Bayesian Neural Neworks in Data-Intensive High Energy Physics Applications

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

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

Abstract

Bayesian Neural Networks (BNNs) are applied to data-intensive High Energy Physics (HEP) applications for classification and regression. Neural networks are non-linear functions that can be used to model (in principle) any mapping of $\mathbf{N}$ continuous real variables to $\mathbf{M}$ real variables. Where traditional neural networks use optimization techniques to find an optimum set of neural network parameters, ω_0, BNNs assign a probability density to every set of network parameters, ω_k, in the parameter space. However, BNNs are significantly more computationally intensive to construct than neural networks. The goal of this work is to develop efficient implementations of the training of BNNs on Graphical Processing Units (GPUs).

Our preliminary studies with a GPU indicate that speed improvements of at least 80 are already possible with relatively modest optimization. We are therefore confident that the outcome of the proposed work could be extremely far-reaching once it becomes possible to fit complicated multivariate functions in minutes rather than in hours or days.

Bayesian Neural Networks

A neural network with at least one hidden layer of nodes is proven to be a "universal approximator," meaning that it can reproduce *any* smooth mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$. In this work we focus on mapping from many inputs to $\mathbf{m = 1}$ Such a network can be represented in equation form as:

$$f(X,\omega) = a + \sum_{j=1}^{H} b_j \tanh(c_j + \sum_{i=1}^{N} d_{ji}\, x_i) \tag{1}$$

where X is the set of training data, ω are the neural network parameters, H is the number of hidden nodes in the network, and N is the number of input variables. This network employs one hidden layer, and **tanh** is the chosen "activation function" for the neurons.

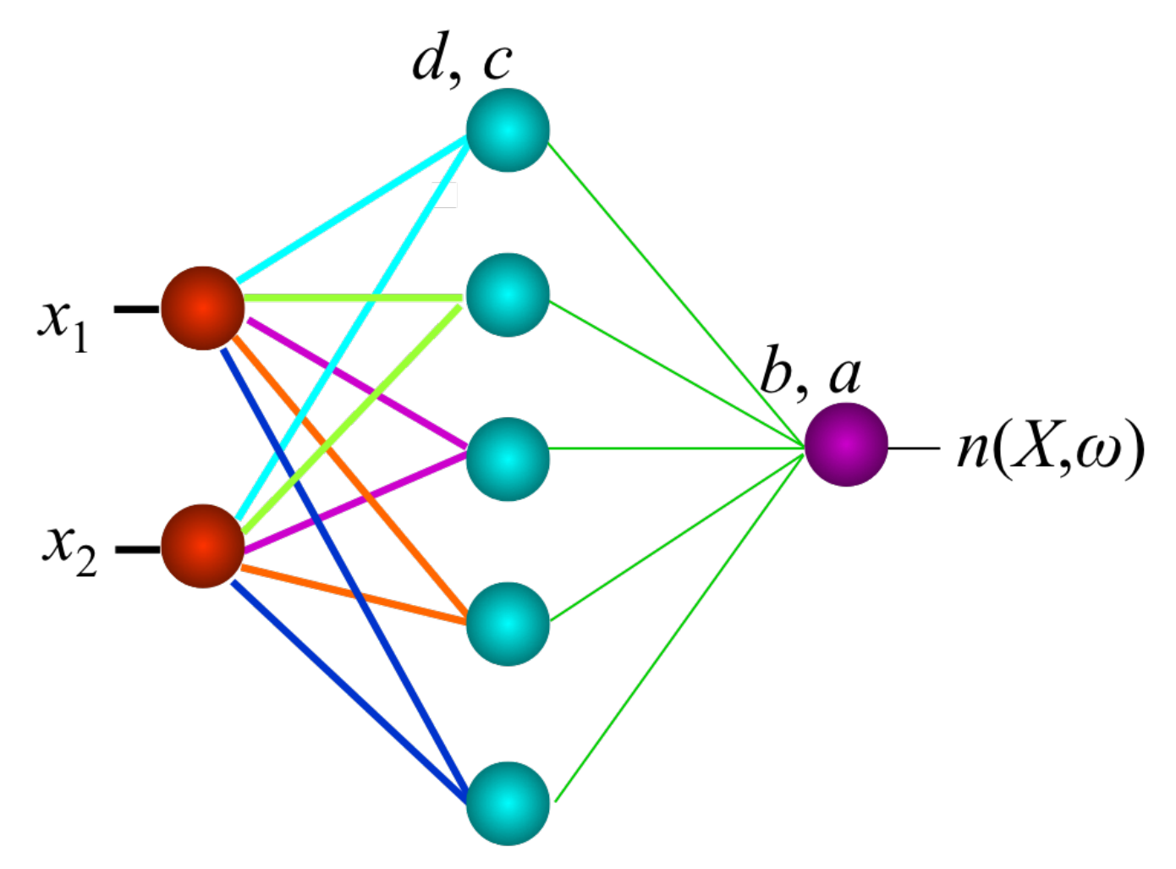

Figure: Graphical representation of a neural network with a single hidden layer.

For classification,

$$n(X,\omega) = 1/[1 + \exp(-f(X,\omega)], \tag{2}$$

yielding $n(X,\omega) \in (0,1)$.

The training of neural networks and Bayesian neural networks differ because of the differing perspectives with which the network parameters are viewed. Most training algorithms for neural networks view their parameters as parameters to be fitted to the training data using, for example, gradient descent to a single "best" fit point in the network parameter space. By contrast, 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. Given the likelihood of the training data, which we denote by $p(X|\omega)$, where ω denotes a point in the neural network parameter space, we invoke Bayes theorem

$$p(\omega|X) = p(X|\omega)\, p(\omega)/p(X), \tag{3}$$

to compute the probability density $p(\omega|X)$ where $\ln p(X|\omega)$ is given by:

$$-\sum_{n=1}^{T} w_n \left[t_n \ln n(X_n,\omega) + (1-t_n)\ln[1 - n(X_n,\omega)]\right]. \tag{4}$$

$\mathbf{T}$ is the number of examples in the training data, w_n is a weight associated with each example, and $t_n = 1, 0$ are the targets for classification into two classes. The training of a Bayesian neural network entails sampling points ω_k from $p(\omega|X)$. We do so using Markov Chain Monte Carlo. For the applications we have in mind, $T \sim 10^5 - 10^6$, which renders these calculations a formidable challenge.

High Energy Physics Application

The CERN Large Hadron Collider (LHC) is the most powerful scientific instrument to date. It was built for two main purposes: to study the origins of mass by searching for the Higgs boson and to search for physics outside the Standard Model. One unconfirmed prediction of the SM is that the mass of fundamental particles arise through interaction with an energy field, the Higgs field. The existence of the Higgs field would require a mediating particle, the Higgs boson. The FSU group in collaboration with the Politecnico di Bari, Italy are searching for the Higgs boson via the $pp \longrightarrow H \longrightarrow ZZ^* \longrightarrow 4l + X$ and $pp \longrightarrow H \longrightarrow WW \longrightarrow 2l + X$ decay channels. After event selection, an estimated 11.2 events in the ZZ^* channel and 38.7 events in the $2l$ channel, assuming the Higgs has a mass of 130 GeV. This signal to background ratio in the ZZ^* mode is $\mathbf{1:5}$, whereas in the WW mode the ratio is $\mathbf{1:20}$.

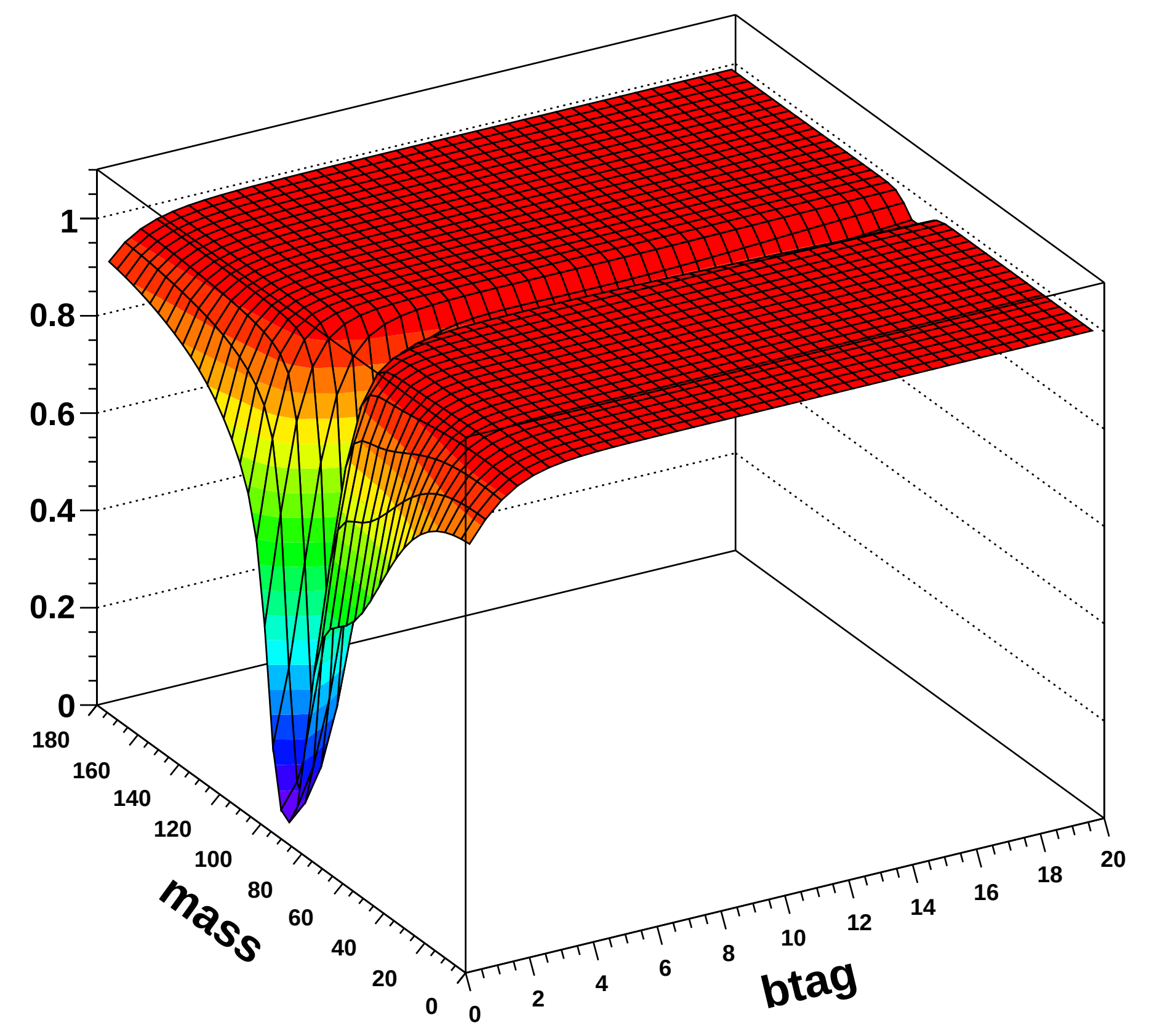

Figure: Classification probability $P(\mathrm{top}|\mathrm{mass},\mathrm{btag})$.

The figure shows an example of the BNN modeling of the classification probability $P(\mathrm{top}|\mathrm{mass},\mathrm{btag})$, where the mass is that of muon-pairs created at the LHC and the variable btag characterizes the number of so-called **b**-quarks created. This function discriminates between reactions at the LHC containing top quarks that decay to muon-pairs from reactions that contain **Z** bosons also decaying to muon-pairs. The function, $P(\mathrm{top}|\mathrm{mass},\mathrm{btag})$, is modeled by a BNN with two input variables (mass and btag), 15 hidden nodes, 10,000 training events, and a sampling of about a million network parameter points. This required about 4 hours on a fast laptop. However, realistic applications may need $\sim 10^6$ training events and may use ~ 15 input variables. Since training time is proportional to the number of training events, this motivates the development of computational methods to reduce the training time significantly. We propose using GPUs to achieve this goal.

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. 4). Graphical Processing Units (GPUs) are ideal for this application because the calculations of each event of training data, $\mathbf{T}$, 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

Abdelhak Djouadi
Higgs Phenomenology: A Short Review
Acta Physica Polonica A arXiv:hep-ph/9612361v1. December, 2009.

Jouko Lampinen and Aki Vehtari
Bayesian Approach for neural networks–review and case studies
Neural Networks 14. 2001.