Bayesian Deep Learning

Solving Problems in Computational Chemical Biology

Gabriel Ong[1]

[1] Karanicolas Lab
Program in Molecular Therapeutics - Fox Chase Cancer Center

Group Meeting, 28th October 2020

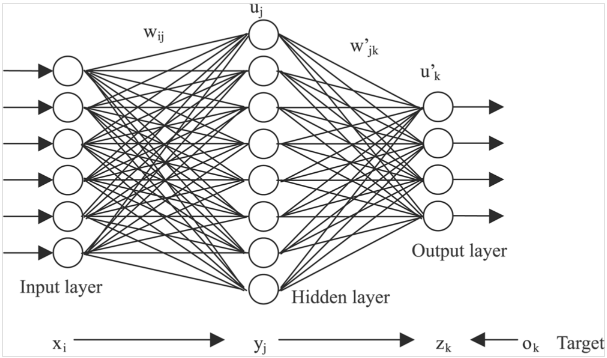
Table of Contents

Table of Contents

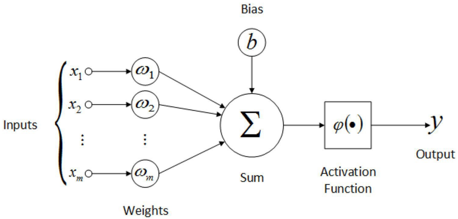
1. A Review of Deep Learning
2. What are Bayesian Neural Networks?
3. Applications to Computational Chemical Biology
4. Conclusions and Thanks

What are Neural Networks?

Neural networks are large networks of artificial "neurons" inspired by the human brain.



What are Neural Networks?



How do neural networks work?

- Aggregate inputs from all neurons in the previous layer.
- Alter the aggregated inputs by some bias term.
- Pass this data through an activation function - this way an output is 0 if the data is irrelevant.

Neural Networks: A Different Perspective

Let's think of neural networks another way: recognizing and reproducing high dimensional patterns.

Neural Networks as Functions

$$F_{\mathcal{N}} : \mathbb{R}^n \to \mathbb{R}^m$$

Given an *n*-dimensional input, find the corresponding *m*-dimensional output.

Neural Networks: A Different Perspective

So let's think of neural networks as a function depending on two objects:

- x_i, the n-dimensional input
- θ, the weights and biases of the neural network

Ideally this function would return the correct output, y_i so

$$F_{\mathcal{N}}(\theta, x_i) = y_i.$$

However, we would have to be really lucky for the function to return the correct y_i for a randomly initialized θ so we need to "teach" the machine the correct set of weights and biases.

A "Functional" View of Machines Learning

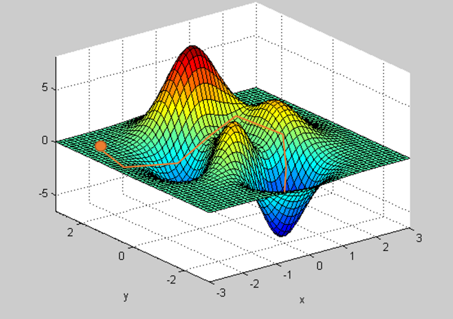
The process of learning is then choosing the correct θ, the correct sets of weights and biases, that accurately approximates y_i for a given input x_i.

We can phrase this as an optimization problem: choose $\hat{\theta}$ that minimizes error over a training set $(x_1, y_1), \ldots, (x_s, y_s)$.

$$\text{error} = \frac{\sum_{i=1}^{s} |y_i - F_{\mathcal{N}}(\theta, x_i)|^2}{s}$$

A “Functional” View of Machines Learning

How do we choose $\hat{\theta}$?



More recent work (Kingma and Ba, 2014) has outlined Adam, or stochastic gradient descent with momentum, as the most reliable way to achieve $\hat{\theta}$.

Recalling Sophomore Linear Algebra

Let's use some intuition from linear algebra (even though neural networks are non-linear).

A Severely Under-determined System

$$\dim \hat{\theta} >>> s$$

The number of tunable weights and biases is far larger than the size of our training data set. We often train networks with millions of tunable parameters with only tens of thousands of datapoints.

Recalling Sophomore Linear Algebra

Think of this as trying to solve a system of tens of thousands of equations with millions of variables.

Recalling Sophomore Linear Algebra

Think of this as trying to solve a system of tens of thousands of equations with millions of variables.

- The system is under-determined: we do not have enough constraints (training data) for a unique solution.

Recalling Sophomore Linear Algebra

Think of this as trying to solve a system of tens of thousands of equations with millions of variables.

- The system is under-determined: we do not have enough constraints (training data) for a unique solution.
- In other words, there are many possible minima.

A Caveat: The Problem of Generalization

- Since $\hat{\theta}$ gave minimum average error on the training set, $|F_{\mathcal{N}}(\hat{\theta}, x_i) - y_i|^2$ is small for $(x_i, y_i) \in (x_1, y_1), \ldots, (x_s, y_s)$.

A Caveat: The Problem of Generalization

- Since $\hat{\theta}$ gave minimum average error on the training set, $|F_{\mathcal{N}}(\hat{\theta}, x_i) - y_i|^2$ is small for $(x_i, y_i) \in (x_1, y_1), \ldots, (x_s, y_s)$.
- Is $|F_{\mathcal{N}}(\hat{\theta}, x_j) - y_j|^2$ also small for $(x_j, y_j) \notin (x_1, y_1), \ldots, (x_s, y_s)$?

A Caveat: The Problem of Generalization

- Since $\hat{\theta}$ gave minimum average error on the training set, $|F_{\mathcal{N}}(\hat{\theta}, x_i) - y_i|^2$ is small for $(x_i, y_i) \in (x_1, y_1), \ldots, (x_s, y_s)$.
- Is $|F_{\mathcal{N}}(\hat{\theta}, x_j) - y_j|^2$ also small for $(x_j, y_j) \notin (x_1, y_1), \ldots, (x_s, y_s)$?
- In other words, does this hold for previously unseen data?

A Caveat: The Problem of Generalization

- Since $\hat{\theta}$ gave minimum average error on the training set, $|F_{\mathcal{N}}(\hat{\theta}, x_i) - y_i|^2$ is small for $(x_i, y_i) \in (x_1, y_1), \ldots, (x_s, y_s)$.
- Is $|F_{\mathcal{N}}(\hat{\theta}, x_j) - y_j|^2$ also small for $(x_j, y_j) \notin (x_1, y_1), \ldots, (x_s, y_s)$?
- In other words, does this hold for previously unseen data?

Answer: Sometimes...

Motivating Question

Can we do better?

Table of Contents

1 A Review of Deep Learning

2 What are Bayesian Neural Networks?

3 Applications to Computational Chemical Biology

4 Conclusions and Thanks

Bayesian Neural Networks: An Introduction

Bayesian Neural Networks (BNNs) are approaches where uncertainty of model parameters are directly incorporated into the model. Fundamentally, Bayesian Deep Learning considers *marginalization* over *optimization*:

- Represents not a single setting of weights and biases but all possible settings weighted by posterior probability in a Bayesian model average.
- Accounts for complementary explanations of data by characterizing epistemic uncertainty. Many parameter settings fit data but which is the right (generalizing) explanation?
- Better parametrization of noise in data.

[AW20]

Bayesian Neural Networks: An Introduction

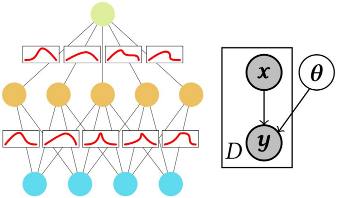
Often the predictive distribution we wish to calculate is given by the following integral:

$$p(y|x, \mathcal{D}) = \int p(y|x, \theta) p(\theta|\mathcal{D}) d\theta.$$

For inputs x outputs y and data size $\mathcal{D}$, the integral represents the Bayesian model average where all settings θ are weighted by posterior probabilities. This integral is usually calculated numerically - often impossible to solve analytically.

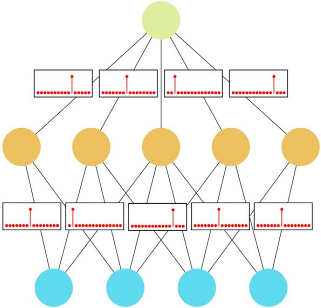
[AW20]

Bayesian Neural Networks: An Introduction



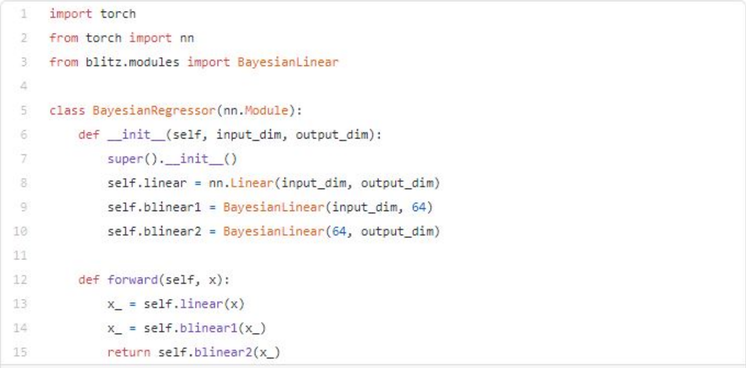
Bayesian neural networks consider all possible parameters over a probability distribution.

[LVJ20]



Traditional neural networks only consider one set of parameters.

Bayesian Neural Networks: Implementation



```python
import torch
from torch import nn
from blitz.modules import BayesianLinear

class BayesianRegressor(nn.Module):
    def __init__(self, input_dim, output_dim):
        super().__init__()
        self.linear = nn.Linear(input_dim, output_dim)
        self.blinear1 = BayesianLinear(input_dim, 64)
        self.blinear2 = BayesianLinear(64, output_dim)

    def forward(self, x):
        x_ = self.linear(x)
        x_ = self.blinear1(x_)
        return self.blinear2(x_)
```

Packages for Bayesian deep learning are available in PyTorch making the technique easily accessible.

Table of Contents

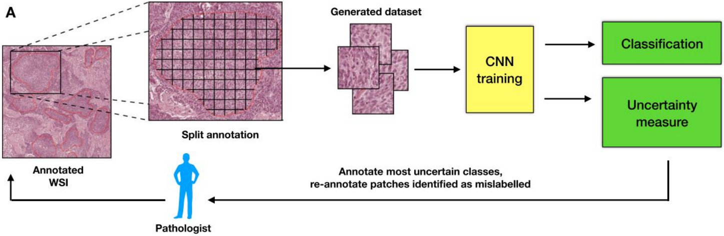
1. A Review of Deep Learning
2. What are Bayesian Neural Networks?
3. Applications to Computational Chemical Biology
4. Conclusions and Thanks

How has Bayesian Deep Learning been applied to computational chemical biology tasks?

Looking at some previous applications:

- Histopathological Image Classification (Raczkowski, *et.al.*)
- Somatic Variant Calling in Cancer (Dubourg-Felonneau, *et.al.*)

Histopathological Image Classification (Raczkowski, *et.al.*)



Histopathological Image Classification (Raczkowski, *et.al.*)

“This model achieves exceptional classification accuracy, outperforming models trained on the same dataset. The network outputs an uncertainty measurement for each tested image.”

- Bayesian deep learning in combination with dropout and batch normalization provide state of the art classification accuracy.

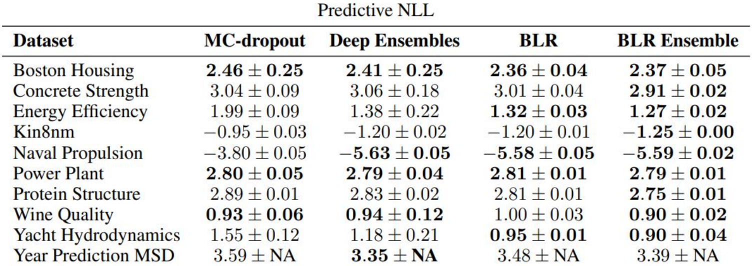
Somatic Variant Calling (Dubourg-Felonneau, *et.al.*)

“In addition to demonstrating similar performance in comparison to standard neural networks, we show that the resultant output probabilities make these better suited to the disparate and highly-variable sequencing data-sets these models are likely to encounter in the real world.”

- Biological data is often highly variable yet arise from connected patterns. Bayesian neural networks with its capacity to represent epistemic uncertainty allows for a better reproduction of underlying patterns in data.

Can we use this in DTI regression tasks?

Promising results from Bayesian regression (Moberg, *et.al.*):



Predictive NLL

Dataset	MC-dropout	Deep Ensembles	BLR	BLR Ensemble
Boston Housing	**2.46 ± 0.25**	**2.41 ± 0.25**	**2.36 ± 0.04**	**2.37 ± 0.05**
Concrete Strength	3.04 ± 0.09	3.06 ± 0.18	3.01 ± 0.04	**2.91 ± 0.02**
Energy Efficiency	1.99 ± 0.09	1.38 ± 0.22	**1.32 ± 0.03**	**1.27 ± 0.02**
Kin8nm	−0.95 ± 0.03	−1.20 ± 0.02	−1.20 ± 0.01	**−1.25 ± 0.00**
Naval Propulsion	−3.80 ± 0.05	**−5.63 ± 0.05**	**−5.58 ± 0.05**	**−5.59 ± 0.02**
Power Plant	**2.80 ± 0.05**	**2.79 ± 0.04**	**2.81 ± 0.01**	**2.79 ± 0.01**
Protein Structure	2.89 ± 0.01	2.83 ± 0.02	2.81 ± 0.01	**2.75 ± 0.01**
Wine Quality	**0.93 ± 0.06**	**0.94 ± 0.12**	1.00 ± 0.03	**0.90 ± 0.02**
Yacht Hydrodynamics	1.55 ± 0.12	1.18 ± 0.21	**0.95 ± 0.01**	**0.90 ± 0.04**
Year Prediction MSD	3.59 ± NA	**3.35 ± NA**	3.48 ± NA	3.39 ± NA

BNN regression outperforms traditional NN and traditional deep ensemble approaches.

Considerations for Further Work

How can we use this in our kinase binding affinity model?

Considerations for Further Work

How can we use this in our kinase binding affinity model?

- Consider implementing an *n*-last layer BNN in our regression model. It is often not needed to propagate epistemic uncertainty across the entire model.

Considerations for Further Work

How can we use this in our kinase binding affinity model?

- Consider implementing an *n*-last layer BNN in our regression model. It is often not needed to propagate epistemic uncertainty across the entire model.
- Consider deep ensemble or Bayesian deep ensemble approaches where multiple models are trained and predictions are averaged out, though this may be computationally expensive.

Table of Contents

1 A Review of Deep Learning

2 What are Bayesian Neural Networks?

3 Applications to Computational Chemical Biology

4 Conclusions and Thanks

Thanks and Acknowledgements

Thanks to:

- John Karanicolas
- Kiruba Palani
- Jake Khowsathit
- Chris Parry
- Grigorii Andrianov

FOX CHASE
CANCER CENTER
TEMPLE HEALTH

- Daniel Yeggoni
- Sven Miller
- Lei Ke

References

[AW20] Wilson, Andrew Gordon. *Bayesian Deep Learning and a Probabalistic Perspective of Model Construction*. International Conference of Machine Learning, 2020

[LVJ20] Jospin, Laurent Valentin, et. al. *Hands-on Bayesian Neural Networks*. ArXiv Preprint 2007.06823v1, 2020