XIIIth Quark Confinement and the Hadron Spectrum

Section on Statistics for the 21st Century

Maynooth, Ireland

The Bayesian Interpretation of Deep Neural Networks: What Is It Good For?

Harrison B. Prosper

Department of Physics, Florida State University

August 1, 2018

Outline

Outline

The basic structure of neural networks was inspired by biological research during World War II[1] and biology remains a source of inspiration. It has inspired, for example, state-of-the-art models such as convolutional neural networks (CNN)[2], which in turn has led to extraordinary breakthroughs such as DeepMind's AlphaGo Zero[3] Go-playing program.

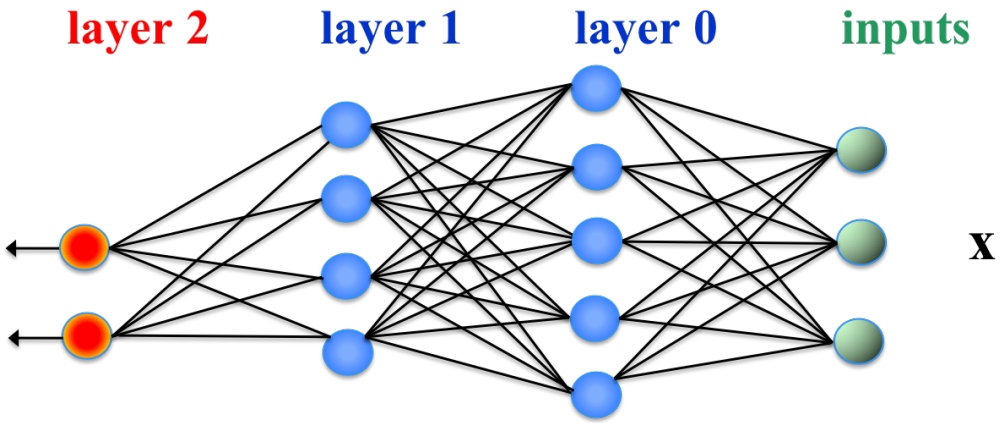
However, while the inspiration from biology should not be dismissed, the most relevant point to be made is that these models–deep neural networks (DNN)–are simply highly flexible, parametrized, real-valued functions $f(x, \theta)$ of real-valued attributes x (often referred to as features) and free parameters θ. In some applications the number parameters is now in the tens of millions.

[1] McCulloch and Pitts, Bulletin of Mathematical Biophysics, **5**, 115-133 (1943); Rosenblatt Psychological Review **65** (1958)

[2] LeCun, Bottou, Bengio, and Haffner, Proc. of the IEEE, 1-46 (1998); LeCun, Bengio and Hinton, Nature **521** 436-444 (2015)

[3] Silver et al., Nature **550**, 354-359 (2017)

Example (A (3,5,4,2) multi-layer neural network)



$$\mathbf{h}_1 = h(\mathbf{b}_0 + \mathbf{w}_0\mathbf{x})$$
$$\mathbf{h}_2 = h(\mathbf{b}_1 + \mathbf{w}_1\mathbf{h}_1)$$
$$\boldsymbol{f} = g(\mathbf{b}_2 + \mathbf{w}_2\mathbf{h}_2)$$

$h(z)$

$$\mathrm{ReLU}(z) = \max(0, z),$$
$$\tanh(z)$$

$g(z)$

$$\mathrm{identity}(z) = z,$$
$$\mathrm{logistic}(z) = \frac{1}{1 + \exp(-z)},$$
$$\mathrm{softmax}(z) = \frac{\exp(z)}{\sum_i \exp(z_i)}$$

$$f(\mathbf{x}, \theta) = g(\mathbf{b}_2 + \mathbf{w}_2 h(\mathbf{b}_1 + \mathbf{w}_1 h(\mathbf{b}_0 + \mathbf{w}_0 \mathbf{x}))),$$
$$\theta \equiv \mathbf{b}_i, \mathbf{w}_j$$

$h(z)$ and $g(z)$ are applied element-wise to the matrices $\mathbf{b}_j$ and $\mathbf{w}_j$.

By 2050, it is likely that the following ethical problem will occur.

A self-driving vehicle is in a no-win situation: whatever decision it makes, someone will suffer a terrible injury or be killed. The vehicle decides to injure the aged passenger within the car and thereby avoid seriously injuring or killing a young girl who has just stepped into the street. It seems that it ought to be possible to understand why the car decided to injure the old man rather than the young girl by delving into the inner workings of the vehicle's deep neural network.

In my view, trying to delve into the inner workings of what could be a non-linear, 100-million parameter, function is likely to be a futile exercise.

It is more fruitful to understand what these complicated functions *approximate* and use that understanding to build higher level decision making systems.

To that end, a significant conceptual advance occurred in 1990[4] when it was shown that a neural network trained as a classifier approximates the class probabilities

$$p(k \,|\, x) = \frac{p(x|k)\, p(k)}{\sum_k p(x|k)\, p(k)}, \quad k = 1, \cdots, K,$$

where K is the number of disjoint classes.

The following year, Richard and Lippmann[5] pointed out some useful consequences of this important result.

It turns out that the result holds for *any* class of functions provided that

- the training data are sufficiently numerous,
- the function $f(x, \theta)$ is sufficiently flexible, and
- the appropriate loss function is used.

[4] Ruck et al., IEEE Trans. Neural Networks **4**, 296-298 (1990); Wan, IEEE Trans. Neural Networks **4**, 303-305 (1990)

[5] Richard and Lippmann, Neural Computation **3**, 461-483 (1991)

Outline

A Disclaimer The term "Bayesian interpretation" of $f(x, \theta)$, when describing the result

$$f(x, \theta) \approx p(k \,|\, x) = \frac{p(x|k)\, p(k)}{\sum_k p(x|k)\, p(k)}, \quad k = 1, \cdots, K,$$

is seriously misleading! It would be correct to say that $f(x, \theta)$ approximates Bayes theorem.

The interpretation would be Bayesian if and only if the prior probabilities $p(k)$ were degrees of belief, which in most applications they are not.

A truly Bayesian approach would require calculation of the predictive distribution of the target value y given input features x.

$$p(y|x, T) = \int p(y|x, f)\, p(f\,|\,T)\, df,$$

where $p(y|x, f) \equiv p(y|x, \theta)$, is the likelihood of y given x,
$p(f\,|\,T) \equiv p(\theta|\,T)$ is the posterior density of θ given training data T.

Every standard machine learning method, even one as cryptic as AdaBoost, can be cast as an optimization problem whose goal is to minimize the average

$$R(f) = \frac{1}{N}\sum_{i=1}^{N} L(y, f(x_i, \theta)) + C(\theta)$$

of a suitable loss function $L(y, f)$ subject to some constraint $C(\theta)$.

The key point to note is that this sum approximates the *functional*

$$\begin{aligned} R[f] &= \int \left[\int dy L(y, f(x_i, \theta))\, p(y, x) \right] dx, \\ &\equiv \int G(f)\, dx, \end{aligned}$$

where $p(y, x)$ is the probability density of the targets y and features x.

A minimization algorithm, typically a variation of stochastic gradient descent, searches a function space by varying the function $f(x, \theta)$ by small amounts $\delta f(x)$ until a function is found for which

$$\begin{aligned} \delta R[f] &= \int \frac{\delta R}{\delta f(x)} \, \delta f(x) \, dx = 0, \forall x, \\ &= \int \frac{\partial G}{\partial f} \, \delta f(x) \, dx = 0, \end{aligned}$$

or, equivalently,

$$\frac{\delta R}{\delta f(x)} = \frac{\delta G}{\delta f} = 0.$$

In general, however, it is not possible to set the functional derivative $\delta R/\delta f$ to zero, unless that is $f(x, \theta)$ is sufficiently flexible in which case $\delta R/\delta f$ can be set arbitrarily close to zero. There is considerable empirical evidence that deep neural networks possess the requisite degree of flexibility. Indeed, it has been shown that a network with a single hidden layer and a sufficient number of nodes is a universal approximator.

We now consider three standard loss functions

- Quadratic loss
- Absolute loss
- Exponential loss

Example (Quadratic Loss: $L(y, f) = (y - f)^2$)

For the quadratic loss,

$$G(f) = \int (y - f)^2 \, p(y, x) \, dy$$

$$= p(x) \int (y - f)^2 \, p(y|x) \, dy,$$

$$\frac{\partial G}{\partial f} = -2p(x) \int (y - f) \, p(y|x) \, dy = 0,$$

which implies $\boxed{f(x, \theta) = \int y \, p(y|x) \, dy, \text{ for some value of } \theta.}$

Conclusion If 1) the training data are sufficient and 2) $f(x, \theta)$ is sufficiently flexible and 3) we use the quadratic loss then $f(x, \theta)$ will approximate the *mean* of the conditional density $p(y|x)$.

Example (Absolute Loss: $L(y, f) = |y - f| \equiv \sqrt{(y-f)^2}$)

For the absolute loss,

$$G(f) = p(x) \int \sqrt{(y-f)^2}\, p(y|x)\, dy,$$

$$\frac{\partial G}{\partial f} = p(x) \int \frac{(y-f)}{|y-f|}\, p(y|x)\, dy = 0.$$

Noting that $(y-x)/|y-x| = 2H(y-f) - 1$, where $H(z)$ is the Heaviside function, $f(x, \theta)$ is the solution of $\boxed{\int_{y>f} p(y|x)\, dy = \frac{1}{2}}$.

Conclusion If 1) the training data are sufficient and 2) $f(x, \theta)$ is sufficiently flexible and 3) we use the absolute loss then $f(x, \theta)$ will approximate the *median* of $p(y|x)$.

Example (Exponential Loss: $L(y, f) = \exp(-y\, f)$)

For exponential loss, for which the loss is large if $y\, f < 0$ and small otherwise,

$$G(f) = p(x) \int \exp(-y\, f)\, p(y|x)\, dy,$$

$$\frac{\partial G}{\partial f} = -p(x) \int y\, \exp(-y\, f)\, p(y|x)\, dy = 0.$$

Suppose $y \in \{-1, +1\}$, then

$$\exp(-f)\, p(+1|x) - \exp(f)\, p(-1|x) = 0,$$

which implies $\boxed{f(x, \theta) = \frac{1}{2} \log p(+1|x) \,/\, p(-1|x)}$.

Comment The exponential loss is the basis of the AdaBoost algorithm (see Friedman, Hastie and Tibshirani, "Additive logistic regression: a statistical view of boosting", The Annals of Statistics, **28**, 377-386 (2000)).

Outline

- Deep neural networks are "just" one class of highly flexible functions of which perhaps many others remain to be discovered. Biology may yet again provide the inspiration.
- Since these functions are exceedingly complicated, I argue that the time spent deconstructing them in order to "understand what they are doing" would be better spent thinking of ways to use our understanding of what they approximate.
- We shall briefly consider two examples:
  1. Approximating likelihoods
  2. Background subtraction in blind

Apart from the discovery of the Higgs boson in 2012, the most significant result from the LHC is the complete absence of evidence of new physics so far, in particular, of supersymmetry.

One might therefore be inclined to conclude that the 124-parameter MSSM is approaching its demise. While this may well be true of the CMSSM, a once popular sub-model of the MSSM, this is far from being true of another sub-model, namely, the 19-parameter pMSSM[6].

In the course of his work on the pMSSM, a former student of mine, Sam Bein, was faced with the task of modeling the 19-parameter CMS likelihood, $L(x)$, of the pMSSM, which was available only as a swarm of points, where here x denotes the 19 parameters of the pMSSM.

[6] ATLAS collaboration, "Summary of the ATLAS experiments sensitivity to supersymmetry after LHC Run 1 – interpreted in the phenomenological MSSM", JHEP **1510**, 134 (2015); CMS collaboration, "Phenomenological MSSM interpretation of CMS searches in pp collisions at sqrt(s) = 7 and 8 TeV", JHEP **1610**, 129 (2016)

When a function $f(x, \theta)$ is trained (that is, fitted) using a training sample comprising two classes with target values $y = 1$ or $y = 0$, with equal numbers of elements in each class, and using, for example, the quadratic loss, the approximation to Bayes theorem yields the well-known result

$$f(x, \theta) \approx D(x) = \frac{p(x|1)}{p(x|1) + p(x|0)},$$

where we have labeled the classes by their target values. From this elegant result a wealth of applications spring to mind.

In particular, given $D(x)$ it follows that

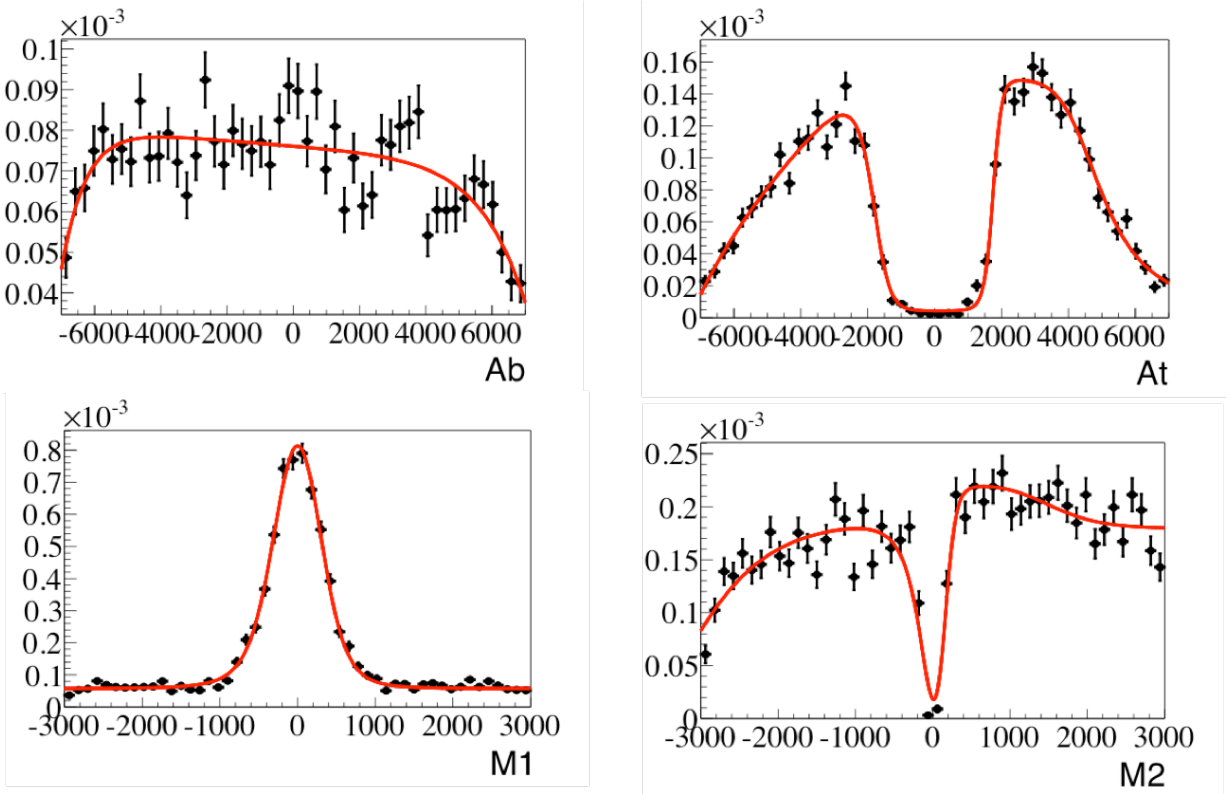
$$p(x|1) = \left[\frac{D(x)}{1 - D(x)}\right] p(x|0).$$

Therefore, given a model for $p(x|0)$ that approximates the likelihood, that approximation can be fined-tuned using $f(x, \theta) \approx D(x)$.

In the pMSSM application, the function $p(x|0)$ was chosen to be

$$p(x|0) = \prod_{i=1}^{19} g_i(x_i),$$

where the g_i are the 1-D marginals of the 19-dimensional likelihood, four of which are shown below:



The training data for class 1 consisted of a random sub-set of the original swarm of points that form the stochastic representation of the likelihood, while that for class 0 consisted of an equal number of points sampled from the product density $p(x|0)$ using standard tools from `ROOT`, thereby yielding the approximation

$$L(x) = \left[\frac{D(x)}{1 - D(x)}\right] \prod_{i=1}^{19} g_i(x_i),$$

for the 19-dimensional pMSSM likelihood $L(x)$.

Both the 1-D functions $g_i(x_i)$ as well as $D(x)$ were approximated with neural networks.

Consider the following task: find a signal with unknown strength μ and unknown density $s(x)$ knowing only the background density $b(x)$ and the data density $d(x)$, with the caveat that both are known only as large swarms of points in n-dimensions. The data density is presumed to be

$$d(x) = \mu s(x) + (1 - \mu) b(x),$$

Alas, so far at the LHC, $\mu \approx 0$, but this does not necessarily imply $\mu = 0$!

The result

$$D(x) = \frac{p(x|1)}{p(x|1) + p(x|0)},$$

suggests a possible way forwards.

Let $p(x|1) = d(x)$ and $p(x|0) = b(x)$ and train the function $f(x, \theta)$ using the same protocol as before.

From

$$D(x) = \frac{d(x)}{d(x) + b(x)},$$

it follows that

$$\frac{(1-\mu')b}{d} = (1-\mu')(D^{-1} - 1),$$

where $0 \leq \mu' < 1$. Define the weighting function

$$w(x, \mu') \equiv \frac{d - (1-\mu')b}{d} = 1 - (1-\mu')(D^{-1} - 1),$$

and repeatedly weight the data by $w(x, \mu')$ for a sequence of values of μ' between zero and one. If a signal exists, then when $\mu' = \mu$, the weighted data will recover the unknown signal!

On the other hand, if $\mu = 0$ the weighted data will simply return the background for all values of μ'.

Outline

- As we race towards a world of AI-enabled devices and services, we clearly need to build in the ability for such systems to explain their decisions and reasoning.

- However, in my view, it may be more fruitful to regard the highly flexible functions $f(x, \theta)$ of which deep neural networks are but one class as the basic building blocks of reasoning systems that may one day comprise thousands of such blocks in complex decision trees.

- As physicists, it may be more useful for us to exploit our understanding of what these functions approximate than to try to discern how they arrived at their "decisions".