An Exploration of Bayesian Neural Networks

Evan Hubinger

Harvey Mudd College

(Dated: May 6, 2018)

I. INTRODUCTION

The idea of a Bayesian neural network dates back to Neal (1995)[1], but has only relatively recently come to prominence with the rise of the use of deep neural networks by organizations such as DeepMind, Google Brain, and OpenAI. Bayesian neural networks are based on the idea of turning a standard neural network into a hierarchical Bayesian model. Thus, we will begin our exploration of Bayesian neural networks by delving into the workings of a traditional neural network.

II. TRADITIONAL NEURAL NETWORKS

Conceptually, a neural network is simply a particular type of universal function approximator, which takes in some data $\boldsymbol{x}$ and outputs some $\boldsymbol{y}$. There are many different types of neural networks, but for the purposes of this paper we will focus on standard feed-forward networks, which operate as follows.

Suppose we are given some input data $\boldsymbol{x}$. Let $\boldsymbol{a}_0 = \boldsymbol{x}$. Then, where W_0 is some weight matrix and $\boldsymbol{b}_0$ is some bias vector, compute

$$\boldsymbol{z}_1 = \boldsymbol{x}^T W_0 + \boldsymbol{b}_0$$

and, where f is some nonlinear function (called an activation function), compute

$$\boldsymbol{a}_1 = f(\boldsymbol{z}_1)$$

where f is applied element-wise. We call the dimensionality of $\boldsymbol{z}_1$ the number of "neurons" in the layer. Then, we repeat the above process on $\boldsymbol{a}_1$ using new weights W_1 and biases $\boldsymbol{b}_0$. Finally, let $\boldsymbol{y} = \boldsymbol{a}_n$ where n is the number of layers in the neural network.

This produces a model which maps input data $\boldsymbol{x}$ to output $\boldsymbol{y}$. Tuning the parameters of this sort of model is typically done via some variant of stochastic gradient descent.

III. BAYESIAN NEURAL NETWORKS

We can turn the above traditional neural network into a Bayesian neural network by putting priors on the weights and biases in the model. For example, let

$$W_i \sim N(0, \mathbb{1})$$
$$\boldsymbol{b}_i \sim N(\mathbf{0}, \mathbf{1})$$

Additionally, to allow for noise in the model, let

$$\boldsymbol{y} \sim N(\boldsymbol{a}_n, \sigma(\mathbf{1}))$$

where σ is a scalar representing the amount of noise in the model.

This produces a hierarchical Bayesian model. Updating this model with training data produces a posterior distribution over $\boldsymbol{y}$ which can be used to model the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Since Bayesian neural networks are full, hierarchical Bayesian models, they can be used to produce more than simply point estimates. In particular, the posterior model can be sampled from to produce estimates of the model's uncertainty.

IV. EXAMPLE

To showcase Bayesian neural networks, we will explore a toy Bayesian neural network. We will be using Edward for this, a Python library for probabilistic programming built on TensorFlow.[2] The example we will be using is modified from Edward's tutorials.[3] The full code for everything we will be doing can be accessed online on GitHub.[4]

We will be attempting to model the true data distribution

$$y \sim N(\sin(\pi(x+1)), 0.05)$$

For our training data, we sampled 100 data points uniformly on the range $x \in [-1, -0.1]$ and 100 data points uniformly on the range $x \in [0.1, 1]$. This allows us to leave the range $x \in [-0.1, 0.1]$ empty to test our model's ability at interpolation. See Figure 1 for the sampled training data.

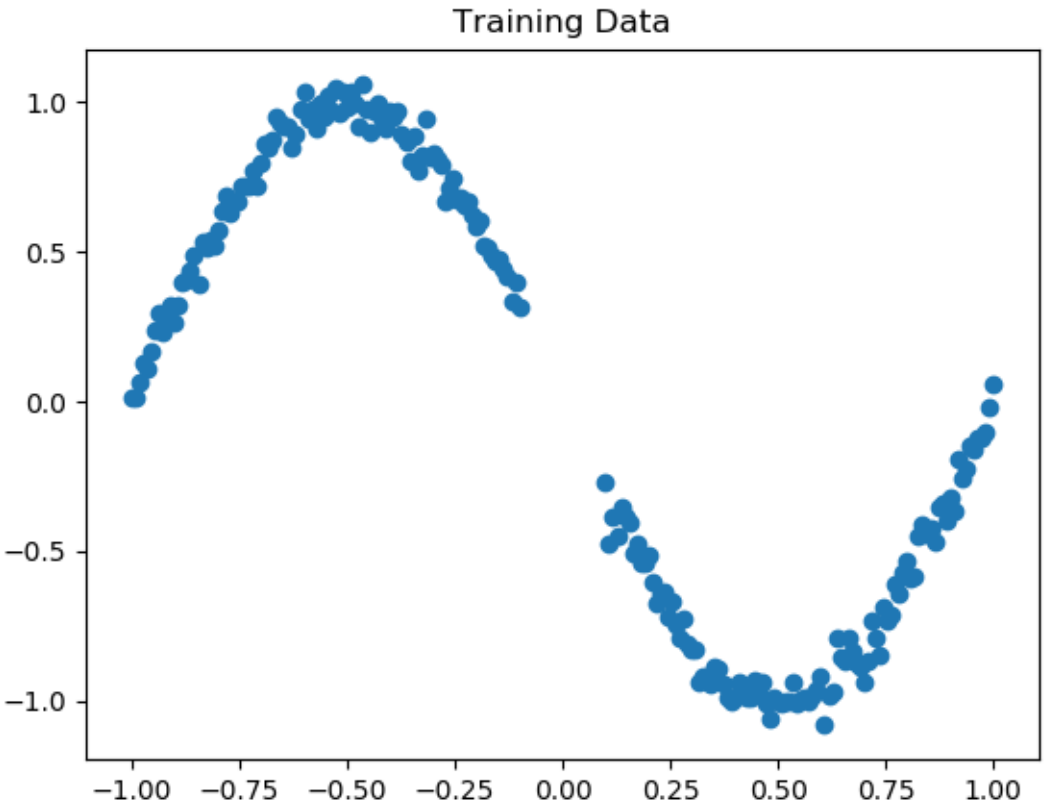

FIG. 1. Training data sampled using the methodology given above.

We then constructed a three-layer feed-forward Bayesian neural network with 10 neurons per layer and activation function $f(x) = \tanh(x)$. We used standard normal priors for the weights and biases and modeled $y \sim N(\boldsymbol{a}_n, \sigma)$. We estimated σ from the data using a finite difference method, which gave the approximation $\sigma \approx 0.0427$, which is very close to the true value $\sigma = 0.05$. To visualize our prior model, we sampled 10 different sets of weights and biases from the prior distribution and plotted the resulting models in Figure 2.

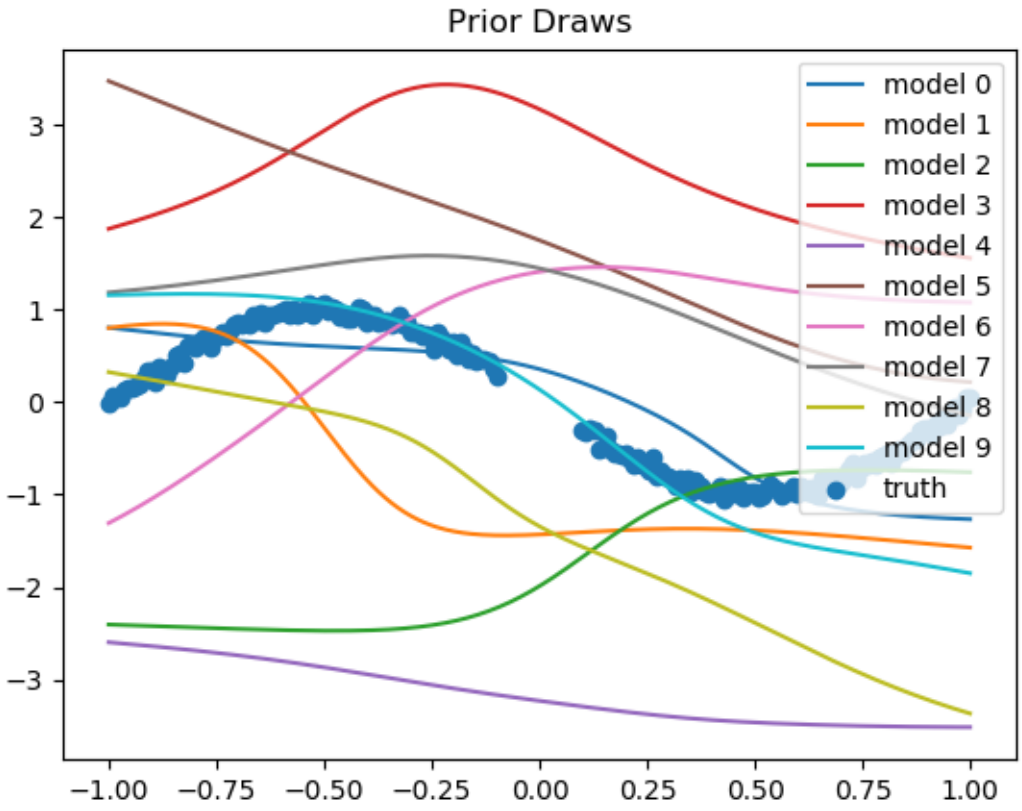

FIG. 2. 10 draws from the prior model distribution plotted alongside the actual data.

We then updated our network using the training data given above. Since an analytical approach to computing the posterior is impossible, we approximated the posterior using variational inference. Specifically, we used variational expectation maximization with a Kullback-Leibler divergence loss function to approximate model parameters.[5] We ran 1500 iterations of variational inference with 15 samples per iteration.

To visualize our posterior, we again sampled 10 different sets of weights and biases, the resulting models of which are plotted in Figure 3.

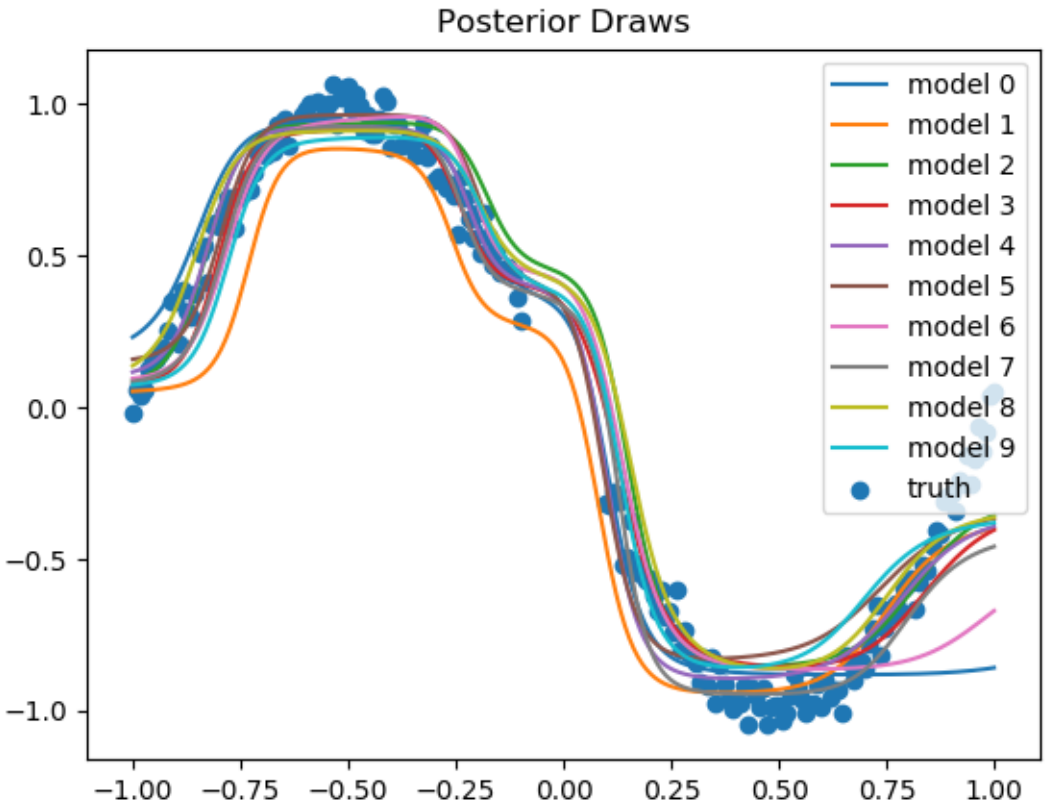

FIG. 3. 10 draws from the posterior model distribution plotted alongside the actual data.

If we take the posterior mean $\hat{y} = \bar{y}$ to be our point estimate for y, we can compare our model's mean estimate to the actual X, y data. See Figure 4 for the posterior means plotted alongside the ground truth.

Since we have a full posterior distribution, we can also compare samples from the posterior to the actual data using various different metrics. One example metric is simply $\bar{y}$, which should be

$$\begin{aligned}
&\mathbb{E}\left[\bar{y}\right] \\
&= E\left[\sin(\pi(x+1)) \mid x \in [-1,1]\right] \\
&= 0
\end{aligned}$$

A plot of the posterior $\bar{y}$ distribution estimated based on 400 samples alongside the actual data $\bar{y}$ is given in Figure 5.

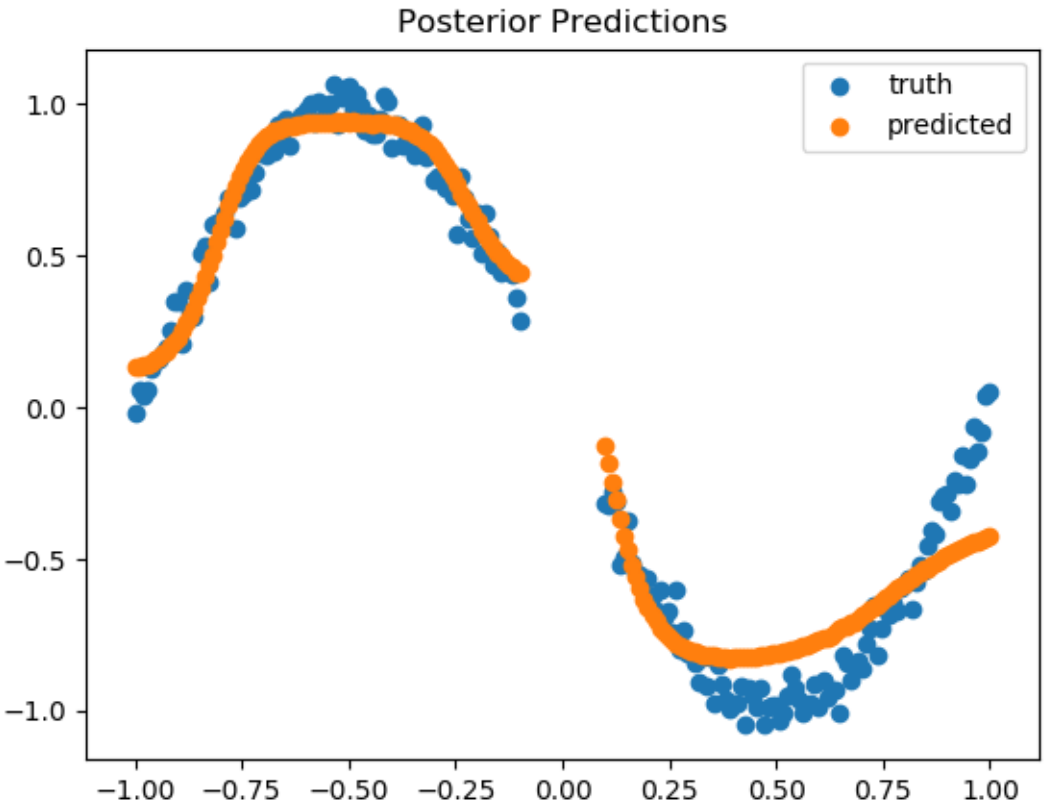

FIG. 4. Posterior mean estimates plotted alongside the actual data.

V. CONCLUSION

[1] Radford Neal, "Bayesian learning for neural networks," (1995).

[2] David Blei et al., "Edward: A probabilistic programming language in tensorflow. deep generative models, variational inference." (2018).

[3] David Blei et al., "Edward tutorial: Bayesian neural network," (2018).

[4] Evan Hubinger, "A toy bayesian neural network example," (2018).

[5] David Blei et al., "Edward api reference: Class klpq," (2018).

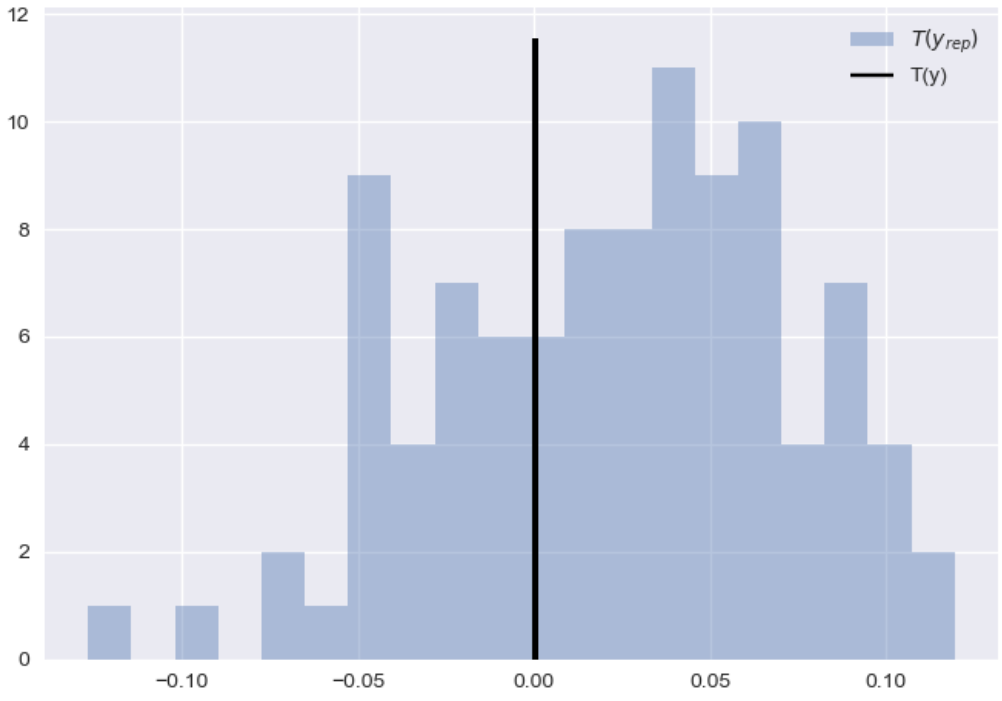

FIG. 5. Actual $\bar{y}$ plotted alongside an estimate of the posterior $\bar{y}$ distribution.