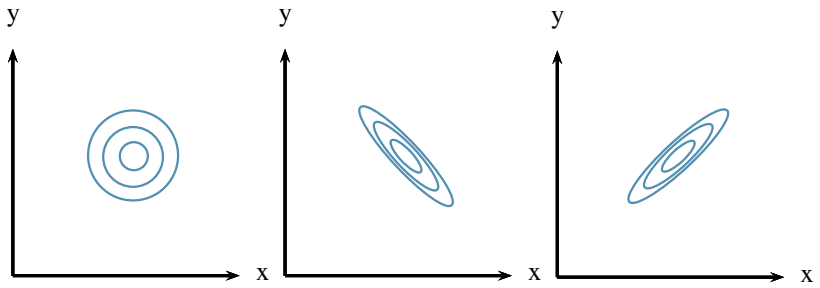


Figure 3.3: A schematic illustration of covariance and correlation. The contours of equiprobability of a posterior with zero covariance (left), where the inferred values of X and Y are uncorrelated. The corresponding plot when the covariance is large and negative (middle), and large and positive (right).

of X_q, the covariance is positive. An example of positive covariance is shown in the third panel of Fig. 3.3. The covariance is also positive if a low value for X_q leads to a low value for X_q. If the value for X_p has little or no effect on X_q, the covariance is negligible or zero $|\sigma_{X_p X_q}| \ll \sqrt{\sigma^2_{X_p}\sigma^2_{X_q}}$, as seen in the first panel of Fig. 3.3. The second panel shows negative covariance, where a high value for X_p leads to a low value for X_q.

Multivariate Best Estimate

For multivariate distributions the best estimate is not as simple to solve for as in Eq. (3.9). A set of *simultaneous equations* must be solved to get the best estimate

$$\left.\frac{dP}{dX_i}\right|_{X_{j0}} = 0, \qquad \left.\frac{d^2P}{dX_i^2}\right|_{X_{j0}} < 0, \tag{3.14}$$

for all $i = 1, ..., N$.

For simplicity the multivariate distribution is considered in two dimensions, where $\{X_i\} = (X, Y)$. As in Eq. (3.9), the Taylor expansion of the logarithm of the posterior, $L = \log_e\Big[P(X, Y|\mathcal{D}, I)\Big]$, is used to determine the reliabilities of the best estimates

$$\begin{aligned} L =& L(X_0, Y_0) + \frac{1}{2}\Big[\left.\frac{d^2L}{dX^2}\right|_{X_0,Y_0}(X - X_0)^2 \\ &+ \left.\frac{d^2L}{dY^2}\right|_{X_0,Y_0}(Y - Y_0)^2 + 2\left.\frac{d^2L}{dXdY}\right|_{X_0,Y_0}(X - X_0)(Y - Y_0)\Big] + ... \end{aligned} \tag{3.15}$$

Again, the term $L(X_0, Y_0)$ is a constant, and the first derivatives are zero from the condition in Eq. (3.14), so the reliability is given by the second derivatives.

There are four partial derivatives, reduced to three using the rules for mixed partial derivatives $\frac{\partial^2}{\partial X \partial Y} = \frac{\partial^2}{\partial Y \partial X}$. Writing the quadratic terms of Eq. (3.15) in matrix form gives

$$Q = \begin{pmatrix} X - X_0 & Y - Y_0 \end{pmatrix} \begin{pmatrix} A & C \\ C & B \end{pmatrix} \begin{pmatrix} X - X_0 \\ Y - Y_0 \end{pmatrix}, \tag{3.16}$$

where the matrix elements are

$$A = \frac{\partial^2 L}{\partial X^2}\Big|_{X_0,Y_0}, \qquad B = \frac{\partial^2 L}{\partial Y^2}\Big|_{X_0,Y_0}, \qquad C = \frac{\partial^2 L}{\partial X \partial Y}\Big|_{X_0,Y_0}. \tag{3.17}$$

The widths, or reliabilities, can now be placed in the *covariance matrix*.

Covariance Matrix

The variances and covariances are the elements of the *covariance matrix*. For N quantities of interest $X_1, ..., X_N$ the covariance matrix is an $N \times N$-matrix. The covariance matrix for X and Y is denoted $\mathrm{cov}(X, Y)$, and it can easily be shown by exponentiation and comparison with the two-dimensional Gaussian that [24]

$$\mathrm{cov}(X, Y) = \begin{pmatrix} \sigma_X^2 & \sigma_{XY}^2 \\ \sigma_{XY}^2 & \sigma_Y^2 \end{pmatrix} = - \begin{pmatrix} A & C \\ C & B \end{pmatrix}^{-1}, \tag{3.18}$$

where A, B and C are the matrix elements of Q defined in Eq. (3.17), σ_X^2 and σ_Y^2 are the variances of X and Y, respectively, and σ_{XY}^2 is the covariance of X and Y.

In short, the posterior probability distribution over X, $P(X|\mathcal{D}, I)$, can be approximated as a Gaussian distribution $\mathcal{N}(m[X], \mathbb{V}[X])$, where the mean is the best estimate and the variance is the associated reliability. The Gaussian distribution is defined by the mean value $m[X]$ and the variance $\mathbb{V}[X]$. For multivariate distributions, the covariance $\sigma_{X_i X_j}^2$ can also be calculated for the variables X_i, and all variances and covariances are contained in the covariance matrix $\mathrm{cov}(X_i, X_j)$. When the exact form of the probability distribution is not known, the elements of a covariance matrix can also be calculated using assumed *covariance functions*, which lies at the heart of Gaussian processes.

3.2 Covariance Functions

The elements of a covariance matrix can be given by *covariance functions*, or *kernels*. In Gaussian processes these are used to calculate the covariance between function values, as will be shown in Eq. (3.29). Since functions can have vectors as inputs, $\mathbf{x}$, the covariance functions are also functions of input vectors $\mathbf{x}$. A function that maps two arguments $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ into $\mathbb{R}$, where $\mathcal{X}$ is the input space, is

generally called a kernel k. Covariance functions are symmetric kernels, meaning that $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$. In this thesis both terms, kernel and covariance function, refer to the covariance function.

A kernel function that only depends on the difference between two points, $\mathbf{x} - \mathbf{x}'$, is called *stationary*, *i.e.* invariant to translations in input space. If, in addition, it only depends on the length $r = |\mathbf{x} - \mathbf{x}'|$, the function is *isotropic*[2].

The covariance function can also depend on the dot product, $\mathbf{x} \cdot \mathbf{x}'$, and is then called a *dot product* covariance function. The most important covariance functions for this thesis are the *squared exponential covariance function* and the *Matérn class of covariance functions*.

3.2.1 The Squared Exponential Covariance Function

The *squared exponential covariance function* has the form

$$k_{\mathrm{SE}}(r) = \exp\Big(-\frac{r^2}{2\ell^2}\Big), \tag{3.19}$$

where ℓ is the *characteristic length scale*. Equation (3.19) is sometimes also called the *radial basis function* (RBF). The length scale, ℓ, determines the smoothness of the function. It can be loosely interpreted as how far you need to move (along a particular axis) in input space for the function values to become uncorrelated. For a large length scale one should expect a very slowly varying function, while a shorter length scale means a more rapidly varying function, see the illustration in Fig. 3.4. The RBF is infinitely differentiable and therefore very smooth. Stein [26] argues that such strong smoothness assumptions are unrealistic for most physical problems, and recommends another class of kernels, namely the *Matérn class of covariance functions*.

3.2.2 The Matérn Class of Covariance Functions

The *Matérn class of covariance functions* is given by

$$k_{\mathrm{Matérn}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big)^{\nu} K_\nu\Big(\frac{\sqrt{2\nu}r}{\ell}\Big), \tag{3.20}$$

where $\nu, \ell > 0$, K_ν is a modified Bessel function [27], and $\Gamma(\nu)$ is the gamma function. The hyperparameter ν controls the smoothness of the function. For $\nu \to \infty$ this becomes the RBF, and for $\nu = 1/2$ it becomes the very rough absolute exponential kernel $k(r) = \exp(-r/\ell)$. In the case of half integer ν, $\nu = p + \frac{1}{2}$ for $p \in \mathbb{N}$, the covariance function is simply the product of an exponential and a

[2]Invariant to rigid rotations in input space.

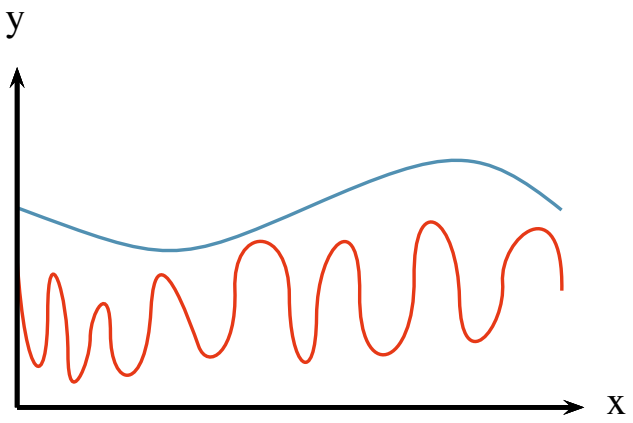


Figure 3.4: The effect of varying the length scale ℓ. A long length scale (blue) gives a slowly varying function, while a short length scale (red) gives a more staccato, quickly varying function.

polynomial

$$k_{\nu=p+\frac{1}{2}} = \exp\Big(-\frac{\sqrt{2\nu}r}{\ell}\Big)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\Big(\frac{\sqrt{8\nu}r}{\ell}\Big)^{p-i}. \tag{3.21}$$

In statistical learning the two most commonly used cases are for $\nu = 3/2$ and $\nu = 5/2$

$$k_{\nu=3/2}(r) = \Big(1+\frac{\sqrt{3}r}{\ell}\Big)\exp\Big(-\frac{\sqrt{3}r}{\ell}\Big), \tag{3.22}$$

$$k_{\nu=5/2}(r) = \Big(1+\frac{\sqrt{5}r}{\ell}+\frac{5r^2}{3\ell^2}\Big)\exp\Big(-\frac{\sqrt{5}r}{\ell}\Big). \tag{3.23}$$

3.2.3 Noise

The covariance function can also contain information about noise in the data. If the noise ε follows a Gaussian distribution $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$, the noise can be represented by adding the variance, σ_n^2, to the diagonal of the covariance matrix

$$k(\mathbf{x}_p, \mathbf{x}_q)_{noise} = \sigma_n^2 \delta_{pq}, \tag{3.24}$$

where δ_{ij} is the Kronecker delta. The model where the noise is assumed to follow a Gaussian distribution is called the *Gaussian noise model*, and is discussed further in Sec. 3.3.2.

3.2.4 Hyperparameters

The RBF kernel in Eq. (3.19) and the Matérn kernel in Eq. (3.20) are both isotropic, *i.e.* they are functions only of the distance between input points, r. Isotropic kernels can, however, be generalised to an anisotropic form by setting

$$r^2(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T M (\mathbf{x} - \mathbf{x}'), \tag{3.25}$$

where M is some positive-definite matrix. An $n \times n$ matrix is said to be positive-definite if the scalar zMz^T is strictly positive for every non-zero column vector z of n real numbers. The matrix M can take multiple forms, such as

$$M_1 = \ell^{-2}\mathbb{I} \quad \text{or} \quad M_2 = \text{diag}(\boldsymbol{\ell})^{-2}, \tag{3.26}$$

where ℓ^2 is a scalar, and $\boldsymbol{\ell}$ is a vector of the same dimension as the input vector $\mathbf{x}$. Choosing $\boldsymbol{\ell}$ to be a vector instead of a scalar is in many cases useful, especially if the input vector $\mathbf{x}$ contains values of different scales.

The matrix M is now part of the *hyperparameters* of the kernel. Each kernel has a vector of hyperparameters, denoted $\boldsymbol{\theta}$. An example is the following generalisation of the RBF kernel,

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\big(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^T M (\mathbf{x}_p - \mathbf{x}_q)\big) + \sigma_n^2 \delta_{pq}, \tag{3.27}$$

where σ_f^2 is a scaling factor and σ_n^2 is the Gaussian noise parameter, discussed in Sec. 3.2.3. The hyperparameters of this kernel are $\boldsymbol{\theta} = (\{M\}, \sigma_f^2, \sigma_n^2)^T$, where $\{M\}$ denotes the parameters in the symmetric matrix M.

Other Covariance Functions

There are several other types of covariance functions that are not discussed here. In addition, kernels can be multiplied and summed to form new kernels, making the space of possible kernels infinite. For further details see Chapter 4 in [28].

3.3 Gaussian Process Regression

Gaussian processes (GP) is a supervised statistical learning method, designed to solve regression and probabilistic classification problems. Only regression is explored in this thesis. This section begins by introducing Gaussian processes and important notation. Then the prior and posterior distributions in Gaussian processes are considered, followed by a short discussion on how functions can be drawn from these distributions. Finally, a quick overview of the noise-free model and the Gaussian noise model are given.

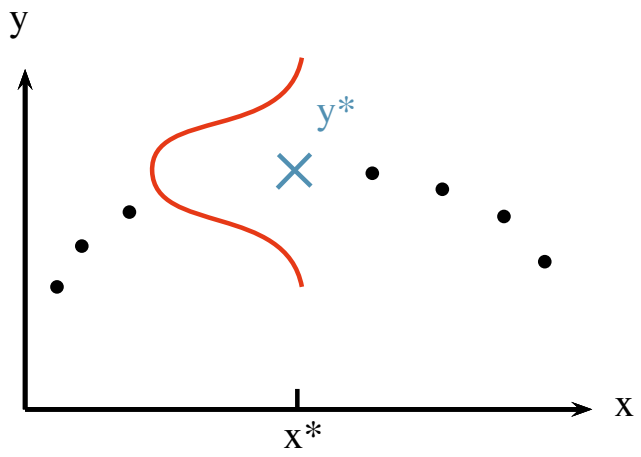


Figure 3.5: An illustration of a GP prediction of the target value y^* (blue cross), given the known set of points $\{x_i, y_i\}$ (black dots). The prediction is a Gaussian distribution in y with mean y^* and variance σ^2. The Gaussian distribution is drawn in red with y on the vertical axis, with uncertainty in the y-direction.

3.3.1 Basics and Notation

Gaussian Processes

Consider a set of points $\mathcal{D} = \{\mathbf{x}_i, y_i\}$, where y is some (possibly noisy) function of $\mathbf{x}$, $y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon$. Here, ε is the noise and $f(\mathbf{x})$ is the true value of the function. These points are illustrated by the black dots in Fig. 3.5. In statistical learning $\mathcal{D}$ is the *training data*, as it is used to train the model. It consists of *features*, which are the components of the input vectors $\mathbf{x}_i$, and *targets*, which are the function values y_i. The set of points is discrete, so there is some $\mathbf{x}^*$ for which the target y^* is unknown. The test point $\mathbf{x}^*$ is marked on the x-axis in Fig. 3.5.

Gaussian processes (GP) predict a Gaussian distribution *over function values* at this point $\mathbf{x}^*$. The distribution for a single test point $\mathbf{x}^*$ has a corresponding mean $m(\mathbf{x}^*)$ and variance $\mathbb{V}(\mathbf{x}^*)$. Note that the mean $m(\mathbf{x}^*)$ *is not the mean of the input vector* $\mathbf{x}^*$, but rather *the mean of function values* $f(\mathbf{x})$ *evaluated at* $\mathbf{x}^*$.

The GP prediction for the target value $y^* = f(\mathbf{x}^*)$ is the mean $m(\mathbf{x}^*)$. Similarly, the variance, $\mathbb{V}(\mathbf{x}^*)$, is in fact the variance in function values $f(\mathbf{x}^*)$, or the width of the Gaussian distribution (red line) in the y-direction in Fig. 3.5. The mean value y^* is drawn as a blue cross in Fig. 3.5. As will be shown, the predicted target y^* is a linear combination of the known targets y_i, where the weights are controlled by the covariances between training points $\mathbf{x}_i$ and the test point $\mathbf{x}^*$.

Some Notation

As Gaussian process regression is notoriously confusing, it is helpful to begin with some notation.

As introduced in Sec. 3.3, the Gaussian distribution with a mean μ and variance σ^2 is denoted $\mathcal{N}(\mu, \sigma^2)$. The distribution can be over a single random variable, f, or a finite set of random variables, f_i. A Gaussian distribution over several random variables is called a *multivariate Gaussian distribution*. The n random variables $f_i,\ i = 1, ..., n$ drawn from a multivariate Gaussian distribution make up the n-dimensional vector $\mathbf{f}$. The mean values, μ_i, are then contained in the n-dimensional *mean vector* $m[\mathbf{f}]$. The variance, σ^2, is replaced by an $n \times n$-dimensional covariance matrix, $\text{cov}(\mathbf{f})$. The multivariate Gaussian distribution over $\mathbf{f}$ is written as

$$\mathbf{f} \sim \mathcal{N}\big(m[\mathbf{f}],\ \text{cov}(\mathbf{f})\big). \tag{3.28}$$

For n points $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i$ is an m-dimensional feature vector and y_i is the target, the features comprise the $n \times m$-matrix X. The targets make up the corresponding n-dimensional vector $\mathbf{y}$. The vector containing the latent function values, *i.e.* $f_i = f(\mathbf{x}_i)$, is denoted $\mathbf{f}$. A central assumption in Gaussian process regression is that the covariance between targets y_i, y_j is given by the kernel of their features $\mathbf{x}_i$, $\mathbf{x}_j$

$$\text{cov}(y_i, y_j) = k(\mathbf{x}_i, \mathbf{x}_j), \tag{3.29}$$

where $k(\mathbf{x}_i, \mathbf{x}_j)$ is a covariance function, as described in Sec. 3.2. In Gaussian processes the distribution over target values $\mathbf{y}$ with feature matrix X will therefore be written

$$\mathbf{y} \sim \mathcal{N}\big(\ m[\mathbf{y}], K(X, X)\ \big), \tag{3.30}$$

where $K(X, X)$ is the covariance matrix containing the covariances of the features in X, calculated using the covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$.

Finally, a Gaussian process will be denoted $\mathcal{GP}$. It may be difficult to distinguish between a Gaussian *distribution*, $\mathcal{N}$, and a Gaussian *process*, $\mathcal{GP}$. The difference can be thought of as the difference between a finite collection of function values $f_i = f(\mathbf{x}_i)$, and the continuous function, $f(\mathbf{x})$. The former can be viewed as a vector $\mathbf{f}$, and can be drawn from a distribution such as the one in Eq. (3.30), $\mathbf{f} \sim \mathcal{N}(\ m[\mathbf{f}], K(X, X)\ )$. The latter is a *function*, drawn from a distribution *over functions*, where the mean $m(\mathbf{x})$[3] and covariances $k(\mathbf{x}, \mathbf{x}')$ are functions as well. A function $f(\mathbf{x})$ drawn from a Gaussian process is written as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{3.31}$$

[3]Note that the mean *function* of $\mathbf{x}$ uses round brackets, $m(\mathbf{x})$, while the mean *value* of $\mathbf{x}$

Function Space View

Gaussian processes are distributions over functions. It is therefore useful to consider the problem in the function space view introduced in [28]. For a function $f(\mathbf{x})$ the Gaussian process *mean function*, $m(\mathbf{x})$, and *covariance function*, $k(\mathbf{x}, \mathbf{x}')$, are defined as

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{3.32}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{3.33}$$

In other words, the mean $m(\mathbf{x})$ is the expected value of the function $f(\mathbf{x})$ at $\mathbf{x}$, and the covariance function is the expected value of the simultaneous deviations of $f(\mathbf{x})$ from the mean $m(\mathbf{x})$ at $\mathbf{x}$, and $f(\mathbf{x}')$ from the mean $m(\mathbf{x}')$ at $\mathbf{x}'$. As mentioned, the mean and covariance are now *functions* of the input vector $\mathbf{x}$. This means that for every input vector $\mathbf{x}$, there is a Gaussian distribution over function values with a mean $m(\mathbf{x})$ and covariance with the function value at $\mathbf{x}'$ given by $k(\mathbf{x}, \mathbf{x}')$. This is a generalization of the single test point in Fig. 3.5, where every point $\mathbf{x}^*$ gets a similar distribution.

The collection of Gaussian distributions over functions that are now a function of the input vector $\mathbf{x}$ are the Gaussian processes, $\mathcal{GP}$. In the same way that a random variable is drawn from a distribution, random functions can be drawn from the $\mathcal{GP}$, as shown in Eq. (3.31). How functions are drawn from the $\mathcal{GP}$ is discussed in more detail in Sec. 3.3.2.

The covariance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ is determined by a covariance function $k(\mathbf{x}, \mathbf{x}')$. As mentioned, the covariance function calculates the covariance between the *function values*, $f(\mathbf{x})$ and $f(\mathbf{x}')$, and *not* the input vectors, $\mathbf{x}$ and $\mathbf{x}$. Rather, the covariance between two function values is a function of the input vectors. Consider again the illustration in Fig. 3.4. The variation in function values $f(\mathbf{x})$ and $f(\mathbf{x}')$ depends on the distance between the points $\mathbf{x}$ and $\mathbf{x}'$, and the characteristic length scale of the process. If the length scale is large, the two input vectors $\mathbf{x}$ and $\mathbf{x}'$ can be far away, and still have similar function values $f(\mathbf{x})$ and $f(\mathbf{x}')$. For short length scales, however, nearby points can have very different function values, because the function varies rapidly.

3.3.2 The Prior and Posterior Distribution

Gaussian processes are Bayesian in that there is a prior and a posterior distribution over functions, where the posterior is obtained by conditioning the prior on the training data. Consider the $n^* \times m$-matrix of test points X^*, containing n^* test points $\mathbf{x}_i^*$, with unknown function values $f(\mathbf{x}^*)$. Using the kernel function, $k(\mathbf{x}_i^*, \mathbf{x}_j^*)$, on the matrix X^* gives the covariance matrix, $K(X^*, X^*)$, as discussed in Sec. 3.2. The covariance matrix now contains the covariance of all test points

$\mathbf{x}_i^*$. Combined with an initial mean of zero[4] one obtains the *prior* distribution of predicted target values $\mathbf{f}^*$

$$\mathbf{f}^* \sim \mathcal{N}(\mathbf{0}, K(X^*, X^*)). \tag{3.34}$$

This distribution contains the prior assumptions about the function values $f(\mathbf{x})$, in that the smoothness of the function and the correlation between function values are encoded in the covariance matrix. This is the prior probability distribution discussed in Sec. 3.1.2, that will be modified by the data to provide the posterior probability distribution. The choice of kernel is therefore one of the most important steps in learning with GPs.

Noise-Free Model

Now consider the addition of training data to the prior distribution in Eq. (3.34), in the form of a noise-free set of n training points $\{\mathbf{x}_i, y_i\}$, so that $y = f(\mathbf{x})$. The input vectors $\mathbf{x}_i$ form the $n \times m$-matrix X, where the rows are the input vectors. The training targets y_i form a corresponding n-dimensional vector $\mathbf{y}$, which in the noise free case is equal to the latent function value vector $\mathbf{f}$. The test points are still contained in the $n^* \times m$ matrix X^*. The goal is to predict an n^*-dimensional vector $\mathbf{f}^*$ containing the predictions of the function values at the points $\mathbf{x}_i^*$, which is conditioned on the known function values $\mathbf{f}$.

The joint distribution of training outputs, $\mathbf{f}$, and test outputs, $\mathbf{f}^*$, according to the prior in Eq. (3.34) is

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X^*) \\ K(X, X^*) & K(X^*, X^*) \end{bmatrix}\right), \tag{3.35}$$

where, as before, $K(X_i, X_j')$ is the covariance matrix between the sets of points $\{\mathbf{x}_i\}$ and $\{\mathbf{x}_j'\}$ calculated using the covariance function $k(\mathbf{x}_i, \mathbf{x}_j')$. By conditioning the distribution of $\mathbf{f}^*$ on the observations $\mathbf{f}$, the posterior distribution over $\mathbf{f}^*$ is obtained[5] [28]

$$\begin{aligned} \mathbf{f}^* \big| X^*, X, \mathbf{f} \sim \mathcal{N}(&K(X^*, X)K(X, X)^{-1}\mathbf{f}, \\ &K(X^*, X^*) - K(X^*, X)K(X, X)^{-1}K(X, X^*)), \end{aligned} \tag{3.36}$$

where $\mathbf{f}^*|X^*, X, \mathbf{f}$ means the target values $\mathbf{f}^*$ for the features X^*, given the known features X and targets $\mathbf{f}$. The prior in Eq. (3.34) has been modified by the data, X and $\mathbf{f}$, to give the posterior in Eq. (3.36), leaving two different distributions from which function values can be drawn before and after the data.

uses square brackets $m[\mathbf{x}]$.

[4]The mean does not have to be zero, it could for example be the mean of the training data.

[5]For more details, see Appendix A.2 in [28].

Drawing Samples

All functions $f(\mathbf{x})$ that are drawn from the Gaussian process $\mathcal{GP}$ must have a covariance between function values $f(\mathbf{x})$ and $f(\mathbf{x}')$ determined by the covariance function $k(\mathbf{x}, \mathbf{x}')$. The mean of the distribution, $m(\mathbf{x})$, is *not* the mean value of each function $f(\mathbf{x})$ drawn from $\mathcal{GP}$, but rather the mean function value you would get at $\mathbf{x}$ if you drew enough functions from the $\mathcal{GP}$. Drawing functions from a distribution in this way will be referred to as *drawing samples*. The samples drawn here will in fact be function vectors $\mathbf{f}$ drawn from the normal distributions in Eq. (3.34) and Eq. (3.36), and not functions $f(\mathbf{x})$ from a Gaussian process $\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$.

To generate samples $\mathbf{f} \sim \mathcal{N}(\mathbf{m}, K)$ with a mean vector $\mathbf{m}$ and covariance matrix K using a scalar Gaussian generator[6], one proceeds as follows [28]: first the Cholesky decomposition L — also known as the matrix square root — of the covariance matrix is found using $K = LL^T$, where L is a lower triangular matrix. The Cholesky decomposition requires that the matrix K be Hermitian and positive-definite. A vector $\mathbf{u}$ is then generated by multiple calls to the scalar Gaussian generator $\mathbf{u} \sim \mathcal{N}(0, \mathbb{I})$. Then $\mathbf{f} = \mathbf{m} + L\mathbf{u}$ has the desired distribution with mean $\mathbf{m}$ and covariance

$$\text{cov}(\mathbf{f}) = \mathbb{E}[(\mathbf{f} - \mathbf{m})(\mathbf{f} - \mathbf{m})^T] = \mathbb{E}[(L\mathbf{u})(L\mathbf{u})^T] = L\mathbb{E}[\mathbf{u}\mathbf{u}^T]L^T = LL^T = K \tag{3.37}$$

by the independence of the elements of $\mathbf{u}$.

Using the method described above, random function vectors have been drawn from the prior and posterior in Eq. (3.34) and Eq. (3.36), respectively, and shown in Fig. 3.6. As discussed, the samples drawn from the prior have mean equal to zero $m[\mathbf{x}] = \mathbf{0}$, and constant covariance, $K(X^*, X^*)$, meaning that if you drew enough function vectors the mean of all function values at every $\mathbf{x}$ would be zero. The prior is shown in the upper panel of Fig. 3.6, where the mean of the distribution is represented by the thick, black line, and the covariance is the light blue band around the mean. In the posterior distribution the mean values and covariances have been modified by the training data, represented by red dots in the lower panel of Fig. 3.6. In a point where there is training data the uncertainty is zero[7], and so all samples drawn from the posterior distribution must pass through this point. Far away from training points the covariance is large. The mean has also been modified to pass through the training points, as seen by the thick black line in the lower panel.

[6] A scalar Gaussian generator generates random numbers from a Gaussian distributions, and can be found in most programming enviroments, such as the `random` environment in `Python`.

[7] Assuming there is no noise in the data.

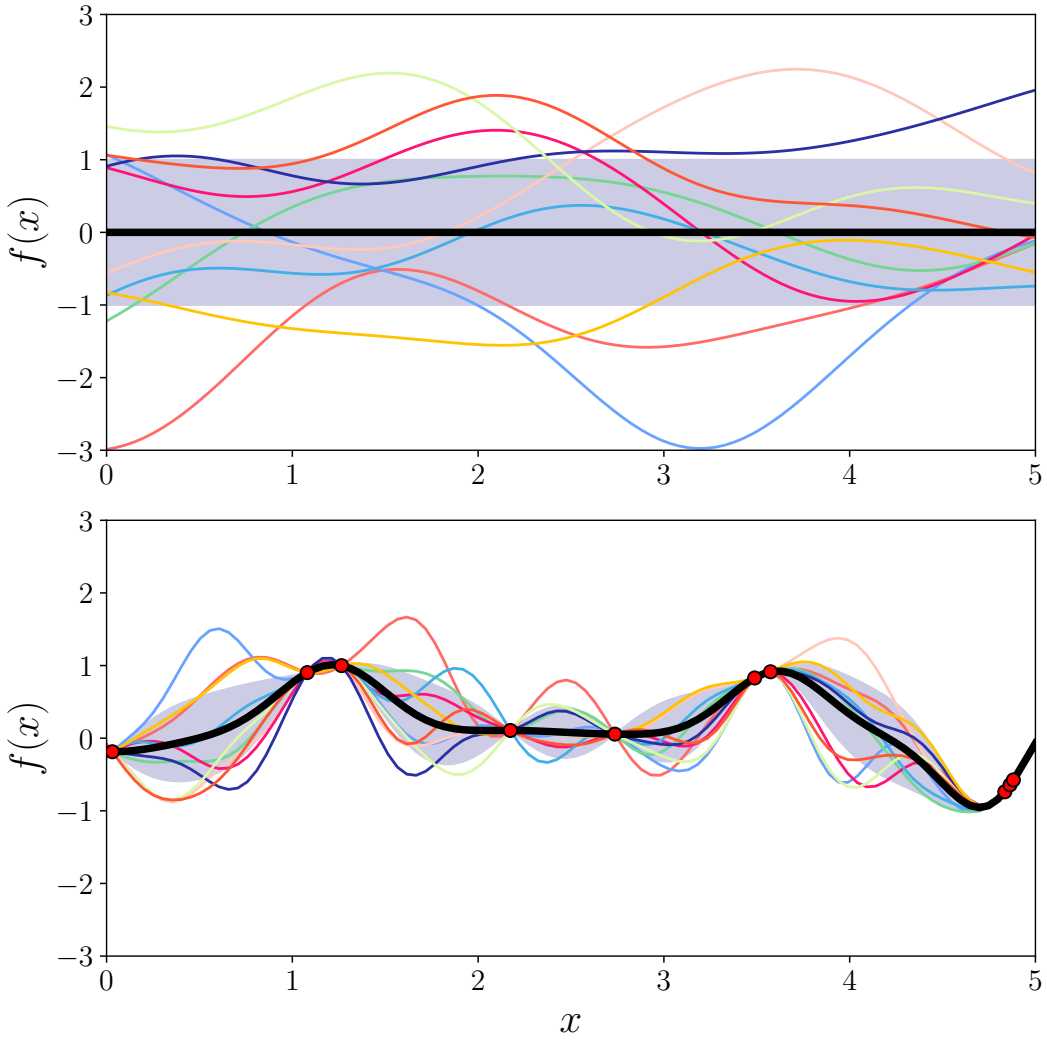


Figure 3.6: Drawing functions from the prior (top) and posterior (bottom) distributions as per the method described in Sec. 3.3.2. The thick, black line represents the mean $m[\mathbf{f}^*]$, while the shaded, blue area is the variance $K(X^*, X^*)$. The multiple coloured lines are functions drawn randomly from the prior and posterior distributions, whose correlation are dictated by the covariance function. Covariances are given by the RBF kernel $k(r) = \sigma_f^2 \exp(-r^2/\ell^2)$, and the prior has mean $\mathbf{0}$. The posterior (bottom panel) has been modified by training points (red dots), giving rise to zero uncertainty at the points where training data exists, and an altered mean value for the distribution. The kernel has hyperparameters $\sigma_f = 1$ and $\ell = 1$ for the prior, and $\sigma_f = 0.594$ and $\ell = 0.279$ for the posterior. Figure generated using `scikit-learn`.

Gaussian Noise Model

Noise-free observations are rare. In most cases targets will contain some noise $y = f(\mathbf{x}) + \varepsilon$, where the noise ε is here assumed to follow a Gaussian distribution $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. This is the *Gaussian noise model*. As discussed in Sec. 3.2 the covariance of a function with Gaussian noise can be expressed as

$$\operatorname{cov}(y_i, y_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 \delta_{ij}. \tag{3.38}$$

Training targets are contained in the n-dimensional vector $\mathbf{y}$, which is now different from the latent function vector $\mathbf{f}$, while training features are contained in the $n \times m$-matrix X, test features in the $n^* \times m$-matrix X^* and predicted targets in the n^*-dimensional vector $\mathbf{f}^*$, as before. Note that the goal is to predict the function values $\mathbf{f}^*$ and *not* the noisy values $\mathbf{y}^*$.

With the addition of noise the prior distribution becomes

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 \mathbb{I} & K(X,X^*) \\ K(X,X^*) & K(X^*,X^*) \end{bmatrix} \right). \tag{3.39}$$

The conditioned distribution is then

$$\mathbf{f}^* \big| X^*, X, \mathbf{y} \sim \mathcal{N}(m[\mathbf{f}^*], \operatorname{cov}(\mathbf{f}^*)) \tag{3.40}$$

where

$$m[\mathbf{f}^*] = K(X^*,X)[K(X,X) + \sigma_n^2 \mathbb{I}]^{-1} \mathbf{y}, \tag{3.41}$$

$$\operatorname{cov}(\mathbf{f}^*) = K(X^*,X^*) - K(X^*,X)[K(X,X) + \sigma_n^2 \mathbb{I}]^{-1} K(X,X^*), \tag{3.42}$$

which are similar to the mean and covariance in Eq. (3.36), but include noise in the covariance matrix.

Equations (3.41) and (3.42) are the key predictive equations for Gaussian process regression. The calculation requires inverting the $n \times n$-matrix containing the covariance $[K(X,X) + \sigma_n^2 \mathbb{I}]$, which for large n becomes computationally unviable. This is discussed further in Sec. 4.4, where distributed Gaussian processes are proposed as a way of scaling Gaussian processes to larger training sets.

To tidy up the expressions in Eqs. (3.41) and (3.42) the covariance matrices $K \equiv K(X,X)$ and $K^* \equiv K(X,X^*)$ are defined. For a single test point $\mathbf{x}^*$ the covariance matrix K^* is written as a vector $\mathbf{k}(X,\mathbf{x}^*) = \mathbf{k}^*$ to denote the covariances between the n training points and the test point $\mathbf{x}^*$. Using this compact notation the GP prediction of $f^* = f(\mathbf{x}^*)$ for a single test point $\mathbf{x}^*$ is

$$m[f^*] = \mathbf{k}^{*T} (K + \sigma_n^2 \mathbb{I})^{-1} \mathbf{y}, \tag{3.43}$$

$$\mathbb{V}[f^*] = k(\mathbf{x}^*,\mathbf{x}^*) - \mathbf{k}^{*T} (K + \sigma_n^2 \mathbb{I})^{-1} \mathbf{k}^*. \tag{3.44}$$

Note that, as intimated in Sec. 3.3.1, the predicted mean value $m[f^*]$ can be viewed as a linear combination of y_i of the form $\alpha\mathbf{y}$, where $\alpha = \mathbf{k}^{*T}(K + \sigma_n^2\mathbb{I})^{-1}$. The vector α then only contains the covariance between features.

Equations (3.43) and (3.44) form the basis for GP prediction in `scikit-learn` [29]. The algorithm for Gaussian process regression on a single test point $\mathbf{x}^*$, with training data X, $\mathbf{y}$ is outlined in Algorithm 1. The algorithm uses the Cholesky decomposition, L, of the covariance matrix to find the weights $\boldsymbol{\alpha}$ used to calculate the predictive mean f^*. The variance $\mathbb{V}[\mathbf{x}^*]$ is calculated using L and the covariance of the test point with the training points, $\mathbf{k}^* = k(X, \mathbf{x}^*)$.

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

Algorithm 1: Algorithm 2.1 from [28].

3.4 Model Selection

Choosing the right kernel and hyperparameters is an important part of Gaussian process regression. Finding the kernel and corresponding hyperparameters that best fit the data is often called *model selection*. Model selection is also referred to as *training* in statistical learning. In this section Bayesian model selection for the marginal likelihood is quickly overviewed, and the log marginal likelihood and cross validation are introduced as tools to optimise Gaussian process estimators.

3.4.1 Log Marginal Likelihood

The *marginal likelihood* can be used to find the optimal hyperparameters of covariance functions. Gaussian process regression models with Gaussian noise have the wonderful trait of analytically tractable integrals for the marginal likelihood, $P(\mathbf{y}|X, \boldsymbol{\theta})$. Note that the marginal likelihood defined here looks different than in Eq. (3.4) — $\boldsymbol{\theta}$ are the hyperparameters of the covariance functions introduced in Sec. 3.2.4, $\mathbf{y}$ are the training outputs and X are the training inputs. The parameters in Eq. (3.4), denoted Θ, are now the true — or *latent* — function values, $\mathbf{f}$ where $\mathbf{y} = \mathbf{f} + \varepsilon$. The marginal likelihood is the integral of the likelihood

times the prior over the latent function values $\mathbf{f}$, given by

$$P(\mathbf{y}|X,\boldsymbol{\theta}) = \int P(\mathbf{y}|\mathbf{f},X,\boldsymbol{\theta})P(\mathbf{f}|X,\boldsymbol{\theta})d\mathbf{f}, \tag{3.45}$$

where $P(\mathbf{f}|X,\boldsymbol{\theta})$ is the prior and $P(\mathbf{y}|\mathbf{f},X,\boldsymbol{\theta})$ is the likelihood. Under the Gaussian process model the prior is a Gaussian, $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2\mathbb{I})$, so the logarithm of the marginal likelihood in Eq. (3.45) gives for the *log marginal likelihood* (LML) [28]

$$\log P(\mathbf{y}|X,\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log\left|K_y\right| - \frac{n}{2}\log 2\pi, \tag{3.46}$$

where n is the number of training points, $K_y = K_f + \sigma_n^2\mathbb{I}$ is the covariance matrix for the noisy targets $\mathbf{y}$, and K_f is the covariance matrix for the latent function values $\mathbf{f}$.. Each term in Eq. (3.46) has an interpretation: $-\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2\mathbb{I})^{-1}\mathbf{y}$ is the only term involving the data, and is therefore the data-fit; $-\frac{1}{2}\log\left|K + \sigma_n^2\mathbb{I}\right|$ is the complexity penalty depending only on the covariance function and the inputs; and $-\frac{n}{2}\log 2\pi$ is a normalization term.

As mentioned, the goal is to use the marginal likelihood to determine the optimal hyperparameters of the covariance function, $\boldsymbol{\theta}$. Note that the marginal likelihood in Eq. (3.45) is the probability of the training outputs, $\mathbf{y}$, given the training inputs X and the hyperparameters $\boldsymbol{\theta}$. Maximizing the log marginal likelihood for the hyperparameters will therefore give the best estimate for the data $\mathbf{y}$ with respect to the hyperparameters $\boldsymbol{\theta}$, as per the discussion in Sec. 3.1.3. Maximizing the LML requires finding the partial derivatives

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|X,\boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^T K_y^{-1}\frac{\partial K_y}{\partial\theta_j}K_y^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(K_y^{-1}\frac{\partial K_y}{\partial\theta_j}), \tag{3.47}$$

Using partial derivatives, or the gradients, to find the optimal hyperparameters is called a *gradient based optimiser*. Computing the inverse of a matrix, K^{-1}, is computationally complex, and for n training points goes as $\mathcal{O}(n^3)$. Once this is done, however, finding the partial derivatives only requires complexity $\mathcal{O}(n^2)$, and so gradient based optimisers are advantageous.

The LML can have several local optima, as seen in the contour plot of the log marginal likelihood in Fig. 3.7. These correspond to different interpretations of the data. The leftmost optimum in Fig. 3.7, for example, favors a small length scale and smaller noise level. This means that little of the data is considered to be noise. The rightmost optimum has a higher noise level, and allows for a range of length scales, as most of the data is considered to be noise. Features with very large length scales are considered superfluous, as the function value depends little on them. To avoid ending up in a local optima, it can be wise to restart the optimiser a few times during learning.

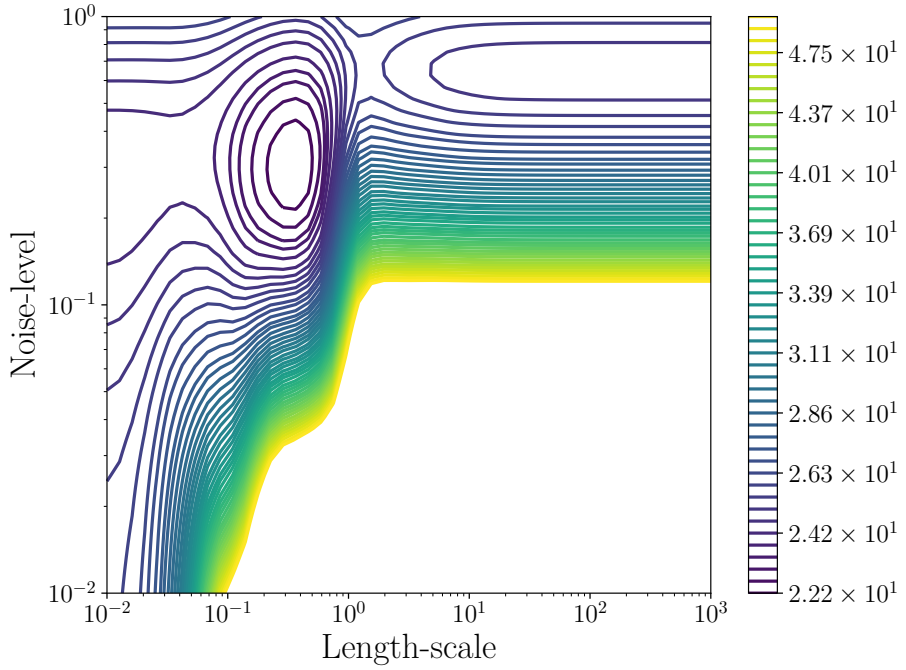


Figure 3.7: A contour plot of the log marginal likelihood with two local optima for a Gaussian process with kernel $k(r) = \sigma_f^2 \exp(-r^2/\ell^2) + \sigma_n^2 \delta_{ij}$. The rightmost optima favours a short length scale and low noise, with $\sigma_f = 0.64$, $\ell = 0.365$ and $\sigma_n^2 = 0.29$, while the leftmost favors a high noise level and therefore several large length scales, with $\sigma_f = 0.00316$, $\ell = 109$ and $\sigma_n^2 = 0.6$. The optimum to the right has LML -21.8 and the optimum to the right has LML -23.87. Plots were generated using `scikit-learn`.

3.4.2 Cross Validation

Cross validation is a means of monitoring the performance of a model. In k-fold validation this is done by dividing the data into k subsets and using $k-1$ folds to train the model, and a single fold to validate it. This is repeated k times, one for each possible validation set. Cross-validation requires a scoring function, such as the R^2 score. The R^2-score is given by

$$R^2 = 1 - \frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{N-1}(y_i - \bar{y})^2}, \tag{3.48}$$

where $\hat{y}_i$ is the predicted value of the ith sample, y_i is the true value and $\bar{y} = \frac{1}{N}\sum_{i=0}^{N-1} y_i$ for N samples. This is the score used for cross validation in this thesis.

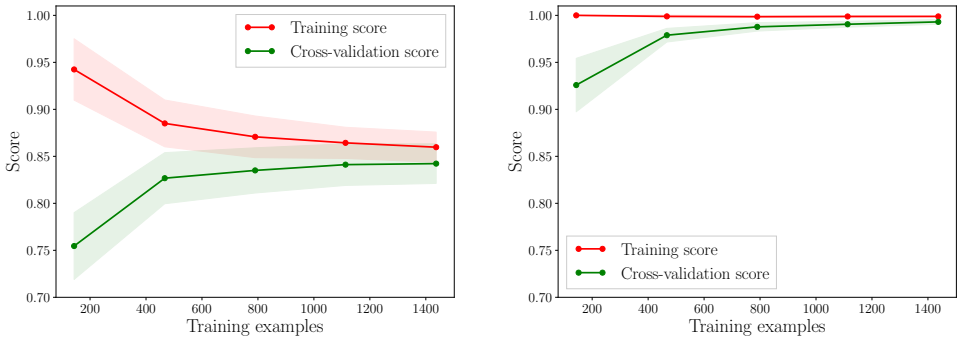
Cross-validation can be used to plot *learning curves*, which are used to find out whether the estimator benefits from adding more data. The learning curve plots the *training score* and *validation score*. The training score is the mean R^2-score for the k predictions of the training points, *i.e.* where the $k-1$ folds used for training are also used as test points. The validation score is the mean R^2 score for the k predictions of validation points, *i.e.* the one fold that is not a part of the training data. The variances of the R^2-scores are plotted as error bands. The test- and validation scores are used to find out if the model is *overfitting* or *underfitting*. *Overfitting* means that the model is a perfect fit to the training data, but predicts poorly for test data because it is not general. *Underfitting* occurs when the model is not able to capture the underlying structure of the data. Ideally, the training score should be 1, and the validation score should approach 1 with the addition of data.

Examples of learning curves are shown in Fig. 3.8 as a function of training points, for two statistical learning techniques called Naive Bayes and Support Vector Machine on an example problem. In **(a)** both the training score and cross-validation score tend to a value below 1, which indicates underfitting. This model will not benefit from more data. The example in **(b)** shows a training score of approximately 1, and a cross validation score that converges towards 1. This model could benefit from more data.

3.4.3 Relative Deviance

In this thesis predictions are compared using the *relative deviance*. For true values y_i and values predicted by the estimator $\hat{y}_i$ the relative deviance is given by

$$\varepsilon_i = \frac{y_i - \hat{y}_i}{y_i}. \tag{3.49}$$



(a) Underfitting. **(b)** Good fit.

Figure 3.8: Learning curves for two different estimators. The training scores are shown as red lines, with uncertainty bands in light red. The validation scores are shown as green lines, with uncertainty bands in light green. The estimator in **(a)** is underfitting, as both the training and validation tend to a value less than one. The estimator in **(b)** is a good fit, and could benefit from more data.

In this thesis the target values have a very wide span, ranging from about 10^{-30} fb to 10^{9} fb. The data is therefore divided into decades, meaning one set contains $\sigma \in [10^i, 10^{i+1}]$, where σ is the cross section. Then a distribution over the relative deviances within each decade is found, with a mean value, $m(\varepsilon_i)$, and standard deviation, $\text{std}(\varepsilon_i)$. These are plotted as a function of i, and denoted

$$m(\varepsilon_i) = m\left(\frac{y_i - \hat{y}_i}{y_i}\right), \tag{3.50}$$

$$\sigma^2(\varepsilon_i) = \mathbb{V}\left(\frac{y_i - \hat{y}_i}{y_i}\right), \tag{3.51}$$

$$\text{std}(\varepsilon_i) = \sqrt{\sigma^2(\varepsilon)}. \tag{3.52}$$

4

Evaluating Cross Sections using Gaussian Processes

This chapter is dedicated to improving the evaluation of cross sections for squark pair production with Gaussian processes. A benchmark Gaussian process estimator is considered and compared to estimators with possible improvements in the dataset, in the kernel and in the features. Thereafter the possibility of scaling Gaussian processes to larger datasets using distributed Gaussian processes is explored.

4.1 Data Generation

In this section the generation of MSSM-24 training and test data is discussed, following closely the discussion in [30].

Sampling of Data

An MPI parallelised `Python` script generates a sample point in the MSSM-24 parameter space by drawing random values from the distributions in Table 4.1. For the CMSSM test set the sample point is drawn from the distributions in Tab. 4.2. When a parameter point has been sampled, it is run through the program `softpoint.x` which calculates its supersymmetric spectrum using the `Softsusy 3.6.2`-package [31]. The spectrum is then written to a SLHA-file [32] that is given as input to `Prospino 2.1`, which subsequently calculates the LO and NLO cross sections according to the method outlined in Section 3.4.1, and the results are written to the SLHA-file. The relevant masses and NLO cross sections are later harvested to input files, which are used by the Gaussian processes.

The weak scale MSSM model used in this thesis, MSSM-24, requires a soft breaking scale Q for its definition, as discussed in Sec. 1.3.3. This scale is set to $Q = 1$ TeV. It is worth noting that the parameter space for the squark cross

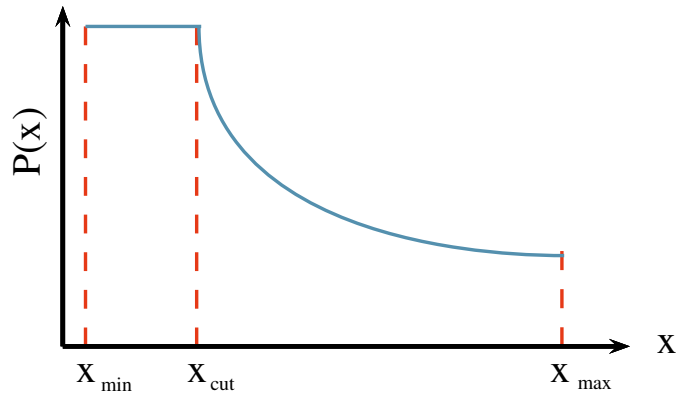


Figure 4.1: Illustration of the log prior distribution $P(x)$. Around $x = 0$ the prior would blow up, so for $x < x_{cut}$ a flat prior is used.

sections is significantly reduced from that of the MSSM-24. The cross sections depend on the values of the masses $m_{\tilde{q}}$ and $m_{\tilde{g}}$. Since only first and second generation squarks are considered, $m_{\tilde{q}}$ contributes with eight masses[1] which are combined with $m_{\tilde{g}}$ to reduce the parameter space to nine dimensions. The parameter space is therefore better sampled than it would appear from the 24 parameters in MSSM-24. The cross sections for the individual squark production processes only depend on $m_{\tilde{g}}$, the masses of the final state squarks, and the mean squark mass, thus reducing the parameter space to four dimensions.

Priors

To get a reasonable coverage of parameter space, a flat and a log prior distribution are used when sampling the parameter space. A flat prior from *e.g.* 0 TeV to 4 TeV will draw approximately the same number of samples in the interval $[0, 100]$ GeV as in the intervals $[1, 1.1]$ TeV, $[1.1, 1.2]$ TeV, $[1.2, 1.3]$ TeV and so on. Almost all samples drawn from this distribution will be of the order of $\mathcal{O}(1\text{ TeV})$, and samples of the order of $\mathcal{O}(1\text{ GeV})$ will be poorly represented.

A log prior, on the other hand, gives an equal probability to each *order of magnitude*. Consequently, values between $[10, 100]$ GeV have the same probability as values between $[0.1, 1]$ TeV. Small masses give the largest cross sections, so sampling of the small mass space is important. On the other hand, there have been no signs of supersymmetry in results from the LHC data analysis, indicating that squark and gluino masses are of the order of $\mathcal{O}(1\text{ TeV})$. Large masses should therefore also be well sampled.

As a result, a combination of the log and flat priors is used in order to properly cover parameter space. To avoid divergence of the log prior close

[1] Four flavours with a pair of lefthanded and righthanded squarks each.

Parameter	Log prior range	Flat prior range
M_1	[0,100,4000]	[0,4000]
M_2	[0,100,4000]	[0,4000]
M_3	[0,100,4000]	[0,4000]
A_t	[-4000, -100, 100, 4000]	[-4000, 4000]
A_b	[-4000, -100, 100, 4000]	[-4000, 4000]
A_τ	[-4000, -100, 100, 4000]	[-4000, 4000]
μ	[-4000, -100, 100, 4000]	[-4000, 4000]
m_A^{pole}	[0,100,4000]	[0,4000]
$\tan\beta$	[2, 60]	[2, 60]
m_{L_1}	[0, 100, 4000]	[0, 4000]
m_{L_2}	[0, 100, 4000]	[0, 4000]
m_{L_3}	[0, 100, 4000]	[0, 4000]
m_{e_1}	[0, 100, 4000]	[0, 4000]
m_{e_2}	[0, 100, 4000]	[0, 4000]
m_{e_3}	[0, 100, 4000]	[0, 4000]
m_{Q_1}	[0, 100, 4000]	[0, 4000]
m_{Q_2}	[0, 100, 4000]	[0, 4000]
m_{Q_3}	[0, 100, 4000]	[0, 4000]
m_{u_1}	[0, 100, 4000]	[0, 4000]
m_{u_2}	[0, 100, 4000]	[0, 4000]
m_{u_3}	[0, 100, 4000]	[0, 4000]
m_{d_1}	[0, 100, 4000]	[0, 4000]
m_{d_2}	[0, 100, 4000]	[0, 4000]
m_{d_3}	[0, 100, 4000]	[0, 4000]

Table 4.1: Table showing the sampling intervals used for the parameters when sampling the MSSM-24, where the soft breaking scale is set to $Q = 1$ TeV. The log priors have three and four limit values, which are of the form `[flat_start, start, end]` and `[start, flat_start, flat_end, end]`. All values in GeV except $\tan\beta$ which is unitless. Table from [30].

Parameter	Flat prior range
m_0	$[0, 5000]$
$m_{1/2}$	$[0, 5000]$
A_0	$[-5000, 5000]$
$\tan\beta$	$[2, 60]$
sgn μ	$\div$

Table 4.2: Parameters sampled for the CMSSM data. All values except sgn μ and $\tan\beta$ are given in GeV. The sign of μ has an equal probability of being positive or negative, and is here chosen to be negative. Table from [30].

to zero this region is covered by a flat prior. The limits on the priors are `[start, flat_start, flat_end, end]` for priors that include negative values, and `[flat_start, start, end]` for priors with only positive values. An illustration of a log prior with positive values is shown in Fig. 4.1, where a flat distribution is used close to $x = 0$ to avoid divergences.

Data Quality

Data quality plots are used to ensure that the sampled data is properly distributed in parameter space. In Fig. 4.2 scatter plot examples for $m_{\tilde{g}}$, $m_{\tilde{d}_L}$, $m_{\tilde{u}_L}$ and $m_{\tilde{u}_R}$ are shown. Mass distributions for the other squark masses are similar, and can be found in Appendix B. The scatter plots use 4000 points. All parts of parameter space seem to be covered, although the density of points is higher for small masses, as can be expected from the log prior. In the top panel of Fig. 4.2 the gluino mass is plotted against $m_{\tilde{d}_L}$, and the cross sections in different decades are shown in different colors. The largest cross sections have very little spread as a function of the gluino mass, while cross sections larger than $\sigma > 10^{-4}$ fb are quite evenly spread as functions of the gluino mass.

The bottom panel in Fig. 4.2 shows scatter plots of both $m_{\tilde{u}_L}$ and $m_{\tilde{u}_R}$ versus $m_{\tilde{d}_L}$. As discussed in Sec. 1.4.6, same generation left-handed squarks come in $SU(2)_L$ doublets, and their masses are predominantly determined by *one* mass parameter because they must form a complete $SU(2)_L$ representation. The mass parameter responsible is the soft mass $m^2_{Q_i}$ for generation i. Right-handed squarks, on the other hand, get their masses from different parameters $m^2_{\tilde{d}_i}$ and $m^2_{\tilde{u}_i}$, and the masses are therefore independent of each other in the MSSM-24. The mass splitting between same-generation left-handed squarks, *e.g.* $\widetilde{d}_L$ and $\widetilde{u}_L$, was given in Sec. 1.4.5, and is

$$m^2_{\tilde{d}_L} - m^2_{\tilde{u}_L} = -\cos 2\beta \; m^2_W,$$

where $m^2_W \approx 80$ GeV. The mass splitting is relatively small for large masses,

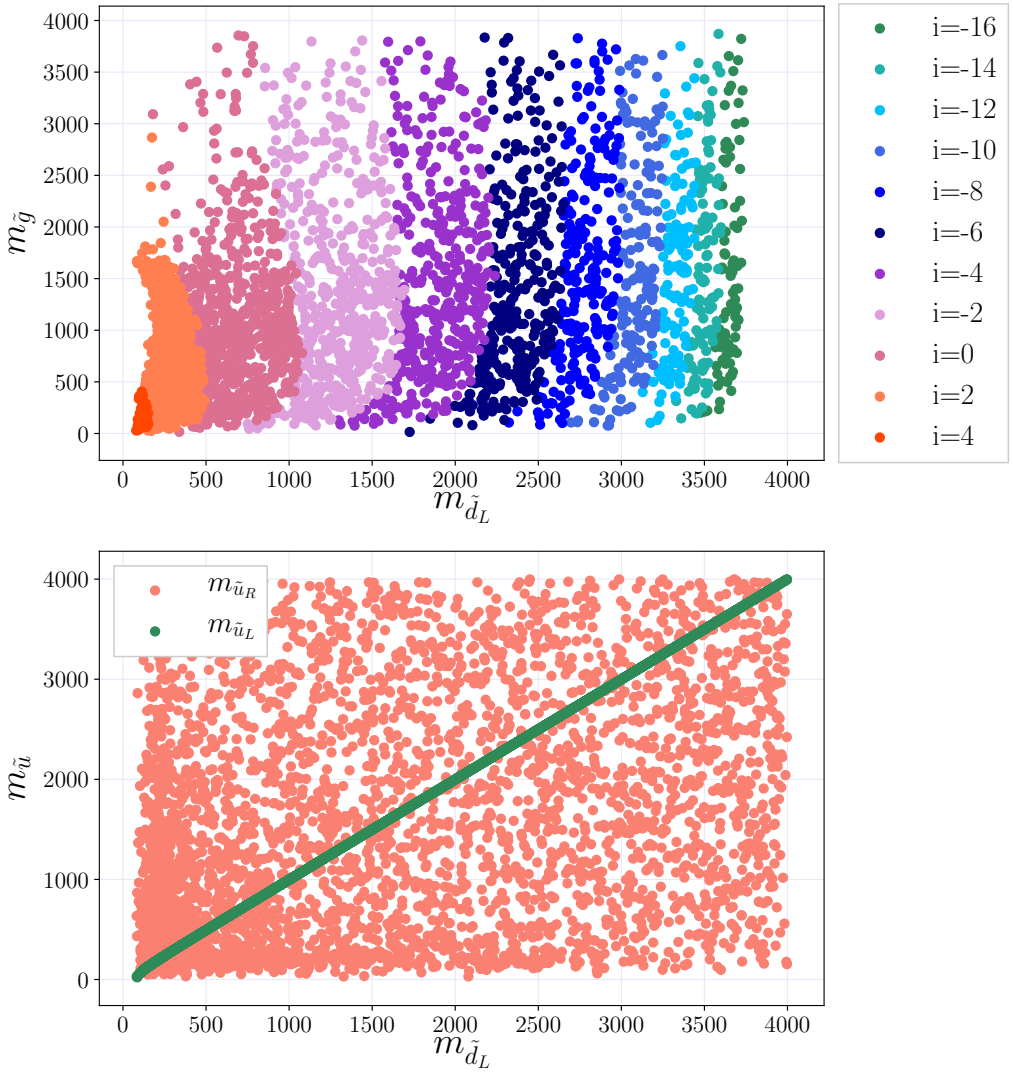


Figure 4.2: Data quality plots of the distribution of mass parameters $m_{\widetilde{g}}$, $m_{\widetilde{d}_L}$, $m_{\widetilde{u}_L}$ and $m_{\widetilde{u}_R}$ for 4000 points. In the top panel the decades of the cross sections, $i = \log_{10} \sigma/\sigma_0$ for $\sigma_0 = 1$ fb, for $\widetilde{d}_L\widetilde{d}_L$ are shown in different colours. There are few points for the largest cross sections, and smaller cross sections are more spread across the gluino mass spectrum.

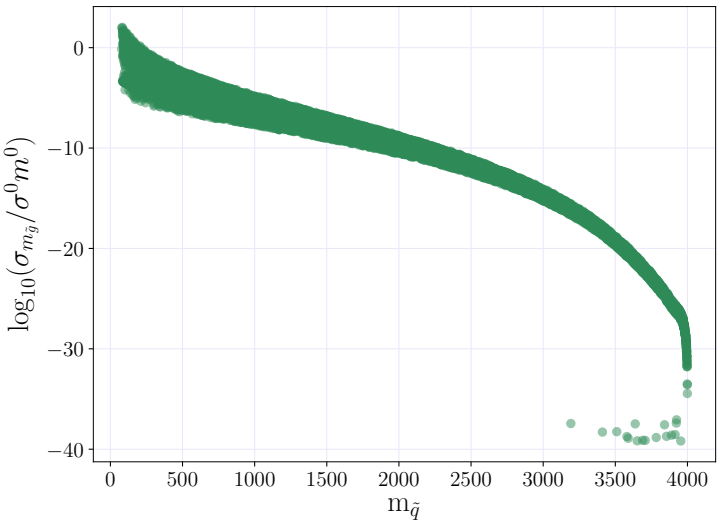


Figure 4.3: Plot of $\log(\sigma_{m_{\tilde{g}}})$ for $\widetilde{d}_L\widetilde{d}_L$ as a function of $m_{\widetilde{d}_L}$, where outliers are included. The outliers are originally $\sigma = 0$ fb, but are set to $\sigma = 10^{-32}$ fb so as to avoid infinities. The outliers have different values here because the cross sections have been divided by $m_{\tilde{g}}^2$.

but becomes significant for small $m_{\widetilde{d}_L}$ and $m_{\widetilde{u}_L}$. It is therefore a possibility that training on processes with same-generation left-handed squarks, $\widetilde{u}_L\widetilde{d}_L$, $\widetilde{s}_L\widetilde{c}_L$, will effectively be training on a single squark mass.

As discussed in Sec. 2.4.1, calculations in `Prospino 2.1` set some NLO terms to zero. These points become outliers in the dataset. The outliers can be seen as a cluster of points well below the others for large masses in Fig. 4.3, where $\sigma_{m_{\tilde{g}}}$ is plotted as a function of $m_{\tilde{q}}$. These points have zero cross sections, which are set to 10^{-32} fb in to avoid divergences when taking the logarithm. In Fig. 4.3 they have different values because the cross sections have been divided by the gluino mass squared.

Noise

The data contains some noise that originates in the numerical `Prospino 2.1` calculation, see Sec. 2.4.1. In a parameter point chosen at random, the relative error ϵ_i has a typical variance of $\sigma_\epsilon^2 \simeq 4.0{\cdot}10^{-6}$. This error fluctuates little between parameter points because there is a convergence criterium in `Prospino 2.1`. The variance $\sigma_\epsilon^2 \simeq 4.0 \cdot 10^{-6}$ is therefore considered a good approximation for the order of magnitude of errors in all points. The goal is now to incorporate this information in the Gaussian processes. For that purpose the following relation is

considered,

$$Y_i = y_i^{true} + \Delta y_i = y_i^{true}(1 + \epsilon_i), \tag{4.1}$$

where Y_i are cross sections provided by `Prospino 2.1`, y_i^{true} are true cross sections and Δy_i is the numerical error. The *relative error* from the calculation is assumed to follow a Gaussian distribution $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$.

The target values used in this thesis will in fact be the logarithm of the cross sections,

$$\begin{aligned} X_i &= \log_{10} Y_i \\ &= \log_{10}(y_i^{true}) + \log_{10}(1 + \epsilon_i). \end{aligned} \tag{4.2}$$

For small ϵ_i the second term in Eq. (4.2) can be expanded in a Taylor series

$$\log_{10}(1 + \epsilon_i) = \frac{\epsilon_i}{\log 10} - \frac{\epsilon_i^2}{2 \log 10} + \mathcal{O}(\epsilon_i^3). \tag{4.3}$$

The first term in Eq. (4.3) is dominant, and for small enough ϵ_i the following approximation can be made

$$\log_{10}(1 + \epsilon_i) \approx \frac{1}{\log 10}\ \epsilon_i. \tag{4.4}$$

The targets in Eq. (4.2) can now be written as

$$X_i \approx \log_{10}(y_i^{true}) + \frac{1}{\log 10}\ \epsilon_i = \log_{10}(y_i^{true}) + \varepsilon_i \tag{4.5}$$

where the *total error*, ε_i, follows a Gaussian distribution $\mathcal{N}(0, \sigma_\varepsilon^2)$. For a random variable X multiplied by a number c the variance is given by

$$\mathbb{V}[cX] = c^2 \cdot \mathbb{V}[X], \tag{4.6}$$

so the variance σ_ε^2 is given by

$$\sigma_\varepsilon^2 = \frac{\sigma_\epsilon^2}{(\log 10)^2}. \tag{4.7}$$

Equation (4.5) is now on the form of the Gaussian noise model discussed in Sec. 3.3.2. The distribution of the relative error ϵ_i has variance $\sigma_\epsilon^2 = 4.0 \cdot 10^{-6}$, so the variance of the Gaussian distributed noise should be

$$\sigma_\varepsilon^2 = \frac{4.0 \cdot 10^{-6}}{(\log 10)^2} \simeq 7.5 \cdot 10^{-7}.$$

The noise term predicted by the Gaussian process estimator should therefore have a variance of the order $\sigma_\varepsilon^2 \sim \mathcal{O}(10^{-7} - 10^{-6})$.

4.2 Dataset Transformations

The plot in Fig. 4.2 indicates that cross sections, in particular of the order of $\mathcal{O}(1\text{ fb})$ and lower, are very spread as functions of $m_{\tilde{g}}$. The spread is also evident in the upper left panel of Fig. 4.4, where the logarithm of the cross section for the production of $\widetilde{d}_L\widetilde{d}_L$ is plotted as a function of the gluino mass. Spread function values are problematic because they make it difficult for the Gaussian process regressor to recognise the shape of the function. The upper left panel shows σ as a function of the squark mass $m_{\tilde{q}}$, which is also quite spread. In addition, the cross section is a very steep function of $m_{\tilde{q}}$ for small cross sections, and relatively flat for large large cross sections. High derivatives in the function means that training points need to be close for a regression algorithm to estimate the actual shape of the function, which presents another difficulty. Some of the steepness and spread is remedied by using $\log_{10}\sigma$ instead of σ. This section will be devoted to reducing the spread caused by the gluino mass dependency.

Scaling Functions

As discussed in Sec. 2.4, the partonic cross sections can be written in terms of scaling functions f, see Eq. (2.19). The scaling functions are the different contributions to the cross section, as explained in Sec. 2.4. The total cross section only differs from the partonic cross section by an integral over parton distribution functions, so the mass dependencies in Eq. (2.19) are relevant for the total cross sections as well.

The main contributions to the cross section come from the scaling functions at the threshold energy given in Eq. (2.22), which makes possible to remove some of the complexity of the function. Note that all terms are proportional to f^B_{qq} ($f^B_{q'q}$), which is again proportional to $m^2_{\tilde{q}}m^2_{\tilde{g}}/(m^2_{\tilde{g}}+m^2_{\tilde{q}})^2$,

$$\sigma \propto \frac{m^2_{\tilde{q}}m^2_{\tilde{g}}}{(m^2_{\tilde{g}}+m^2_{\tilde{q}})^2}. \tag{4.8}$$

All partonic cross sections are proportional to $\sigma \propto 1/m^2$, where m is the average mass of the final state particles. In squark pair production $m^2 = m^2_{\tilde{q}}$, so the proportional squark mass dependency in Eq. (4.8) is automatically cancelled. However, the following transformation can be made

$$\sigma \to \sigma_{m_{\tilde{g}}} = \frac{\sigma}{m^2_{\tilde{g}}}, \tag{4.9}$$

reducing the gluino mass dependency. Another possibility is to further reduce the mass dependecy by defining σ_{fac} as

$$\sigma_{fac} = \sigma\frac{(m^2_{\tilde{g}}+m^2_{\tilde{q}})^2}{m^2_{\tilde{g}}}. \tag{4.10}$$

The middle and bottom panels in Fig. 4.4 show $\sigma_{m_{\tilde{g}}}$ and σ_{fac} as functions of $m_{\tilde{g}}$ and $m_{\tilde{q}}$. The spread as a function of the squark mass is reduced for both expressions, but notably more for σ_{fac} than for $\sigma_{m_{\tilde{g}}}$. However, for large cross sections the shape of $\sigma_{m_{\tilde{g}}}$ as a function of squark and gluino mass are very similar, while both σ and σ_{fac} are very flat functions of $m_{\tilde{g}}$. The similarity in $\sigma_{m_{\tilde{g}}}$ may facilitate finding a kernel that fits well in both dimensions. Both $\sigma_{m_{\tilde{g}}}$ and σ_{fac} were tested as target values, and $\sigma_{m_{\tilde{g}}}$ gave the best results. The target value in this thesis is therefore the logarithm of $\sigma_{m_{\tilde{g}}}$,

$$\log_{10} \sigma_{m_{\tilde{g}}}. \tag{4.11}$$

In the threshold region the partonic version of $\sigma_{m_{\tilde{g}}}$ is given by

$$\hat{\sigma}_{ij,m_{\tilde{g}}} = \frac{8\pi\beta\alpha_s^2(Q^2)}{27(m_{\tilde{q}}^2+m_{\tilde{g}}^2)^2}\Bigg\{1+4\pi\alpha_s(Q^2)\Bigg[\frac{1}{24\beta}+\frac{2}{3\pi^2}\log^2(8\beta^2) - \frac{7}{2\pi^2}\log(8\beta^2)-\frac{2}{3\pi^2}\log(8\beta^2)\log\Big(\frac{Q^2}{m^2}\Big)\Bigg]\Bigg\}. \tag{4.12}$$

4.3 Training the Gaussian Process

In this section a Gaussian process estimator is trained with benchmark settings for sample processes, and a selected set of modifications to the benchmark estimator are introduced. The quality of estimators is quantified in plots of the relative deviance distributions for deviances between the GP predictions and `Prospino 2.1` calculations, as defined in Sec. 3.4.3. Note that in many cases the hyperparameter σ_f^2 is at its upper limit, $\sigma_{f\,\mathrm{max}}^2 = 1000$. Increasing the upper limit on σ_f^2 alters the optimised length scale hyperparameters, but does not change the prediced cross sections.

4.3.1 The Benchmark

The sample processes in this section are the pair production of $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{d}_L\widetilde{u}_R$. The benchmark settings (BM) for the Gaussian process (GP) estimator are

- A GP estimator with 2000 training points.
- 20 000 test points.
- Features $m_{\tilde{g}}, m_{\widetilde{d}_L}$ ($m_{\tilde{g}}, m_{\widetilde{d}_L}, m_{\widetilde{u}_R}$) for $\widetilde{d}_L\widetilde{d}_L$ ($\widetilde{d}_L\widetilde{u}_R$) production.
- The exponential squared kernel (RBF) with a white noise term,

$$k(\mathbf{x}_i,\mathbf{x}_j) \;=\; \sigma_f^2 \exp\big(-\frac{1}{2}(\mathbf{x}_i-\mathbf{x}_j)^T M(\mathbf{x}-\mathbf{x}')\big)+\sigma_n^2\delta_{ij}, \tag{4.13}$$

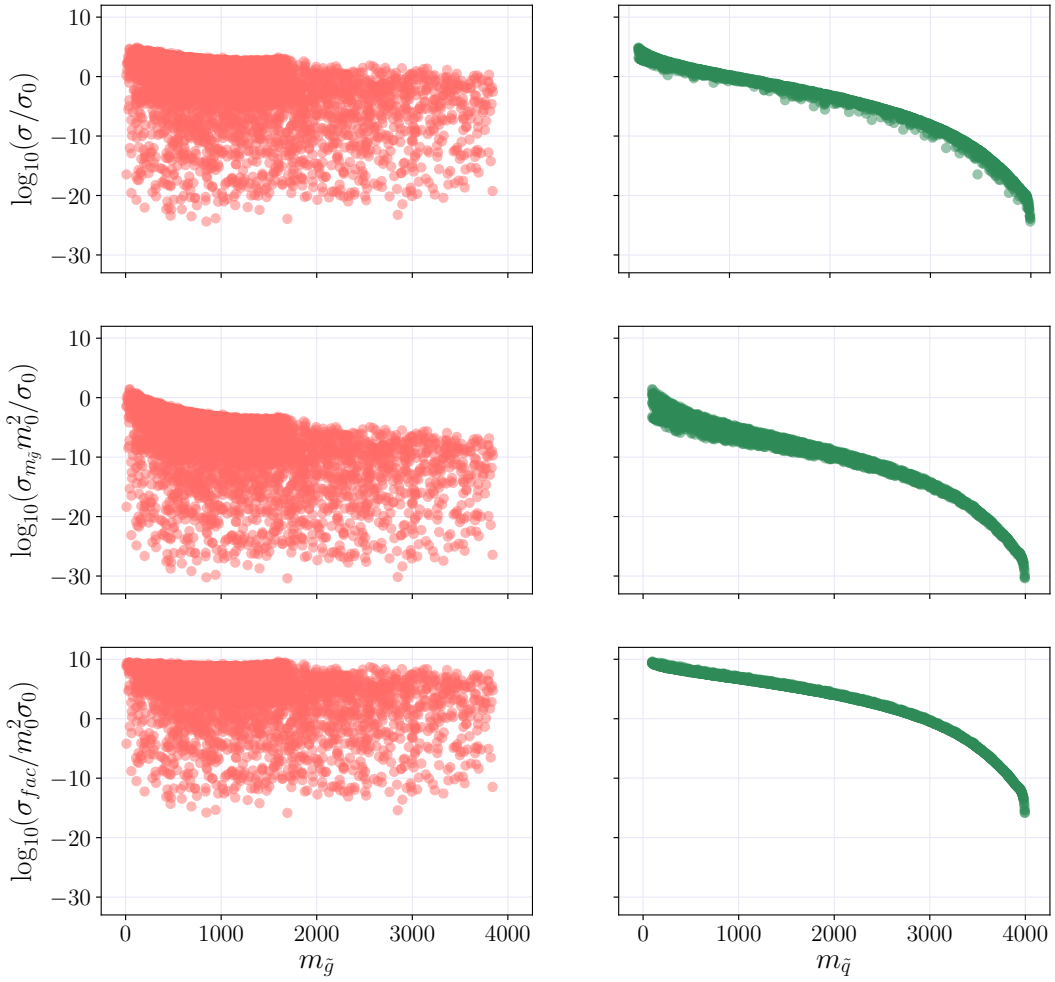


Figure 4.4: The cross section, σ, and the modified cross sections, $\sigma_{m_{\tilde{g}}}$ and σ_{fac}, as functions of the gluino mass $m_{\tilde{g}}$ and squark mass $m_{\tilde{q}} = m_{\widetilde{d}_L}$ for the production of $\widetilde{d}_L\widetilde{d}_L$. The distributions are less spread when some of the mass dependency is removed.

	σ_f	$\ell_{m_{\tilde{g}}}$	$\ell_{m_{\tilde{d}_L}}$	$\ell_{\bar{m}}$	σ_n^2
BM	31.6	4600	1880		0.47
No outliers	31.6	3920	182		0.00372
$\sigma > 10^{-16}$ fb	22.7	1170	998		0.00336
$\bar{m}$	31.6	1180	199	893	0.0000119
Matern	31.6	30200	8600		0.462

Table 4.3: Optimised kernel hyperparameters of different estimators for the $\widetilde{d}_L\widetilde{d}_L$ cross section.

where $M = \text{diag}(\boldsymbol{\ell})^{-2}$, and σ_n^2 is the variance of the distribution of white noise $\varepsilon_{WK} \sim \mathcal{N}(0, \sigma_n^2)$. The hyperparameters are $\boldsymbol{\theta} = \{\sigma_f^2, \boldsymbol{\ell}, \sigma_n^2\}$.

The kernel is implemented in `scikit-learn` in the following way

```
kernel_BM =  ConstantKernel(constant_value=10,
    constant_value_bounds=(1e-3, 1e4)) * RBF(length_scale =
    np.array([1000, 1000]), length_scale_bounds=(1, 1e6)) +
    WhiteKernel(noise_level=1, noise_level_bounds=(2e-10,1e2))
```

Note that the length scale of the RBF is given as a vector of the same dimension as the feature vector $\mathbf{x}, \boldsymbol{\ell} \in \mathbb{R}^D$. This is to allow different features to have different characteristic length scales, as they appear to have from the lower panels in Fig. 4.4. The target values are $\log_{10} \sigma_{m_{\tilde{g}}}$, where $\sigma_{m_{\tilde{g}}}$ was defined in Sec. 4.2.

The means and standard deviations of the relative deviance distributions, $m(\varepsilon_i)$ and $\text{std}(\varepsilon_i)$ as defined in Eqs. (3.50) – (3.51), are plotted for the BM settings in Fig. 4.5 for $\widetilde{d}_L\widetilde{d}_L$. For $\widetilde{d}_L\widetilde{u}_R$ the results are very similar, and therefore not shown here. The optimised kernel hyperparameters of the GP are listed in Tables 4.3 – 4.4, while computation times on an intel i5 CPU are listed in Table 4.5. The estimated noise level variances, σ_n^2, are large for both processes, at $\sigma_n^2 = 0.47$ for $\widetilde{d}_L\widetilde{d}_L$ and $\sigma_n^2 = 0.65$ for $\widetilde{d}_L\widetilde{u}_R$, which is far from the value $\sigma_\varepsilon^2 \sim 7.5 \cdot 10^{-7}$ predicted in Sec. 4.1. In addition, the GP estimator is unstable. The mean relative deviances oscillate around zero, as can be seen in Fig. 4.5.

4.3.2 Outliers

The data contains outliers that originate in the `Prospino 2.1` calculations, as discussed in Sec. 4.1. To investigate the effect of outlier points on the GP predicted cross sections, a GP estimator with the BM settings is trained on a dataset where outliers have been removed.

In Fig. 4.6a the BM estimators with and without outliers are compared. With the removal of outliers, the relative deviance distribution means, $m(\varepsilon_i)$, are close

	σ_f	$\ell_{m_{\tilde{g}}}$	$\ell_{m_{\tilde{d}_L}}$	$\ell_{m_{\tilde{u}_R}}$	$\ell_{\bar{m}}$	σ_n^2
BM	22.7	3070	2700	3770		0.65
No outliers	31.6	1010	3030	3240		0.00316
$\sigma > 10^{-16}$ fb	31.6	1150	3230	3310		0.00276
$\bar{m}$	166	795	5010	3910	742	0.0000697
Matérn	10.9	3220	1700	6950		0.402

Table 4.4: Optimised kernel hyperparameters of different estimators for the $\widetilde{d}_L\widetilde{u}_R$ cross section.

	Time $\widetilde{d}_L\widetilde{d}_L$	Time $\widetilde{d}_L\widetilde{u}_R$
BM	00:07:48	00:08:40
No outliers	00:08:42	00:11:20
$\sigma > 10^{-16}$ fb	00:07:24	00:11:07
$\bar{m}$	00:11:20	00:16:37
Matérn	00:07:28	00:13:05

Table 4.5: Computation times with 2000 training points and 20 000 test points on an intel i5 CPU.

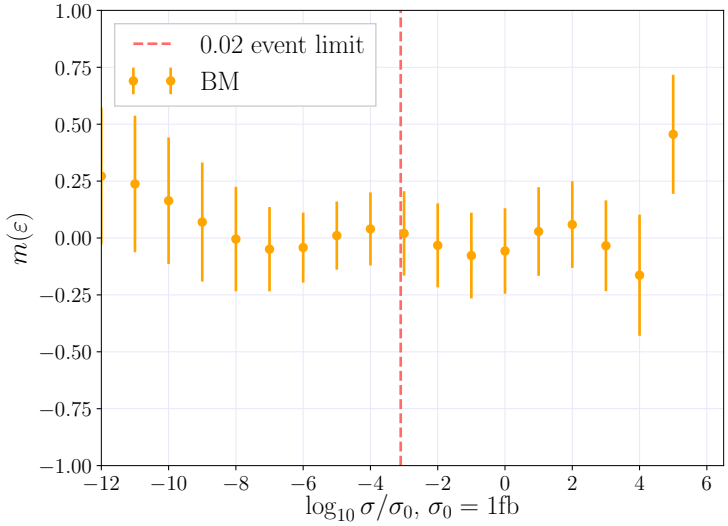


Figure 4.5: The mean and standard deviation of the relative deviance distributions, $m(\varepsilon_i)$ and std(ε_i), as a function of $i = \log_{10}\sigma/\sigma_0$, for the process $\widetilde{d}_L\widetilde{d}_L$ with benchmark settings. The optimised kernel hyperparameters are listed in Table 4.3.

to zero for all orders of magnitude. The standard deviations, std(ε_i), are reduced for most orders of magnitude, but are larger for *e.g.* $i = 0$ and $i = 2$ in Fig. 4.6a. The gluino length scale $\ell_{m_{\tilde{g}}}$ is reduced for both processes, going from $\ell_{m_{\tilde{g}}} = 4600$ to $\ell_{m_{\tilde{g}}} = 3920$ for $\widetilde{d}_L\widetilde{d}_L$, and from $\ell_{m_{\tilde{g}}} = 3070$ to $\ell_{m_{\tilde{g}}} = 1010$ for $\widetilde{d}_L\widetilde{u}_R$. For $\widetilde{d}_L\widetilde{d}_L$ the length scale of the squark mass is also reduced, from $\ell_{m_{\tilde{d}_L}} = 1880$ to $\ell_{m_{\tilde{d}_L}} = 182$. However, as the length scales change when the upper bounds on the hyperparameter σ_f^2 is raised, without changing the predicted cross sections, they are not as reliable as the noise levels to indicate the quality of the estimator. The estimated noise levels are reduced significantly with the removal of outliers, from $\sigma_n^2 = 0.47$ to $\sigma_n^2 = 0.00372$ for $\widetilde{d}_L\widetilde{d}_L$ and from $\sigma_n^2 = 0.65$ to $\sigma_n^2 = 0.00316$ for $\widetilde{d}_L\widetilde{u}_R$. The reduced noise levels imply that the GP estimator considers the outliers to be noise. However, as the noise level is the noise distribution for *all* datapoints[2], much of the signal is also considered noise, which could explain the large length scales. Including outliers therefore leads the estimator to underfit.

4.3.3 Cuts on Cross Sections

Smooth functions are easier to fit with Gaussian processes. For small cross sections the target values are very steep functions of the squark mass, as can be seen from the righthand panels in Fig. 4.4. In addition, small target values comprise the regions with the most spread as a function of the gluino mass. Since the limit for 0.02 events at the LHC with 20 fb^{-1} of integrated luminosity is at $\sigma = 10^{-3}$ fb,[3] a lower cut is introduced for the cross sections $\sigma_{cut} = 10^{-16}$ fb, well below the 0.02 event limit. The cut excludes all cross sections lower than σ_{cut} from both training and testing.

The resulting relative deviance distributions for $\widetilde{d}_L\widetilde{d}_L$ are shown in Fig. 4.6b, and the optimised kernel hyperparameters are listed in Tables 4.3 – 4.4. The estimated noise levels are lower than for the estimator in Sec. 4.3.2, with the variance going from $\sigma_n^2 = 0.00372$ to $\sigma_n^2 = 0.00336$ for $\widetilde{d}_L\widetilde{d}_L$ and from $\sigma_n^2 = 0.00316$ to $\sigma_n^2 = 0.00276$ for $\widetilde{d}_L\widetilde{u}_R$. The GP estimated noise levels seem to be approaching the value predicted in Sec. 4.1, indicating that the exclusion of low cross sections improves the GP estimator.

The relative deviance distributions of the largest cross sections are significantly improved with respect to the BM with and without outliers. The mean values are close to zero for all orders of magnitude, and the standard deviations are around 15%. Training and testing exclusively on large cross sections renders a very stable prediction of cross sections for all (included) orders of magnitude. Surprisingly, the estimator is improved for small cross sections $\mathcal{O}(10^{-12}$ fb) with

[2]Recall that the variance is added to the diagonal of the training data covariance matrix.

[3]Cross sections with lower values than this predict less than 0.02 events, and are therefore less interesting in this thesis.

the exclusion of cross sections below σ_{cut}, indicating that the functional expression is different for very small cross sections $\sigma < 10^{-16}$ fb.

4.3.4 Features

`Prospino 2.1` calculates the K-factor for the mean of the squark masses, $\bar{m}$. The LO cross sections for non-degenerate squark masses are then multiplied by the K-factor to find the individual NLO cross sections, as discussed in Sec. 2.4.1. Adding the mean squark mass as a feature could therefore improve the GP estimator. In this section a GP estimator is tested using the BM settings with the addition of $\bar{m}$ as a feature.

The optimised kernel hyperparameters are listed in Tables 4.3 – 4.4. For the GP estimator with the mean mass as a feature, raising the upper limit on σ_f^2 did have an effect on the predicted cross sections, and the upper limit was therefore raised. Adding the mean mass as a feature reduces the estimated noise level variance considerably, to $\sigma_n^2 = 1.19 \cdot 10^{-5}$ for $\widetilde{d}_L\widetilde{d}_L$, and $\sigma_n^2 = 6.97 \cdot 10^{-5}$ for $\widetilde{d}_L\widetilde{u}_R$. The noise variances are approximating the variance predicted in Sec. 4.1, which is an indication that the GP estimator is improved.

Further, the GP estimates a shorter length scale for $\bar{m}$ than for $m_{\widetilde{g}}$. This implies that there is a higher dependency on the mean mass than on the gluino mass, since GPs attribute large length scales to irrelevant features. The resulting relative deviance distributions for $\widetilde{d}_L\widetilde{d}_L$ are shown in Fig. 4.6c and compared to the BM. With some exceptions at $\log_{10}\sigma/\sigma_0 \in [2, 4]$, where the variances are very large, adding $\bar{m}$ as a feature gives relative deviance means of almost zero and very small standard deviations for cross sections above the 0.02 event limit.

4.3.5 Kernel

The RBF kernel is intrinsically smooth, while the Matérn kernel has a hyperparameter ν to control its smoothness. It is sometimes argued that this makes Matérn a better kernel for physical processes, as mentioned in Sec. 3.2.2. For this reason the Matérn kernel is tested. In `scikit-learn` the hyperparameter ν is not optimised during training, and must be determined beforehand. In this section an estimator with the BM settings and the Matérn kernel with $\nu = \frac{3}{2}$ is compared to the BM with the RBF kernel, and the resulting relative deviance distributions for $\widetilde{d}_L\widetilde{d}_L$ are found in Fig. 4.6d.

The hyperparameter ν is set to $\frac{3}{2}$ as this is one of the values for which covariances are quick to calculate. For values not in $\nu \in [\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \infty]$ `scikit-learn` evaluates Bessel functions explicitly, which increases the computational cost by up to a factor 10.

For large cross sections, the values predicted by the GP with the Matérn kernel are somewhat more stable than with the RBF kernel, in that the relative

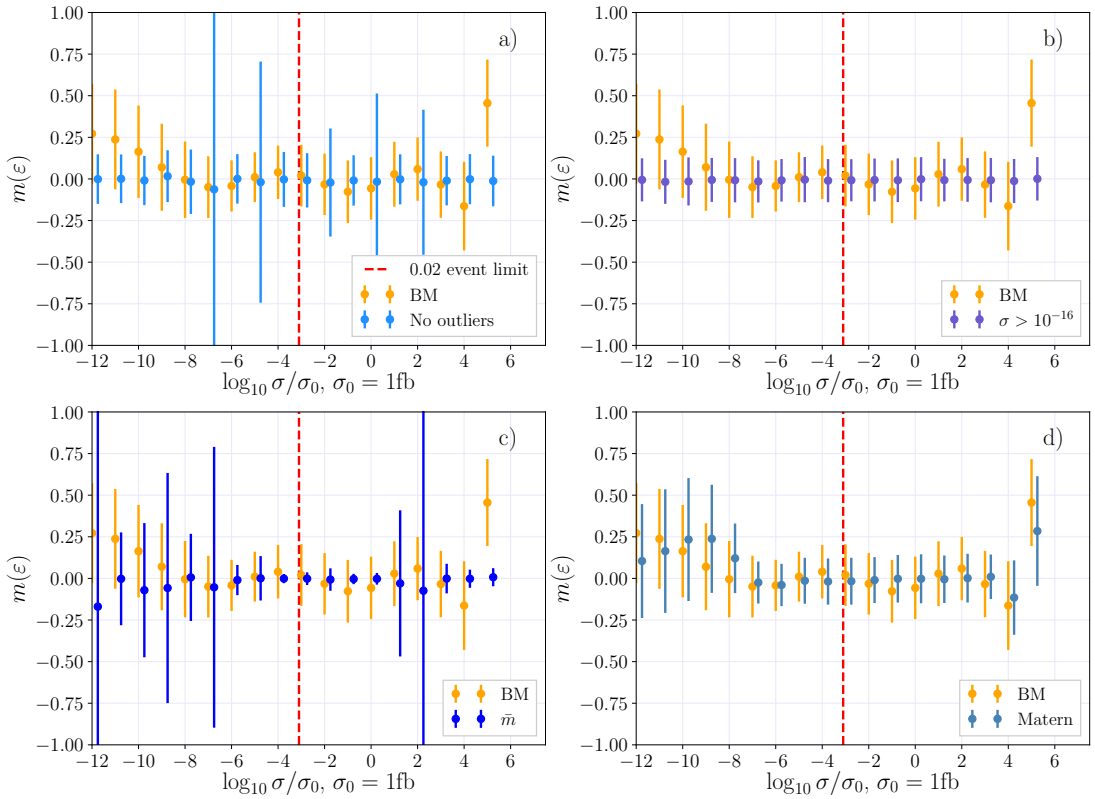


Figure 4.6: The mean and standard deviation of the relative deviance distributions, $m(\varepsilon_i)$ and $\text{std}(\varepsilon_i)$ per decade $i = \log_{10}\sigma/\sigma_0$, for the process $\widetilde{d}_L\widetilde{d}_L$ with **a)** benchmark settings (orange) and removed outliers (blue); **b)** benchmark settings (orange) and a lower cut on cross sections (blue); **c)** benchmark settings (orange) and the added feature $\bar{m}$ (blue); **d)** the benchmark settings (orange) and the Matérn kernel (blue).

deviance means are close to zero. For small cross sections the relative deviance distributions have larger standard deviations. As small cross sections do not comprise the region of interest, the Matérn kernel is considered a better fit than the RBF. As seen from the optimised kernel hyperparameters in Tables 4.3 – 4.4, the Matérn kernel predicts a slightly lower noise level variance than the RBF. The noise variances go from $\sigma_n^2 = 0.47$ with RBF to $\sigma_n^2 = 0.462$ with Matérn for $\widetilde{d}_L\widetilde{d}_L$, and from $\sigma_n^2 = 0.65$ with RBF to $\sigma_n^2 = 0.402$ with Matérn for $\widetilde{d}_L\widetilde{u}_R$.

4.3.6 Optimised Settings

A GP estimator with a combination of the improved settings from the foregoing sections is tested in this section. The optimised settings are

- A GP estimator with 2000 training points.

- 20 000 test points.
- Features $m_{\widetilde{g}}, m_{\widetilde{d}_L}, \bar{m}$ ($m_{\widetilde{g}}, m_{\widetilde{d}_L}, m_{\widetilde{u}_R}, \bar{m}$) for $\widetilde{d}_L\widetilde{d}_L$ ($\widetilde{d}_L\widetilde{u}_R$) production.
- The Matérn kernel with $\nu = 3/2$ and a white noise term

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(1 + \sqrt{3}\big[(\mathbf{x}_i - \mathbf{x}_j)^T M(\mathbf{x}_i - \mathbf{x}_j)\big]^{1/2}\right) \tag{4.14}$$

$$\times \exp\left(\sqrt{3}\big[(\mathbf{x}_i - \mathbf{x}_j)^T M(\mathbf{x}_i - \mathbf{x}_j)\big]^{1/2}\right) + \sigma_n^2\delta_{ij}, \tag{4.15}$$

  where $M = \mathrm{diag}(\boldsymbol{\ell})^{-2}$ and σ_n^2 is the variance of the distribution of white noise $\varepsilon_{WK} \sim \mathcal{N}(0, \sigma_n^2)$. The hyperparameters are $\boldsymbol{\theta} = \{\sigma_f^2, \boldsymbol{\ell}, \sigma_n^2\}$.
- A cut on the cross sections $\sigma > \sigma_{cut} = 10^{-16}$ fb.[4]

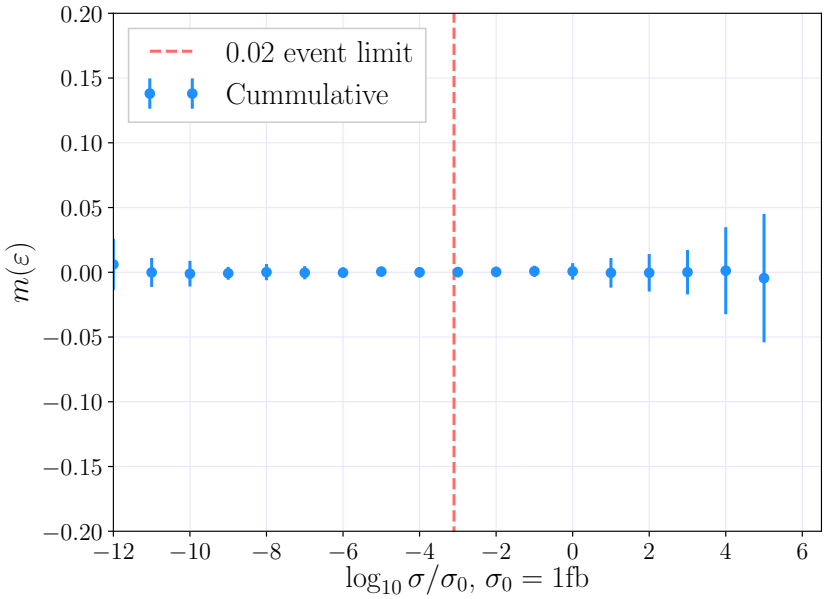
The resulting relative deviance distributions are found in Fig. 4.7. Note that the limits on the y-axis are now $[-0.2, 0.2]$, as opposed to $[-1, 1]$ in Fig. 4.6. With the optimised settings all standard deviations, $\mathrm{std}(\varepsilon_i)$, are less than 5% and all means, $m(\varepsilon_i)$, are close to zero for $\widetilde{d}_L\widetilde{d}_L$. The GP estimator for the process $\widetilde{d}_L\widetilde{u}_R$ performs considerably worse than $\widetilde{d}_L\widetilde{d}_L$, but most standard deviations are within 10%. Training and testing with the optimised settings for 2000 training points and 20 000 test points takes 00:09:30 for $\widetilde{d}_L\widetilde{d}_L$ and 00:10:28 for $\widetilde{d}_L\widetilde{u}_R$ under the same conditions as in Table 4.5.

Noise

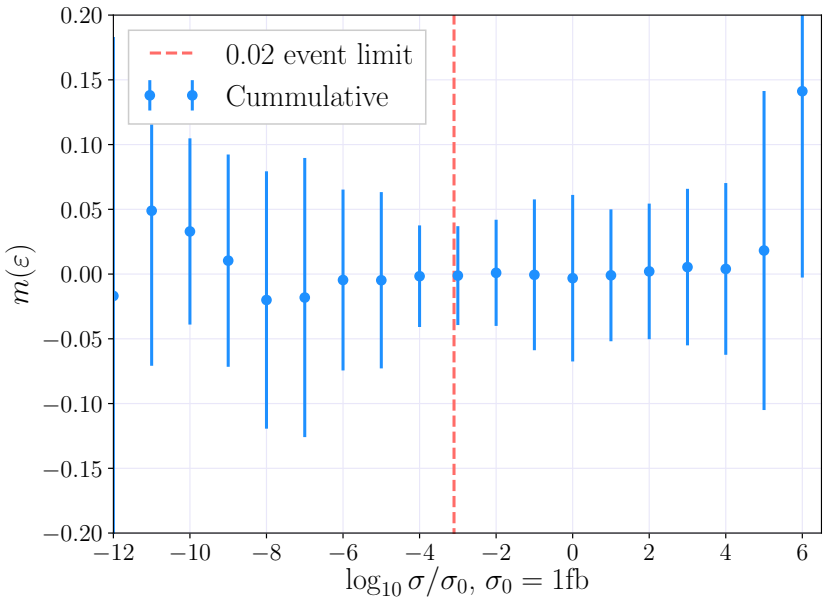
The noise level in the data is known to some degree, as discussed in Sec. 4.1. A more direct approach of explicitly adding noise to the covariance is therefore tested. In `scikit-learn` an option to letting the `WhiteKernel` estimate the level of noise is to specify the noise by passing it to the Gaussian process regressor function. Noise levels estimated by the `WhiteKernel` will be referred to as *free*, and noise levels passed explicitly to the regressor function will be referred to as *fixed*.

When noise levels are free, the GP with the optimised settings estimates noise variances relatively close to the predicted value $\sigma_\varepsilon^2 = 7.5 \cdot 10^{-7}$. The estimated noise variances are $\sigma_n^2 = 1.0 \cdot 10^{-5}$ for $\widetilde{d}_L\widetilde{d}_L$ and $\sigma_n^2 = 5.1 \cdot 10^{-5}$ for $\widetilde{d}_L\widetilde{u}_R$. Apart from the mean mass length scale, $\ell_{\bar{m}}$, that changes from 9500 to 5900 for $\widetilde{d}_L\widetilde{u}_R$, the remaining kernel hyperparameters hardly change when σ_n^2 is fixed at σ_ε^2. The optimised kernel hyperparameters with fixed and free noise levels are listed in Table 4.6, and relative deviance distributions are shown in Fig. 4.8. A marginal improvement for fixed noise levels can be seen in the standard deviations in Fig. 4.8, but the mean values are almost identical.

[4]Effectively also removing the outliers $\sigma = 0 < 10^{-16}$ fb.



(a) The process $\widetilde{d}_L\widetilde{d}_L$



(b) The process $\widetilde{d}_L\widetilde{u}_R$

Figure 4.7: The mean and standard deviation of the relative deviance distributions, $m(\varepsilon_i)$ and $\text{std}(\varepsilon_i)$ per decade $i = \log_{10}\sigma/\sigma_0$ for $\widetilde{d}_L\widetilde{d}_L$, with the optimised settings from Sec. 4.3.6. For $\widetilde{d}_L\widetilde{d}_L$, all orders of magnitude have standard deviations smaller than 5%, and for $\widetilde{d}_L\widetilde{u}_R$ most standard deviations are smaller than 10%.

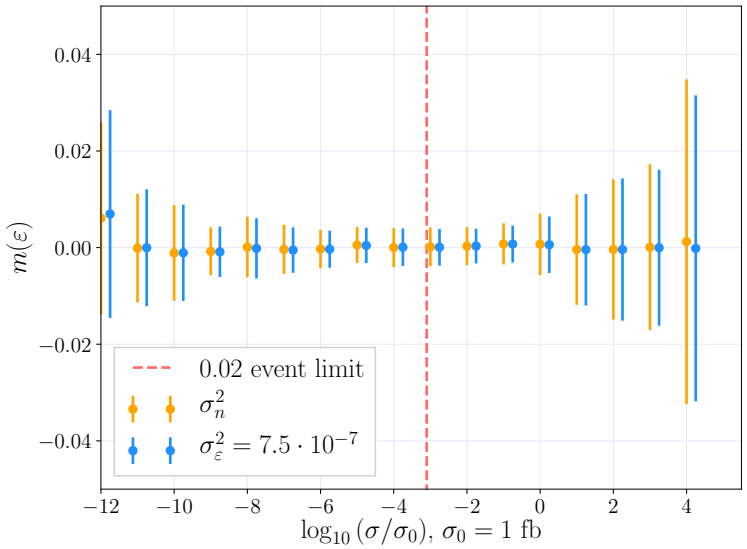


Figure 4.8: The mean and standard deviation of the relative deviation distributions, $m(\varepsilon_i)$ and $\text{std}(\varepsilon_i)$, per decade $i = \log_{10}\sigma/\sigma_0$ for $\widetilde{d}_L\widetilde{d}_L$ with the cummulative settings from Sec. 4.3.6. The noise level σ_n^2 is estimated by the GP (orange), and in given explicitly as $\sigma_\varepsilon^2 = 7.5 \cdot 10^{-7}$ (blue).

For calculations with few training points the computation time is hardly affected by holding σ_n^2 fixed. For larger datasets, however, removing `WhiteKernel` from the total kernel could reduce the computation time, as it leaves one less hyperparameter to optimise. At any rate, since training times have been kept at a reasonable level in this thesis, the `WhiteKernel` with a free σ_n^2 is used for the remainder of the project.

	σ_f	$\ell_{m_{\tilde{g}}}$	$\ell_{m_{\tilde{d}_L}}$	$\ell_{m_{\tilde{u}_R}}$	$\ell_{\bar{m}}$	σ_n^2
$\widetilde{d}_L\widetilde{d}_L$	27.4	30500	17400		74800	$1 \cdot 10^{-5}$
$\widetilde{d}_L\widetilde{d}_L$	31.6	28300	18300		69900	$7.5 \cdot 10^{-7}$ (fixed)
$\widetilde{d}_L\widetilde{u}_R$	31.6	1890	4100	4140	9500	$5.1 \cdot 10^{-5}$
$\widetilde{d}_L\widetilde{u}_R$	31.6	1780	4020	4030	5900	$7.5 \cdot 10^{-7}$ (fixed)

Table 4.6: Optimised kernel parameters for the optimised settings with the noise level estimated by the GP, and given explicitly to the GP estimator $\sigma_\varepsilon^2 = 7.5 \cdot 10^{-7}$.

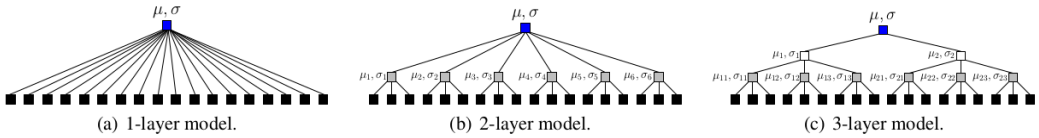


Figure 4.9: Computational graphs of hierarchical product-of-expert models. Main computations are at the leaf nodes (GP experts in black). All other nodes recombine computations from their children nodes. The blue node at the top represents the final prediction. Figure from [6].

4.4 Distributed Gaussian Processes

A significant disadvantage of Gaussian processes is that the computation times scale poorly with the size of the dataset, as discussed in Sec. 3.3. For n training points, training time scales as $\mathcal{O}(n^3)$ and the time for prediction scales as $\mathcal{O}(n^2)$. Consequently, there is a practical limit of $\mathcal{O}(10^4)$ to the size of datasets used to train a GP. The bottleneck of the computation is the inversion of an $n \times n$-matrix, as mentioned in Sec. 3.3. This section is therefore dedicated to block diagonalizing the matrix using distributed Gaussian processes.

In [6] a method of scaling GPs to large datasets is proposed, in the form of a robust Bayesian Committee Machine (rBCM). This method is based on the concepts of product-of-experts and Bayesian Committee Machine introduced below, and has the advantage of providing an uncertainty for the prediction.

4.4.1 Product-of-Experts

Product-of-expert (PoE) models are a way of parallelising large computations. They combine several independent computations on subsets of the total data, called 'experts'. In the case of distributed Gaussian processes each expert performs GP on a subset of the training data, and the predictions on a common test set are combined.

Consider the training dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, partitioned into M subsets $\mathcal{D}^k = \{X^k, \mathbf{y}^k\}$, $k = 1, ..., M$. The feature vectors are D-dimensional, $\mathbf{x} \in \mathbb{R}^D$. Each GP expert trains on its training dataset $\mathcal{D}^k$, then predictions are combined at the parent node. This node could also be considered an expert for a PoE with several layers, see Fig. 4.9.

4.4.2 Bayesian Committee Machine

The Bayesian Comittee Machine (BCM) [33] is a type of PoE where a new weighting scheme is introduced. Consider the training dataset from the previous section partitioned into M subsets, $\mathcal{D}^k = \{X^k, \mathbf{y}^k\}$, $k = 1, ..., M$. For simplicity a single test point, $\mathbf{x}^*$, is considered, where f^* is the unknown target value.

M estimators are trained separately, one for each training dataset. Now let $\mathbf{D}^i = \{\mathcal{D}^1, ..., \mathcal{D}^i\}$ denote the datasets with indices smaller than or equal to i for $i = 1, ..., M$. For the ith estimator the posterior probability distribution for the test point $\mathbf{x}^*$ is $P(f^*|\mathcal{D}^i)$. All prior and posterior distributions are conditioned on the information I, $P(f^*|\mathcal{D}^i, I)$, which is omitted from the expressions for purely aesthetic reasons. Then in general

$$P(f^*|\mathbf{D}^{i-1}, \mathcal{D}^i) \propto P(f^*)P(\mathbf{D}^{i-1}|f^*)P(\mathcal{D}^i|\mathbf{D}^{i-1}, f^*), \tag{4.16}$$

meaning that the posterior distribution for the datasets $\{\mathcal{D}^1, ..., \mathcal{D}^i\}$ is proportional to the posterior for the dataset $\mathcal{D}^i$ given the datasets $\{\mathcal{D}^1, ..., \mathcal{D}^{i-1}\}$, times the posterior of the datasets $\{\mathcal{D}^1, ..., \mathcal{D}^{i-1}\}$, multiplied by the prior $P(f^*)$. Note that this is simply a generalization of the product rule in Eq. (3.2).

Assuming the datasets $\mathbf{D}^{i-1}$ and $\mathcal{D}^i$ are independent — or at least have a very small correlation — given f^*, the following approximation is made

$$P(\mathcal{D}^i|\mathbf{D}^{i-1}, f^*) \approx P(\mathcal{D}^i|f^*). \tag{4.17}$$

Using this approximation and applying Bayes' theorem gives

$$P(f^*|\mathbf{D}^{i-1}, \mathcal{D}^i) \propto \frac{P(f^*|\mathbf{D}^{i-1})P(f^*|\mathcal{D}^i)}{P(f^*)}. \tag{4.18}$$

The approximate posterior distribution using the entire dataset, $\mathcal{D}$, is then

$$\hat{P}(f^*|\mathcal{D}) \propto \frac{\prod_{i=1}^{M} P(f^*|\mathcal{D}^i)}{P(f^*)^{M-1}}, \tag{4.19}$$

where the hat over $\hat{P}$ means that the distribution is approximate. The posterior distributions of each expert are simply multiplied, and divided by the prior $M-1$ times.

4.4.3 Robust Bayesian Comittee Machine

The Bayesian Comittee Machine can be modified to a *robust Bayesian Comittee Machine* (rBCM), which is the scheme that will be used in this thesis for distributed Gaussian processes (DGP). The M experts are assumed to be independent [6] as in Eq. (4.17), effectively block-diagonalizing the covariance matrix. The marginal likelihood from Eq. (3.46) is now factorised into the product of M individual terms because of the independence assumption. For the training data of one dataset $\mathcal{D}^k = \{X^k, \mathbf{y}^k\}$ the log marginal likelihood is given by Eq. (3.46)

$$\log P(\mathbf{y}^k|X^k, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{kT}(K_\psi^k + \sigma_\varepsilon^2\mathbb{I})^{-1}\mathbf{y}^k - \frac{1}{2}\log|K_\psi^k + \sigma_\varepsilon^2\mathbb{I}|, \tag{4.20}$$

where K_ψ^k is the covariance matrix of the training features X^k, with the kernel hyperparameters given by $\boldsymbol{\theta} = \{\boldsymbol{\psi}, \sigma_\varepsilon^2\}$. Computing the LML now entails inverting a $n_k \times n_k$ matrix, $K_\psi^k + \sigma_\varepsilon^2 \mathbb{I}$, which requires time $\mathcal{O}(n_k^3)$ and memory consumption $\mathcal{O}(n_k^2)$. For $n_k \ll N$, this reduces the computation time and memory use considerably, and allows for parallel computing.

The rBCM predicts a function value f^* at a corresponding test input $\mathbf{x}^*$ according to a predictive distribution that is similar to the BCM in Eq. (4.19), with the addition of powers β_k,

$$P(f^*|\mathbf{x}^*, \mathcal{D}) = \frac{\prod_{k=1}^M P_k^{\beta_k}(f^*|\mathbf{x}^*, \mathcal{D}^{(k)})}{P^{-1+\sum_k \beta_k}(f^*|\mathbf{x}^*)}, \tag{4.21}$$

where the powers β_k control the importance of the individual experts, but also how strong the influence of the prior is. In [6] these are chosen according to the predictive power of each expert at $\mathbf{x}^*$. More specifically, β_k is the change in differential entropy between the prior $P(f^*|\mathbf{x}^*)$ and the posterior $P(f^*|\mathbf{x}^*, \mathcal{D}^{(k)})$, which can be calculated as

$$\beta_k = \frac{1}{2}(\log \sigma_{**}^2 - \log \sigma_k^2(\mathbf{x}^*)), \tag{4.22}$$

where $\sigma_{**}^2 = k(\mathbf{x}^*, \mathbf{x}^*)$ is the prior variance of the test point, and $\sigma_k^2(\mathbf{x}^*)$ is the predictive variance of the kth expert given by Eq. 3.44.

The combined predictive mean and variance are denoted μ_{rbcm}^* and σ_{rbcm}^{*2}, and are given by

$$\mu_{rbcm}^* = (\sigma_{rbcm}^*)^2 \sum_k \beta_k \sigma_k^{-2}(\mathbf{x}^*)\mu_k(\mathbf{x}^*), \tag{4.23}$$

$$(\sigma_{rbcm}^*)^{-2} = \sum_{k=1}^M \beta_k \sigma_k^{-2}(\mathbf{x}^*) + \big(1 - \sum_{k=1}^M \beta_k\big)\sigma_{**}^{-2}. \tag{4.24}$$

Implementation

There is currently no functionality for an rBCM in the `scikit-learn` library, so the combined predictive mean and variance in Eqs. (4.23) – (4.24) were implemented in a `Python` routine. The `scikit-learn` library's existing framework for regular Gaussian processes were used for the individual experts. The algorithm was parallelised, so that each expert can learn and predict in parallel, before being combined to the final prediction. The program for training models in parallel is listen in Appendix A.1, and the class used to combine experts is listed in Appendix A.2. The benchmark test of distributed Gaussian processes is found in Appendix C.

For parallelisation the `scikit-learn` function `Parallel` from `joblib` was used, which runs `Python` functions as pipeline jobs. It uses the `Python` function

`multiprocessing` as a backend. An example of usage with 3 parallel jobs is as follows

```
>>> from joblib import Parallel, delayed
>>> from math import sqrt
>>> Parallel(n_jobs=3)(delayed(sqrt)(i**2) for i in range(10))
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0]
```

where `delayed` is a simple trick to be able to create a tuple with a function-call syntax.

4.4.4 Evaluating Cross Sections using Distributed Gaussian Processes

In this section the distributed Gaussian processes (DGP) described in the previous section are applied to estimators with the benchmark settings from Sec. 4.3.1 to scale the problem to larger training sets. The prediction for a single expert with 8000 training points is compared to the combined prediction of four experts with 2000 training points each, both in terms of computation time and estimator quality. Larger training datasets per expert could be used, such as four experts with 8000 training points each, but comparing to a single expert with the same amount of data would be unfeasible because of the scale of the computation, as it would require an expert with $4 \times 8000 = 32000$ training points.

Adding Experts

The addition of experts with 2000 training points improves the BM estimator from Sec. 4.3.1 significantly, with very little increase in computational cost. In Fig. 4.10 a comparison of relative deviance distributions between one expert and four experts with 2000 training points each, and one expert with 8000 training points is shown. For comparison the settings are the benchmark settings, and outliers are included in the dataset. With the addition of data, the accuracy and stability of estimators are very much improved. As seen from the distributions in Fig. 4.10 there is little difference between using four experts with 2000 training points each and using a single expert with 8000 training points. The difference in computation times, however, is very large. Four experts with 2000 points take a little under six minutes to compute, while the single expert with 8000 points takes over an hour and a half. Computation times are listed in Table 4.7.

4.4.5 Cross Validation

Since there is no `scikit-learn` functionality for distributed Gaussian processes, an algorithm for k-fold cross validation as a function of experts had to be implemented, according to the outline in Sec. 3.4.2. In the algorithm the `scikit-learn`

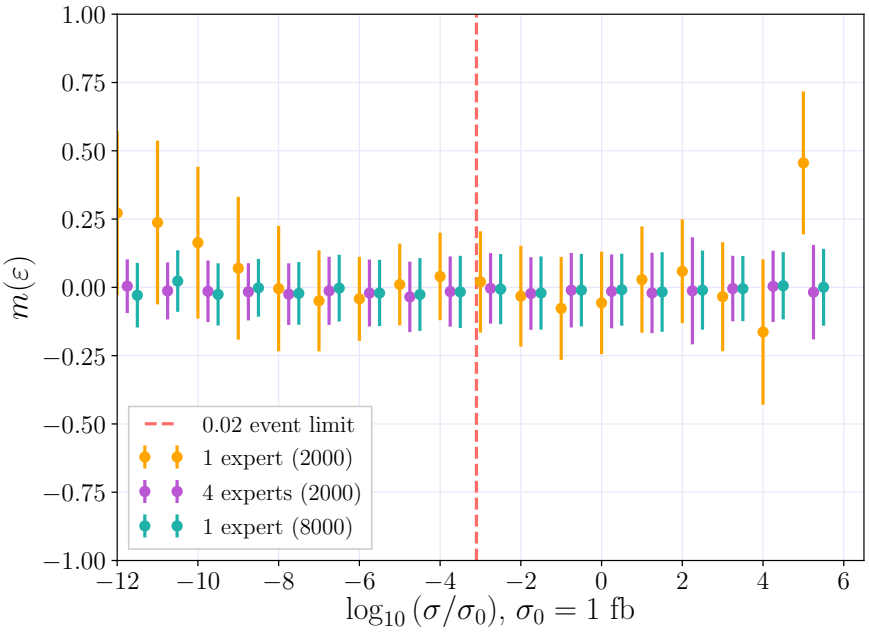


Figure 4.10: The mean and standard deviation of the relative deviance distributions, $m(\varepsilon_i)$ and $\text{std}(\varepsilon_i)$, per decade $i = \log_{10}\sigma/\sigma_0$ for $\widetilde{d}_L\widetilde{d}_L$, using GP with the BM settings, for one (orange) and four (violet) experts with 2000 training points each, and one expert with 8000 training points (green).

Number of experts	Points per expert	Time
1	2000	00:03:32
4	2000	00:05:46
1	8000	01:35:21

Table 4.7: Table of computation times for GP fit and prediction on the Abel HPC cluster.

function `KFold` is used to find training and test indices for k splits of the data. The scoring function used in the CV is the R^2-score introduced in Sec. 3.4.2, and the code is listed in Appendix A.3.

5

Results

In this chapter distributed Gaussian processes (DGP) trained on the MSSM-24 dataset are used to predict cross sections for the MSSM-24 and the CMSSM. The individual GP estimators use the settings from Sec. 4.3.6. Learning curves are plotted as functions of the number of experts and number of points per expert for sample processes. Further, relative deviance distributions are shown for deviances between the DGP predictions and `Prospino 2.1` calculations. Cross sections predicted by the DGP are plotted as functions of squark and gluino masses, and compared to cross sections calculated using `Prospino 2.1` and `NLL-fast 2.1`. Finally, the optimization of the model with respect to predictive capabilities, model size and computation times is discussed.

5.1 Learning Curves

Learning curves as a function of the number of experts for the settings in Sec. 4.3.6 are shown in Fig. 5.1 for $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{d}_L\widetilde{u}_R$. The experts are trained with 500, 1000 and 2000 points each, and learning curves are calculated according to the cross validation method in Sec. 3.4.2.

The training scores for the estimators of $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{d}_L\widetilde{u}_R$ are very close to 1, indicating that neither model is underfitting. The validation curves for both processes converge towards 1, albeit faster and for less training points per expert for $\widetilde{d}_L\widetilde{d}_L$ than for $\widetilde{d}_L\widetilde{u}_R$. This is expected, as $\widetilde{d}_L\widetilde{d}_L$ has less features than $\widetilde{d}_L\widetilde{u}_R$ and performed much better for a single expert with 2000 points, as shown in Fig. 4.7. In both cases the training and validation scores are very high, even for few experts. Adding more data, both in the form of more experts and more points per expert, gives higher validation scores. Including a 'bad'[1] expert sometimes affects the validation score negatively and increases the uncertainty in the score, *e.g.* in Fig. 5.1b going from nine to ten experts with 1000 training points for $\widetilde{d}_L\widetilde{d}_L$, but this is consistent with the uncertainty band shown. Nevertheless,

[1]Bad in the sense that the individual R^2-score is low.

the addition of data generally improves the score. Predictions in this chapter therefore use the largest reasonable[2] models, of ten experts with 5000 and 8000 training points each, depending on whether or not the models need to be stored.

5.2 Comparison with Prospino and NLL-fast

In this section plots of the relative deviance distributions defined in Sec. 3.4.3 are shown for the squark pair-production cross sections from the MSSM-24 and CMSSM datasets. Cross sections predicted by the DGP are compared to cross sections calculated using `Prospino 2.1`, and `NLL-fast 2.1` where this is possible.

All first and second generation squarks are considered; $\widetilde{u}_L$, $\widetilde{d}_L$, $\widetilde{s}_L$, $\widetilde{c}_L$, $\widetilde{u}_R$, $\widetilde{d}_R$, $\widetilde{s}_R$ and $\widetilde{c}_R$. These make up 36 different processes for squark pair production. A separate distributed Gaussian processes estimator (DGP) is trained for each squark process, resulting in $36 \times 10 = 360$ trained experts.

The relative deviance distributions for the MSSM-24 datasets use cross sections predicted by DGPs consisting of ten experts with 8000 training points per expert. For the CMSSM each expert uses 5000 training points, as the models had to be stored between training and prediction. Model size is discussed in Sec. 5.3.

5.2.1 Relative Deviance

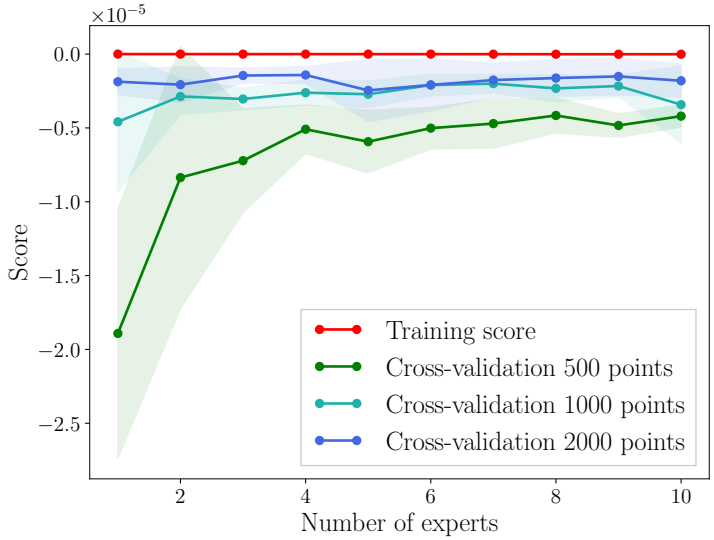
MSSM-24

Distributions of the relative deviance between DGP predicted cross sections and `Prospino 2.1` numbers for 20 000 test points from the MSSM-24 dataset are shown in Fig. 5.2 for selected squark processes. Distributions for the remaining squark processes are found in Appendix D.

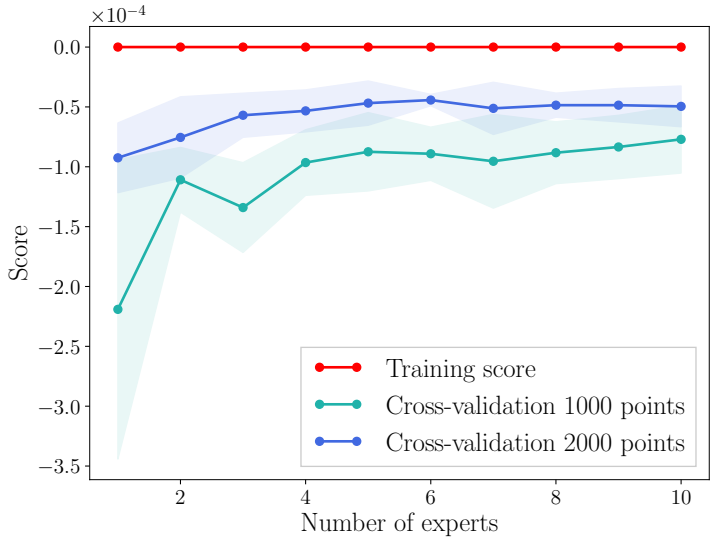
The relative deviance distributions for the DGP prediction of same flavour left-handed squarks are shown in Fig. 5.2a. All distributions have a mean of approximately zero, and a standard deviation well within the desired value of 10%. The largest cross sections above 10^4 fb have been excluded from the plots, as the small number of training and test points here leave distributions with large uncertainties. These squark processes have only three features, which appears to make them easy to learn for the DGP. The corresponding same flavour right-handed processes show similar results.

In Fig. 5.2b relative deviance distributions are shown for the process $\widetilde{u}\widetilde{d}$ for different chirality combinations. The different chirality combinations yield different expressions for the cross section, as shown in Sec. 2.3.1. All squark processes in Fig. 5.2b have four features due to the two different squark masses, which appears to make prediction more difficult for the DGPs. Nevertheless, mean relative

[2]Taking into account computation times and model sizes, given in Sec 5.3.

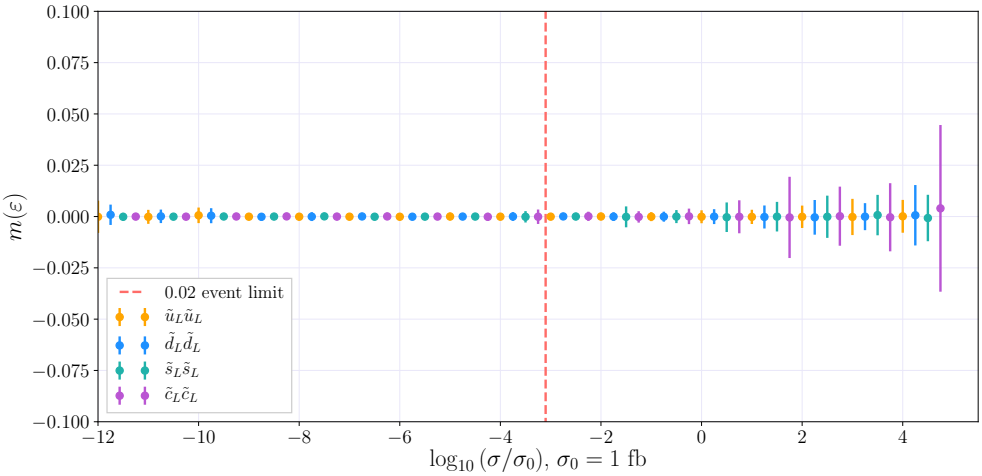


(a) The process $\widetilde{d}_L\widetilde{d}_L$

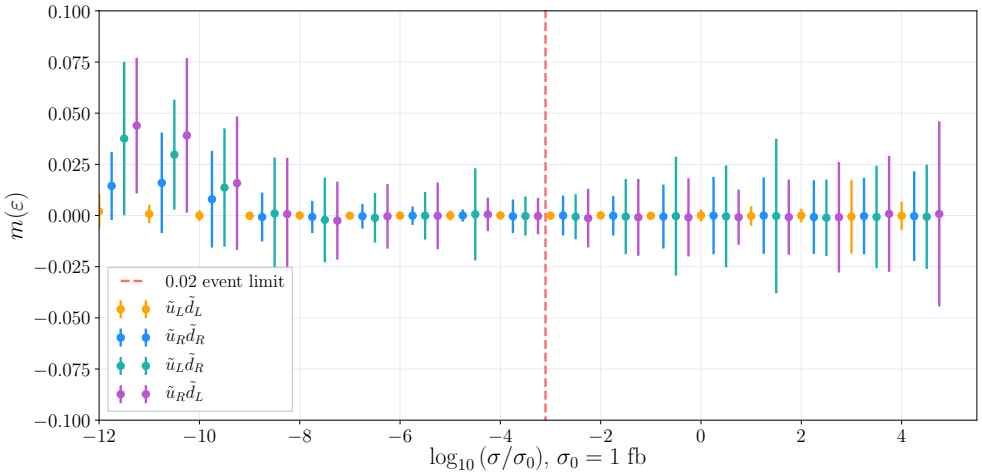


(b) The process $\widetilde{d}_L\widetilde{u}_R$

Figure 5.1: Learning curves as a function of number of experts, with 500, 1000 and 2000 training points per expert for the processes **(a)** $\widetilde{d}_L\widetilde{d}_L$ and **(b)** $\widetilde{d}_L\widetilde{u}_R$. The validation curve for $\widetilde{d}_L\widetilde{u}_R$ with 500 training points per expert is omitted because the uncertainties in the validation scores are very large. The k-fold cross validation uses R^2-score as desribed in Sec. 3.4.2, and here $R^2 - 1$ is plotted. Note that the units on the y-axes are different for **(a)** and **(b)**.



(a) The processes $\widetilde{u}_L\widetilde{u}_L$, $\widetilde{d}_L\widetilde{d}_L$, $\widetilde{s}_L\widetilde{s}_L$ and $\widetilde{c}_L\widetilde{c}_L$.



(b) The processes $\widetilde{u}_L\widetilde{d}_L$, $\widetilde{u}_R\widetilde{d}_R$, $\widetilde{u}_L\widetilde{d}_R$ and $\widetilde{u}_R\widetilde{d}_L$.

Figure 5.2: The mean and standard deviation of the relative deviance distributions, $m(\varepsilon_i)$ and $\text{std}(\varepsilon_i)$, per decade $i = \log_{10}\sigma/\sigma_0$. Ten experts with 8000 training points from the MSSM-24 dataset were used to predict cross sections for 20 000 test points from the MSSM-24 dataset for processes **(a)** $\widetilde{u}_L\widetilde{u}_L$, $\widetilde{d}_L\widetilde{d}_L$, $\widetilde{s}_L\widetilde{s}_L$ and $\widetilde{c}_L\widetilde{c}_L$; and **(b)** $\widetilde{u}_L\widetilde{d}_L$, $\widetilde{u}_R\widetilde{d}_R$, $\widetilde{u}_L\widetilde{d}_R$ and $\widetilde{u}_R\widetilde{d}_L$. The 0.02 event limit is represented by the dotted red line.

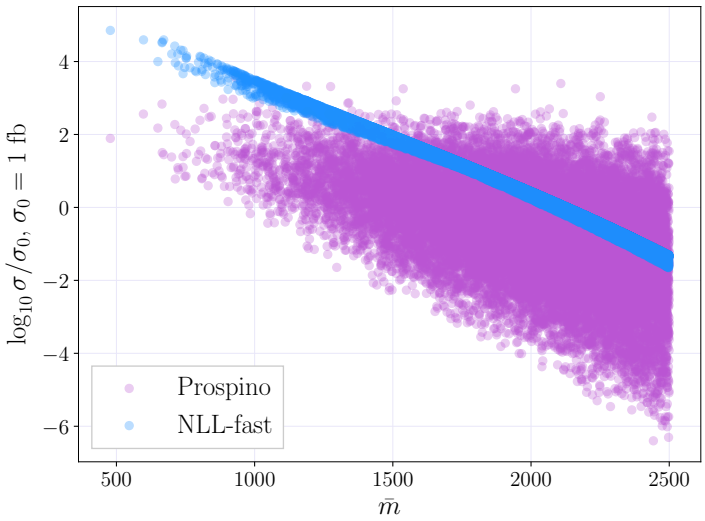


Figure 5.3: Scatter plot of cross sections calculated by `Prospino 2.1` (violet) and `NLL-fast 2.1` (blue) for MSSM-24 data points.

deviances are close to zero, particularly for cross sections above the 0.02 event limit. Standard deviations are larger than for left-handed equal-flavour chirality processes, but within 2.5% for most cross sections larger than $> 10^{-8}$ fb.

The DGPs for $\widetilde{d}_L\widetilde{u}_L$ show superior performance to the other chirality combinations, presumably because the strong correlation between $m_{\widetilde{d}_L}$ and $m_{\widetilde{u}_L}$ discussed in Sec. 4.1 means the DGPs effectively train on three features. The DGPs for $\widetilde{d}_R\widetilde{u}_R$ perform better than for different chiralities, but worse than for $\widetilde{u}_L\widetilde{d}_L$.

The relative deviance between `NLL-fast 2.1` and `Prospino 2.1` is not shown for the MSSM-24 points, because of the former's poor ability to predict squark pair production cross sections. A scatter plot of MSSM-24 cross sections calculated by `Prospino 2.1` and `NLL-fast` is shown in Fig. 5.3. For small mean masses `NLL-fast 2.1` predicts cross sections are too large. This is because the relative mass splitting between right- and left-handed squarks becomes larger for small masses. Because the difference in cross sections is so large — ranging up to four orders of magnitude, as seen in Fig. 5.3 — the relative deviance plots are not shown.

CMSSM

The DGPs trained on the MSSM-24 data are also tested on the CMSSM dataset and compared to cross sections calculated by `NLL-fast 2.1` for the same parameter points. Since `NLL-fast 2.1` can only give the total squark-squark cross section, the true values used are the sums of cross sections calculated by `Prospino 2.1` for all 36 individual processes for each parameter point. These are

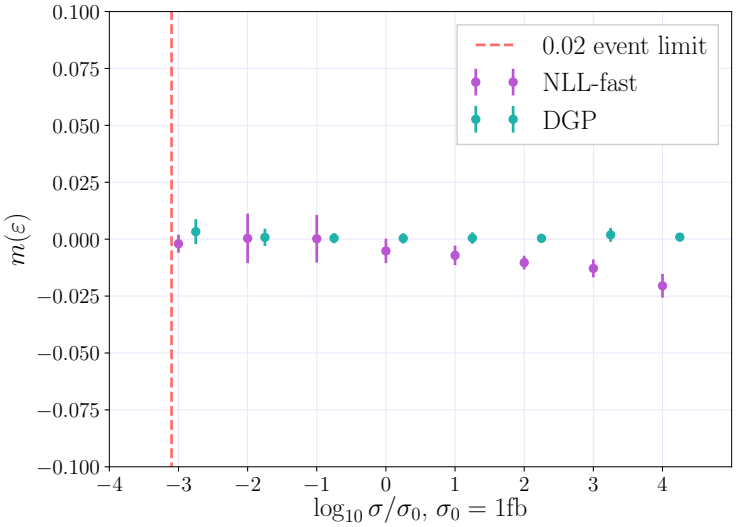


Figure 5.4: The mean and standard deviation of the relative deviance distributions, $m(\varepsilon_i)$ and $\mathrm{std}(\varepsilon_i)$, per decade $\log_{10}\sigma/\sigma_0$, for the total cross section for all squark pair production processes calculated by `NLL-fast 2.1` and predicted by DGPs for CMSSM data. The 'true' values are in both cases the cross sections calculated by `Prospino 2.1`. For the DGP ten experts with 5000 training points each were trained on MSSM-24 data for each process $\widetilde{q}_i\widetilde{q}_j$.

compared to sums over the corresponding 36 cross sections estimated using the DGPs and the `NLL-fast 2.1` numbers. `NLL-fast 2.1`, which calculates cross sections at $\sqrt{s} = 8$ TeV, is limited to the mass ranges $m_{\widetilde{q}} \in [200, 2500]$ GeV and $m_{\widetilde{g}} \in [200, 2500]$ GeV.

The resulting relative deviance distributions are shown in Fig. 5.4. Cross sections calculated by `NLL-fast 2.1` are close to the true values for the CMSSM data. This is because the squark masses in the CMSSM have much smaller splittings than in the MSSM-24, as discussed in Sec. 1.4.8, and `NLL-fast 2.1` assumes degenerate squark masses. For large cross sections `NLL-fast 2.1` predicts values that are too large, as for the MSSM-24 data shown in Fig. 5.3, while the DGPs predict cross sections very close to the true value. `NLL-fast 2.1` overestimates the cross sections for small masses because the hyperfine splitting and the RGE running of different gauge couplings between right- and left-handed squarks make the relative mass splittings larger for small masses.

5.2.2 Cross Sections

In this section the DGPs trained on the MSSM-24 dataset are used to predict cross sections as a function of a single mass, either squark or gluino, with the

remaining masses held fixed. Cross sections for individual processes, such as $\widetilde{d}_L\widetilde{d}_L$, and the total cross section for squark pair production are plotted both as functions of the squark mass and the gluino mass. The predictions are then compared to cross sections calculated by `Prospino 2.1` and `NLL-fast 2.1` for the same masses.

Cross sections from the DGPs and `Prospino 2.1`, and from `NLL-fast 2.1` where these are included, are plotted with the following uncertainties: `Prospino 2.1` gets uncertainties from the scale dependence described in Sec. 2.4.1, DGP uncertainties are the predictive variances in Eq. (4.24), and `NLL-fast 2.1` gets uncertainties from scale dependence, PDFs and α_s. This permits a comparison between the regression error in the DGP and other error sources.

Cross Sections for Two Processes

Figure 5.5a shows the cross sections for $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{d}_L\widetilde{u}_R$ plotted as functions of the squark mass $m_{\widetilde{d}_L}$ for $m_{\widetilde{d}_L} \in [200, 2500]$ GeV. The mass splitting with $\widetilde{u}_L$ is given in Eq. (1.83). All other squark masses are held at 1000 GeV, and the gluino mass is held at 500 GeV. The cross sections predicted by the DGPs have small variances, as can be seen from the 20σ uncertainty bands plotted in Fig. 5.5a. The predictions overlap with `Prospino 2.1` calculations for small $m_{\widetilde{d}_L}$, and are slightly — but systematically — underestimated for large $m_{\widetilde{d}_L}$.

Figure 5.5b shows the cross sections for $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{d}_L\widetilde{u}_R$ plotted as functions of the gluino mass in the range $[200, 2500]$ GeV, while all the squark masses are held at 1000 GeV. The DGP predicted cross sections coincide with the numbers calculated by `Prospino 2.1` for all cross sections. The uncertainty in the predicted cross sections for $\widetilde{d}_L\widetilde{d}_L$ increase with the gluino mass, while the uncertainty for $\widetilde{d}_L\widetilde{u}_R$ is relatively stable as a function of the gluino mass.

Total Cross Section as a Function of the Squark Mass

Figure 5.6a shows the total cross section for squark pair production as a function of $m_{\widetilde{d}_L}$, for the same mass values as in Fig. 5.5a. The cross sections predicted by the DGPs again coincide with the true values for small $m_{\widetilde{d}_L}$, and the underestimation for large masses from Fig. 5.5a is repeated here. In comparison, the `NLL-fast 2.1` cross sections are far from the values calculated by `Prospino 2.1` for all values of $m_{\widetilde{d}_L}$, save one. For $m_{\widetilde{d}_L} = 1000$ GeV, where the squark masses are in fact degenerate, the calculations from `NLL-fast 2.1` and `Prospino 2.1` coincide.

Total Cross Section as a Function of the Gluino Mass

Figure 5.6b shows the total cross section for squark pair production as a function of the gluino mass, for $m_{\widetilde{g}} \in [200, 2400]$ GeV. All squark masses are held at

1000 GeV. Cross sections predicted by the DGPs and calculated by `Prospino 2.1` and `NLL-fast 2.1` coincide for all values of the gluino mass. It can be expected that `NLL-fast 2.1` and `Prospino 2.1` coincide when the squark masses are equal, as `NLL-fast 2.1` assumes degenerate squark masses. The uncertainties in the predictions from the DGPs are fairly small, but increase with the gluino mass.

5.3 Optimizing the Model

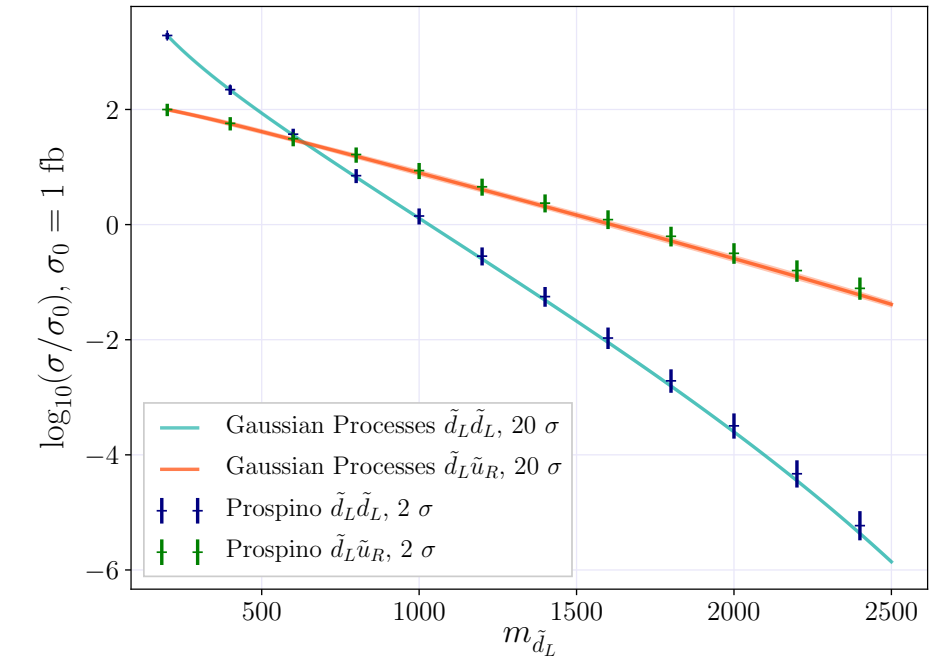
The distributed Gaussian process predictions with ten experts using 8000 training points each are very accurate. Unfortunately, the Gaussian process models require a lot of space when stored. The GP models are stored using the `Python` library `joblib`. In addition, larger models take longer to predict values, and prediction speed is the main motivation for using DGPs. In this section model sizes and computation times are addressed, in order to find a compromise between prediction quality, size and computation time.

Number of Models

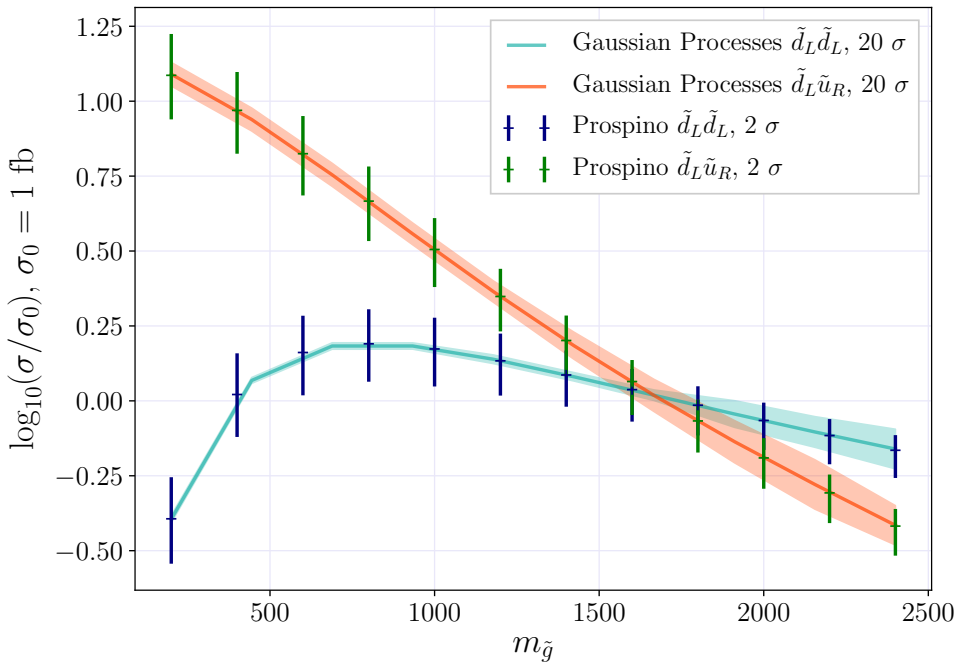
In this thesis 36 Gaussian process models were trained with ten experts each, resulting in a model of in total 360 experts. However, the effective number of experts may be reduced as the cross sections of some processes are equivalent to leading order.

Cross sections for processes where the squarks have the same flavour and same chirality (SFSC) are to leading order identical to the processes with opposite chirality. For example $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{d}_R\widetilde{d}_R$ are identical as functions of the squark mass, as shown in Sec. 2.3.1. Furthermore, `Prospino 2.1` calculates cross sections for strong interactions, and the NLO terms only contain QCD corrections, as discussed in Sec. 2.4. Since there is no electroweak correction, there is no distinction between left- and right-handed squarks in the calculation. They also share the same underlying pdf from the quark, *e.g.* the pdf's used for calculating $\widetilde{d}_R\widetilde{d}_R$ and $\widetilde{d}_L\widetilde{d}_L$ is the one for the d-flavour squark. The SFSC processes for the same flavour, *e.g.* $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{d}_R\widetilde{d}_R$, are therefore very similar, and one can in fact be used to predict values for the other as a function of the pertaining squark mass, *i.e.* $m_{\widetilde{d}_L}$ and $m_{\widetilde{d}_R}$. Since there are eight SFSC processes, the total number of models can be reduced from 36 to 32.

Further, the masses of different generation left-handed squarks are independent. Following the same reasoning as for SFSC processes, the process $\widetilde{u}_L\widetilde{s}_L$ is, to leading order, identical to $\widetilde{u}_R\widetilde{s}_R$ as a function of $m_{\widetilde{u}_L}$ and $m_{\widetilde{s}_L}$, and $m_{\widetilde{u}_R}$ and $m_{\widetilde{s}_R}$, respectively. There are eight processes with left-handed squarks of different generations, so the number of models can therefore reduced further, from 32 to 28.

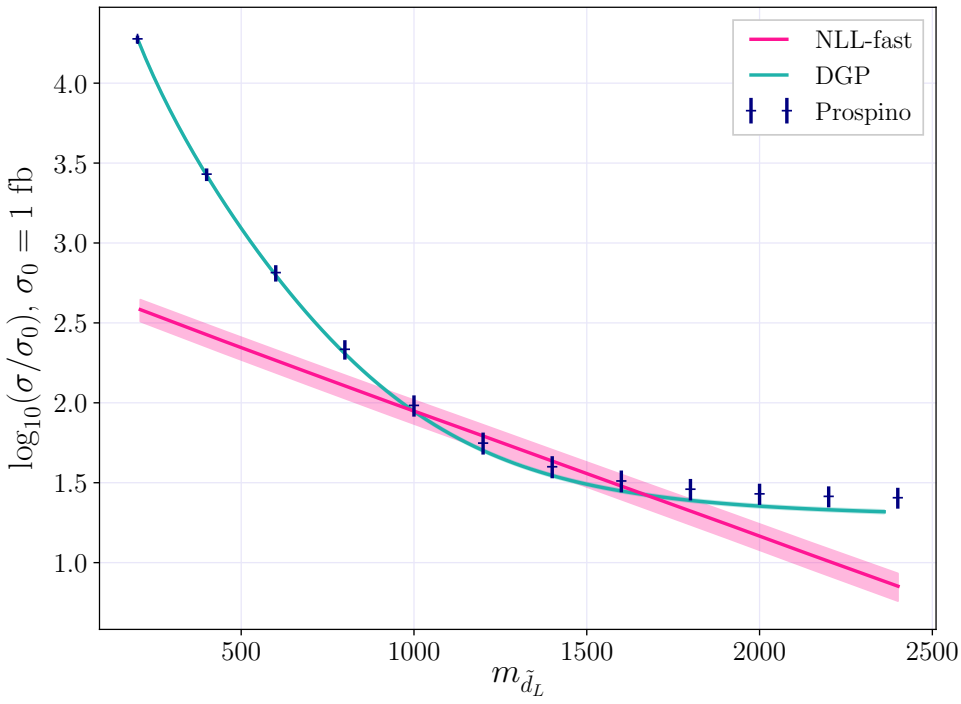


(a) $m_{\tilde{d}_L} \in [200, 2400]$ GeV.

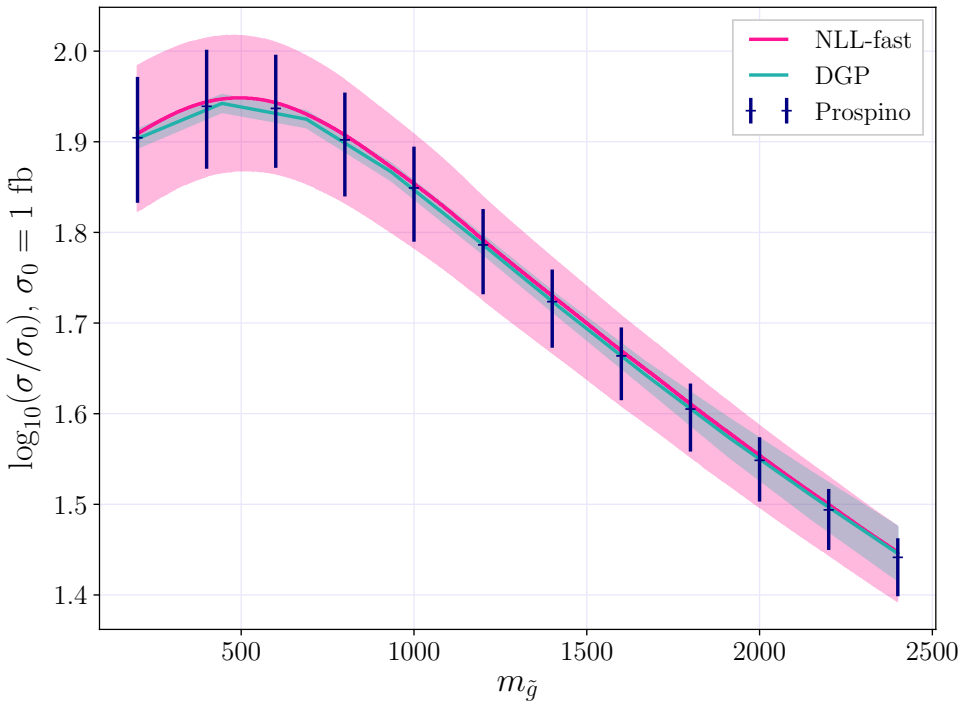


(b) $m_{\tilde{g}} \in [200, 2400]$ GeV.

Figure 5.5: Cross sections for $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{u}_R\widetilde{d}_L$ production, generated by `Prospino 2.1` (points with error bars) and predicted by the DGP (lines with error bands). Cross sections are calculated for **(a)** $m_{\widetilde{d}_L}, m_{\widetilde{u}_L} \in [200, 2400]$ GeV and all other squark masses held at 1000 GeV and $m_{\tilde{g}} = 500$ GeV, and **(b)** $m_{\tilde{g}} \in [200, 2400]$ GeV with all squark masses held at 1000 GeV. Note that the units on the y-axes are different in **(a)** and **(b)**.



(a) $m_{\tilde{d}_L}, m_{\tilde{u}_R} \in [200, 2400]$.



(b) $m_{\tilde{g}} \in [200, 2400]$.

Figure 5.6: Total cross sections for all 36 processes calculated by `Prospino 2.1` and `NLL-fast 2.1`, and estimated by Gaussian processes. In **(a)** The masses $m_{\tilde{d}_L}$ and $m_{\tilde{u}_L}$ are varied from $[200, 2400]$ GeV while all other squark masses are held at 1000 GeV, and $m_{\tilde{g}} = 500$ GeV, and **(b)** the mass m_g is varied $[200, 2400]$ GeV while all squark masses are held at 1000 GeV. All errors shown 1σ. Note that the units on the y-axes are different in **(a)** and **(b)**.

Processes with different flavours and different chiralities are also identical to leading order, such as $\widetilde{d}_L\widetilde{u}_R$ and $\widetilde{u}_L\widetilde{d}_R$ as functions of $m_{\widetilde{d}_L}$ and $m_{\widetilde{d}_R}$, respectively. As there are 12 such processes, six can be removed. The number of models is then reduced from 28 to 22.

The smallest possible set of model then consists of 22 separate DGPs, one for each of the following processes

				$\widetilde{u}_L\widetilde{d}_R$
$\widetilde{u}_L\widetilde{u}_L$	$\widetilde{u}_L\widetilde{u}_R$	$\widetilde{u}_L\widetilde{d}_L$	$\widetilde{u}_L\widetilde{s}_L$	$\widetilde{u}_L\widetilde{s}_R$
$\widetilde{d}_L\widetilde{d}_L$	$\widetilde{d}_L\widetilde{d}_R$	$\widetilde{u}_R\widetilde{d}_R$	$\widetilde{u}_L\widetilde{c}_L$	$\widetilde{u}_L\widetilde{c}_R$
$\widetilde{s}_L\widetilde{s}_L$	$\widetilde{s}_L\widetilde{s}_R$	$\widetilde{s}_L\widetilde{c}_L$	$\widetilde{d}_L\widetilde{s}_L$	$\widetilde{d}_L\widetilde{s}_R$
$\widetilde{c}_L\widetilde{c}_L$	$\widetilde{c}_L\widetilde{c}_R$	$\widetilde{s}_R\widetilde{c}_R$	$\widetilde{d}_L\widetilde{c}_L$	$\widetilde{d}_L\widetilde{c}_R$
				$\widetilde{s}_L\widetilde{c}_R$,

where the models for the processes in the first, second, fourth and fifth columns can be used on the remaining processes with $R \leftrightarrow L$.

It is worth stressing that the processes described in this section are *not identical* — the leading order cross sections are indeed identical, and no included corrections differ between left- and right-handed squarks, but the next-to-leading order cross sections depend on the mean mass. The cross sections have different mean mass dependencies, because of the correlation between same-generation left-handed squarks. For example, $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{d}_R\widetilde{d}_R$ are similar functions of $m_{\widetilde{d}_L}$ and $m_{\widetilde{d}_R}$, but for $\widetilde{d}_L\widetilde{d}_L$ the masses $m_{\widetilde{d}_L}$ and $m_{\widetilde{u}_L}$ are correlated, and the masses $m_{\widetilde{d}_R}$ and $m_{\widetilde{u}_R}$ are not. This is, however, a relatively small effect. The GP model for $\widetilde{d}_L\widetilde{u}_R$ was used to predict cross sections for $\widetilde{u}_L\widetilde{d}_R$, and the relative deviances were around or smaller than 2%. The effective number of models described above can therefore be used to a good approximation.

Model Size

Storing the distributed Gaussian processes models require a lot of memory. The GP experts are stored using the `Python 2.1` library `joblib`. The functions `joblib.dump()` and `joblib.load()` provide a replacement for pickle to work efficiently on `Python` objects containing large data. The `joblib.dump()` command takes an argument `compress`, which can be an integer between 0 and 9, a boolean or a 2-tuple. For the integers, 0 means no compression, and a higher value means more compression but slower read and write times. If `True` is passed, the integer is set to 3, which is often a good compromise. In Table 5.1 the compressed and uncompressed model sizes for a single expert are shown for various numbers of points per expert.

A compressed single Gaussian process model trained with 2000 training points takes up 16 MB. Seeing as one Gaussian process is needed per process, a relatively

Points per expert	Size (uncompressed) [MB]	Size (compressed) [MB]
500	2.0	0.93
1000	7.7	3.8
2000	31.0	16.0
3000	69.0	34.0
5000	191.0	94.0

Table 5.1: Size of stored models when no compression is done, and when the models are compressed using `compress=3`. Models are stored using the `joblib` function `joblib.dump()`.

small model consisting of a single expert with 2000 training points per process will require

$$36 \times 16 \text{ MB} = 576 \text{ MB}.$$

As seen from Table 5.1 the model size scales approximately as $\mathcal{O}(n^2)$ for a model with n training points. There is very little difference in model sizes for models with three and four features, and a negligible difference between using the RBF kernel and the Matérn kernel.

The main factors that affect the model size are therefore the number of experts and the size of each expert. For M experts with n training points each, the model size scales as $\mathcal{O}(M \cdot n^2)$. Where it is possible with respect to precision it is therefore advisable to add *more* experts as opposed to *larger* experts. This is further illustrated in Fig. 5.8, where lines of constant model size are shown in the plane of number of experts vs points per expert. As seen from the lines showing constant model size, the model size increases rapidly with the number of points per expert. For example, two experts with 3500 training points each require approximately the same storage space as ten experts with 1500 training points.

The model size places limits on the number of training points an optimised model should have. A model for the 36 processes with ten experts per process, where each expert is trained with 5000 points and compressed, would require

$$10 \times 36 \times 94 \text{ MB} \approx 33 \text{ GB}.$$

By comparison, reducing the expert sizes to 3000 training points each would require

$$10 \times 36 \times 34 \text{ MB} \approx 12 \text{ GB}.$$

Comparisons of the relative deviance distributions with 3000, 5000 and 8000 training points per expert for the processes $\widetilde{d}_L\widetilde{d}_L$ and $\widetilde{d}_L\widetilde{u}_R$ are shown in Fig. 5.7.

Points per expert	Prediction time (uncompressed)	Prediction time (compressed)
500	00:00:24	00:00:25
1000	00:00:43	00:00:43
2000	00:02:22	00:02:22
3000	00:03:58	00:04:05
5000	00:10:24	00:10:30

Table 5.2: Computation time on an intel i5 CPU to evaluate 10 000 cross sections for a single process, for the uncompressed and compressed models using `compress=3`. Models are stored using the `joblib` function `joblib.dump()`.

The improvement in the prediction for $\widetilde{d}_L\widetilde{u}_R$ from 3000 to 8000 training points per expert is fairly large for all σ, and the goal of standard deviations below 10% is not compromised anywhere. Processes with different squark flavours and chiralities should therefore use experts with larger number of training points, as the main contribution to the prediction error will come from these processes.

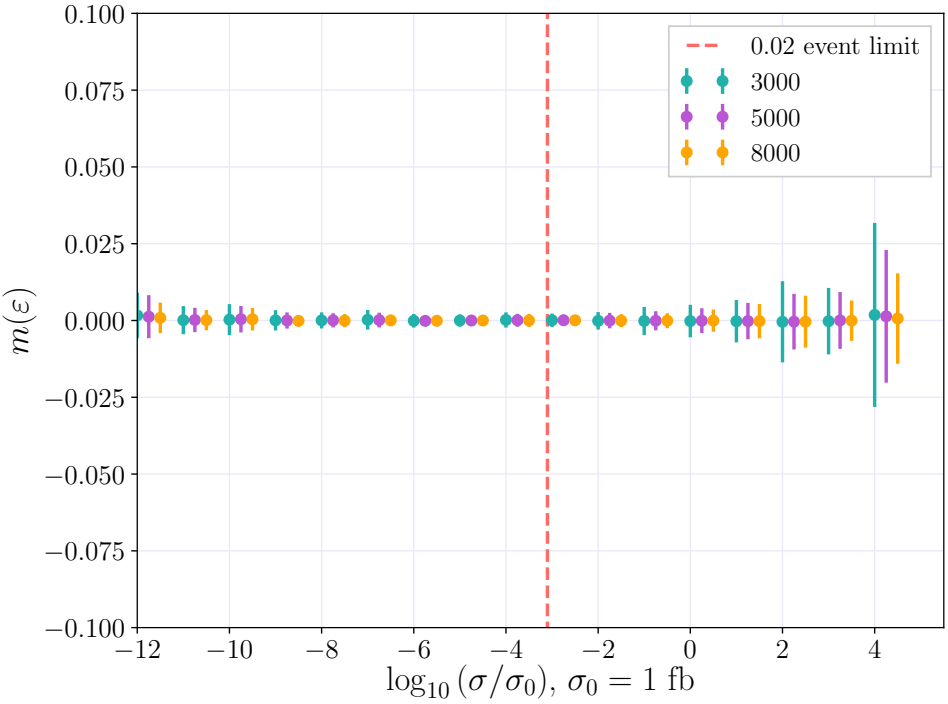
In contrast, the process $\widetilde{d}_L\widetilde{d}_L$ is modelled very accurately even for 3000 training points per expert. For cross sections larger than 10^2 fb there is a visible improvement from 3000 to 8000 training points per expert, likely rooted in the addition of data in this sparse region, in the form of larger experts. For 3000 training points per expert this could potentially be remedied by adding more experts trained on large cross sections. In conclusion, processes with squarks of equal flavour and chirality can use smaller experts, while processes with squarks of different flavour or different chirality, or both, benefit from larger experts.

Computation Times

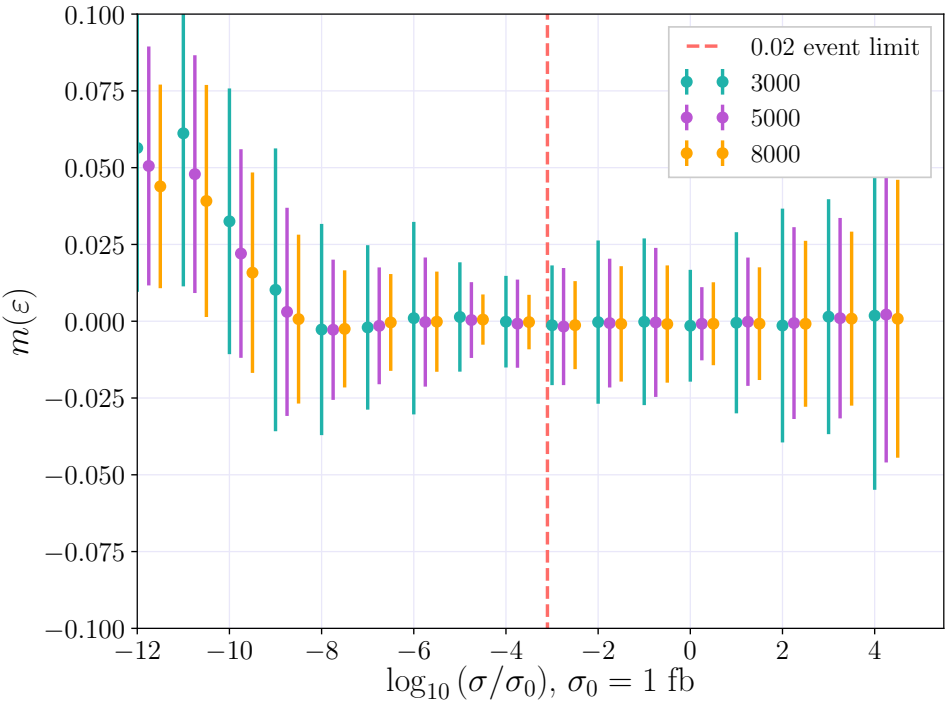
Table 5.2 shows the computation times for evaluating 10 000 cross sections for a single process on an intel i5 CPU, using the uncompressed and compressed models discussed in the previous section. The difference in computation times is very small between the compressed and uncompressed models, so the compressed models are used for further testing.

Figure 5.8 shows lines of constant time in the space of number of experts versus points per expert. The prediction time increases rapidly with the number of points per expert. For example, ten experts with 1000 training points each need as much time as two experts with 3000 training points each. In theory, the prediction time scales as $\mathcal{O}(n^2)$ where n is the number of training points. From Table 5.2, however, the scaling appears to be closer to $\mathcal{O}(1.1n)$. This could be because of the implementation of DGPs, or because loading the models takes time.

Training ten experts with 5000 training points each, in parallel, takes approx-



(a) The process $\widetilde{d}_L\widetilde{d}_L$.



(b) The process $\widetilde{d}_L\widetilde{u}_R$.

Figure 5.7: The mean and standard deviation of the relative deviance distributions, $m(\varepsilon_i)$ and $\text{std}(\varepsilon_i)$, per decade $i = \log_{10}\sigma/\sigma_0$. Ten experts with 3000 (green), 5000 (violet) and 8000 (orange) training points from the MSSM-24 dataset each were tested on 20 000 test points from the MSSM-24 dataset for processes **(a)** $\widetilde{d}_L\widetilde{d}_L$ and **(b)** $\widetilde{d}_L\widetilde{u}_R$.

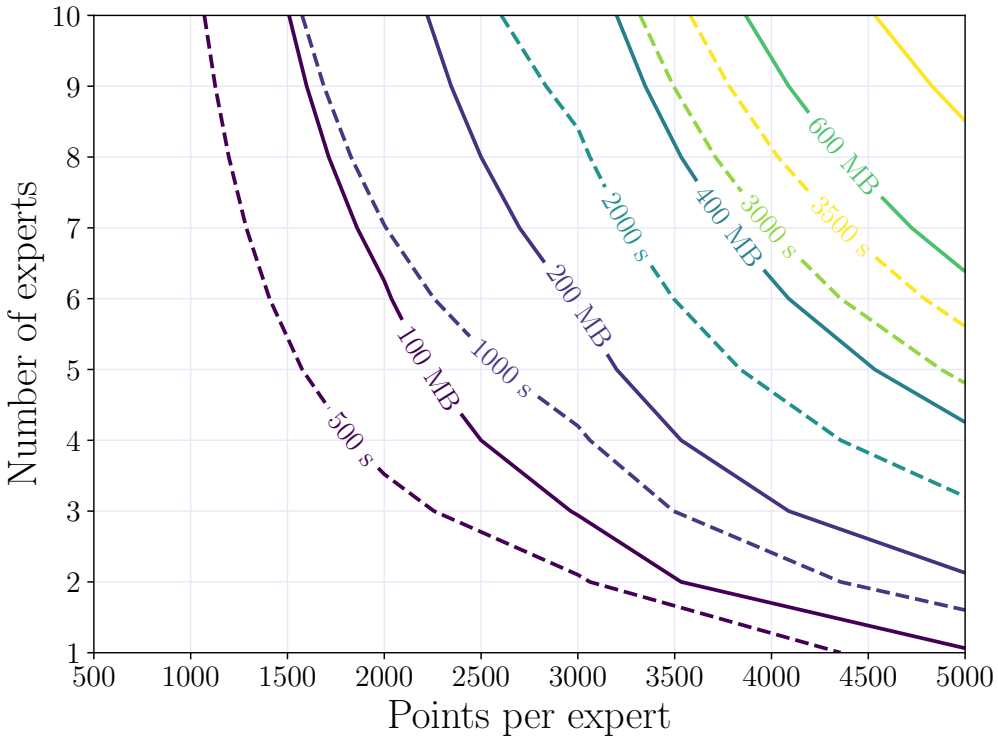


Figure 5.8: Contours showing constant computation time in seconds (dashed lines) and constant compressed model size in MB (solid lines) in the space spanned by number of experts and points per expert. The times are listed in Table 5.2 and compressed model sizes are listed in Table 5.1. To find the computation times and model sizes for more experts than one, the numbers in Tables 5.2 – 5.1 were multiplied by the number of experts.

	Computation time [s]
Prospino	570
NLL-fast	0.00739
Distributed Gaussian Processes	16.9

Table 5.3: Computation times for 1 parameter point for all 36 squark pair production processes.

imately 40 minutes for a single process on the Abel HPC cluster. The prediction time for the DGP was calculated by letting the model predict values for 2000 test points for a single process on the Abel HPC cluster, and dividing the total computation time by 2000. The average computation time for predicting the cross section for a single process at a single parameter point is

$$0.469 \text{ s}.$$

The computation time for predicting all 36 cross sections in series at a single parameter point is shown in Table 5.3, along with the corresponding computation times for `Prospino 2.1` and `NLL-fast 2.1`. `NLL-fast 2.1` is considerably faster than both DGP and `Prospino 2.1`, but has a very large error relative to the `Prospino 2.1` computation, as seen in Fig. 5.6a. Although it is much slower than `NLL-fast 2.1`, the DGP is faster than `Prospino 2.1` by a factor of approximately 34.

The predictions of each DGP expert and each process are *not* done in parallel. Parallelising the prediction algorithm could reduce the computational time in Table 5.3 by a factor of approximately 10×36 for ten experts per process, giving a total time of

$$16.9 \text{ s} : 360 = 0.0469 \text{ s}.$$

Note that these are idealised times, meant to illustrate how the current algorithm can be improved. In theory, the prediction time of Gaussian processes goes as $\mathcal{O}(n^2)$ for n training points, so using smaller experts would also reduce the computation time.

Conclusions

This thesis explored the use of Gaussian processes on next-to-leading order supersymmetric cross sections. More specifically, Gaussian process regression was used to predict cross sections for squark pair production in quark-quark collisions at hadron colliders. Several aspects of the Gaussian process training were investigated.

The algorithm proved sensitive to outliers in the dataset with cross sections calculated by `Prospino 2.1`. The outliers came from `Prospino 2.1` setting next-to-leading order cross sections to zero when the mean squark mass was above threshold. For individual masses *below* the threshold, with nonzero leading order cross sections, this meant setting the next-to-leading order cross section to zero, which created illogical points — or outliers.

The spread in cross sections as a function of the gluino mass made it difficult for the Gaussian processes to recognise the underlying function. The next-to-leading order cross section can be written in terms of scaling functions. For squark pair production in the threshold region these scaling functions are proportional to the square of the gluino mass. Dividing the cross sections by the square of the gluino mass reduces the spread, and this quantity was therefore used as the target value.

Having removed some of the problematic spread, the steepness of the target function for large squark masses was addressed. Large squark masses yield low cross sections, considered less important to this thesis. A lower cut on cross sections was therefore used for the training and test data, which made the function easier to learn for the Gaussian processes.

Given the expression for the leading order partonic cross section in Eq. (2.18) and the next-to-leading order calculation in `Prospino 2.1` — which calculates the K-factor for the mean squark mass and multiplies it by the individual leading order cross sections — the features used were the gluino mass, the final state squark masses, and the mean squark mass. The kernel used to calculate the covariance matrix was the Matérn kernel. In addition, a Gaussian distributed noise level was added to the covariance function, where the noise level variance was estimated by the Gaussian process.

Gaussian processes scale poorly with the size of the training dataset. As a consequence, distributed Gaussian processes in the form of a robust Bayesian

committee machine were used. The robust Bayesian Committee Machine effectively block diagonalises the covariance matrix of the Gaussian process, and trains one Gaussian process expert on each block of the matrix. Distributing the training set among experts allowed for the use of larger training datasets, which improved the Gaussian processes considerably.

A model with distributed Gaussian processes was trained for each of the 36 squark pair production processes with MSSM-24 data for the two first squark generations. The relative deviance distributions for all processes in Fig. 5.2 and Appendix D show that the predicted cross sections have standard deviations well within the desired 10% discussed in Sec. 2.4.1, and in most cases within 5%. Processes with equal flavour squarks with the same chirality proved easier to learn for the Gaussian processes, possibly because they require one less feature than the other processes.

The distributed Gaussian processes trained on the MSSM-24 data were also used to predict cross sections for the CMSSM data. Although the difference is small, the relative deviances of the Gaussian process predicted cross sections relative to the `Prospino 2.1` cross sections are smaller than the `NLL-fast 2.1` relative to `Prospino 2.1`, as shown in Fig. 5.4.

The distributed Gaussian process models are large. The model used in this thesis, of ten experts with 5000 training points each for all squark pair processes is 33 GB large. Since models for some of the processes are almost equivalent, the size of the model can be reduced and the full model of 36 distributed Gaussian processes is not strictly necessary. Additionally, processes with equal flavour squarks with the same chirality require less experts than the other processes.

The computation time for predicting cross sections using Gaussian processes is shorter than for `Prospino 2.1`, but much longer than `NLL-fast`. This time can be reduced by parallelising the prediction for each process and for each expert.

As further work with Gaussian processes, models for other supersymmetric strong processes could be developed. The target values will then have to be reevaluated. The proportionality to the gluino mass is unique for the squark scaling functions, so a simplification of target values may not be possible for other processes. Difficulties can also arise for processes that require large feature vectors.

Appendices

Appendix A

Code Listings

A.1 Distributed Gaussian Process Training

```
"""
Program that trains experts in parallel.
The program takes input arguments as command line
arguments:

[training size] [number of experts] [number of features]
    [features] [target]

@author : Ingrid A. V. Holm
"""

import sys
import numpy as np
import sklearn as sk
from sklearn.gaussian_process.kernels import RBF,
    WhiteKernel, Matern, ConstantKernel as C
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.cross_validation import train_test_split
import pandas as pd

from sklearn.externals import joblib
from sklearn.externals.joblib import Parallel, delayed

# Set training size
if len(sys.argv) >= 2:
    trainsize = float(sys.argv[1])
    if trainsize >= 1:
        trainsize = int(trainsize)
else:
    trainsize = 0.001

# Choose number of experts
```

```
if len(sys.argv) >= 3:
    n_experts = int(sys.argv[2])
else:
    n_experts = 1

if len(sys.argv) >= 4:
    if int(sys.argv[3]) == 2:
        feature_list = [sys.argv[4], sys.argv[5]]
        target_list = sys.argv[6]
    elif int(sys.argv[3]) == 3:
        feature_list = [sys.argv[4], sys.argv[5], sys.argv[6]]
        target_list = sys.argv[9]
else:
    print "Error! No feature list was provided!"
    sys.exit(1)


##########################################################################
# Read in and treat data (mostly) using pandas.                          #
##########################################################################

# Read in data (last column is BS)
data_lin = pd.read_csv('data_290k.dat', sep=' ',
   skipinitialspace=True)

#Create data frames in pandas
df_lin = pd.DataFrame(data_lin)

#Drop BS column
df_lin = df_lin.drop('Unnamed: 46', axis=1)

# Remove zero-cross sections
mask1 = df_lin[target_list[0]] != 0
df_lin = df_lin[mask1]

# Lower limit
mask_lower = df_lin[target_list] > 1e-16
df_lin = df_lin[mask_lower]

mean_list = ["39.mcL", "40.mdL", "41.mdR", "42.muL",
   "43.msL", "44.muR", "45.msR", "46.mcR"]
means = df_lin[mean_list].mean(axis=1).values.ravel()

# Define features to train on and target nubers (used below)
# Convert to array for scikit using values
# ravel is used to avoid [[n]], i.e. n x 1 arrays
features = df_lin[feature_list].values
target = df_lin[target_list].values.ravel()
target_mg = target/features[:,0]**2
```

```
long_features = np.zeros(( len(features), len(features[0])+1
    ))
long_features[:,0] = features[:,0]
long_features[:,1] = features[:,1]

if len(features[0]) == 2:
    long_features[:,2] = means[:,]
    kernel_matern1 = C(10, (1e-3, 1e3)) *
        Matern(np.array([10, 10, 10]), (1, 1e6), nu=1.5)+
        WhiteKernel(0.0001, (1e-8,1e-2))

elif len(features[0]) == 3:
    long_features[:,2] = features[:,2]
    long_features[:,3] = means[:,]
    kernel_matern1 = C(10, (1e-3, 1e3)) *
        Matern(np.array([10, 10, 10, 10]), (1, 1e6), nu=1.5)+
        WhiteKernel(0.0001, (1e-8,1e-2))

# Covert to the logarithm of sigma_mg
target_mg = np.log10(target_mg)

# Split data into training and test set
X_train, X_test, y_train, y_test =
    train_test_split(long_features, target_m2,
    random_state=42, train_size = trainsize)

# Define function to train GP experts
def train_experts(X_train, y_train, train_start, train_end,
    i):
    # Choose subset to train on
    X_train_1 = X_train[train_start:train_end]
    y_train_1 = y_train[train_start:train_end]

    gp1 = GaussianProcessRegressor(kernel=kernel_matern1)
    gp1.fit(X_train_1, y_train_1)

    print "Expert number ", i, " with kernel: ", gp1.kernel_
    j = i+1
    name = process+'_'+str(train_start-train_end)+'_'+str(j)
    joblib.dump(gp1, name)

    return 1

# Train experts in parallel
out = Parallel(n_jobs=n_experts,
    verbose=4)(delayed(train_experts)(X_train, y_train,
    i*(trainsize/n_experts), (i+1)*(trainsize/n_experts), i)
    for i in range(n_experts))
```

A.2 Distributed Gaussian Process Prediction

```
"""
A class that takes trained GP-models from joblib, and
combines them into a rBCM for prediction.

@author: Ingrid A. V. Holm
"""
import numpy as np
import sklearn as sk
from sklearn.externals import joblib
from joblib import Parallel, delayed

class dgp_predict:
    """
    Combines Gaussian process experts into a
    robust Bayesian Comittee Machine.
    """
    def __init__(self, models):

    """
    Takes as argument the list 'models', that contains
    the names of the GP-experts,
    e.g. models=['uLuL_3000_1', 'uLuL_3000_2']
    """
        self.models = models
        self.n_experts = len(models)

    def predict(self, X, kernel=None):
        """
        Takes as arguments

        'X': test features

        and 'kernel': initial kernel used for all GPs.
        If 'kernel' is None, the kernel from each
        expert is used.
        """
        models = self.models
        self.X = X
        self.N = len(X)

        # Remove prefix for simplicity
        N = self.N
        n_experts = self.n_experts

        # Define empty arrays to save predicted means and
            variances
        mus = np.zeros((n_experts, N))
        sigmas = np.zeros((n_experts, N))
```

```
        priors = np.zeros((n_experts, N))

        # Loop over experts in parallel
        out = Parallel(njobs =
            n_experts)(delayed(prediction)(models[j], X,
            kernel) for j in range(n_experts))

        # Retrieve values from 'out'
        for i in range(n_experts):
            mus[i] = out[i][0]
            sigmas[i] = out[i][1]
            priors[i] = out[i][2]


        # Define empty arrays for the combined mean and
            variance
        sigma_rbcm_neg = np.zeros(N)
        mu_rbcm = np.zeros(N)

        # Sum over experts to find sigma_rbcm
        for j in range(n_experts):
            sigma_rbcm_temp = self.fill_sigmas(X, mus[j],
                sigmas[j], priors[j])
            sigma_rbcm_neg += sigma_rbcm_temp

        self.sigma_rbcm_neg = sigma_rbcm_neg

        # Sum over experts to find mu_rbcm
        for j in range(n_experts):
            mu_rbcm_temp = self.fill_mus(X, mus[j],
                sigmas[j], priors[j])
            mu_rbcm += mu_rbcm_temp

        # Return predictive mean and standard deviation
        return mu_rbcm, np.sqrt(sigma_rbcm_neg**(-1))


    def fill_sigmas(self, X, mu, sigma, prior):
        N = self.N
        n_experts = self.n_experts

        sigma_rbcm_neg = np.zeros(N)

        for k in range(N):
            mu_star = mu[k]
            sigma_star_mean = sigma[k]
            prior_cov = prior[k]

            beta =
                0.5*(np.log(prior_cov)-np.log(sigma_star_mean))
```

```
            sigma_rbcm_neg[k] =
                beta*sigma_star_mean**(-1)+(1./n_experts -
                beta)*prior_cov**(-1)

        # Return estimated sigma_rbcm
        return sigma_rbcm_neg

    def fill_mus(self, X, mu, sigma, prior):
        sigma_rbcm_neg = self.sigma_rbcm_neg
        N = self.N
        n_experts = self.n_experts

        mu_rbcm = np.zeros(N)

        for k in range(N):
            mu_star = mu[k]
            sigma_star_mean = sigma[k]
            prior_cov = prior[k]

            beta =
                0.5*(np.log(prior_cov)-np.log(sigma_star_mean))
            mu_rbcm[k] =
                sigma_rbcm_neg[k]**(-1)*(beta*sigma_star_mean**(-1)*mu_star)

        # Return estimated mu_rbcm
        return mu_rbcm


def prediction(model, X, kernel):

    N = len(X)
    gp = joblib.load(model)

    if kernel == None:
        kernel = gp.kernel_

    # Define empty array for this expert's means and variances
    mu_temp = np.zeros(N)
    sigma_temp = np.zeros(N)
    prior_temp = np.zeros(N)

    # Predict mean and variance for each test point
    for k in range(N):
        x = X[k].reshape(1,-1)
        mu_temp[k], sigma_temp[k] = gp.predict(x,
            return_cov=True)

        # Save the prior variance sigma_**
        prior_temp[k] = kernel(x)

    # Save predicted values
```

```
    mus[j] = mu_temp
    sigmas[j] = sigma_temp
    priors[j] = prior_temp

    return mus, sigmas, priors
```

A.3 Cross Validation

```
"""
A program that does cross validation for distributed
Gaussian processes.

@author: Ingrid A. V. Holm
"""

from dgp_parallel import dgp_parallel
import sys
import numpy as np
import sklearn as sk
from sklearn.gaussian_process.kernels import RBF,
    WhiteKernel, Matern, ConstantKernel as C
from sklearn.metrics import r2_score
from sklearn.cross_validation import train_test_split
from sklearn.model_selection import KFold

import matplotlib.pyplot as plt
import pandas as pd

##########################################################################
# Read in and treat data (mostly) using pandas.                          #
##########################################################################

feature_list = ['38.mGluino','40.mdL']
target_list = ['17.dLdL_NLO']

# Read in data (last column is BS)
data_lin =
    pd.read_csv('/usit/abel/u1/ingraho/combined_files_all/data_290k.dat',
    sep=' ', skipinitialspace=True)

#Create data frames in pandas
df_lin = pd.DataFrame(data_lin)

# Remove outliers
mask = df_lin['17.dLdL_NLO'] != 0
df_lin = df_lin[mask]
```

```
# Lower limit
mask_lower = df_lin['17.dLdL_NLO'] > 1e-16
df_lin = df_lin[mask_lower]

# Define features to train on and target nubers (used below)
# Convert to array for scikit using values
# ravel is used to avoid [[n]], i.e. n x 1 arrays
mean_list = [ "39.mcL", "40.mdL", "41.mdR", "42.muL",
    "43.msL", "44.muR", "45.msR", "46.mcR"]
means = df_lin[mean_list].mean(axis=1).values.ravel()

n_features = len(feature_list)
features = df_lin[feature_list].values
target = df_lin['17.dLdL_NLO'].values.ravel()

long_features = np.zeros(( len(features), len(features[0])+1
    ))
if len(long_features[0]) == 3:
    long_features[:,0] = features[:,0]
    long_features[:,1] = features[:,1]
    long_features[:,2] = means[:,]
elif len(long_features[0]) == 4:
    long_features[:,0] = features[:,0]
    long_features[:,1] = features[:,1]
    long_features[:,2] = features[:,2]
    long_features[:,3] = means[:,]

# Change target to sigma_mg
target_m2 = target/features[:,0]**2

##########################################################################
# Distributed Gaussian Processes                                         #
##########################################################################

# Define kernel to be used in GP
kernel_matern_3 = C(10, (1e-3, 100)) * Matern(np.array([10,
    10, 10]), (1e-4, 1e6), nu=1.5)


N_experts = 10
R2_scores_cv_test_mean = np.zeros(N_experts)
R2_scores_cv_train_mean = np.zeros(N_experts)
R2_scores_cv_test_std = np.zeros(N_experts)
R2_scores_cv_train_std = np.zeros(N_experts)

N_per_expert = 4000

n_experts = 1
my_njobs = 1

train_sizes = []
```

```
k = 5

for i in range(N_experts):

    trainsize = N_per_expert*(i+1)
    tot_size = int(float(k)/(k-1)*trainsize)

    train_sizes.append(trainsize)

    print "Size of training set: ", trainsize
    print "Size of test set: ", tot_size/k

    X, X_o, y, y_o = train_test_split(long_features,
        target_m2, random_state=42, train_size = tot_size)
    kf = KFold(n_splits=k, random_state=42)


    n_experts = i+1
    my_njobs = i+1

    r2_scores_test = []
    r2_scores_train = []

    for train_index, test_index in kf.split(X):
        X_train = X[train_index]
        X_test = X[test_index]
        y_train = y[train_index]
        y_test = y[test_index]

        print len(X_train), len(X_test)

        my_dgp = dgp_parallel(n_experts, output_name=False,
            kernel=kernel_matern_3, verbose=False,
            njobs=my_njobs)

        # Predict values for test data
        X_test_dgp, y_test_dgp, mu_dgp_test, sigma_dgp_test,
            rel_err_dgp =
            my_dgp.fit_and_predict(long_features, target_m2,
            trainsize=trainsize, alpha=7.544e-07, X_train_ =
            X_train, X_test_ = X_test, y_train_ = y_train,
            y_test_ = y_test)

        # Predict values for training data
        X_test_dgp, y_train_dgp, mu_dgp_train,
            sigma_dgp_train, rel_err_dgp =
            my_dgp.fit_and_predict(long_features, target_m2,
            trainsize=trainsize, alpha=7.544e-07, X_train_ =
            X_train, X_test_ = X_train, y_train_ = y_train,
            y_test_ = y_train)
```

```
            # Calculate and store R**2-scores
            r2_scores_test.append(r2_score(y_test_dgp,
                mu_dgp_test) )
            r2_scores_train.append(r2_score(y_train_dgp,
                mu_dgp_train) )


        r2_scores_test = np.asarray([r2_scores_test])
        R2_scores_cv_test_mean[i] = np.mean(r2_scores_test)
        R2_scores_cv_test_std[i] = np.std(r2_scores_test)

        r2_scores_train = np.asarray([r2_scores_train])
        R2_scores_cv_train_mean[i] = np.mean(r2_scores_train)
        R2_scores_cv_train_std[i] = np.std(r2_scores_train)

train_scores_mean = R2_scores_cv_train_mean
train_scores_std = R2_scores_cv_train_std
test_scores_mean = R2_scores_cv_test_mean
test_scores_std = R2_scores_cv_test_std


# Write training and validation scores to file
d = {'train_sizes' : train_sizes, 'train_scores_mean' :
    train_scores_mean, 'train_scores_std' : train_scores_std,
    'test_scores_mean' : test_scores_mean, 'test_scores_std' :
    test_scores_std}
df = pd.DataFrame(d)
df.to_csv('cv_scores_dLdL_experts_10x4000_nu15.dat', sep=' ')
```

Appendix B

Data Quality

Data quality plots for all squark masses are found here.

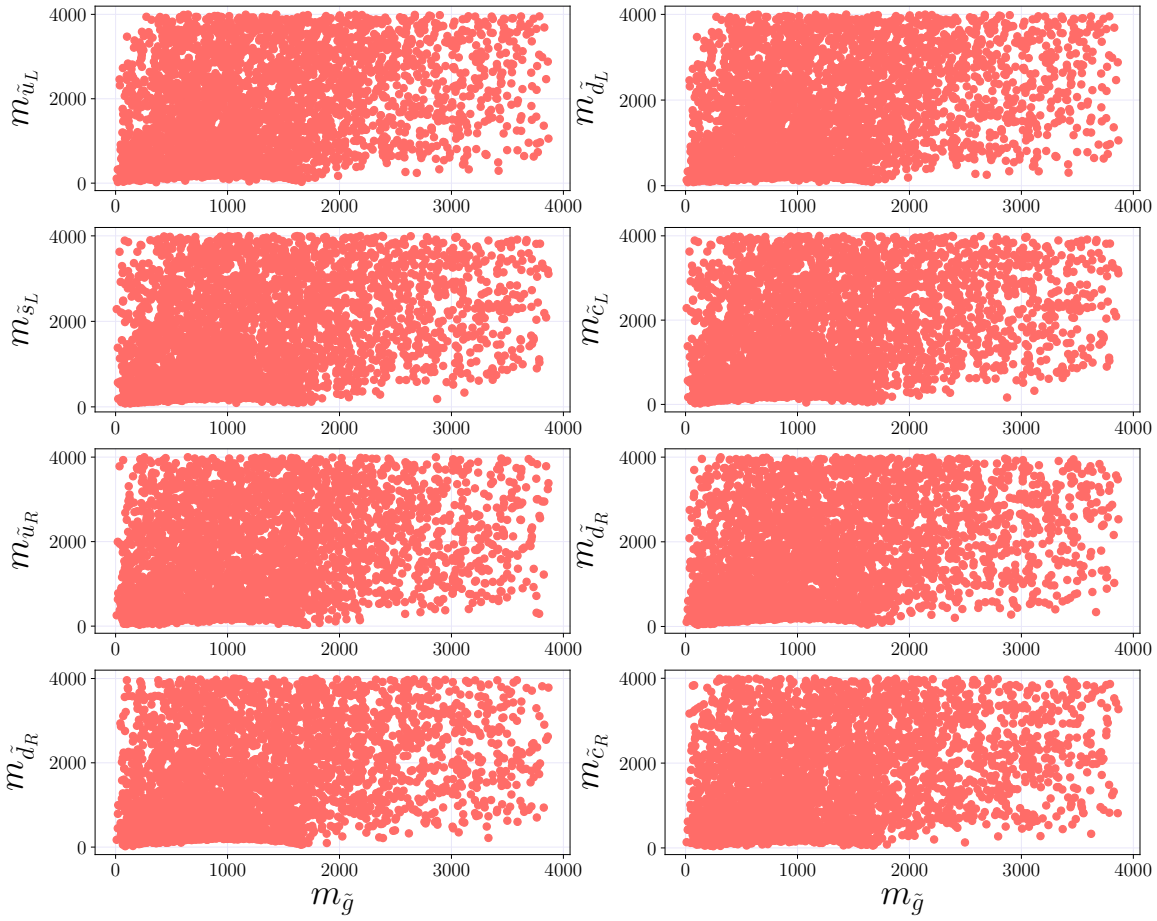


Figure B.1: Data quality plots of all squark masses versus the gluino mass.

Appendix C

Benchmark for Distributed Gaussian Processes

The benchmark function for parallelised distributed Gaussian processes is

$$f(x_1, x_2) = 4x_1x_2,$$

where the vectors $\mathbf{x} = (x_1, x_2)$ were drawn from a random normal distribution using the `numpy` function `random.randn`. Gaussian processes implemented by `scikit-learn` in the function `GaussianProcessRegressor` were compared to distributed Gaussian processes with 4 experts. 2000 training points and 1000 test points were used, and the resulting times for the GP and DGP were

$$\text{Gaussian processes time: } 154.12 \text{ s} \tag{C.1}$$

$$\text{Distributed Gaussian processes time: } 5.61 \text{ s} \tag{C.2}$$

Histograms of the relative deviances for Gaussian processes (GP) and Distributed Gaussian processes (DGP) are found in Fig. (C.1).

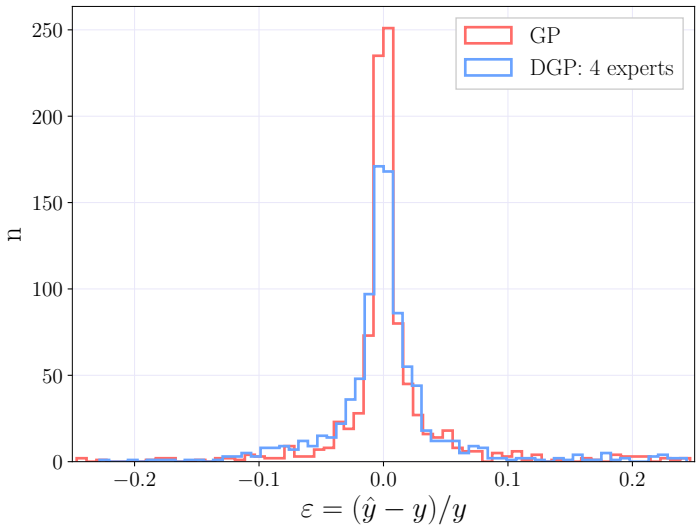


Figure C.1: Histogram of the relative deviance between true value y and predicted value $\hat{y}$ for Gaussian process regression (GP) and Distributed gaussian process regression (DGP) for the function $f(x_1, x_2) = 4x_1x_2$.

Appendix D

Relative Deviance Distributions

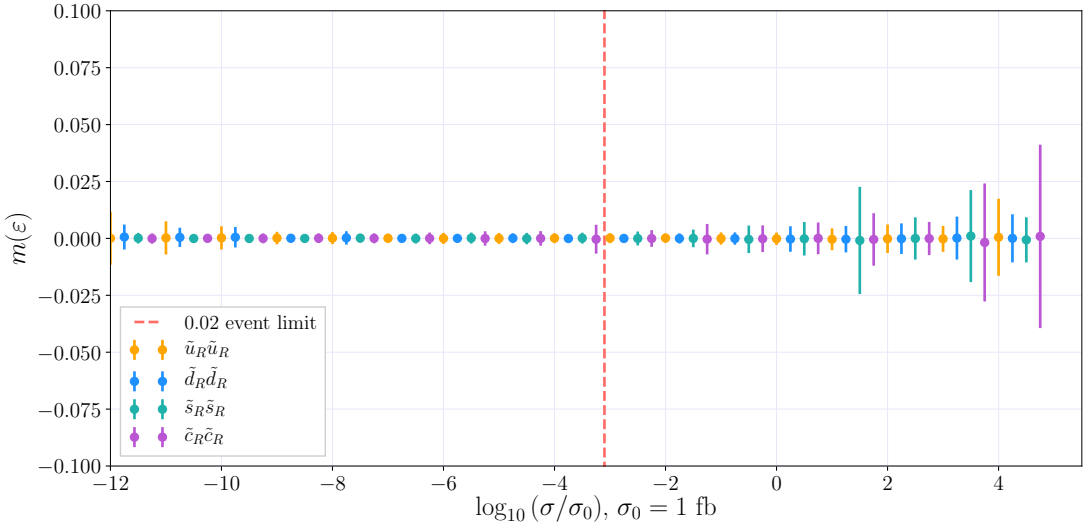


Figure D.1: Relative deviance distributions for $\widetilde{u}_R\widetilde{u}_R$, $\widetilde{d}_R\widetilde{d}_R$, $\widetilde{s}_R\widetilde{s}_R$ and $\widetilde{c}_R\widetilde{c}_R$.

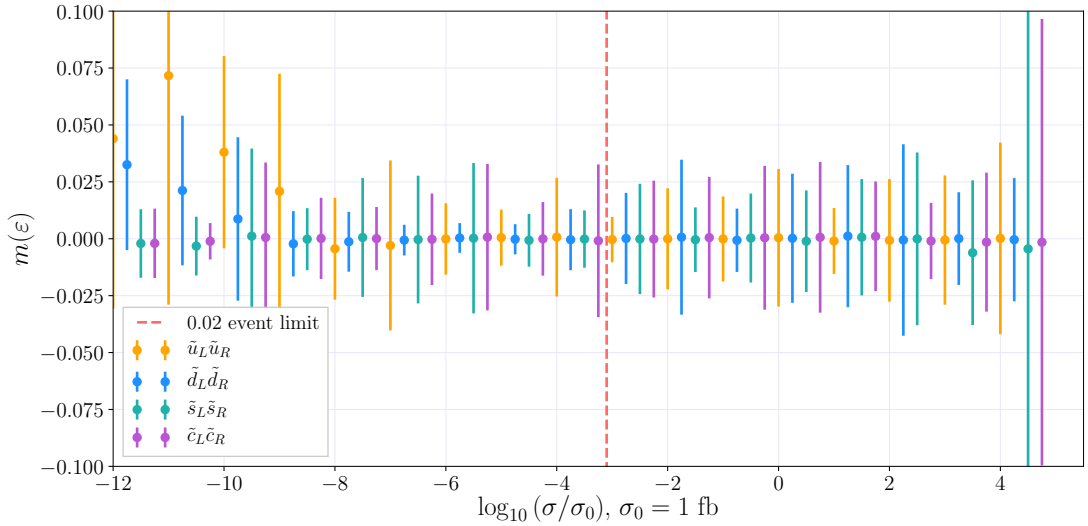


Figure D.2: Relative deviance distributions for $\widetilde{u}_L\widetilde{u}_R$, $\widetilde{d}_L\widetilde{d}_R$, $\widetilde{s}_L\widetilde{s}_R$ and $\widetilde{c}_L\widetilde{c}_R$.

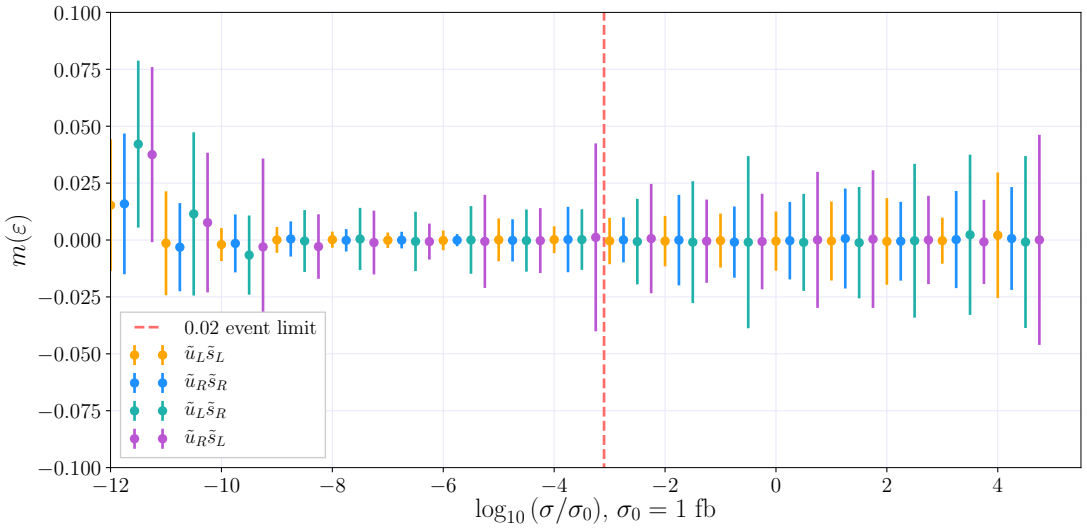


Figure D.3: Relative deviance distributions for $\widetilde{u}_L\widetilde{s}_L$, $\widetilde{u}_R\widetilde{s}_R$, $\widetilde{u}_L\widetilde{s}_R$ and $\widetilde{u}_R\widetilde{s}_L$.

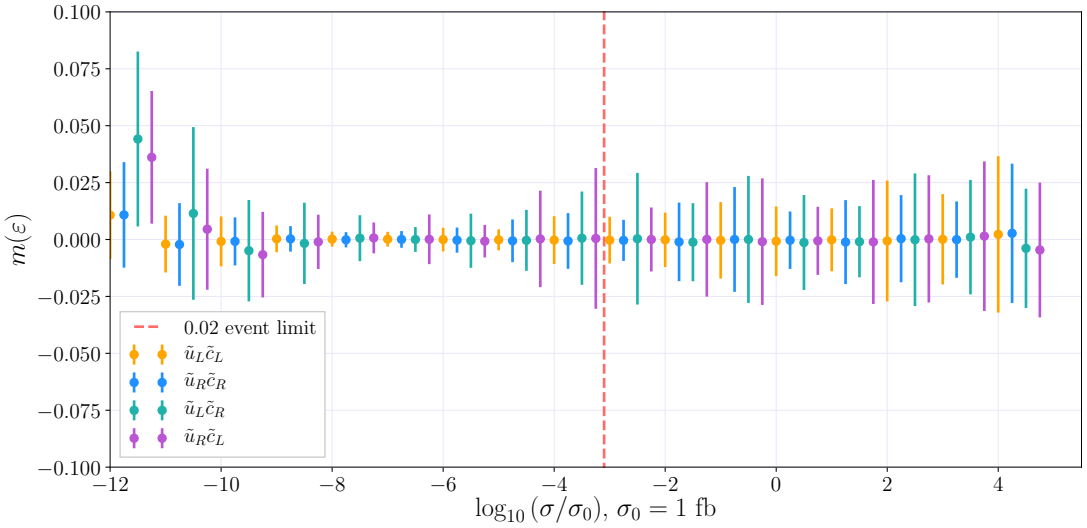


Figure D.4: Relative deviance distributions for $\widetilde{u}_L\widetilde{c}_L$, $\widetilde{u}_R\widetilde{c}_R$, $\widetilde{u}_L\widetilde{c}_R$ and $\widetilde{u}_R\widetilde{c}_L$.

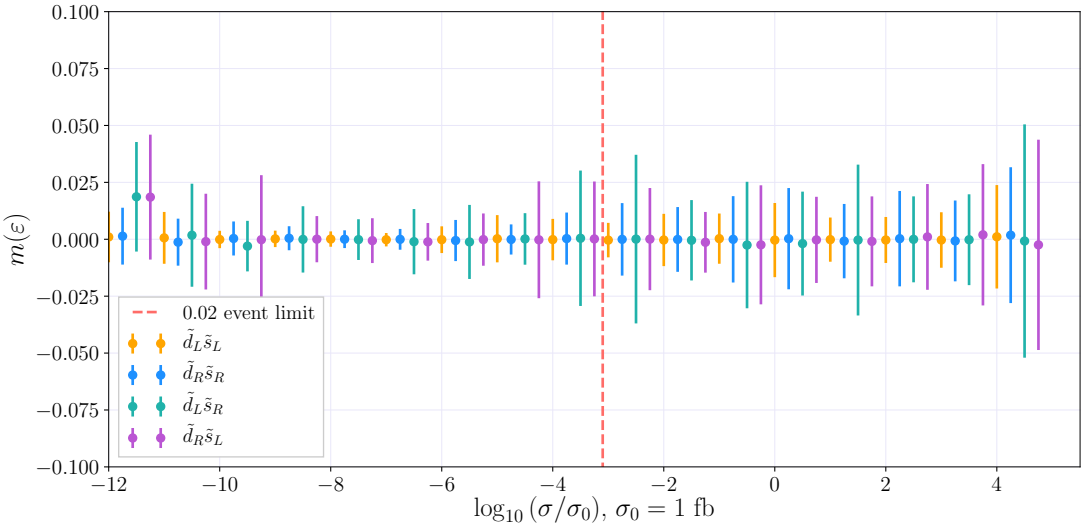


Figure D.5: Relative deviance distributions for $\widetilde{d}_L\widetilde{s}_L$, $\widetilde{d}_R\widetilde{s}_R$, $\widetilde{d}_L\widetilde{s}_R$ and $\widetilde{d}_R\widetilde{s}_L$.

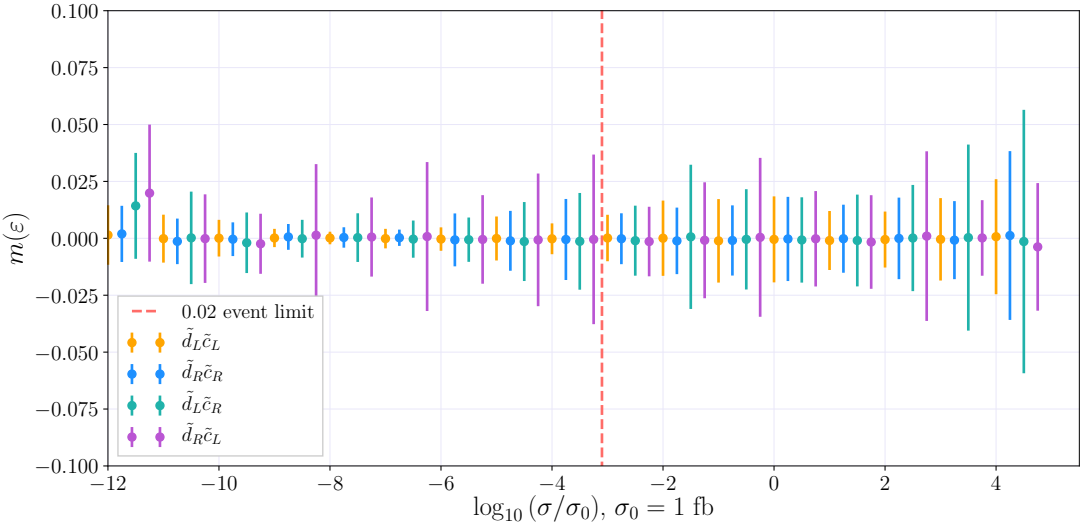


Figure D.6: Relative deviance distributions for $\widetilde{d}_L\widetilde{c}_L$, $\widetilde{d}_R\widetilde{c}_R$, $\widetilde{d}_L\widetilde{c}_R$ and $\widetilde{d}_R\widetilde{c}_L$.

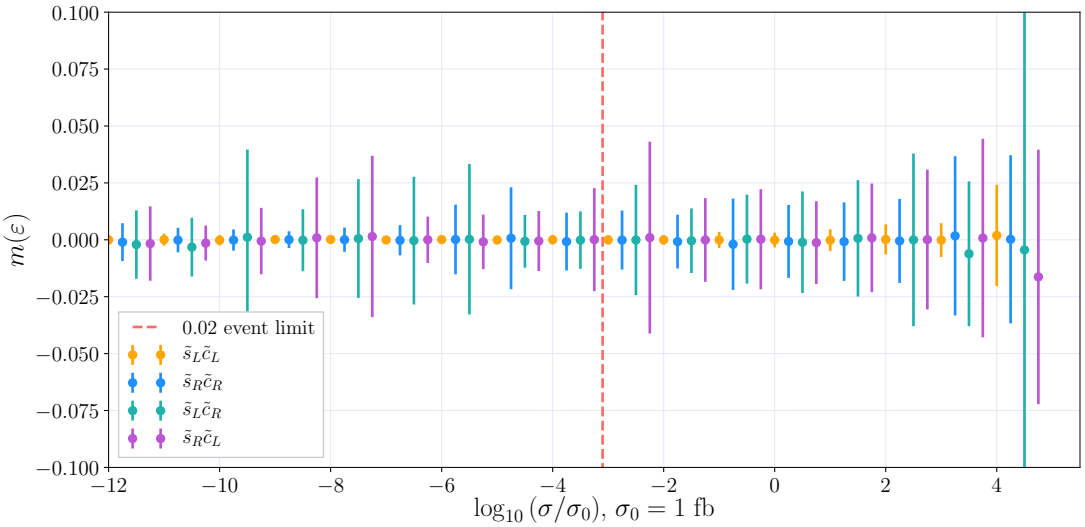


Figure D.7: Relative deviance distributions for $\widetilde{s}_L\widetilde{c}_L$, $\widetilde{s}_R\widetilde{c}_R$, $\widetilde{s}_L\widetilde{c}_R$ and $\widetilde{d}_R\widetilde{c}_L$.

Bibliography

[1] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. A. Khalek, A. Abdelalim et al., *Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc*, *Physics Letters B* **716** (2012) 1 – 29.

[2] S. Chatrchyan, V. Khachatryan, A. Sirunyan, A. Tumasyan, W. Adam, E. Aguilo et al., *Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc*, *Physics Letters B* **716** (2012) 30 – 61.

[3] W. Beenakker, R. Höpker, M. Spira and P. Zerwas, *Squark and gluino production at hadron colliders*, *Nuclear Physics B* **492** (1997) 51–103.

[4] W. Beenakker, R. Höpker and M. Spira, *Prospino: a program for the production of supersymmetric particles in next-to-leading order qcd*, *arXiv preprint hep-ph/9611232* (1996) .

[5] W. Beenakker, C. Borschensky, M. Krämer, A. Kulesza, E. Laenen, S. Marzani et al., *Nlo+ nll squark and gluino production cross sections with threshold-improved parton distributions*, *The European Physical Journal C* **76** (2016) 53.

[6] M. P. Deisenroth and J. W. Ng, *Distributed gaussian processes*, *arXiv preprint arXiv:1502.02843* (2015) .

[7] P. Batzing and A. Raklev, *Lecture notes for fys5190/fys9190*, .

[8] A. Kvellestad, *Chasing susy through parameter space*, .

[9] S. P. Martin, *A supersymmetry primer*, in *Perspectives on supersymmetry II*, pp. 1–153. World Scientific, 2010.

[10] G. R. Farrar and P. Fayet, *Phenomenology of the production, decay, and detection of new hadronic states associated with supersymmetry*, *Physics Letters B* **76** (1978) 575–579.

[11] S. Weinberg, *The Quantum Theory of Fields*, vol. 1. Cambridge University Press, 1995, 10.1017/CBO9781139644167.

[12] J. Larmor, *Lxiii. on the theory of the magnetic influence on spectra; and on the radiation from moving ions*, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **44** (1897) 503–512.

[13] P. M. Nadolsky, H.-L. Lai, Q.-H. Cao, J. Huston, J. Pumplin, D. Stump et al., *Implications of cteq global analysis for collider observables*, *Phys. Rev. D* **78** (Jul, 2008) 013004.

[14] S. Dulat, T.-J. Hou, J. Gao, M. Guzzi, J. Huston, P. Nadolsky et al., *New parton distribution functions from a global analysis of quantum chromodynamics*, *Physical Review D* **93** (2016) 033006.

[15] M. Aaboud, G. Aad, B. Abbott, J. Abdallah, O. Abdinov, B. Abeloos et al., *Search for squarks and gluinos in final states with jets and missing transverse momentum at \sqrt {s} s= 13{\mathrm {TeV}} tevwith the atlas detector*, *The European Physical Journal C* **76** (2016) 392.

[16] R. K. Ellis, W. J. Stirling and B. R. Webber, *QCD and collider physics*. Cambridge university press, 2003.

[17] G. P. Lepage, *A new algorithm for adaptive multidimensional integration*, *Journal of Computational Physics* **27** (1978) 192 – 203.

[18] C. Balázs, A. Buckley, L. A. Dal, B. Farmer, P. Jackson, A. Krislock et al., *Colliderbit: a gambit module for the calculation of high-energy collider observables and likelihoods*, *The European Physical Journal C* **77** (2017) 795.

[19] A. Kulesza and L. Motyka, *Threshold resummation for squark-antisquark and gluino-pair production at the lhc*, *Physical review letters* **102** (2009) 111802.

[20] A. Kulesza and L. Motyka, *Soft gluon resummation for the production of gluino-gluino and squark-antisquark pairs at the lhc*, *Physical Review D* **80** (2009) 095004.

[21] W. Beenakker, S. Brensing, M. Krämer, A. Kulesza, E. Laenen and I. Niessen, *Soft-gluon resummation for squark and gluino hadroproduction*, *Journal of High Energy Physics* **2009** (2009) 041.

[22] W. Beenakker, S. Brensing, M. Krämer, A. Kulesza, E. Laenen, L. Motyka et al., *Squark and gluino hadroproduction*, *International Journal of Modern Physics A* **26** (2011) 2637–2664.

[23] M. Bayes and M. Price, *An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a*

letter to john canton, amfrs, *Philosophical Transactions (1683-1775)* (1763) 370–418.

[24] D. Sivia and J. Skilling, *Data analysis: a Bayesian tutorial*. OUP Oxford, 2006.

[25] P. S. Laplace, *Théorie analytique des probabilités*. Courcier, 1820.

[26] M. Stein, "Interpolation of spatial data: some theory for kriging. 1999."

[27] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, vol. 55. Courier Corporation, 1964.

[28] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*, vol. 1. MIT press Cambridge, 2006.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., *Scikit-learn: Machine learning in Python*, *Journal of Machine Learning Research* **12** (2011) 2825–2830.

[30] J. V. Sparre, *Fast evaluation of supersymmetric cross sections*, .

[31] B. Allanach, *Softsusy: A program for calculating supersymmetric spectra*, *Computer Physics Communications* **143** (2002) 305 – 331.

[32] P. Skands, B. C. Allanach, H. Baer, C. Balázs, G. Bélanger, F. Boudjema et al., *Susy les houches accord: Interfacing susy spectrum calculators, decay packages, and event generators*, *Journal of High Energy Physics* **2004** (2004) 036.

[33] V. Tresp, *A bayesian committee machine*, *Neural computation* **12** (2000) 2719–2741.