Contents

Chapter 1

Evaluating Cross Sections with Gaussian Processes

This chapter is dedicated to evaluating cross sections for squark pair production with Gaussian processes. The learning of a benchmark process is considered, and compared to Gaussian processes with changes in the data set, the kernel and different features. One change to the benchmark settings is considered at a time. Then the effect of adding more data in the form of experts is investigated, and finally learning curves for the optimal parameters are calculated to decide whether the method benefits from adding more data.

1.1 Data Generation

In this section the generation of MSSM-24 training and test data is discussed, following closely the discussion in [4].

Sampling of Data

An MPI parallelized script generates a sample point in parameter space by drawing random values from the distributions in Tab. (1.1). The table contains log and flat priors, and a combination of these is used to cover more of the parameter space. When a parameter point has been sampled, it is run through the program `softpoint.x` which calculates its SUSY spectrum using the `Softsusy 3.6.2`-package [1]. The spectrum is then written to a `slha`-file and given as input to `Prospino 2.1`, which calculates the LO and NLO cross sections according to the method outlined in Section 3.4.1. The relevant features and NLO cross sections are harvested to `.dat`-files, which are used by the Gaussian processes.

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 1.1: Table showing the sampling intervals used for the parameters when sampling the MSSM-24 model, 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 [start_flat, start_log, end_log] and [start_log, start_flat, start_log, end_log]. All values in GeV except $\tan\beta$ which is unitless. Table from [4].

Priors

To get a reasonable distribution in parameter space it is necessary to use an objective prior distribution of parameters. This means that priors are assigned according to a set of principles of how information should be encoded in a probability distribution. More details on objective priors can be found in [3].

The first of these principles is the *transformation group invariance*, which states that the probability distribution should be invariant under any transformation that is considered irrelevant to the problem. In other words, the *pdf* should satisfy

$$\pi(x|I)dx = \pi(x+a|I)d(x+a) \qquad \Rightarrow \qquad \pi(x|I) = \pi(x+a|I), \tag{1.1}$$

where a is some translation. This is often referred to as a uniform or *flat prior*. Invariance under scale transformations, which are transformations that introduce a scale to the problem $m \to m' = cm$, requires

$$\pi(m|I)dm = \pi(cm|I)cdm, \tag{1.2}$$

which is satisfied if $\pi(m|I) \propto 1/m$. Since this corresponds to $\pi(\log m|I)$ being flat, it is called the *log prior*.

The flat prior covers the edges of parameter space well, while a log prior covers the innermost points [1]. Therefore, 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 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 prior with positive values is shown in Fig. (1.1), where a flat distribution is used close to $x = 0$ to avoid divergences.

The weak scale MSSM model used in this project, MSSM-24, requires a soft breaking scale Q. This scale is set to $Q = 1$ TeV. It is also worth noting that the parameter space for the cross sections is significantly reduced from that of the MSSM-24. The cross sections depend on the values $m_{\tilde{g}}$, $m_{\tilde{q}}$, $\tilde{g}_s$, $\hat{g}_s$ and s. Since only first and second generation squarks are considered, this contributes with 8 masses [2] which combined with the 4 other parameters reduces the parameter space to 12 dimensions. The parameter space is thus better sampled than it would appear from the 24 parameters in MSSM-24.

Data Quality

To ensure the data is properly distributed in parameter space data quality plots are generated. In Fig. (1.2) distributions for $m_{\tilde{g}}$, $m_{\tilde{d}_L}$ and $m_{\tilde{u}_L}$ are shown, as

[1] The points close to 0.

[2] 4 quarks with a pair of lefthanded and righthanded squarks each.

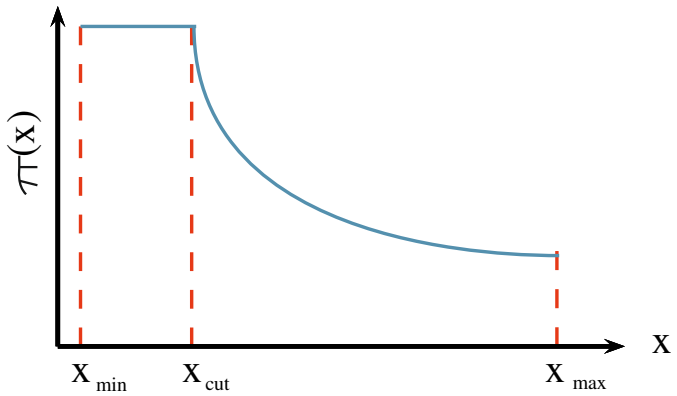


Figure 1.1: Illustration of the log prior distribution $\pi(x)$. Around $x = 0$ the prior would blow up, so at a limit x_{cut} a flat prior is used.

these are the most relevant for this thesis. The scatter plots use 2000 points, which is the number of training points used in the benchmark settings in the following sections. All parts of the parameter space seem to be covered, although the density of points is higher for small masses. This could affect predictions with few training points.

Noise

The observations contain some noise that originates in the `Prospino 2.1` calculation. In a parameter point chosen at random the relative error has a standard deviation of $\varepsilon = 0.002$. This error fluctuates little between parameter points, so it is 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 process. For that purpose the following relation is considered,

$$Y_i = y_i^{true} + \epsilon_i = y_i^{true}(1 + \varepsilon_i), \tag{1.3}$$

where cross sections provided by Prospino are denoted as Y_i, real cross sections as y_i^{true} and $\varepsilon_i \sim \mathcal{N}(0, \varepsilon)$. The distribution of Y_i can thus be written as

$$Y_i = \mathcal{N}(y_i^{true}, \varepsilon y_i^{true}), \tag{1.4}$$

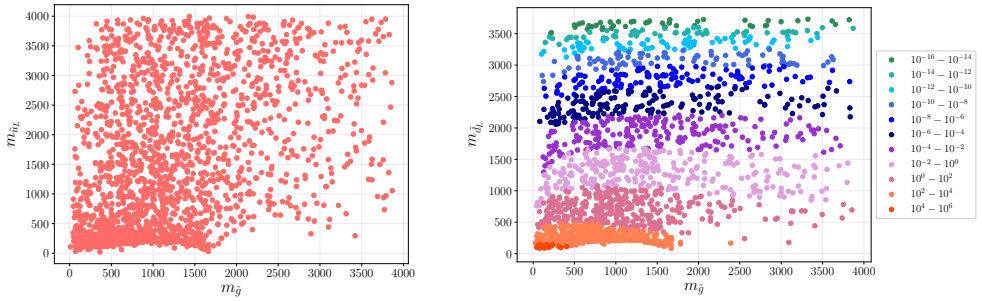
where the only random variable is ε. Changing variables to $\log_{10}$ gives

$$X_i = \log_{10} Y_i \to Y_i = 10^{X_i} \tag{1.5}$$

$$P_{X_i}(X_i) = P_{Y_i}(Y_i)\left|\frac{\partial Y_i}{\partial X_i}\right| \tag{1.6}$$

$$= P_{Y_i}(y_i) 10^{X^i} \log 10 \tag{1.7}$$

$$= \mathcal{N}(10^{x_i^{true}}, \varepsilon 10^{x_i^{true}}) \cdot 10^{X_i} \cdot \log 10. \tag{1.8}$$



(a) $m_{\tilde{g}}$-$m_{\tilde{u}_L}$ **(b)** $m_{\tilde{g}}$-$m_{\tilde{d}_L}$

Figure 1.2: Data quality plots of the distribution of mass parameters $m_{\tilde{d}_L}$, $m_{\tilde{u}_L}$ and $m_{\tilde{g}}$ for 2000 points. In **(b)** different orders of magnitude of the cross sections for $\tilde{d}_L\tilde{d}_L$ are shown in different colors, indicating that there are few points in the very high cross sections $\sigma \propto 10^4 - 10^6$, and that lower cross sections are more spread across the gluino mass spectrum.

So the relevant distribution is in fact

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

where the following expansion can be made

$$\log_{10}(1 + \mathcal{N}(0, \varepsilon)) \simeq \frac{\mathcal{N}(0, \varepsilon)}{\log 10} - \frac{\mathcal{N}(0, \varepsilon)^2}{\log 100} + ... \tag{1.9}$$

Since the leading order term is dominant for small ε the logarithm of the cross section may be approximated as

$$X_i \simeq \log_{10} y_i^{true} + \frac{1}{\log 10}\mathcal{N}(0, \varepsilon) \tag{1.10}$$

Eq. (**??**) is now on the form of the Gaussian noise model. The error distribution has a standard deviation $\varepsilon = 2 \cdot 10^{-3}$, so noise variance of the Gaussian process should be

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

The noise term predicted by the GP should therefore be of the order of $\mathcal{O}(10^{-7})$.

1.2 Dataset Transformations

The plot in Fig. (1.2b) indicates that cross sections, especially those of the order $\mathcal{O}(1$ fb) and lower, are very spread as a function of $m_{\tilde{g}}$. This is more evident

in the upper left panel of Fig. (1.3), where the cross section σ is plotted as a function of the gluino mass. The upper left panel shows σ as a function of the squark mass $m_{\tilde{q}}$, which is more defined but still has some spread. This section will be devoted to reducing the spread caused by the gluino mass.

Scaling Functions

As previously mentioned, the partonic cross sections can be written in terms of scaling functions f

$$\hat{\sigma}_{ij} = \frac{\alpha_s^2(Q^2)}{m^2}\left\{ f_{ij}^B(\eta, r) + 4\pi\alpha_s(Q^2)\left[f_{ij}^{V+S}(\eta, r, r_t) + f_{ij}^H(\eta, r) + \bar{f}_{ij}(\eta, r)\log\left(\frac{Q^2}{m^2}\right)\right]\right\}, \tag{1.11}$$

where

$$\eta = \frac{s}{m^2} - 1, \qquad r = \frac{m_{\tilde{g}}^2}{m_{\tilde{q}}^2}, \qquad r_t = \frac{m_t^2}{m^2},$$

where $m = (\sqrt{p_1^2} + \sqrt{p_2^2})/2$ is the average mass of the particles produced. The scaling functions are the different contributions to the cross section, as explained in Chapter 2. As the total cross section only differs from the partonic cross section by an integral over parton distribution functions, the mass dependencies in Eq. (1.11) are relevant for the total cross sections as well.

The energy near the threshold is the base for an important part of the contributions to the cross section [2]. In this region the scaling functions can be expanded in the low velocity of produced particles β, leading to the following expressions [2]

$$\begin{aligned}
f_{qq}^B &= \frac{8\pi\beta m_{\tilde{q}}^2 m_{\tilde{g}}^2}{27(m_{\tilde{q}}^2 + m_{\tilde{g}}^2)^2}, & f_{q'q}^B &= \frac{8\pi\beta m_{\tilde{q}}^2 m_{\tilde{g}}^2}{9(m_{\tilde{q}}^2 + m_{\tilde{g}}^2)^2} \\
f_{qq}^{V+S} &= f_{qq}^B \frac{1}{24\beta} & f_{q'q}^{V+S} &= f_{q'q}^B \frac{1}{24\beta} \\
f_{qq}^H &= f_{qq}^B\Big[\frac{2}{3\pi^2}\log^2(8\beta^2) - \frac{7}{2\pi^2}\log(8\beta^2)\Big] & f_{q'q}^H &= f_{q'q}^B\Big[\frac{2}{3\pi^2}\log^2(8\beta^2) - \frac{19}{6\pi^2}\log(8\beta^2)\Big] \\
\bar{f}_{qq} &= -f_{qq}^B \frac{2}{3\pi^2}\log(8\beta^2) & \bar{f}_{q'q} &= -f_{q'q}^B \frac{2}{3\pi^2}\log(8\beta^2).
\end{aligned} \tag{1.12}$$

As the main contributions come from this energy region, it may be possible to remove some of the complexity of the function using the expressions in Eq. 1.12. Note that all terms are proportional to f_{qq}^B ($f_{q'q}^B$), which is again proportional to $m_{\tilde{q}}^2 m_{\tilde{g}}^2$. Since the partonic cross section is proportional to $\sigma \propto 1/m^2$, and

$m^2 = m_{\tilde{q}}^2$ in the case of squark production this $m_{\tilde{q}}^2$-dependency is automatically cancelled. However, the following transformation can be made

$$\sigma \to \sigma_{m_{\tilde{g}}} = \frac{\sigma}{m_{\tilde{g}}^2}, \tag{1.13}$$

reducing the gluino mass dependency. The lower panels in Fig. (1.3) show $\sigma_{m_{\tilde{g}}}$ as a function of $m_{\tilde{g}}$ and $m_{\tilde{q}}$. The spread as a function of the squark mass is much smaller, and for high cross sections the shape of $\sigma_{m_{\tilde{g}}}$ as a function of squark and gluino mass are very similar, which may make it easier to find a kernel that fits well in both dimensions. Therefore, the target value in this thesis will be the logarithm of $\sigma_{m_{\tilde{g}}}$. The logarithm is used because of the large span in function values. In the threshold region, the partonic version of this expression is

$$\hat{\sigma}_{ij,m_{\tilde{g}}} = \frac{8\pi\beta\alpha_s^2(Q^2)}{27(m_{\tilde{q}}^2 + m_{\tilde{g}}^2)^2}\Bigg\{1 + 4\pi\alpha_s(Q^2)\Bigg[\frac{1}{24\beta} + \frac{2}{3\pi^2}\log^2(8\beta^2) \tag{1.14}$$

$$- \frac{7}{2\pi^2}\log(8\beta^2) - \frac{2}{3\pi^2}\log(8\beta^2)\log\left(\frac{Q^2}{m^2}\right)\Bigg]\Bigg\}. \tag{1.15}$$

Another possibility is to further reduce the mass dependecy, by defining σ_{fac} as

$$\sigma_{fac} = \sigma\frac{(m_{\tilde{g}}^2 + m_{\tilde{q}}^2)^2}{m_{\tilde{g}}^2}. \tag{1.16}$$

This transformation provides an even smoother function, but has been left as further work with GPs.

1.3 Learning the Gaussian Process

In this section a single Gaussian process is learned with benchmark settings, and then a selected set of improvements are introduced. The results are quantified using plots of the relative deviance defined in Chapter 4 (REFERER TIL LIKNINGEN).

1.3.1 The Benchmark

In this thesis the benchmark processes will be the production of $\tilde{d}_L\tilde{d}_L$ and $\tilde{d}_L\tilde{u}_L$. The benchmark settings are a single GP with 2000 training points and 20 000 test points that uses $m_{\tilde{g}}$ and $m_{\tilde{d}_L}$ as features (as well as $m_{\tilde{u}_L}$ for $\tilde{d}_L\tilde{u}_L$ production). The exponential squared kernel (RBF [3]) with a white noise term is used. The kernel is implemented in `scikit-learn` in the following way

[3]The exponential squared kernel is often called the radial basis function, hence the abbreviation.

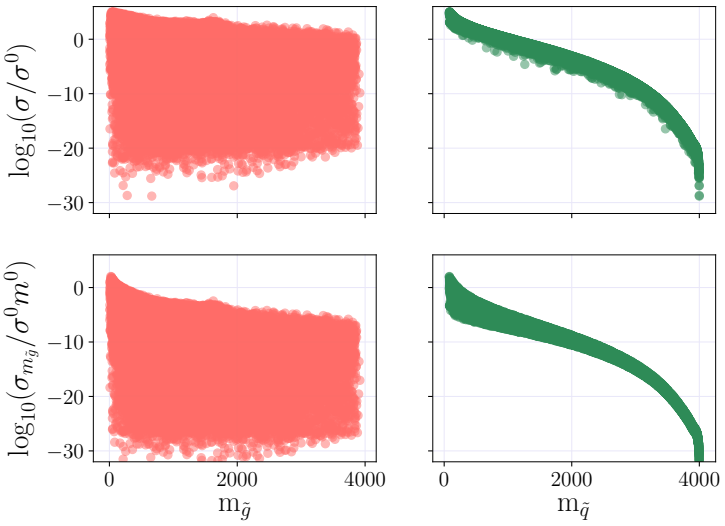


Figure 1.3: The quantities σ, $\sigma_{m_{\tilde{g}}}$ and $\sigma_{\tilde{q}}$ as functions of the gluino mass $m_{\tilde{q}}$ and squark mass $m_{\tilde{q}}$ for the production of $\tilde{d}_L\tilde{d}_L$. Here, $m_{\tilde{q}} = m_{\tilde{d}_L}$. The functions have less spread when some of the mass dependency is removed, which may make learning easier.

```
kernel_BM = C(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))
```

where the features are `(m_gluino, m_dL)` for $\tilde{d}_L\tilde{d}_L$ and `(m_gluino, m_dL, m_uL)` for $\tilde{d}_L\tilde{u}_L$. Note that the length scale of the RBF is given a a vector of the same dimension as the feature vector $\mathbf{x}, \boldsymbol{\ell} \in \mathbb{R}^D$ [4]. This is in case the different masses have different covariances, as they appear to do from the lower panels in Fig. (1.3).

Because of the large span of $\sigma_{m_{\tilde{g}}}$-values (ranging from 10^{-32} to 10^6), values are divided into decades. This means that for each interval $10^i - 10^{i+1}$ an error is calculated. The error is the mean and variance of the distribution of relative deviations in each decade

$$\mu = \mathrm{m}\left(\frac{y-\hat{y}}{y}\right), \qquad \sigma^2 = \mathrm{var}\left(\frac{y-\hat{y}}{y}\right). \tag{1.17}$$

The means and variances of the logarithms are plotted for the BM settings in Fig. (1.4) for $\tilde{d}_L\tilde{d}_L$. The plot for $\tilde{d}_L\tilde{u}_L$ is very similar, and is therefore not shown here.

[4] This example is for $\tilde{d}_L\tilde{d}_L$ production, for $\tilde{d}_L\tilde{u}_L$ production $D = 3$, as $m_{\tilde{u}_L}$ is inluded as well.

	C	$\ell_{m_{\tilde{g}}}$	$\ell_{m_{\tilde{d}_L}}$	$\ell_{\bar{m}}$	α
BM	2981	5470	2190		0.47
No outliers	9702	5740	215		0.00372
$\sigma > 10^{-16}$ fb	515	1170	998		0.0036
$\bar{m}$	1095	1190	200	846	0.0000119
Matern	1000	30200	8600		0.462

Table 1.2: Optimal kernel parameters for different settings for $\tilde{d}_L\tilde{d}_L$.

	C	$\ell_{m_{\tilde{g}}}$	$\ell_{m_{\tilde{d}_L}}$	$\ell_{m_{\tilde{u}_L}}$	$\ell_{\bar{m}}$	α
BM	2490	5870	4000	2220		0.593
No outliers	6400	4420	3100	240		0.00348
$\sigma > 10^{-16}$ fb	10000	1540	2900	2150		0.0031
$\bar{m}$	3806	1340	3590	251	748	0.0000119
Matern	1000	31100	1000000	8260		0.585

Table 1.3: Optimal kernel parameters for different settings for $\tilde{d}_L\tilde{u}_L$.

The optimal kernel parameters found by the GP are found in Tab. (1.2)-(1.3), while computation times on a laptop are found in Tab. (1.4). The predicted noise levels α are very high for both processes, at $\alpha = 0.47$ for $\tilde{d}_L\tilde{d}_L$ and $\alpha = 0.593$ for $\tilde{d}_L\tilde{u}_L$, which is far from the expected value of $\sim 7.544 \cdot 10^{-7}$. In addition, the prediction is very unstable, with a over prediction for low cross sections.

1.3.2 Outliers

Calculations in `Prospino 2.1` set some NLO terms to zero because $K = 0$ for numerical reasons, as discussed in **??**. This leads to outliers in the dataset, as shown in Fig. (1.5), where they can be seen as a cluster of points well below the other cross sections. These points have zero cross sections, which are set to $\epsilon = 10^{-32}$ fb in the calculation to avoid divergences. Removing the ouliers makes the prediction much better for small cross sections, but also stabilizes the

	Time $\tilde{d}_L\tilde{d}_L$	Time $\tilde{d}_L\tilde{u}_L$
BM	00:07:48	00:08:40
Outliers	00:08:42	00:11:20
Cut	00:07:24	00:11:07
Features	00:11:20	00:16:37
Kernel	00:07:28	00:13:05

Table 1.4: Computation times for GP with 2000 training points and 20 000 test points on a laptop with 4 cores.

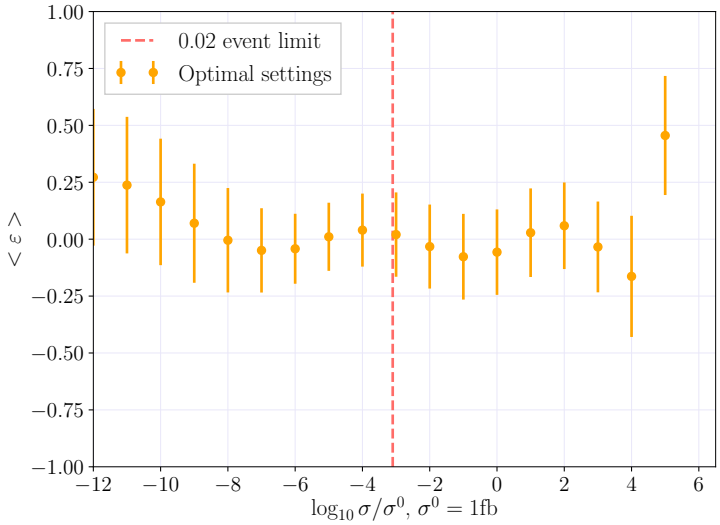


Figure 1.4: The relative deviance ε as a function of the logarithm of the normalized cross section for $\tilde{d}_L\tilde{d}_L$ with benchmark settings. 2000 training points and 20 000 test points were used on a regular GP, with the final kernel as described in the text. Features are the physical masses $m_{\tilde{g}}$ and $m_{\tilde{d}_L}$.

prediction for larger values, as can be seen in Fig. (1.7 a) where the prediction without outliers is compared to the BM. The predicted noise levels are significantly reduced with the removal of outliers for both processes, from $\alpha = 0.47$ to $\alpha = 0.00372$ for $\tilde{d}_L\tilde{d}_L$ and from $\alpha = 0.593$ to $\alpha = 0.00348$ for $\tilde{d}_L\tilde{u}_L$. This implies that including the outliers leads the GP to underfit, which means that much of the signal is considered noise.

1.3.3 Cuts on Cross Sections

Smooth functions are easier to fit with Gaussian processes, and the target values are very steep functions of large squark masses. This can be seen from the righthand panels in Fig. (1.3). 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 is at $\sigma = 10^{-3}$ fb [5], a lower cut is set at $\sigma_{cut} = 10^{-16}$ fb. The cut excludes all cross sections lower than σ_{cut} from both training and testing. The resulting error distributions for $\tilde{d}_L\tilde{d}_L$ are shown in Fig. (1.7 b), and the optimal kernel parameters are found in Tab. (1.2)-(1.3). Noise levels are further reduced from the case where outliers are removed, with the variance going from $\alpha = 0.00372$ to $\alpha = 0.00336$ for $\tilde{d}_L\tilde{d}_L$ and from $\alpha = 0.00348$ to $\alpha = 0.0031$ for

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

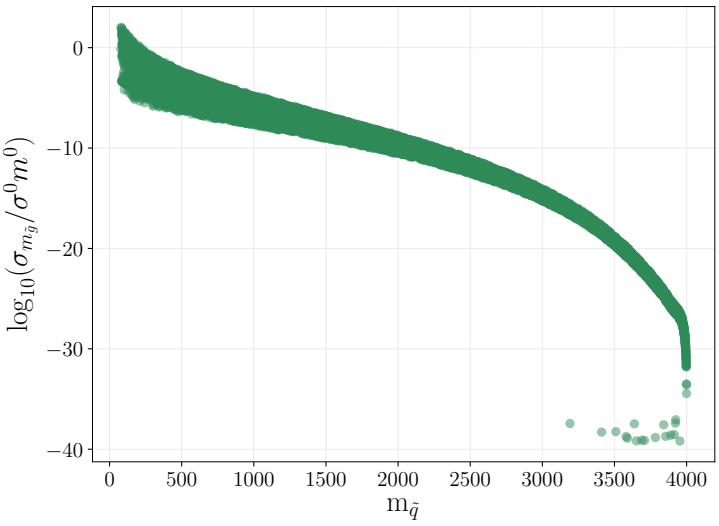


Figure 1.5: Plot of $\log(\sigma_{m_{\tilde{g}}})$ for $\tilde{d}_L\tilde{d}_L$ as a function of $m_{\tilde{d}_L}$, where outliers are included. The outliers are originally $\sigma = 0$ fb, but are set to $\sigma = 10^{-32}$ fb so as to avoid infinities.

$\tilde{d}_L\tilde{u}_L$. The estimated noise level seems to be approaching the expected theoretical value, indicating that the exclusion of low cross sections improves the prediction. The error in the prediction of the highest targets is significantly improved from the cases of the BM and the removed outliers. Training and testing only on high values renders the prediction very stable for all (included) orders of magnitude.

1.3.4 Features

`Prospino 2.1` calculates the NLO cross section for the mean of the squark masses, $\bar{m}$, and uses this to find the K-factor. The K-factor is then multplied with the LO cross sections for non-degenerate squark masses to find the individual NLO terms. There is therefore reason to believe that adding the mean squark mass as a feature will improve the prediction. The features used in this section are

```
features_dLdL = (m_gluino, m_dL, m_mean)
features_dLuL = (m_gluino, m_dL, m_uL, m_mean)
```

The optimal kernel values are found in Tab. (1.2)-(1.3). Adding the mean mass as a feature reduces the estimated noise level considerably, and estimates the same level for $\tilde{d}_L\tilde{d}_L$ and $\tilde{d}_L\tilde{d}_L$, at $\alpha = 1.19 \cdot 10^{-5}$. Since the two processes should have approximately the same level of noise this is an indication that the prediction is improved. Further, the length scale for $\bar{m}$ is smaller than for $m_{\tilde{g}}$.

This implies that there is a higher dependence on the mean mass than the gluino mass, since GPs attribute large length scales to irrelevant features. The resulting error distributions for $\tilde{d}_L\tilde{d}_L$ are shown in Fig. (1.7 c) and compared to the BM. With some exceptions where the variances are very large adding the new feature gives a mean of almost zero and very small variances for cross sections over the 0.02 event limit.

1.3.5 Kernel

While the exponential squared kernel is very smooth, 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. In this section the Matérn kernel with $\nu = 1.5$ is compared to the BM RBF kernel, and the resulting error distributions for $\tilde{d}_L\tilde{d}_L$ are found in the lower right panel of Fig. (1.4). The choice of hyperparameter is explained below. The predictions are somewhat more stable for high cross sections than with the RBF kernel. For low cross sections the error distributions have larger variances, but as this is currently not the region of interest the Matérn kernel is considered a better fit than the RBF. As can be seen from the optimal kernel values in Tab. (1.2)-(1.3) the Matérn kernel predicts a slightly lower noise level than the RBF. The noise variances go from $\alpha = 0.47$ with RBF to $\alpha = 0.462$ with Matérn for $\tilde{d}_L\tilde{d}_L$, and from $\alpha = 0.593$ with RBF to $\alpha = 0.585$ with Matérn for $\tilde{d}_L\tilde{u}_L$.

Hyperparameters

The Matérn kernel has the aforementioned hyperparameter ν, which controls the smoothness of the function. As $\nu \to \infty$ the function becomes very smooth, and approaches the exponential squared kernel. For half-integer values the Matérn kernel becomes the product of an exponential squared kernel and a polynomial – a simple form which is faster to compute [6]. In order to determine the best value of ν k-fold cross validation was performed as a function of ν for the BM settings. The validation curve uses the R^2-score and $k = 5$ for the cross validation, and is shown in Fig. (1.6). Based on the cross validation, the best value for the hyperparameter is $\nu = 1.5$.

1.3.6 Optimal Settings

A combination of the improved settings found in the foregoing sections is tested in this section. The cummulative settings are; a single GP with outlier points removed; a lower cut on cross sections $\sigma > 10^{-16}$ fb; the Matérn kernel with

[6] Scikit-learn takes around 10 times longer to compute values not in the interval $\nu = [0.5, 1.5, 2.5, \infty]$, because of the evaluation of Bessel functions.

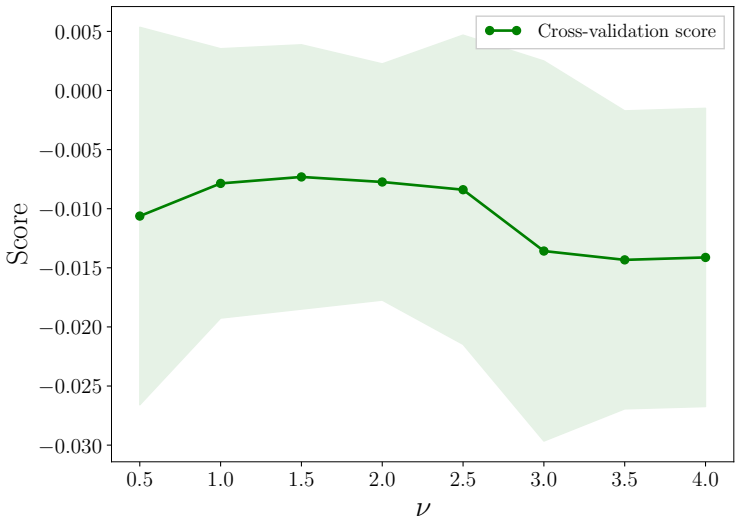


Figure 1.6: Validation curve as a function of the hyperparameter ν of the Matérn kernel, with k-fold validation with $k = 5$. The settings are the BM settings, except for the exchange of the RBF kernel for the Matérn kernel. The scoring parameter is the R^2-score, but here $R^2 - 1$ is plotted.

$\nu = 1.5$; and the gluino mass, the relevant squark mass(es) and the mean of all squark masses as features. The resulting error distributions are found in Fig. (1.8). With all improvements implemented the prediction is very good, as all error distribution variances are less than 5%. The computation with optimal settings takes 00:09:30 for $\tilde{d}_L\tilde{d}_L$ and 00:10:28 for $\tilde{d}_L\tilde{u}_L$ on a regular laptop with 4 cores.

Noise

Since the noise level is known to some degree, it is worth testing whether this knowledge be used in Gaussian processes. As previously shown, the variance of the Gaussian distribution of noise is around $\alpha_{fix} = 7.544 \cdot 10^{-7}$. In `scikit-learn` an option to letting the `WhiteKernel` estimate the level of noise is to specify the noise by passing it as `alpha` to the Gaussian process regressor function

```
gp = GaussianProcessRegressor(kernel=kernel, alpha=7.544e-7)
```

For the optimal settings the `WhiteKernel` predicts values close to α_{fix}, with $\alpha = 10^{-5}$ for $\tilde{d}_L\tilde{d}_L$ and $5.63 \cdot 10^{-6}$ for $\tilde{d}_L\tilde{u}_L$. The remaining kernel parameters therefore hardly change when α is fixed, as can be seen in Tab. (1.5). As expected, the prediction changes very little. A marginal improvement can be seen in the variance for α_{fix} in Fig. (1.9). For calculations with few training points the

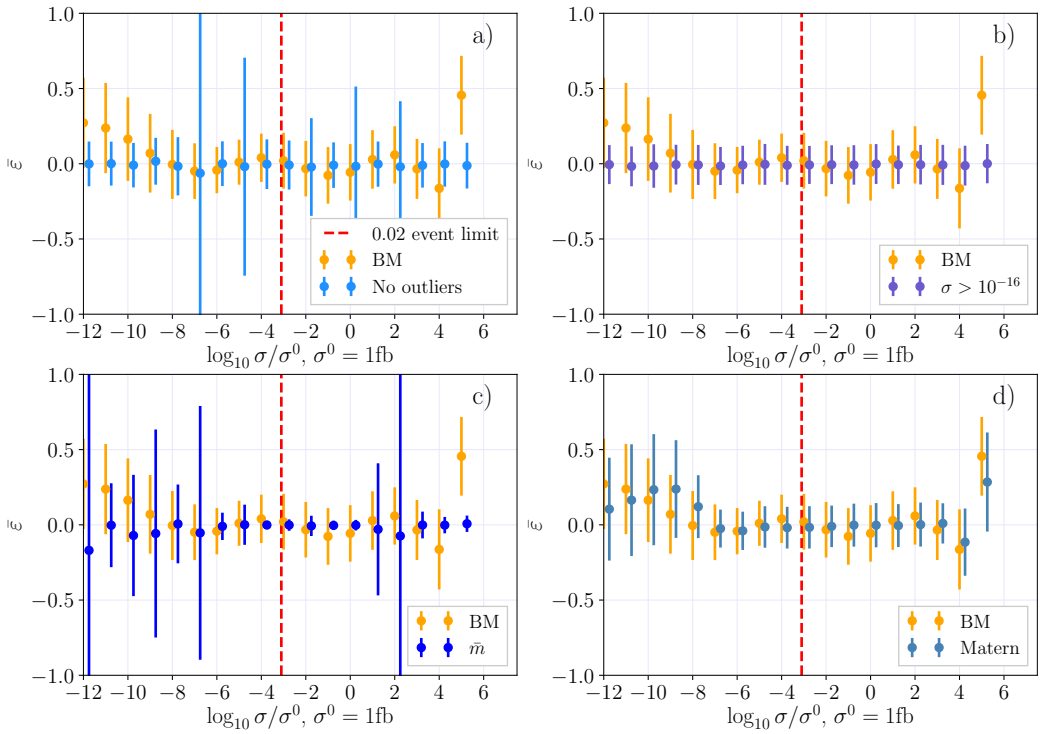


Figure 1.7: The distribution of the relative deviance ε as a function of the logarithm of the normalized cross section for $\tilde{d}_L\tilde{d}_L$ with a) benchmark settings (orange) and removed outliers (blue); b) benchmark settings (orange) and a lower cut on cross sections (blue); c) benchmark settings (orange) and the added feature $\bar{m}$ (blue); d) the benchmark settings (orange) and the Matérn kernel (blue). Benchmark settings are abbreviated BM.

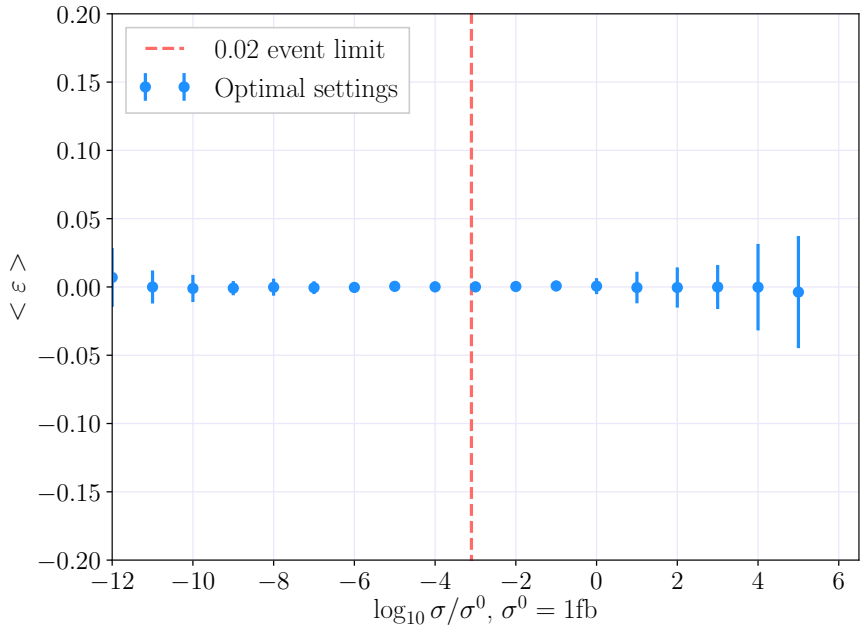


Figure 1.8: The distribution of the relative deviance ε as a function of the logarithm of the normalized cross section for $\tilde{d}_L\tilde{d}_L$ with the optimal settings; a single GP with 2000 training points that uses the Matérn kernel with $\nu = 1.5$. Outliers are removed and a lower cut at $\sigma = 10^{-16}$ fb is introduced. The features are $m_{\tilde{g}}$, $m_{\tilde{d}_L}$ and $\bar{m}$, and the target is $\sigma_{m_{\tilde{g}}}$. All error distibutions have σ_{std} smaller than 5%.

	C	$\ell_{m_{\tilde{g}}}$	$\ell_{m_{\tilde{d}_L}}$	$\ell_{m_{\tilde{u}_L}}$	$\ell_{\bar{m}}$	α
$\tilde{d}_L\tilde{d}_L$	750	30500	17400		74800	$1 \cdot 10^{-5}$
$\tilde{d}_L\tilde{d}_L$	1000	28300	18300		69900	$7.544 \cdot 10^{-7}$ (fixed)
$\tilde{d}_L\tilde{u}_L$	1000	30200	79600	18500	89000	$5.63 \cdot 10^{-6}$
$\tilde{d}_L\tilde{u}_L$	1000	29100	66200	18300	76000	$7.544 \cdot 10^{-7}$ (fixed)

Table 1.5: Kernel parameters for the optimal settings with the noise level estimated by the GP, and given as a constant $\alpha = 7.544 \cdot 10^{-7}$.

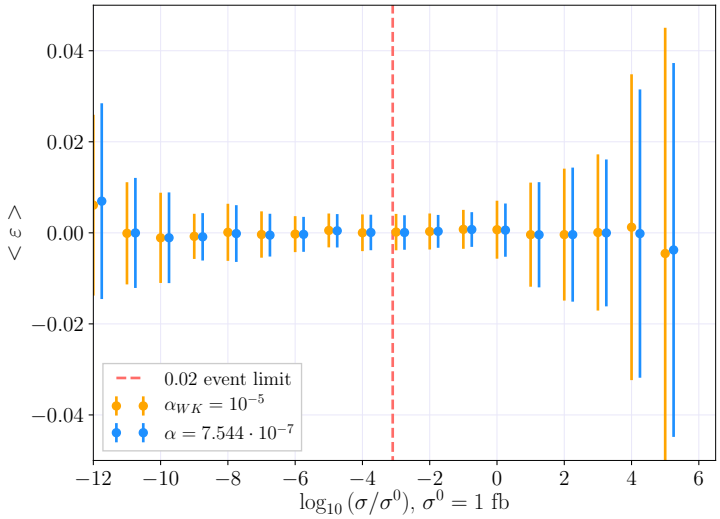


Figure 1.9: The distribution of the relative deviance ε as a function of the logarithm of the normalized cross section for $\tilde{d}_L\tilde{d}_L$ with the optimal settings. In one case the noise level is estimated by the GP (orange), and in the other it is given as a parameter α (blue).

computation time is not affected in any great way, but for larger datasets the removal of α from the kernel may affect the time.

1.4 Distributed Gaussian Processes

Adding experts improves the prediction with very little increase of the computational cost. In Fig. (1.10) a comparison of error 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. The improvement of the prediction with the addition of data is large, along with the stability of predictions. In terms of error distributions there seems to be little difference between using four experts with 2000 training points each and

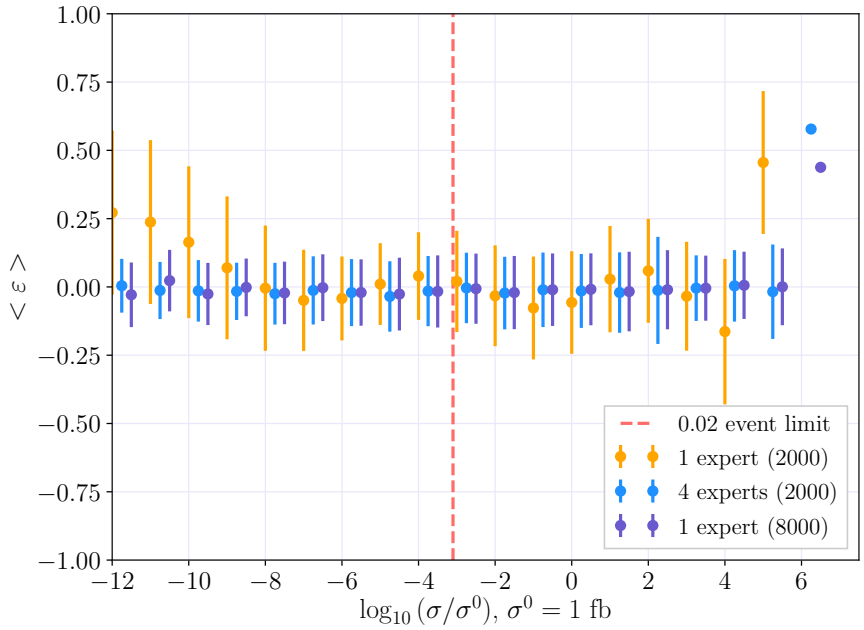


Figure 1.10: The error distributions of GP with the BM settings, using 1 (orange) and 4 (experts).

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

Table 1.6: Table of computation times for GP fit and prediction on Abel.

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 shown in Tab. (1.6).

1.4.1 Cross validation for DGP

There is no `scikit-learn` function for distributed Gaussian processes, so an algorithm for k-fold cross validation as a function of experts was implemented. The algorithm uses the `scikit-learn` function `KFold` to find training and test indices for k splits of the data, and is found in the sub-library `model_selection`. For k folds it is implemented in the following way

```
from sklearn.model_selection import KFold
kf = KFold(n_splits=k, random_state=42)
```

The loss function used in the CV is the R^2-score, introduced earlier, and pseudocode for the algorithm is found in Alg. (1). Learning curves for the optimal settings are found in Fig. (1.11) for $\tilde{d}_L\tilde{d}_L$ and Fig. (1.12) for $\tilde{d}_L\tilde{u}_L$, with 500, 1000 and 2000 points per expert.

Data: $N_{experts}$ (max number of experts), n (training points per expert), X (inputs), $\mathbf{y}$ (targets), k (number of folds for cross validation)
number of experts $\mathbf{n} = [1, ..., N_{experts}]$;
for *each number of experts* i **do**
  Training size $= n \cdot (i + 1)$;
  Total size = training size $\cdot \frac{k}{k-1}$;
  Split training data into subsets;
  Use KFold to create k-fold cross validation instance kf;
  for *training indices, test indices in* kf **do**
    Fit GP to $k - 1$ folds of training data;
    Use 1 fold as test data;
    Predict values $\hat{y}_{train}$ for training data;
    Predict values $\hat{y}_{test}$ for test data;
    $R^2_{\text{train}}(\hat{y}, y) = 1 - \frac{\sum_{i=0}^{\text{Training size}-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{\text{Training size}-1}(y_i - \bar{y})^2}$;
    $R^2_{\text{test}}(\hat{y}, y) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2}$;
  end
  Find mean m and std σ_{std} of R^2_{test}-values and R^2_{train}-values
end
Result: $\mathbf{n}, \mathbf{m}(R^2_{\text{train}})$, $\boldsymbol{\sigma}_{std}(R^2_{\text{train}})$, $\mathbf{m}(R^2_{\text{test}})$, $\boldsymbol{\sigma}_{std}(R^2_{\text{test}})$.

Algorithm 1: Pseudocode for k-fold cross validation of distributed Gaussian processes, which calculates the R^2-scores for training and test data as a fucntion of the number of experts, to be used for *e.g.* learning curves. In the R^2-score calculation, y_i are true values, $\hat{y}_i$ are GP predicted values, and $\bar{y}$ is the mean of all y_i.

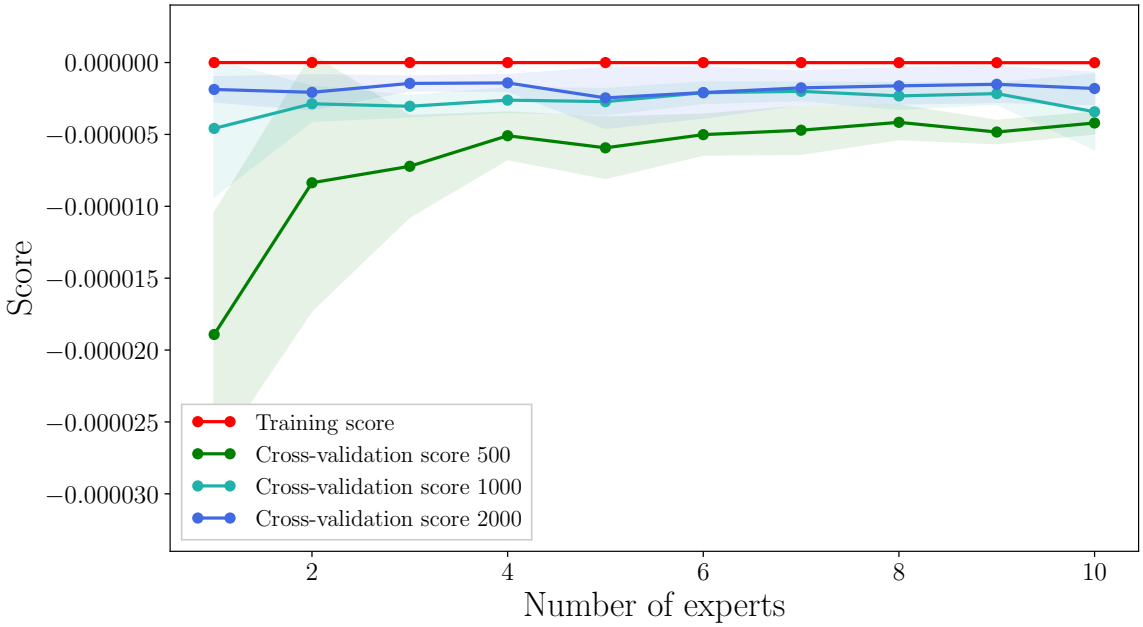


Figure 1.11: Learning curves as a function of number of experts, with 500 and 1000 training points per expert for $\tilde{d}_L\tilde{d}_L$. k-fold cross validation uses R^2-score, but here $R^2 - 1$ is plotted.

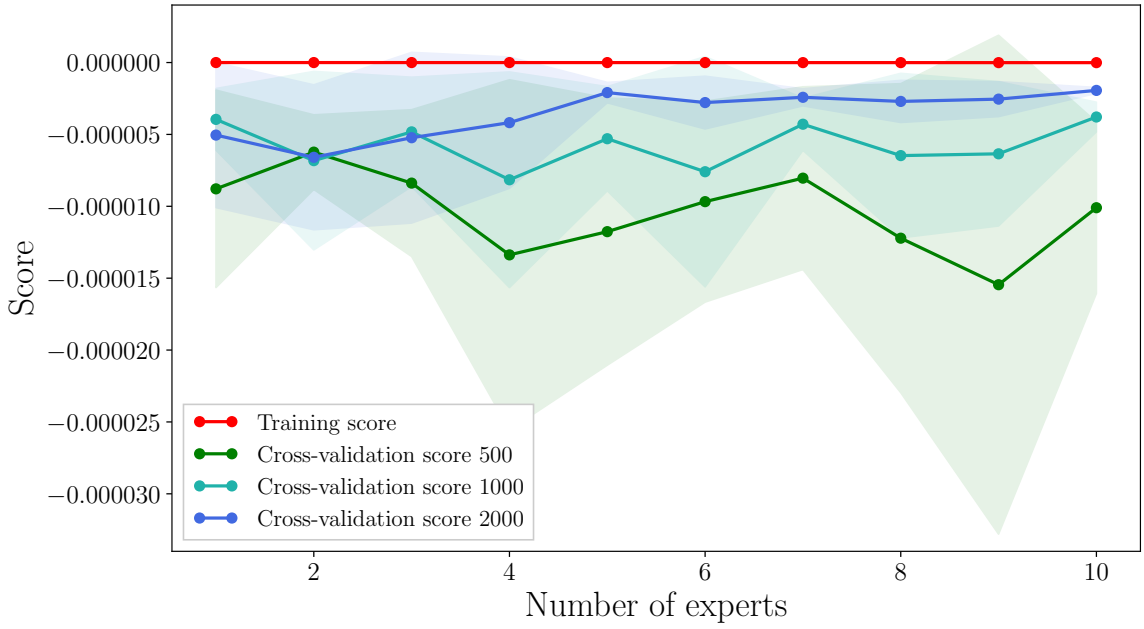


Figure 1.12: Learning curves as a function of number of experts, with 500 and 1000 training points per expert for $\tilde{d}_L\tilde{u}_L$. k-fold cross validation uses R^2-score, but here $R^2 - 1$ is plotted.

Bibliography

[1] B.C. Allanach. Softsusy: A program for calculating supersymmetric spectra. *Computer Physics Communications*, 143(3):305 – 331, 2002.

[2] Wim Beenakker, R Höpker, Michael Spira, and PM Zerwas. Squark and gluino production at hadron colliders. *Nuclear Physics B*, 492(1-2):51–103, 1997.

[3] Anders Kvellestad. Chasing susy through parameter space. 2015.

[4] Jon Vegard Sparre. Fast evaluation of supersymmetric cross sections. 2015.