

BAYESIAN NEURAL NETWORKS

PUSHPALATHA C. BHAT¹ AND HARRISON B. PROSPER²

¹*Fermi National Accelerator Laboratory, Batavia, Illinois, USA*

²*Florida State University, Tallahassee, Florida, USA*

The training of neural networks can be viewed as a problem of inference, which can be addressed from a Bayesian viewpoint. This perspective leads to a method, new to the field of particle physics, called Bayesian neural networks (BNN). After a brief overview of the method we illustrate how it can be usefully deployed in particle physics research.

1. Introduction

Neural networks (NN) ¹ are non-linear functions that, in principle, can model *any* (smooth) map of a set of one or more real input variables to a set of one or more outputs ². In this paper, we consider neural networks for binary classification. The typical application in particle physics is to separate signal from background. We also make some brief remarks on networks for regression, that is, for fitting functions. We end with an example of classification from particle physics.

2. Classification

If a network is trained with events, described by a vector of variables x , such that signal events are labeled by $t = 1$ and background events by $t = 0$, then the network output y approximates the posterior probability ³

$$y \approx \text{Prob}(t = 1|x) = \frac{p(x|1)p(1)}{p(x|1)p(1) + p(x|0)p(0)}, \quad (1)$$

that is, the probability that an event defined by the variables x belongs to the signal class $t = 1$. $p(x|1)$ and $p(x|0)$ are the probability density functions for the signal and background classes, respectively, and $p(1)$ and $p(0)$ are the corresponding class prior probabilities. Typically, one trains with equal numbers of signal and background events, in which case the priors cancel out. The label t is referred to as the *target*.

The idea behind Bayesian neural networks (BNN) is to cast the task of training a network as a problem of inference, which is solved using Bayes' theorem. The latter is used to assign a probability density to each point w in the parameter space of the neural network. Each point w corresponds to a network defined by a specific set of parameter val-

ues. In the standard methods for training neural networks, one finds a *single* point w_0 in the parameter space, that is, a single network. In the Bayesian approach, one performs a weighted average over all points, that is, all networks. As with the standard methods, the BNN methods make use of training data $\{(t_1, x_1), \dots, (t_N, x_N)\}$, where t_i is the *known* label associated with data x_i . The probability density assigned to point w , that is, to a network, is given by Bayes' theorem

$$\begin{aligned} p(w|t, x) &= \frac{p(t, x|w)p(w)}{p(t, x)}, \\ &= \frac{p(t|x, w)p(x|w)p(w)}{p(t|x)p(x)}, \\ &= \frac{p(t|x, w)p(w)}{p(t|x)}, \end{aligned} \quad (2)$$

where we have assumed that the data x do not depend on w , in which case $p(x|w) = p(x)$. Thus, in order to assign a probability density to a network, defined by the point w , we need the likelihood $p(t|x, w)$ and the prior density $p(w)$.

Consider a *class* of neural networks defined by the functional form

$$y(x, w) = \frac{1}{1 + \exp[-f(x, w)]}, \quad (3)$$

where

$$f(x, w) = b + \sum_{j=1}^H v_j \tanh(a_j + \sum_{i=1}^P u_{ij} x_i), \quad (4)$$

having P inputs, a *single* hidden layer of H hidden nodes and a single output. In the particular BNN method described here, every network has the same structure. However, as noted below, the effective number of hidden nodes could be fewer than H , if there are hidden nodes with associated weights near zero and if such networks are assigned higher probability than those with a greater number of active

nodes. The parameters u_{ij} and v_j are called *weights* and a_j and b are called *biases*. Both sets of parameters are usually referred to collectively as weights, w .

Since, for a correctly trained network, the probability that $t = 1$ is $y(x, w)$, and $1 - y$ for $t = 0$, the probability of the set of targets $t = (t_1, t_2, \dots, t_N)$, given the data $x = (x_1, x_2, \dots, x_N)$, is

$$p(t|x, w) = \prod_{i=1}^N y^{t_i} (1 - y)^{1-t_i}, \quad (5)$$

in which we have assumed the events to be independent. Given an event with data x' , a reasonable estimate of the probability that it belongs to the signal class (assuming $p(0) = p(1)$) is given by the weighted average

$$\bar{y}(x'|t, x) = \int y(x', w) p(w|t, x) dw, \quad (6)$$

where the posterior density $p(w|t, x)$, given by Eq. (2), is computed using the likelihood, Eq. (5), and some prior $p(w)$, to be discussed shortly.

Currently, the only feasible way to perform the high-dimensional integral in Eq. (6) is to sample the density $p(w|t, x)$, in some appropriate way, and to approximate the integral using the average

$$\bar{y}(x'|t, x) \approx \frac{1}{K} \sum_{k=1}^K y(x', w_k), \quad (7)$$

where K is the number of points w sampled from $p(w|t, x)$. We note, again, that each point w corresponds to a different neural network function in the class of networks with P inputs and H hidden nodes. The average is therefore an average over *networks*.

It may happen that some of the points w correspond to networks that are tightly fit to the training data. Such networks will typically perform poorly on an independent set of events. However, if one averages over many networks, one expects to produce an estimate of the signal class probability, $y = p(1|x)$, that is less likely to be affected by “over training.” Moreover, in the Bayesian approach, there is less need to limit, severely, the number of hidden nodes because a low probability density will be assigned to points w that correspond to unnecessarily large networks, in effect, pruning them away. Indeed, networks have been trained⁴, successfully, that contain more weights than the number of training data! In this Bayesian approach, the network should be as

large as is computationally feasible so that the class of functions defined by the network parameter space includes a subset that are good approximations to the true mapping.

3. Regression

In a regression problem, the targets t are usually sampled from a continuous set. For example, we may wish to model a function $t = f(x)$ that maps an uncorrected measurement of the transverse momentum of a jet of particles, to a corrected measurement. In this case, the target t would be the *known* “correct” value of the transverse momentum of the jet—perhaps, taken to be the transverse momentum of a Z boson recoiling against the jet, while x would be the measured jet transverse momentum, along with any other measured quantities believed to be relevant. If we wish to fit a function to these data, the form given in Eq. (5) for the likelihood of the targets is inappropriate. A better model, assuming that the noise in the targets can be modeled by a Gaussian—at least approximately, is

$$\begin{aligned} p(t|x, w) &= \prod_{i=1}^N \exp[-(t_i - f(x_i, w))^2/2\sigma^2], \\ &= \exp[-\sum_{i=1}^N (t_i - f(x_i, w))^2/2\sigma^2], \end{aligned} \quad (8)$$

with $f(x_i, w)$ given in Eq. (4). Even if the noise in the target is not Gaussian, Eq. (8) may still yield reasonable results, provided that the value of σ is chosen to match the noise level in the targets.

One advantage of modeling such mappings with neural networks is that the functional form Eq. (4) is flexible enough to model functions $t = f(x)$ in which x is multi-dimensional, and, in which one or more components of x may be statistically dependent.

4. Computing the Posterior Density

In order to compute the average in Eq. (6), it is necessary to generate a sample of points w from the posterior density, Eq. (2). Unfortunately, sampling from the posterior density is not feasible using simple numerical methods. In practice, a sample is generated using Markov Chain Monte Carlo (MCMC) methods⁴. In the MCMC method, one steps through a parameter space in such a way that points are visited with a probability proportional to the density one

wishes to sample, here the posterior density $p(w|t, x)$. Points where $p(w|t, x)$ is large will be visited more often than points where $p(w|t, x)$ is small. The methods of choice for sampling complex densities, such as $p(w|t, x)$, originate in the field of computational statistical physics. The problem of moving through the network parameter space is re-cast as a problem of statistical mechanics, specifically, of a *single* particle moving through a (rather complicated) potential.

The posterior density is written as

$$p(w|t, x) = \exp[-V(q)], \quad (9)$$

where $V(q) = -\ln p(w|t, x)$ (with $q \equiv w$) is interpreted as a spatially varying ‘‘potential’’ through which the ‘‘particle’’ moves. One adds a ‘‘kinetic energy’’ term $T(p) = \frac{1}{2}p^2$, where p is a vector of dimensionality equal to that of the network parameter space. The ‘‘mass’’ of the ‘‘particle’’ can be taken to be unity by appropriate re-scaling. The motion of the particle is governed by its ‘‘Hamiltonian’’ $H = T + V$. For a Hamiltonian system, the particle will, *eventually*, visit every phase space point (q, p) arbitrarily closely in such a way that the density of points in phase space is proportional to $\exp(-H)$. By randomly (and appropriately) injecting or removing ‘‘energy’’ from the system, different constant energy regions of phase space $\{(p, q)\}$ can be explored. A Markov chain q_1, q_2, \dots, q_N is thereby created, which converges (eventually) to a sequence of points that constitute a sample from the density $p(w|t, x)$. Since the correlation between adjacent points is very high, typically 0.9 or higher, one usually saves a point, that is, a network, after every L steps, to lessen the correlation between the saved points. It is also necessary to discard the initial part of the Markov chain because, in general, it will not be a faithful sample of the required density.

4.1. Prior

Every Bayesian inference requires the specification of a prior. Unfortunately, for this problem, the choice of prior is not obvious. However, experience suggests that a reasonable class to choose from is the class of Gaussian priors centered at zero, which favors smaller rather than larger weights. Smaller weights yield smoother fits to data. In the example described next, which uses the BNN package of Radford Neal⁴, a Gaussian prior is specified for each weight. However, the variance for weights belonging to a given

group (either *input-to-hidden weights* (u_{ij}), *hidden-biases* (a_j), *hidden-to-output weights* (v_j) or *output-bias* (b)) is chosen to be the same: σ_u^2 , σ_a^2 , σ_v^2 , or σ_b^2 , respectively. However, since we do not know, *a priori*, what these variances should be, their values are allowed to vary over a large range, while favoring small variances. This is done by assigning each variance a gamma prior

$$p(z) = \left(\frac{\alpha}{\mu}\right)^\alpha \frac{z^{\alpha-1} e^{-z\frac{\alpha}{\mu}}}{\Gamma(\alpha)}, \quad (10)$$

where $z = \sigma^{-2}$, and with the mean μ and shape parameter α set to some fixed plausible values. The inverse of the variance $z = \sigma^{-2}$ is sometimes referred to as the *precision*. The gamma prior is referred to as a *hyperprior* and the parameter (here the precision) for which it is a prior is called a *hyperparameter*.

5. An Example: Single Top Search

The electroweak production of single top quarks, which has not been observed so far, is an important prediction of the Standard Model. Moreover, it is potentially a sensitive probe of new physics. The observation of this process at the Fermilab Tevatron is much more challenging than the observation of top-antitop pairs⁵, because of the much smaller signal to background ratio involved. We have studied the discrimination of the signal, in the channel $p\bar{p} \rightarrow tqb \rightarrow Wbqb$, from the dominant background process, $p\bar{p} \rightarrow Wb\bar{b}$, for the case in which the W boson decays into a muon (μ) and a neutrino (ν). The final state, therefore, contains a high transverse momentum muon, two b -quark jets and significant missing transverse energy due to the neutrino from the W boson.

We considered eleven kinematic variables that involve the transverse energies, spatial separation and invariant masses of the measured final state objects. All eleven variables were used as inputs to the neural networks, each with thirty hidden nodes and a single output. A Markov chain of networks was generated using the BNN software package noted above, with a training sample consisting of 1000 events each of signal and background. Five hundred iterations, of twenty MCMC steps each, were used. Neural network parameters were stored after each iteration. The results of the training are shown in Fig. 1. For both the signal and background samples, the network outputs, averaged over the last 100 networks,

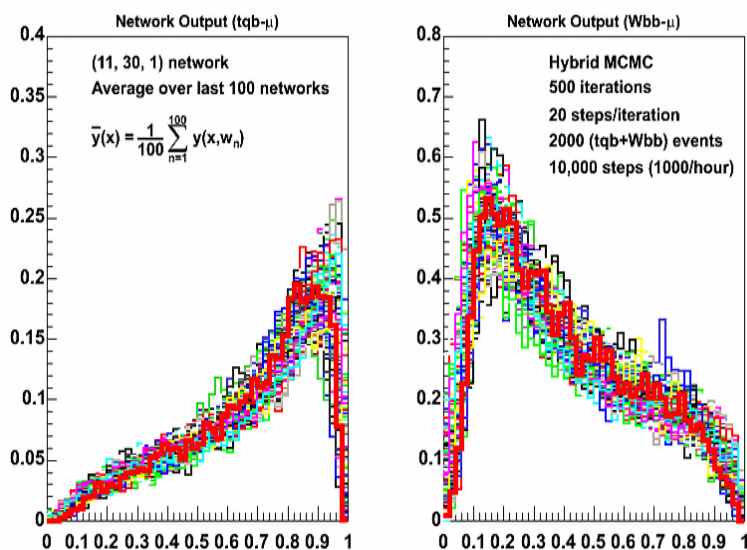


Fig. 1. The output distributions of an ensemble of neural networks, generated using a MCMC method, for Monte Carlo signal (single top – tqb channel) and background (Wbb) events. The thick histogram is the distribution of the network output averaged over the last 100 of a sequence of 500 networks, sampled from the posterior density $p(w|t, x)$.

are shown superposed on the output distributions of each of the individual networks. As one might have expected, the distributions of the 100 networks show some scatter. One expects, however, the Bayesian average to be a more robust estimate of the true signal class probability.

6. Conclusions

Bayesian learning of neural networks could take us another step closer to realizing optimal and robust results in classification problems. It also allows a fully probabilistic approach with proper treatment of uncertainties. But, of course, the key question is: does the averaging help? The answer, in principle, is yes. More to the point, we have found the answer to be yes, in practice. Figure 1 is, in effect, a comparison of 100 *single* neural networks with the Bayesian average over all of them. A study of these distributions reveals that the area under the ROC curve (the plot of the signal efficiency vs. background efficiency) is *larger* for the “Bayesian-averaged network” than for any one of the individual networks, which is an indication that the averaging helps.

The BNN method, however, is computationally demanding. In the example described above, 10 hours were required to sample 10,000 points, that is, networks, from the posterior density. A large number

of points is needed so that one can abstract a subset of (several hundred) networks that are approximately statistically independent.

We have started exploring the application of the BNN method to the analysis of particle physics data. The initial results, as illustrated by the example of the single top quark search by the DØ Collaboration, at Fermilab, are very promising.

References

1. C. M. Bishop, *Neural Networks for Pattern Recognition*, (Clarendon Press, Oxford, 1998); R. Beale and T. Jackson, *Neural Computing: An Introduction*, (Adam Hilger, New York, 1991).
2. E.K. Blum and L.K. Li, “Approximation theory and feedforward networks,” *Neural Networks*, **4**, 511-515 (1991).
3. D.W. Ruck et. al., “The multilayer perceptron as an approximation to a Bayes optimal discriminant function,” *IEEE Trans. Neural Networks* **1** (4), 296-298 (1990); E.A. Wan, “Neural network classification: a Bayesian interpretation,” *IEEE Trans. Neural Networks* **1** (4), 303-305 (1990).
4. R. M. Neal, *Bayesian Learning of Neural Networks*, (Springer-Verlag, New York, 1996). For a review, see: <http://www.cs.utoronto.ca/~radford/ftp/review.pdf>.
5. CDF Collaboration, F. Abe et. al., *Phys. Rev. Lett.* **74**, 2626 (1995); DØ Collaboration, S. Abachi et. al., *Phys. Rev. Lett.* **74**, 2632 (1995); For a review, see P.C. Bhat, H.B. Prosper and S.S. Snyder, *Int. J. Mod. Phys.* **13**, 5113 (1998).