Modeling the Predictions of Supersymmetry at the LHC using Bayesian Neural Networks

Michelle E. Perry, Dr. Anke Meyer-Baese, Dr. Harrison Prosper*

Florida State University, Department of Scientific Computing, *Department of Physics

Physics Motivation

There exist many unanswered questions in the area of High Energy Physics (HEP), even with the recent discovery of the Higgs boson. What is dark matter? Why is the Higgs mass so low? The currently accepted theory, the Standard Model (SM), provides no answers to these questions. Therefore, physicists are working on theories beyond the SM that may provide answers. One well-motivated class of theories is based on supersymmetry (SUSY), which associates each fundamental particle in the SM with a supersymmetric counterpart. A key prediction of SUSY is that supersymmetric particles, known as "sparticles", will be created in pairs in high energy collisions, for example, gluino pairs as illustrated in the Feynman diagrams below. If these SUSY theories are viable, their predictions must be consistent with the observations being made at the Large Hadron Collider (LHC).

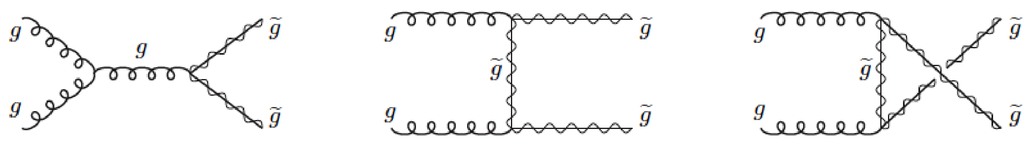

Figure : Feynman diagrams depicting gluino pair production at the LHC.

Testing Supersymmetric Theories

By far the most popular of the SUSY theories is the minimal supersymmetric SM (MSSM), a 119-parameter model that has the ability to address some of the unanswered questions mentioned above. However, it remains nearly impossible to study the validity of the MSSM because of its huge parameter space, so traditionally physicists have imposed constraints on the model. A commonly used model to test the SUSY hypothesis is the constrained minimal supersymmetric SM (CMSSM), a highly constrained sub-model of the MSSM. The constraints make working with this model computationally feasible. But, unfortunately, the constraints are so stringent that the falsification of the CMSSM would not invalidate the MSSM [2]. A less-constrained model of the MSSM, the phenomenological minimal supersymmetric SM (pMSSM), is a sub-model that encapsulate most of the possible physics of the MSSM. However, the computational burden of testing the pMSSM hinders its routine use. Therefore, we propose a parallelized method to create functional mappings of the 19-dimensional model space to observable predictions, which will make it possible to use standard techniques for analyzing this theory.

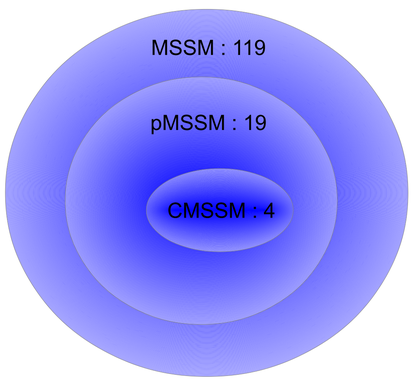

Figure : Graphical representation of the MSSM model space and a couple of the sub models that are being studied. The value adjacent to the model name is the number of free parameters in the space.

Testing the pMSSM at the LHC

For each 19-dimensional point in the parameter space, the pMSSM makes predictions, $y_i = f_i(\theta_1 \ldots \theta_{19})$, for quantities such as sparticle masses and cross sections. However, these predictions are given only at a discrete set of points, which could number in the millions. Statistical approaches that require a smooth mapping can not be applied. Our proposed solution to this problem is to model the pMSSM predictions using Bayesian Neural Networks (BNNs), which for this application, is given by

$$f(\theta) = \int n(\theta, \omega) P(\omega|T) d\omega, \tag{1}$$

where ω are the free parameters of the neural network $n(\theta, \omega)$ and T are the training data [1], consisting of the ensemble of pMSSM parameter points and the associated predictions. This will provide the smooth mappings needed for standard statistical methods. The advantage of BNNs over neural networks is that BNNs are less prone to over-training and they provide estimates of the accuracy of the mappings.

Bayesian Neural Networks

In this work we focus on networks that map many inputs to a single output. Such a network can be represented as:

$$f(\theta, \omega) = a + \sum_{j=1}^{H} b_j \tanh\left(c_j + \sum_{i=1}^{N} d_{ji}\, \theta_i\right) \tag{2}$$

where ω are the neural network parameters, H is the number of hidden nodes in the network, and $N = 19$ is the number of pMSSM parameters. This network employs one hidden layer, and **tanh** is the chosen "activation function" for the neurons. In the Bayesian approach, one assigns a probability density to every point in the parameter space of the neural network so that one can assign meaning to the statement that one point in the neural network parameter space is more probable, given the training data, than another. We can compute the probability density $p(\omega|T)$ where $\ln p(\theta|T)$ is given by

$$\sum_{n=1}^{K} w_n \left[t_n - f(\theta_n, \omega)\right]^2, \tag{3}$$

K is the number of examples in the training data, $T = \{t_n, \theta_n\}$, w_n is a weight associated with each example, and t_n are the targets, that is, predictions y_n associated with the pMSSM parameter points θ_n. The prior for each network parameter is taken to be a Gaussian centered at zero. The training of a Bayesian neural network entails sampling points ω_k from $p(\omega|T)$. We do so using Markov Chain Monte Carlo. For SUSY applications, $K \sim$ millions, which renders these calculations a formidable challenge. In order to demonstrate the viability of this approach, and the challenge, we generated a mapping from the pMSSM parameter space to the mass of one of the sparticles, the gluino. Figure 1 shows an example of the BNN modeling of the function $m_{\tilde{g}} = f(\theta_1, \cdots, \theta_{19})$, where $m_{\tilde{g}}$ is the mass the gluino and $\theta_1 \ldots \theta_{19}$ are the parameters of the pMSSM. This function is based on a BNN with $N = 19$, $H = 40$, and $K = 20,000$ training patterns. This calculation took 20 hours on a single Intel Core i7 CPU @ 2.67 GHz. An implementation of this algorithm with a 100 factor speedup would decrease the computation time to about 12 minutes.

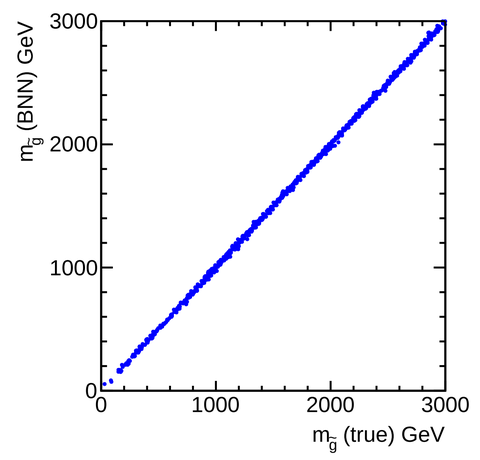

Figure : BNN predicted mass of the gluino vs. mass from the pMSSM predictions.

GPU Implementation

Our goal is to reduce the time required to train multivariate BNNs by at least two orders of magnitude, through careful optimization of the most time-consuming part of the training algorithm, namely, the calculation of the large sum of highly non-linear functions (Eq. 3). Graphical Processing Units (GPUs) are ideal for this application because the calculations of each event of training data, T, are independent. We ideally want to train on orders of $10^5 - 10^6$ events, so the GPU is preferable to parallel CPU implementations due to the many-core nature of the GPUs. A parallel reduction algorithm is then used on the results from each event to give us $P(X|\omega)$. Current work utilizes NVIDIA's CUDA C extension on a single GPU, with hopes of expanding to multiple GPUs on the new FSU SPEAR GPU cluster.

References

Pushpalatha C. Bhat and Harrison B. Prosper.
Bayesian neural networks, Nov 2005.

S. Sekmen et al.
Interpreting lhc susy searches in the phenomenological mssm.
J. High Energy Phys., Jan 2012.