Prepared for submission to JHEP

Fast supersymmetry phenomenology at the Large Hadron Collider using machine learning techniques

A. Buckley,[a] **A. Shilton,**[b] **and M. J. White**[c]

[a] *University of Edinburgh, School of Physics & Astronomy, Mayfield Rd, Edinburgh, EH9 3JZ, UK*
[b] *University of Melbourne, School of Engineering, Department of Electrical and Electronic Engineering, VIC 3010, Australia*
[c] *University of Melbourne, School of Physics, VIC 3010, Australia*

E-mail: mwhi@unimelb.edu.au

Abstract: A pressing problem for supersymmetry (SUSY) phenomenologists is how to incorporate Large Hadron Collider search results into parameter fits designed to measure or constrain the SUSY parameters. Owing to the computational expense of fully simulating lots of points in a generic SUSY space to aid the calculation of the likelihoods, the limits published by experimental collaborations are frequently interpreted in slices of reduced parameter spaces. For example, both ATLAS and CMS have presented results in the Constrained Minimal Supersymmetric Model (CMSSM) by fixing two of four parameters, and generating a coarse grid in the remaining two. We demonstrate that by generating a grid in the full space of the CMSSM, one can interpolate between the output of an LHC detector simulation using machine learning techniques, thus obtaining a superfast likelihood calculator for LHC-based SUSY parameter fits. We further investigate how much training data is required to obtain usable results, finding that approximately 2000 points are required in the CMSSM to get likelihood predictions to an accuracy of a few per cent. The techniques presented here provide a general approach for adding LHC event rate data to SUSY fitting algorithms, and can easily be used to explore other candidate physics models.

Contents

1 Introduction

The ATLAS and CMS experiments [1, 2] at the Large Hadron Collider at CERN, Geneva, have recently published their first results relating to the search for new theories of particle physics. Of the variety of theories developed to explain the deficiencies of the Standard Model, supersymmetry (SUSY) is a leading contender. The absence of evidence for supersymmetric particle production in the first LHC data can be used to derive limits on the parameters of candidate supersymmetric theories [3–13]. This in turn requires one to calculate the number of expected signal events at different points in the SUSY parameter space, before assigning a suitable likelihood based on the number of observed events and the Standard Model background expectation.

To rigorously estimate the number of expected signal events for a point in a SUSY parameter space, one must evaluate the cross-section and then perform a Monte Carlo simulation to obtain a realistic sample of 'reconstructed' candidate events, with correctly modelled acceptance and resolution effects. The time for such a simulation ranges from several minutes for a fast, approximate simulation code to hours for the full GEANT4-based simulations employed by the experimental collaborations. A next-to-leading order evaluation of the total SUSY production cross-section might take a further half an hour or so. Clearly neither of these approaches can be used directly in a fit to supersymmetric parameters where one often requires millions of likelihood evaluations in order to obtain a reasonable result. The ATLAS and CMS collaborations therefore interpret their results in restricted model spaces such as the Constrained Minimal Supersymmetric Model (CMSSM), fixing all parameters except the unified scalar and gaugino mass parameters m_0 and $m_{1/2}$ (the remaining parameters being the trilinear coupling term A_0, the Higgs VEV ratio $\tan\beta$ and the sign of the Higgs mass parameter μ). One can then fully simulate (i.e., simulate both the fully hadronised event and the detector interaction/reconstruction effects) a grid of points in parallel upon which to set an exclusion limit. This limit can be interpreted by phenomenologists, but is only really useful if the number of events in the search channels under study is only weakly dependent on A_0 and $\tan\beta$, whence one can use the measurement in that channel to make more general inferences [1].

This paper will suggest a more general solution to the problem, using as a test case the full 4 parameters of the CMSSM (we have continued to assume $\mu > 0$). If one could successfully interpolate between the simulation output values in the full space of the CMSSM (or any other model space), one would obtain a function that could be evaluated in fractions of a second to give the expected number of signal events in a given search channel, and this could subsequently be used for very fast likelihood calculations. We demonstrate that this is possible using either a Bayesian neural net or a support vector machine as a regressor (in an approach borrowed from the cosmology literature [14]), with training data supplied by a fast, public LHC simulation code of the kind routinely used in the phenomenology literature. Our results could therefore already be used to do LHC parameter fit studies in this context, whilst even the fast simulation would still have proved prohibitively expensive to use directly in a non-parallelised fitting routine that relied on, say, Markov Chain Monte Carlo techniques. We will also argue that this approach to adding LHC event rate data to SUSY parameter fitting algorithms solves many of the problems associated with previous attempts found in the literature, such as in References [15] and [16]. If the LHC sees direct evidence of sparticle production in the near future, one could use an interpolation of inclusive variables to add extra information to SUSY parameter fits alongside other features such as kinematic endpoint information.

In addition, we investigate whether it is feasible for experimental collaborations to try a similar technique on the output of their full detector simulations, thus enabling them to provide a suitable likelihood function for phenomenologists after generating their model grids in the full CMSSM rather than in a 2D parameter slice. Although our work is carried out in the framework of the CMSSM, there should be no barrier to replicating these techniques in other models, though models with more parameters would inevitably require larger training data sets.

[1]Of course, it is also useful in the case that nature has chosen the same values of A_0 and $\tan\beta$ as those found in the ATLAS and CMS papers, but we consider this unlikely.

The paper is structured as follows. In Section 2 we briefly review the machine learning techniques used in our analysis, covering both the Bayesian training of neural networks, and the use of support vector machines. Readers who do not need an introduction to these topics may safely skip this section. In Section 3 we give details of our training data simulation, and demonstrate the use of a neural net to obtain a fast CMSSM event rate calculator using the recent ATLAS zero lepton search as an example. We repeat the study using a support vector machine, and show the optimum results obtained for both algorithms when we did not restrict the size of our training data sample. Section 4 shows the performance for differing amounts of training data in order to examine the feasibility of using similar techniques with slower detector simulations than those utilised here. A detailed discussion of the implications of our results is deferred to Section 5, where we briefly compare our work to previous attempts at the problem, clearly state the current limitations, and share further insight for those wishing to try similar techniques. Finally, we present conclusions in Section 6.

2 Machine learning techniques for regression

The problem we address in this paper can be stated as follows. Assume that we have calculated the number of signal events expected in an LHC detector for n points in the CMSSM space, each defined by a given choice of m_0, $m_{1/2}$, A_0 and $\tan\beta$ (we have fixed μ to be positive, but could repeat our procedure for negative μ). Is it possible to obtain an interpolating function for the output values, such that, for any new points that are not in our simulated sample, we can get the number of events expected in an LHC detector?

This is a standard regression problem, and there are a number of documented methods for addressing it. A popular choice is a Multilayer Perceptron Network, which can be shown to be able to approximate any smooth function if the network has sufficient complexity. Clearly the success in our example will depend on whether the target distribution of events in the CMSSM parameter space is indeed smooth. The quality of our results must serve to demonstrate whether this is true or not. A popular alternative to Multilayer Perceptrons is Support Vector Regression, which applies the principles of structural risk minimization to achieve a trade-off between empirical risk minimisation and over-fitting. We now present a brief introduction to both techniques.

2.1 Multilayer Perceptron Networks

2.1.1 Overview

The inspiration for the Multilayer Perceptron Network (MLP) is the architecture of animal brains. The networks consist of several 'neurons', each of which sums a series of weighted scalar inputs, and produces an output determined by an 'activation function' which is usually chosen to be either a step function or a non-linear function that varies between -1 and 1. The MLP network then consists of a series of linked neurons, as illustrated in Figure 1, and is an example of a 'feed-forward' network in which information is passed through from inputs to outputs. For a detailed guide to neural networks and the Bayesian approach to network training, we refer the reader to [17]. For a convenient, short exposition with an example of application in a completely different context see [18].

In our case, the inputs are values of m_0, $m_{1/2}$, A_0 and $\tan\beta$, represented hereafter by the vector of input parameters $\mathbf{p}$. The LHC target variable is the cross-section of events in a given search channel $y(\mathbf{p})$ (we will assume that the process may be repeated for different search channels, and hence only have one output variable per net). The outputs of the output node and hidden nodes in the network might be calculated as follows:

$$f(\mathbf{p}, \mathbf{w}) = b + \Sigma_j v_j h_j(x) \tag{2.1}$$

$$h_j(x) = \tanh(a_j + \Sigma_i u_{ij} x_i) \tag{2.2}$$

where $f(\mathbf{p})$ gives the output of the output node (which in an ideal world would equal $y(\mathbf{p})$), u_{ij} is the weight on the connection from input unit i to hidden unit j, v_j is the weight on the connection from hidden unit j to the output unit, and b and a_j are bias terms for the output and hidden nodes respectively. $\mathbf{w}$ is used to label the entire set of network parameters for later convenience. Here we see that the output value is simply a weighted sum of the hidden unit values plus a bias term, whilst the hidden units put the weighted sum of their input values plus bias through a non-linear activation function. It is the non-linear nature of this activation function that allows the network to model complex distributions, and the tuning of the weights essentially controls the degree of complexity of the model.

To solve the problem of this paper, then, one needs to define a network architecture and find the values of the weights and biases of the neurons such that the network reproduces the correct value of the LHC target variable when fed the corresponding input data. This is accomplished by 'training' the network with a set of training data $\{\mathbf{p}^{(i)}, y^{(i)}\}$. Several approaches to this problem exist, such as adjusting the weights and biases to minimise the sum of the squared differences between the network outputs $f(\mathbf{p})^{(i)}$ and the correct values $y^{(i)}$. The function we wish to minimise may have many discrete minima, and hence ensuring that a correct solution is found is non-trivial.

2.1.2 Bayesian approach to neural network training

A particularly powerful approach to neural net training uses the methods of Bayesian inference. We first use our MLP network to define a probabilistic model for our regression problem as follows. Regression with a real valued target can be modelled by assuming that the target value is described by the output of our network, $f(\mathbf{p}, \mathbf{w})$, subject to corruption from Gaussian noise of constant standard deviation σ. The probability density for the target is then:

$$p(y|\mathbf{p}, \mathbf{w}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y - f(\mathbf{p}, \mathbf{w}))^2}{2\sigma^2}) \tag{2.3}$$

Let us call the full set of model parameters θ, where $\theta \equiv \mathbf{w}, \sigma$ encompassing both the network parameters and the noise. Our prior knowledge of these parameters before any data have been observed can be represented by a prior distribution $p(\theta)$, and Bayes'

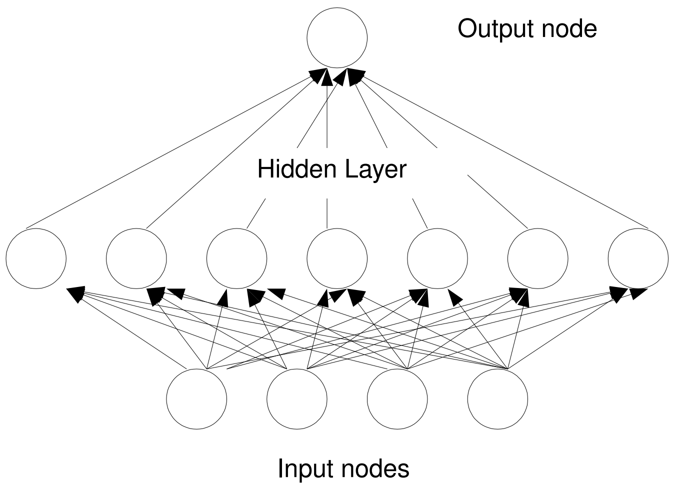


Figure 1: Schematic diagram of a Multilayer Perceptron Network.

theorem then tells us how to update the prior distribution to a posterior distribution after the training data $D = \{\mathbf{p}^{(i)}, y^{(i)}\}$ have been observed:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \propto L(\theta|D)p(\theta) \tag{2.4}$$

Assuming that our training data points are independent, the likelihood $L(\theta|D)$ in this equation is given by a product of terms of the form of equation 2.3, with the product running over all training points:

$$L(\theta|D) = \prod_{i=1}^{n} p(y^{(i)}|\mathbf{p}^{(i)}, \theta) \tag{2.5}$$

If we now wish to obtain a new output $y^{(n+1)}$ from an input vector $\mathbf{p}^{(n+1)}$ that is not in the training sample D, we must integrate the predictions of the model with respect to the posterior of the model parameters:

$$p(y^{(n+1)}|\mathbf{p}^{(n+1)}, D) = \int p(y^{(n+1)}|\mathbf{p}^{(n+1)}, \theta)p(\theta|D)\mathrm{d}\theta \tag{2.6}$$

This defines the predictive distribution for $y^{(n+1)}$. If we want to estimate the value of this distribution that minimises the squared error loss, we can take the mean of this distribution:

$$\hat{y}^{(n+1)} = \int f(\mathbf{p}^{(n+1)}, \theta)p(\theta|D)\mathrm{d}\theta \tag{2.7}$$

2.1.3 Implementation

One can easily pick networks from the prior distribution on the network parameters as defined in our probabilistic model. The difficulty in the Bayesian approach to network training lies in evaluating the integral in equation 2.7 to get our output value estimate. Since it is the expectation value of $f(\mathbf{p}^{(n+1)}, \theta)$ with respect to the posterior distribution of the parameters, however, one can use a Monte Carlo method using a sample of values $\theta^{(t)}$ drawn from the posterior distribution of parameters:

$$\hat{y}^{(n+1)} \approx \frac{1}{N} \sum_{t=1}^{n} f(\mathbf{p}^{(n+1)}, \theta^{(t)}) \tag{2.8}$$

For our analysis, we use the Flexible Bayesian Modelling software (FBM) by Radford Neal [2], which uses a combination of Markov Chain Monte Carlo (MCMC) methods to obtain these posterior samples and to evaluate equation 2.7. It assumes a Gaussian prior on the network weights, but allows the width of the Gaussian to be specified by a hyperparameter that is in turn specified by a vague prior. Hence, one essentially requires no intuitive knowledge of a suitable prior for the network parameters, but can instead let the data tune the complexity of the model. A vague prior is also used for the noise parameter σ. Neal uses the Hybrid Monte Carlo algorithm to explore the posterior of the network parameters, and periodically updates the hyperparameters using Gibbs sampling. This minimises the amount of tuning required to obtain good performance with the Hybrid Monte Carlo method.

2.2 Support Vector Regression

Support vector machines (SVMs) [19–21] are a popular alternative to multi-layer perceptron type networks for both classification and regression problems [22–26]. Support vector regressors [27–29] apply the principles of structural risk minimisation [28] to achieve a trade-off between empirical risk (training set error) minimisation (ERM) and regularisation to avoid the problem of over-fitting. They also have relatively fewer parameters to select in comparison with the MLP and do not suffer from the problem of local minima.

The support vector regression approach seeks to approximate the relation between input parameters $\mathbf{p} \in \mathbb{R}^{d_L}$ (d_L being the number of parameters in the input) and targets y using a trained machine of the form:

$$y(\mathbf{p}) = \langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{p}) \rangle + b$$

where $\boldsymbol{\varphi} : \mathbb{R}^{d_L} \to \mathbb{R}^{d_H}$ is the feature map into a d_H-dimensional *feature space* given a-priori, $\mathbf{w} \in \mathbb{R}^{d_H}$ the weight vector and $b \in \mathbb{R}$ the bias. Given a training set of N pairs $\left(\mathbf{p}^{(i)}, y^{(i)}\right)$ the weight vector $\mathbf{w}$ and bias b are chosen to solve the primal training problem:

$$\begin{aligned} &\min_{\mathbf{w},b} R(\mathbf{w}, b) = \tfrac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle + \tfrac{C}{N} \sum_{i \in \mathbb{Z}_N} \xi_i \\ &\text{such that: } \left\langle \mathbf{w}, \boldsymbol{\varphi}\left(\mathbf{p}^{(i)}\right) \right\rangle + b \le y^{(i)} + \epsilon + \xi_i \ \forall i \in \mathbb{Z}_N \\ &\qquad\qquad\quad \left\langle \mathbf{w}, \boldsymbol{\varphi}\left(\mathbf{p}^{(i)}\right) \right\rangle + b \ge y^{(i)} - \epsilon - \xi_i \ \forall i \in \mathbb{Z}_N \\ &\qquad\qquad\quad \xi_i \ge 0 \ \forall i \in \mathbb{Z}_N \end{aligned} \tag{2.9}$$

where the second term in R, via the slack variables ξ_i and together with the constraints, provides a measure of the training set error or empirical risk and the first term is a regularisation included to minimise over-fitting. The training parameter $C \in \mathbb{R}^+$ controls the trade-off between these dual objectives, while $\epsilon \in \mathbb{R}^+$ controls noise insensitivity.

[2]URL: `http://www.cs.toronto.edu/~radford/fbm.software.html`

In practice the primal training problem of equation 2.9 is rarely solved directly. Instead the Wolfe-dual of equation 2.9:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} Q(\mathbf{w}, b) = \frac{1}{2} \sum_{i,j \in \mathbb{Z}_N} K_{ij} \alpha_i \alpha_j - \sum_{i \in \mathbb{Z}_N} \alpha_i y^{(i)} + \epsilon \sum_{i \in \mathbb{Z}_N} |\alpha_i| \\ &\text{such that: } -\frac{C}{N} \leq \alpha_i \leq \frac{C}{N} \ \forall i \in \mathbb{Z}_N \\ &\qquad \sum_{i \in \mathbb{Z}_N} \alpha_i = 0 \end{aligned} \tag{2.10}$$

is solved [29], where $K_{ij} = K\left(\mathbf{p}^{(i)}, \mathbf{p}^{(j)}\right)$, $K(\mathbf{p}, \mathbf{q}) = \langle \boldsymbol{\varphi}(\mathbf{p}), \boldsymbol{\varphi}(\mathbf{q}) \rangle$ is the kernel function, and each dual variable α_i, $i \in \mathbb{Z}_N$, corresponds to training pair $\left(\mathbf{p}^{(i)}, y^{(i)}\right)$. Note that the dual in equation 2.10 is a convex quadratic programming problem with a positive semi-definite Hessian $\mathbf{K} = [K_{ij}]$, and so has no non-global minima. The trained machine may also be written in terms of the dual variables α_i:

$$y(\mathbf{p}) = \sum_{i \in \mathbb{Z}_N} \alpha_i K\left(\mathbf{p}^{(i)}, \mathbf{p}\right) + b$$

The advantage of the dual form over the primal form is that any function $K : \mathbb{R}^{d_L} \times \mathbb{R}^{d_L} \to \mathbb{R}$ satisfying Mercer's condition [30, 31] may be used directly without explicit knowledge of the feature map $\boldsymbol{\varphi} : \mathbb{R}^{d_L} \to \mathbb{R}^{d_H}$ encompassed therein, allowing $d_H \gg d_L$ without added complexity. This is known as the kernel trick. Some popular kernel functions include [32, 33]:

1. Linear kernel: $K(\mathbf{p}, \mathbf{q}) = \langle \mathbf{p}, \mathbf{q} \rangle$.
2. Polynomial kernel: $K(\mathbf{p}, \mathbf{q}) = (1 + \langle \mathbf{p}, \mathbf{q} \rangle)^d$ (where $d \in \mathbb{Z}^+$).
3. RBF kernel: $K(\mathbf{p}, \mathbf{q}) = \exp\left(-\frac{1}{\sigma} \|\mathbf{p} - \mathbf{q}\|^2\right)$ (where $\sigma \in \mathbb{R}^+$). item Sigmoid kernel: $K(\mathbf{p}, \mathbf{q}) = \tanh(\kappa \langle \mathbf{p}, \mathbf{q} \rangle + \beta)$ (where $\kappa, \beta \in \mathbb{R}^+$).

For simulation purposes polynomial and RBF kernels have been used. Simulations have been carried out using SVMHeavy.[3]

3 ATLAS signatures in the CMSSM

3.1 Signal region and simulation details

In order to demonstrate the utility of machine learning methods in interpolating LHC simulation output, we will develop one example derived from the recent SUSY search results published by the ATLAS experiment. The most constraining search published at the time of writing is that performed in the zero lepton channel, which used four different signal regions to search for an excess of events over the Standard Model background. We will take as a test case 'signal region D', defined by the cuts given in Table 1, where $\Delta\phi(\text{jet}, p_T^{\text{miss}})_{\text{min}}$ is the smallest of the azimuthal separations of p_T^{miss} and the hardest jets with $p_T > 40$ GeV (up to a maximum of three) and m_{eff} is defined as the sum of E_T^{miss} and the magnitudes of the transverse momenta of the three hardest jets. This is sufficient to prove the principle of machine-learning based interpolation, and we leave a detailed examination of all signal regions (including those yet to be published) for future work.

[3]URL: http://people.eng.unimelb.edu.au/shiltona/svm/index.html

Signal region	D
Number of jets	≥ 3
Leading jet p_T[GeV]	> 120
Other jet(s) p_T[GeV]	> 40
$E_T^{\rm miss}$[GeV]	> 100
$\Delta\phi(\text{jet}, p_T^{\rm miss})_{\rm min}$	> 0.4
$E_T^{\rm miss}/m_{\rm eff}$	> 0.25
$m_{\rm eff}$[GeV]	> 1000

Table 1: Event selection cuts used in the ATLAS experiment zero lepton search for supersymmetric particles [5]. See text for variable definitions.

Training data for our support vector machine and neural net runs was generated by picking points at random from a specified distribution in the CMSSM parameter space then, for each point, running `Isajet 7.75` [34] to generate the sparticle mass and decay spectra, followed by `HERWIG 6.505` [35–37] with `AcerDet` [38] to generate a sample of 50,000 events. We then apply the selection cuts given in Table 1 and store the cross-section of events entering signal region D, $\sigma^{(\rm D)}$ (this is equivalent to storing the number of events observed in the search channel once the integrated luminosity is specified). The number of events generated and simulated per point must reduce the statistical error on $\sigma^{(\rm D)}$ to an acceptable level: 50,000 events is more than enough to negate the effect of statistical fluctuations in the training data on the quality of the interpolation results.

In choosing a range for the CMSSM parameters, we took care to ensure that our choice was reasonably large but, nevertheless, that it covered most of the interesting region for early LHC data. Our final choice is given in Table 2. Generating data for 20,000 of these training points took approximately 1.5 days on the University of Melbourne Tier 2 cluster. It would take considerably longer with a full detector simulation, and thus there is a clear need to investigate how big the training data set needs to be to obtain reasonable results. This is considered in the next section.

A histogram of the output values obtained for ATLAS Region D is given in Figure 2, for CMSSM parameters taken from a flat distribution in the range specified in Table 2. One can see that the true cross-section has a tail that is underpopulated by generating training data in this fashion. One can fix this is a variety of ways, of which the simplest options are:

1. Generate extra data in the tail of the distribution and train two nets to cover both regions. For CMSSM data, the points in the tail span the entire range of m_0, A_0 and $\tan\beta$, but have $m_{1/2} \lesssim 500$. We therefore tried two training sets of equal size: one in the full mass range, and one with $m_{1/2} < 500$.

2. Generate training data from a distribution chosen so as to maximise the number of points in the tail. We did this by picking m_0 and $m_{1/2}$ values at random from the

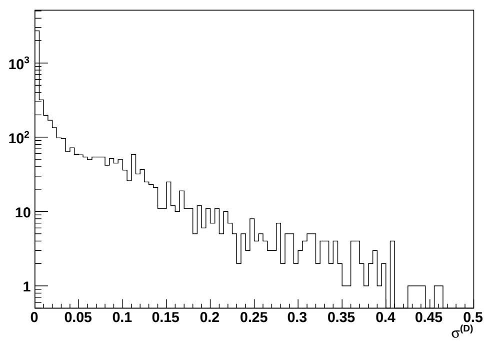


Figure 2: Distribution of the cross-section of events passing the selection cuts for region D (at leading order), for 5000 training points of 50,000 events each.

distribution $f(x;\lambda) = e^{-\frac{x}{\lambda}}/\lambda$, with $\lambda = 200$, whilst retaining a flat distribution in A_0 and $\tan\beta$.

The two methods gave very similar results and hence we only show results for the second case. The ability to train one net instead of two, with a reduced total training set size makes the second option strongly preferred.

In addition to our training data set, we generated an independent test set of 5,000 events in order to present our final results, and to easily compare the support vector machine and Bayesian neural net methods. The support vector machine requires a training and test sample at the training stage, whilst the Bayesian neural net only requires a training sample. By using a final test set that is completely independent of any of this data, we ensure that we get a fair comparison between our two methods. It would not be necessary to generate such a large test set when applying the interpolation method in practice.

3.2 Interpolation results

3.2.1 Results obtained using a Bayesian neural net

We trained an MLP network with 2 hidden layers of 50 neurons each, and assessed convergence of the FBM software by looking at the variation of the squared error on the training sample vs iteration number, as well as the variations in the net weights and biases. Convergence was deemed to have set in once the squared error on the training set stabilised, and once the weights commenced a stable, periodic variation around a fixed value. In general, nets with less training data required more iterations before convergence was observed, with ≈ 1300 iterations required for 5000 training points.

Figure 3a shows our predicted cross-section of events passing the ATLAS Region D selection cuts ($\sigma^{(\mathrm{D})}_{\mathrm{pred}}$) vs the true value from the full Monte Carlo simulation and reconstruction ($\sigma^{(\mathrm{D})}_{\mathrm{true}}$) as evaluated on our independent test sample, using a net trained with 5000 training points. The results demonstrate excellent agreement for low to moderate values of the true cross-section, with the performance degrading towards higher values. Here it is important to note that the error is still reasonable even in the tail, and also that

Parameter	Minimum	Maximum
$m_{1/2}$[GeV]	50	1200
m_0[GeV]	50	1200
A_0[GeV]	-1000	1000
$\tan\beta$	2	60

Table 2: Range used for the CMSSM parameters, in which we work to obtain an optimum interpolation between the output values of the LHC simulation.

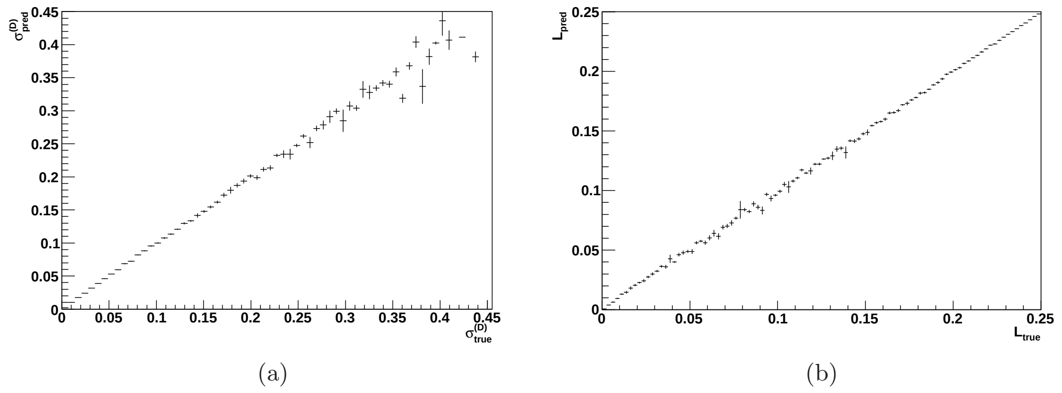


Figure 3: The true cross-section of events passing the ATLAS region D selection cuts (left), and the true Poisson likelihood of points in the CMSSM space (right) shown for the original simulation output vs. the prediction obtained using a Bayesian neural net trained with 5000 simulated data points and evaluated on an independent test sample of 5000 points.

this is the region in which one can tolerate a larger error. Since ATLAS has not seen a significant excess over the Standard Model cross-section, points with relatively large signal cross-sections are already very unlikely, and will not figure prominently in a parameter fit. To demonstrate this, one can evaluate the Poisson likelihood for observing o events with a signal(background) expectation of $s(b)$:

$$L = \frac{e^{-(s+b)}(s+b)^o}{o!} \tag{3.1}$$

Figure 3b shows this likelihood for the values of o and b published in Reference [5], where we neglect the effects of statistical and systematic uncertainties for this illustration. Figure 3b demonstrates that our net output reproduces the ATLAS likelihood within a few percent over the entire range of interest, thus supporting our conclusion that it is not essential to model the entire tail of the signal cross-section distribution to per cent accuracy in order to obtain excellent final results. Indeed, the selection of training data is partly an exercise in using physical intuition to decide where in the CMSSM to expend most effort on detailed simulation.

To further illustrate that our neural net output adequately reproduces the behaviour of the original simulation, we show in Figure 4 the 95% exclusion contour in the CMSSM for a grid of points similar to the ATLAS grid used in Reference [5]. This features a coarse scan over m_0 and $m_{1/2}$, with the other parameters fixed at $A_0 = 0$ and $\tan\beta = 3$. For each point in the grid, we assign a Poisson likelihood as above, then assign each point a value of $\Delta\chi^2 = -2\ln L/L_{\mathrm{max}}$, where L_{max} is the maximum value of the likelihood obtained in the scan. We then smooth a 2D histogram of these values and plot the contour $\Delta\chi^2 = 5.99$, corresponding to the 95% exclusion contour.

The left-hand plot of Figure 4 shows the original simulation output for a grid of points in which m_0 and $m_{1/2}$ are chosen to reproduce the published ATLAS values. Each point in this plot takes approximately 30 minutes to process, this being dominated by the time taken to perform the event generation and detector simulation. The right-hand plot is generated using our neural net trained using 5000 data points. We first fix A_0 and $\tan\beta$ to the ATLAS values and then choose 5000 random points in the $m_0, m_{1/2}$ plane and evaluate the net output. Each of these takes a fraction of a second to generate. The result is a smoother limit, but one that nevertheless agrees very well with the original simulation. The bumpy sections of the left-hand plot are likely to have arisen from the poor performance of the smoothing procedure performed on the coarse grid points. The advantage of using the neural net output is that one can obtain a much finer exploration of the contour (in addition to the fact that one can explore the full space of the CMSSM rather than simply a parameter slice as in this validation example).

Comparison of Figure 4 with the original ATLAS contour in Reference [5] shows that our limit is slightly weaker. This can be understood by the fact that we have not included next-to-leading order effects in the SUSY production cross-section, which would tend to increase the yield of events per point and thus shift the limit to higher values. These could be included by weighting each of our training data points by the relevant k-factor. We have also not included the effect of detector systematic errors, although these could be applied in a later study by comparing our simulation output to the published ATLAS results, as was recently performed in [13]. These do not affect our proof of principle so we defer them to a future study. It is, however, worth noting that the effect of neglecting k-factors is to make our limit more conservative than the ATLAS contour.

3.2.2 Results obtained using a Support Vector Machine

The support vector regressor was trained using two kernel functions, namely the polynomial kernel and the RBF kernel. The following three-stage process was used:

1. Data pre-processing: training data was normalised to zero mean, unit variance on a per-parameter basis using $\mathbf{p}^{(i)} := (\mathrm{diag}\,(\Delta\mathbf{p}))^{-1}\left(\mathbf{p}^{(i)} - \bar{\mathbf{p}}\right)$, $y^{(i)} := (\Delta y)^{-1}\left(y^{(i)} - \bar{y}\right)$ for all $i \in \mathbb{Z}_N$, where $\bar{y}$, $\bar{\mathbf{p}}$ and Δy, $\Delta\mathbf{p}$ are the means and component-wise standard deviations, respectively, of the training targets $y^{(i)}$ and input parameter vectors $\mathbf{p}^{(i)}$.

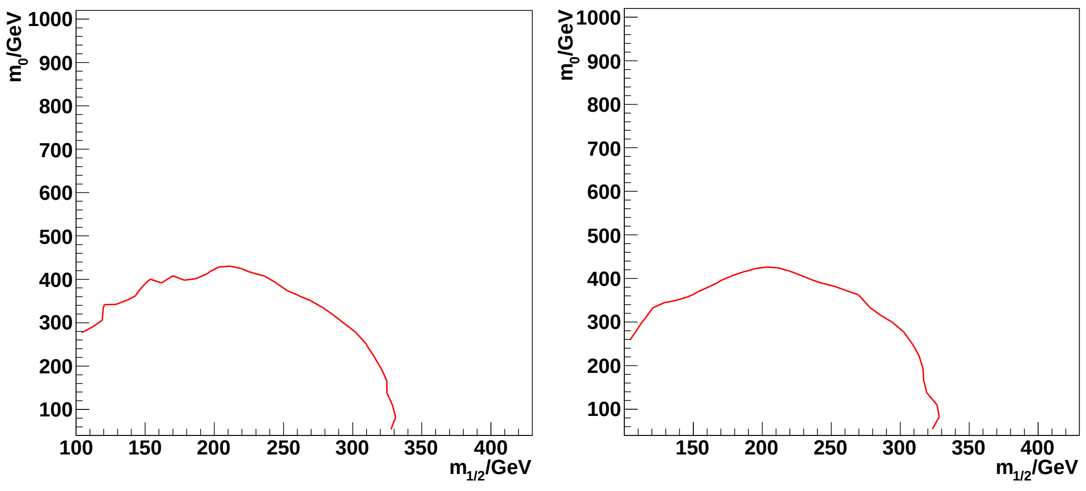


Figure 4: The 95% exclusion contour in the $m_0, m_{1/2}$ plane of the CMSSM for $A_0 = 0$ and $\tan\beta = 3$ as obtained from a grid of `AcerDet`-simulated points (left) and from our neural network output (right). The neural network was trained on 5000 data points, and the exclusion contour was generated by evaluating the network output for 5000 points with randomly chosen m_0 and $m_{1/2}$ values.

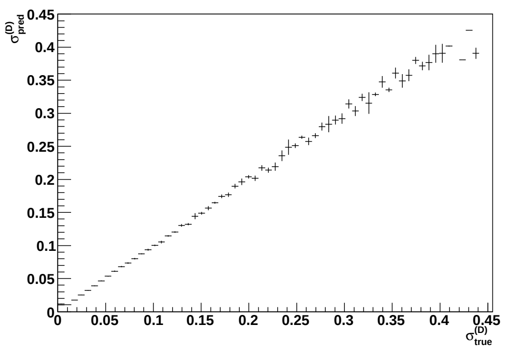
2. Parameter selection: the training set was split into two subsets Θ_T (training) and Θ_E (evaluation) containing, respectively, 2/3 and 1/3 of the total training set. For all combinations of regression parameters $C/N \in \{0.1, 0.5, 1, 5, 10, 50, 100\}$, $K(\mathbf{p}, \mathbf{q}) \in K_p \cup K_R$:

$$\begin{aligned} K_p &= \left\{ \left(1 + \mathbf{p}^T\mathbf{q}\right)^d \middle| \, d \in \{1, 2, 3\} \right\} \\ K_R &= \left\{ \exp\left(-\tfrac{1}{\sigma} \|\mathbf{p} - \mathbf{q}\|^2\right) \middle| \, \sigma \in \{0.1, 0.5, 1, 5, 10, 50, 100\} \right\} \end{aligned}$$

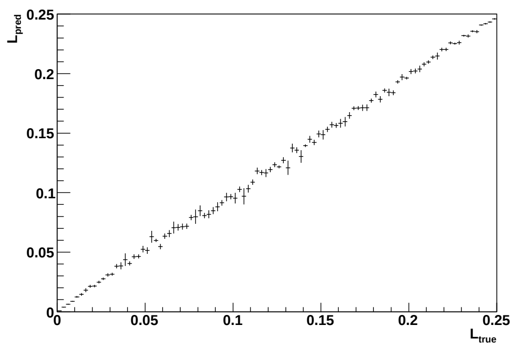
the support vector regressor was trained using Θ_T and its performance evaluated using the mean-squared prediction error on Θ_E to find the regression parameters C/N and K which minimised the mean-squared error.

3. Validation: the support vector regressor was trained using the complete training set and the support vector regression parameters chosen in step 2. The trained regressor was then evaluated on an independent test sample of 5000 points.

Using this process we found regressor parameters $K(\mathbf{p}, \mathbf{q}) = \exp(-\|\mathbf{p} - \mathbf{q}\|^2)$ and $C/N = 5$ were optimal for a training set of 5000 simulated data points. Figure 5a shows the true cross-section of events in the ATLAS region D search channel vs the support vector machine output. Comparison with Figure 3a demonstrates that the results are very similar, though the support vector machine output is better at predicting the value from the original simulation for higher cross-sections. The effect on the Poisson likelihood can be



(a)



(b)

Figure 5: The true cross-section of events passing the ATLAS region D selection cuts (left), and the true Poisson likelihood of points in the CMSSM space (right) shown for the original simulation output vs. the prediction obtained using a support vector machine trained with 5000 simulated data points and evaluated on an independent test sample of 5000 points.

seen in Figure 5b. The advantage of the support vector machine is reduced in the likelihood since the better performance at high cross-sections will translate to only a small change in already unlikely parameter values. The SVM output looks worse at higher likelihoods, but we have checked that increasing the training set further leads to results that are on a par with the Bayesian neural net results. The obvious conclusion is that the Bayesian neural net makes a more efficient use of the training data.

4 Variation of performance with training data

Figure 6 shows the true likelihood value vs the Bayesian neural net prediction for different numbers of training points, evaluated on an independent test sample of 5000 points. The performance clearly degrades for low numbers of points, and gradually approaches the optimum performance obtained with 5000 training points (Figure 3b). The number of training points required to reduce the accuracy of the likelihood prediction to the percent level would seem to be at least 2000, after which the performance increases more slowly.

In Figure 7, we show the equivalent results obtained using a support vector machine. The support vector machine results are noticeably worse than the Bayesian neural net results for smaller numbers of training points, with clear evidence for both a greater spread of predicted values for each true likelihood and a systematic underestimate of the likelihood for the most likely points. The latter is particularly serious given that it is precisely the large likelihood points that one is interested in in practice. This is despite the fact that in the limit of a large amount of training data, the SVM results are competitive with the Bayesian neural net results.

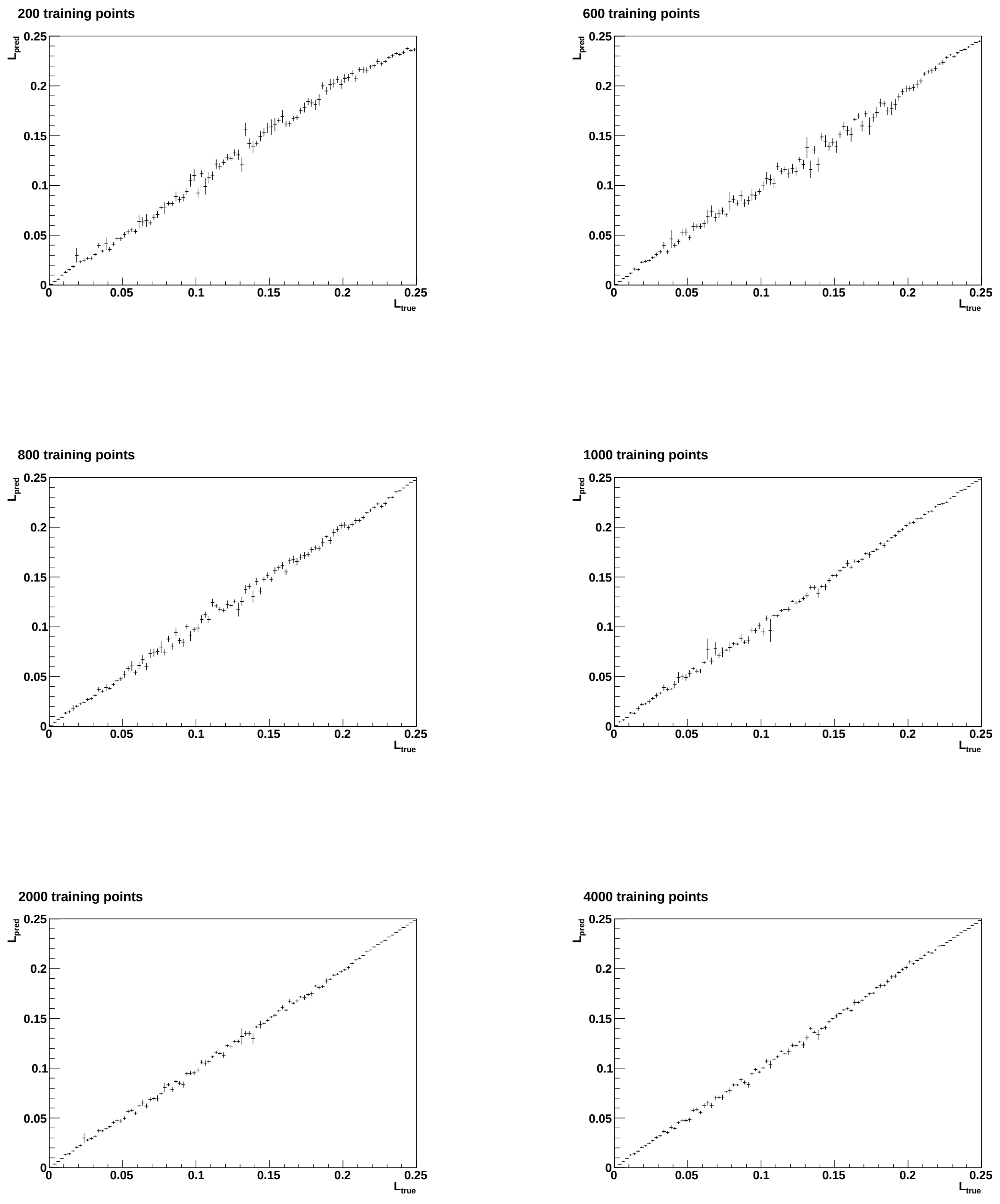


Figure 6: Profile histogram of original likelihood vs Bayesian neural net output for varying amounts of training data.

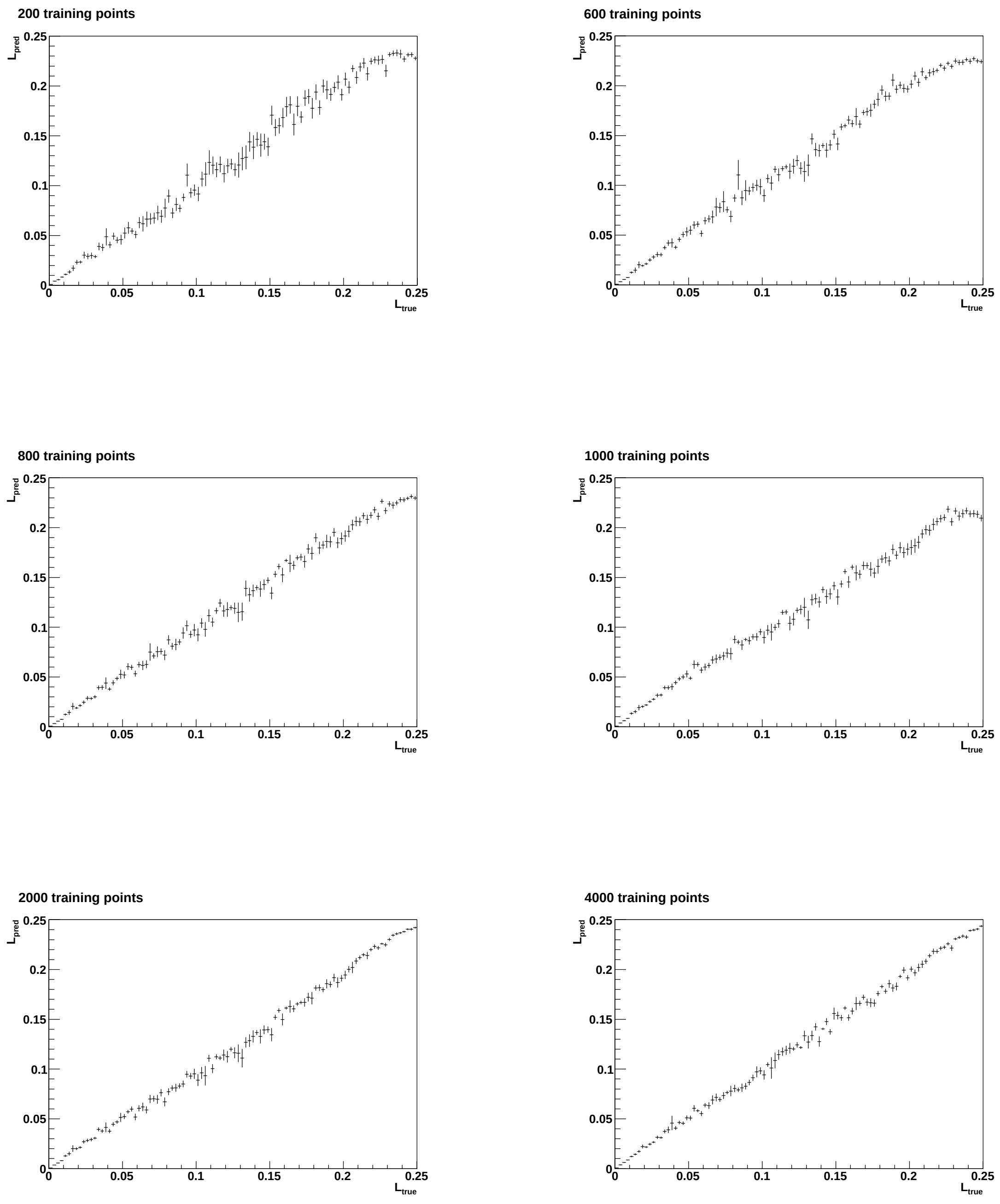


Figure 7: Profile histogram of original likelihood vs support vector machine output for varying amounts of training data.

5 Discussion

The results of the previous section indicate that it is indeed possible to interpolate between a grid of points in the full parameter space of the CMSSM. The training data generated within 36 hours proved easily sufficient to model the most constraining LHC search data and thus provide a quick function for phenomenological studies in the CMSSM. There should be no barrier to either repeating this with different LHC search channels, or to trying different underlying SUSY models.

Our investigation of the number of training points required to observe reasonable performance indicates that a few thousand training points are required to reduce the error in the likelihood to the per cent level when using a Bayesian neural net, and thus to observe identical output from the regressor as from the original simulation. This is ideal for phenomenological studies using fast simulations, particularly given that we managed to generate tens of thousands of training points with 50,000 events per point within 36 hours. For experimental collaborations wishing to use a full simulation, this level of simulation is still achievable given the availability of grid computing resources, but it may cease to be feasible as the integrated luminosity stored by ATLAS and CMS increases and thus the time and disk space allocated to the processing of LHC data removes the resources available for Monte Carlo simulation. Since in practice one would want to test the results on independent samples (such as those we used here), this further increases the stress on limited resources. We therefore suggest that the use of fast and semi-fast simulations would be useful tools if these can be validated against the full simulation within the collaborations.

The support vector machine results are broadly competitive with the Bayesian neural net results in the limit of a large amount of training data, but are noticeably worse for restricted training data sets. We thus conclude that phenomenologists using faster simulations may safely use either approach whilst experimental collaborations with more complex simulation requirements may prefer to use a Bayesian neural net. We further note that generation of the training data is not the only deciding factor in which approach to use. The ease of using a particular code for generating output predictions for input values not in the original sample is an important factor in designing a workable system for non-expert users, and thus having a choice of algorithms and computer codes is useful. In our study, it was indeed easier to generate output values using the support vector machine implementation.

It is useful at this point to compare the approach taken in this publication to other solutions to the problem of using LHC event rate data in SUSY parameter fits. An earlier attempt by the author and collaborators in Reference [15] parallelised the event generation and simulation step, thus reducing the time taken to calculate the likelihood for a given point in parameter space. This allowed a Bayesian analysis of the CMSSM to be performed on a supercomputer using a Markov Chain Monte Carlo sampling algorithm, which is a classic case of an algorithm in which (at least for a given chain), one must generate likelihoods in sequence rather than in parallel. Only 3 parameters were varied, and only 1000 events were generated per point. This has a number of disadvantages over the present method. Firstly, generating only 1000 events per point leads to large statistical fluctuations in the prediction for each point. Secondly, the parallelisation of the code was itself a major project, and the running of the code required access to a suitably fast supercomputer. Thirdly, the results were at leading order only, and adding a next-to-leading order calculation of the cross-section would lead to an unacceptably large computing time. The present method generates the training data in parallel before the parameter fit is carried out, enabling one to use a cluster which is more easily available. One can also add next-to-leading order effects or other computationally expensive calculations to the training data provided one has enough CPU resources in the cluster to generate the training data in a reasonable time, without worrying about having to calculate likelihoods in sequence.

A more recent solution to the problem of incorporating LHC event rate data into SUSY parameter fits is that given in [16]. This uses parameterised cross-sections, branching ratios and acceptances for given search channels. Cross-sections are calculated based on the squark and gluino masses for a given point and are stored in a look-up table. Branching ratios can be evaluated very quickly using a SUSY spectrum generator. Acceptances are evaluated using a combination of look-up tables and calculations of jet and lepton energies in the squark rest frame. The approach taken may be considered similar to that taken here in the sense that one can assign a likelihood for a given point based on results generated previously. However, the parameterisation of cut acceptances must be done separately for processes with different sparticle mass hierarchies, and the simulation does not include effects such as initial and final state radiation that might make an event appear in a different search channel than that naively expected based on the SUSY decay processes that are open at that point. The present method is based on interpolation of simulation results that includes all of these effects. The results of the fast parameterisations can be made available independently of the need to generate training data, and hence can be used to build LHC event rate data into generic fitting algorithms.

Having compared the method presented here favourably to others in the literature, it is only fair to state the limitations. Firstly, one has to generate a new grid of training data if one chooses to look at a different physics model. Given the speed with which we generated our training data this would not seem a major drawback for phenomenological applications, and a similar study in different classes of model is of great interest for future study. Secondly, each search channel must be added as a target variable to the neural net separately. This itself is trivial, though one may find that, e.g., dilepton channels require a different strategy for choosing training data than the zero lepton channel used here. Thirdly, we have not incorporated the effect of systematic errors. This is acceptable if the limit we derive from our results is conservative (as it is here), but would be a major issue to address should an experimental collaboration wish to use these results. In fact, if one were to store four vector data in addition to the yield for each training point, the effect of detector systematics could be determined fairly quickly, and the effects could be added to the parameter space as nuisance parameters. This would have the added advantage that new search channels could be applied to the existing training data without having to regenerate events.

6 Conclusions

We have set out to develop a method of performing fast SUSY phenomenology studies using LHC event rate data, and have presented a toy example that shows successful interpolation of simulated LHC event rate data. This information, combined with published LHC data on the number of observed, and expected background, events allows one to assign a likelihood to points in the full space of the CMSSM in fractions of a second. We find that with ~ 2000 training points, one can approximate the fully simulated likelihood to within a few percent across the range of interest, and our fast simulation results of the ATLAS zero lepton search already approximate the published ATLAS exclusion limit. This proof of principle presents a very promising avenue for future work, and promises a general and intuitive approach for using LHC data in phenomenological studies.

7 Acknowledgements

We are greatly indebted to Mike Hobson for many interesting and useful conversations at the OKC Prospects Workshop, Stockholm in September 2010. We also owe thanks to Tim Dyce of the University of Melbourne for assistance in using the Melbourne Tier 2 cluster, and to the Melbourne members of the Australian Research Council (ARC) Centre of Excellence for Particle Physics at the Terascale for useful discussions during preparation of the final draft. MJW is supported by ARC Discovery Project DP1095099. AS is supported by the ARC Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP). AB is supported by a Scottish Universities Physics Alliance research fellowship.

References

[1] G. Aad et al. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, 3:S08003, 2008.

[2] R. Adolphi et al. The CMS experiment at the CERN LHC. *JINST*, 3:S08004, 2008.

[3] Georges Aad et al. Search for supersymmetry in pp collisions at sqrts = 7TeV in final states with missing transverse momentum and b-jets. 2011.

[4] Georges Aad et al. Search for supersymmetry using final states with one lepton, jets, and missing transverse momentum with the ATLAS detector in sqrts = 7 TeV pp. 2011.

[5] Joao Barreiro Guimaraes da Costa et al. Search for squarks and gluinos using final states with jets and missing transverse momentum with the ATLAS detector in sqrt(s) = 7 TeV proton-proton collisions. 2011.

[6] Serguei Chatrchyan et al. Search for Supersymmetry in pp Collisions at sqrt(s) = 7 TeV in Events with Two Photons and Missing Transverse Energy. 2011.

[7] Vardan Khachatryan et al. Search for Supersymmetry in pp Collisions at 7 TeV in Events with Jets and Missing Transverse Energy. *Phys. Lett.*, B698:196–218, 2011.

[8] Matthew J. Dolan, David Grellscheid, Joerg Jaeckel, Valentin V. Khoze, and Peter Richardson. New Constraints on Gauge Mediation and Beyond from LHC SUSY Searches at 7 TeV. 2011.

[9] O. Buchmueller et al. Implications of Initial LHC Searches for Supersymmetry. *Eur. Phys. J.*, C71:1634, 2011.

[10] O. Buchmueller et al. Supersymmetry and Dark Matter in Light of LHC 2010 and Xenon100 Data. 2011.

[11] P. Bechtle et al. Present and possible future implications for mSUGRA of the non-discovery of SUSY at the LHC. 2011.

[12] Sujeet Akula et al. Interpreting the First CMS and ATLAS SUSY Results. *Phys. Lett.*, B699:377–382, 2011.

[13] B. C. Allanach, T. J. Khoo, C. G. Lester, and S. L. Williams. The impact of the ATLAS zero-lepton, jets and missing momentum search on a CMSSM fit. 2011.

[14] T. Auld, Michael Bridges, M. P. Hobson, and S. F. Gull. Fast cosmological parameter estimation using neural networks. *Mon. Not. Roy. Astron. Soc. Lett.*, 376:L11–L15, 2007.

[15] Christopher G. Lester, Michael Andrew Parker, and Martin J. White, 2. Determining SUSY model parameters and masses at the LHC using cross-sections, kinematic edges and other observables. *JHEP*, 01:080, 2006.

[16] Herbi K. Dreiner, Michael Kramer, Jonas M. Lindert, and Ben O'Leary. SUSY parameter determination at the LHC using cross sections and kinematic edges. *JHEP*, 04:109, 2010.

[17] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[18] Aki Vehtari and Jouko Lampinen. Bayesian mlp neural networks for image analysis. *Pattern Recognition Letters*, 21(13-14):1183 – 1191, 2000. Selected Papers from The 11th Scandinavian Conference on Image.

[19] V. Vapnik. *Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[20] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2):121–167, 1998.

[21] Nello Cristianini and John Shawe-Taylor. *An Introductino to Support Vector Machines and other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK, 2005.

[22] Yongmei Michelle Wang, Robert T. Schultz, Constable R. Todd, and Lawrence H. Staib. Nonlinear estimation and modeling of fMRI data using spatio-temporal support vector regression. *Information Processing in Medical Imaging*, 2732:647–659, 2003.

[23] Lei Xia, Jicheng Meng, Ruimin Xu, Bo Yan, and Guo Yunchuan. Modeling of 3-D vertical interconnect using support vector machine regression. *IEEE Microwave and Wireless Components Letters*, 16(12):639–641, December 2006.

[24] J. C. Mason and D. A. Turner. Applications of support vector machine regression in metrology and data fusion. In P. Ciarlini, M. G. Cox, E. Filipe, F. Pavese, and D. Richter, editors, *Advanced Mathematical Tools in Metrology, vol.57 (proceedings of AMCTM V)*, page 252. World Scientific, 2000.

[25] Jayavardhana Gubbi, Alistair Shilton, Marimuthu Palaniswami, and Michael Parker. Real value solvent accessibility prediction using adaptive support vector regression. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2007)*, pages 395–401, 2007.

[26] D. Lai, A. Khandoker, R. Begg, and M. Palaniswami. A hybrid support vector machine and

autoregressive model for detecting gait disorders in the elderly. In *IEEE International Joint Conference on Neural Networks IJCNN 2007*, August 2007.

[27] Alexander Smola. Regression estimation with support vector learning machines. Master's thesis, Technische Universität Münschen, 1996.

[28] V. Vapnik, S. Golowich, and A. Smola. Support vector methods for function approximation, regression estimation, and signal processing. In *Advances in Neural Information Processing Systems*, volume 9, pages 281–187. MIT Press, 1997.

[29] A. Smola and B. Schölkopf. A tutorial on support vector regression. Technical Report NeuroCOLT2 Technical Report Series, NC2-TR-1998-030, Royal Holloway College, University of London, UK, October 1998.

[30] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Transactions of the Royal Society of London*, 209(A), 1909.

[31] F. G. Tricomi. *Integral Equations*. Interscience, 1970.

[32] Ralf Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, 2002.

[33] Ingo Steinwart and Andreas Christman. *Support Vector Machines*. Springer, 2008.

[34] Frank E. Paige, Serban D. Protopescu, Howard Baer, and Xerxes Tata. Isajet 7.69: A monte carlo event generator for p p, anti-p p, and e+ e- reactions. 2003.

[35] G. Corcella et al. Herwig 6: An event generator for hadron emission reactions with interfering gluons (including supersymmetric processes). *JHEP*, 01:010, 2001.

[36] G. Corcella et al. Herwig 6.5 release note. 2002.

[37] Stefano Moretti, Kosuke Odagiri, Peter Richardson, Michael H. Seymour, and Bryan R. Webber. Implementation of supersymmetric processes in the herwig event generator. *JHEP*, 04:028, 2002.

[38] Elzbieta Richter-Was. AcerDET: A particle level fast simulation and reconstruction package for phenomenological studies on high p(T) physics at LHC. 2002.