Probabilistic Programming

Marius Popescu

popescunmarius@gmail.com

2019 - 2020

Project 2

Bayesian Neural Networks

Bayesian Neural Networks

ACM DL DIGITAL LIBRARY

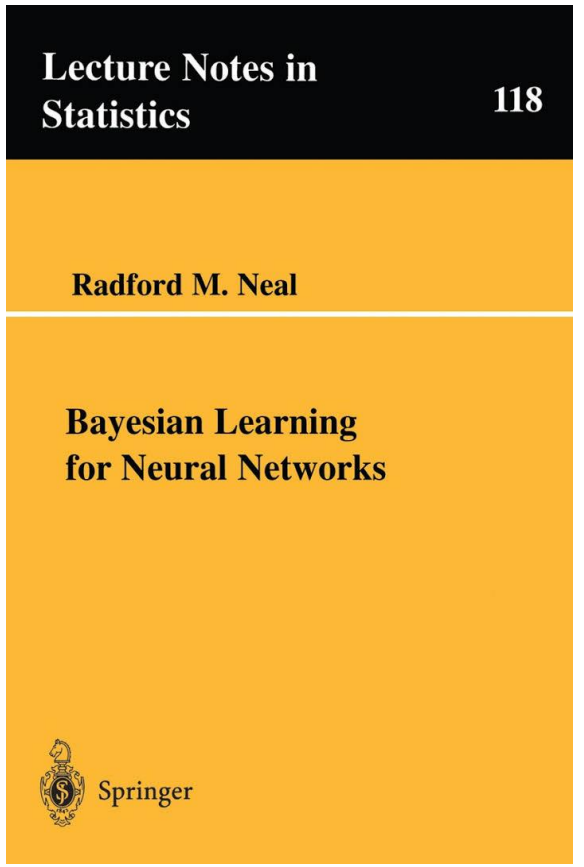
Bayesian learning for neural networks

1995 Doctoral Dissertation

Advisers: Geoffrey Hinton
Authors: Radford M. Neal

Bibliometrics
· Citation Count: 56
· Downloads (cumulative): n/a
· Downloads (12 Months): n/a
· Downloads (6 Weeks): n/a

Publication:
· Doctoral Dissertation
Bayesian learning for neural networks
University of Toronto Toronto, Ont., Canada, Canada ©1995
ISBN:0-612-02676-0



http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.446.9306&rep=rep1&type=pdf

Bayesian Neural Networks

A Bayesian neural network is a neural network with a prior distribution on its weights

Define the prior on the weights and biases $\mathbf{w}$ to be the standard normal:

$$p(\mathbf{w}) = \text{Normal}(\mathbf{0}, \mathbf{I})$$

Define the likelihood for a data point as:

$$p(y|\mathbf{x}, \mathbf{w}) = \text{Bernoulli}(NN(\mathbf{x}, \mathbf{w}))$$

Where $\mathbf{x} \in \mathbb{R}^d$ is the feature vector, $y \in \{0,1\}$ is the output (binary classification), and $NN(\mathbf{x}, \mathbf{w})$ is the function (deterministic) implemented by the neural network whose weights and biases form the latent variables $\mathbf{w}$. The output of the neural network must be a value between 0 and 1 (for example the last layer has only one sigmoid neuron).

Or:

$$p(y|\mathbf{x}, \mathbf{w}) = \text{Categorical}(NN(\mathbf{x}, \mathbf{w}))$$

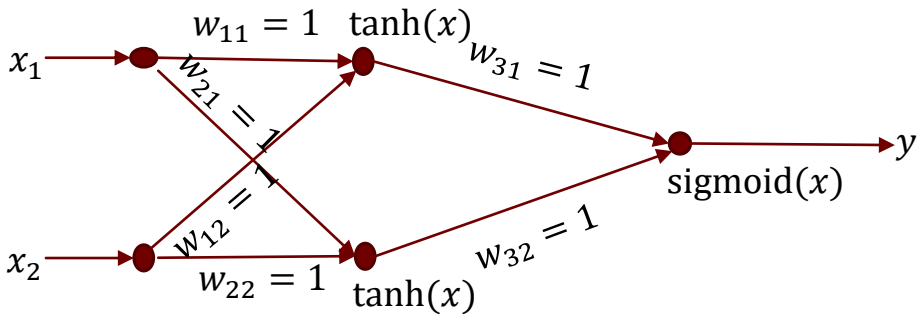
Where $\mathbf{x} \in \mathbb{R}^d$ is the feature vector, $y \in \{1,2, \ldots k\}$ is the output (multiclass classification), and $NN(\mathbf{x}, \mathbf{w})$ is the function (deterministic) implemented by the neural network whose weights and biases form the latent variables $\mathbf{w}$. The outputs of the neural network must sum to 1 (for example the last layer is a softmax layer).

The Task

- Start with a network architecture
- Training: using the training data (observed) infer the posterior distribution of $\mathbf{w}$
- Testing: using this posterior distribution predict the labels of testing data
- Compare to "classic" neural networks results on the same problem
- Do this for both binary classification and multiclass classification

Sanity Check

Take a simple neural network with fixed weights

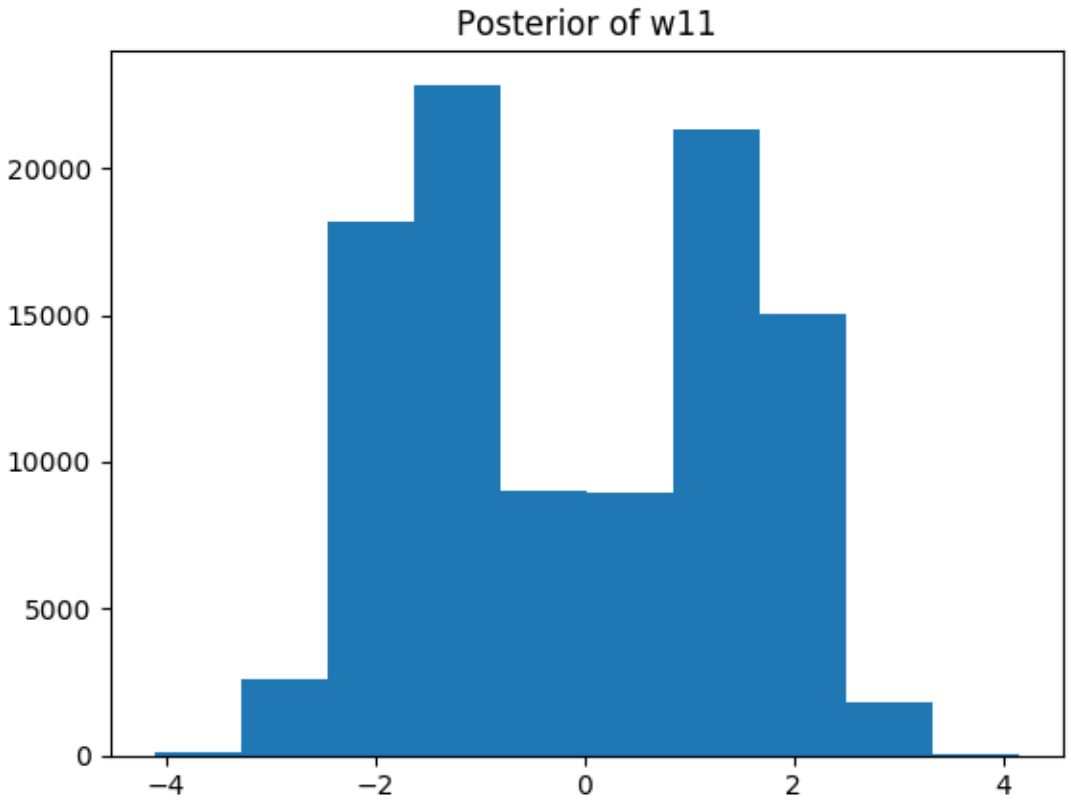


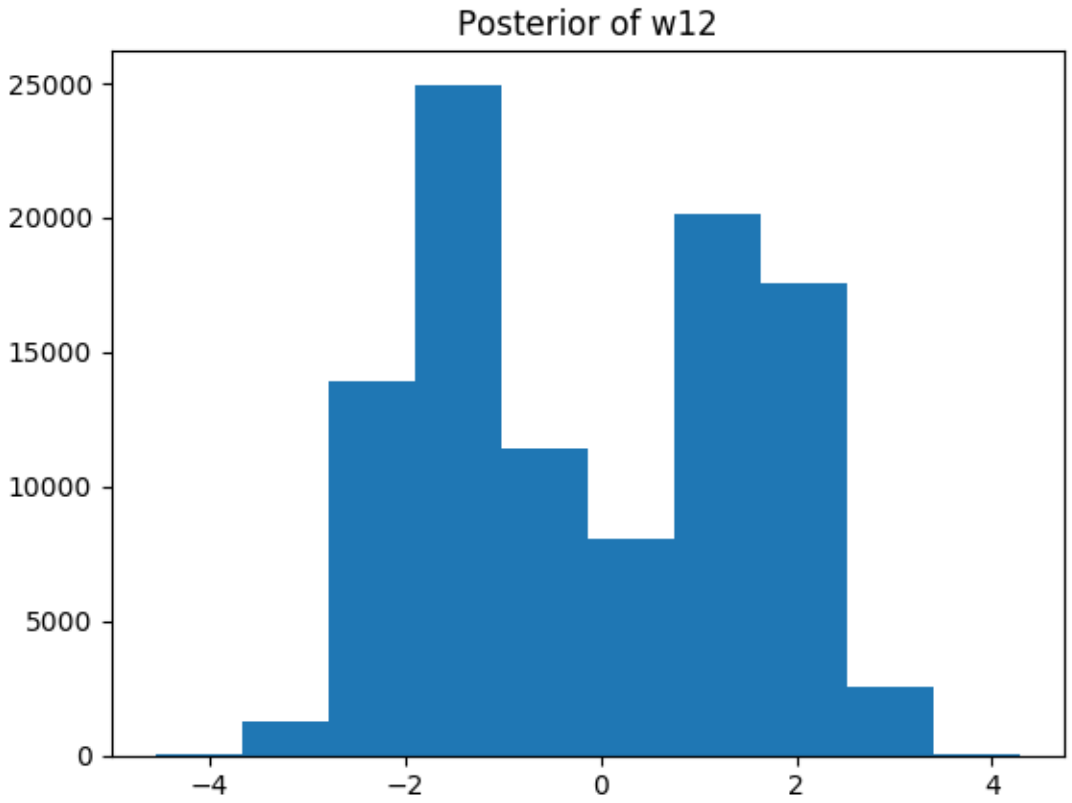
Generate labeled data (100 training, 100 testing) by feeding random input vectors into this network:

```
X = np.random.randn(100, 2)
Y = np.tanh(X[:, 0] + X[:, 1])
Y = 1. / (1. + np.exp(-(Y + Y)))
Y = Y > 0.5
```

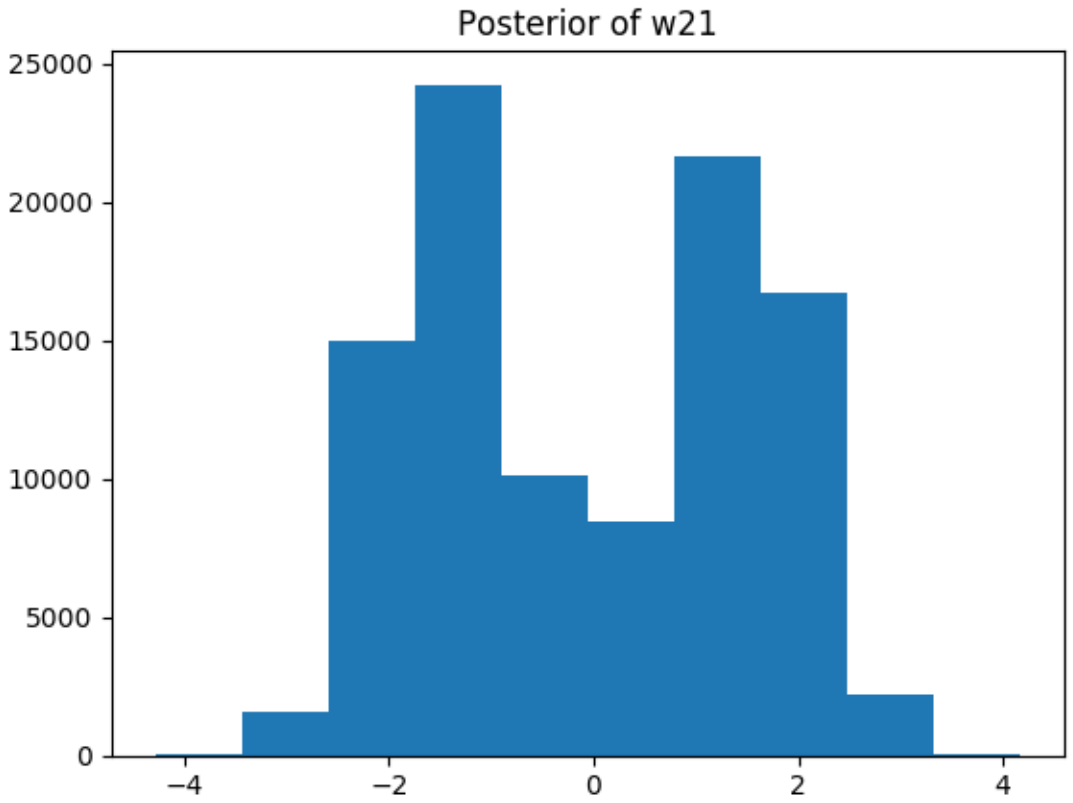
- "Train" a Bayesian neural network with the same architecture
- Analyse the posterior of weights
- Test it

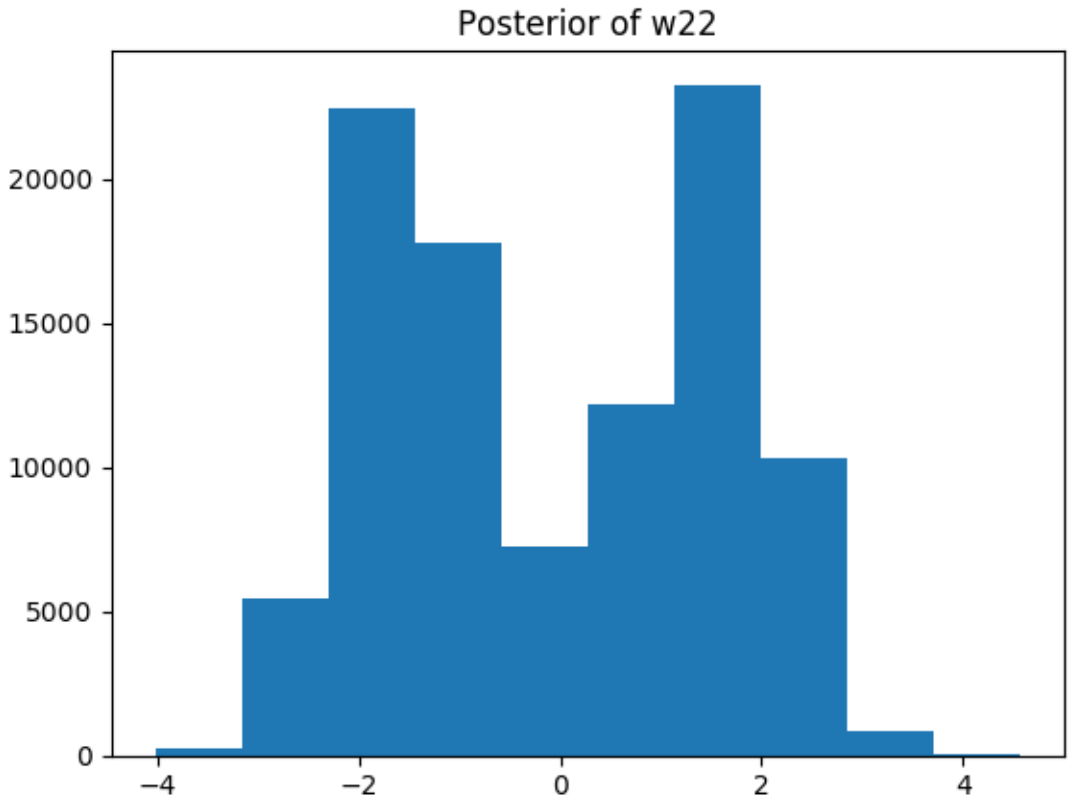
Sanity Check



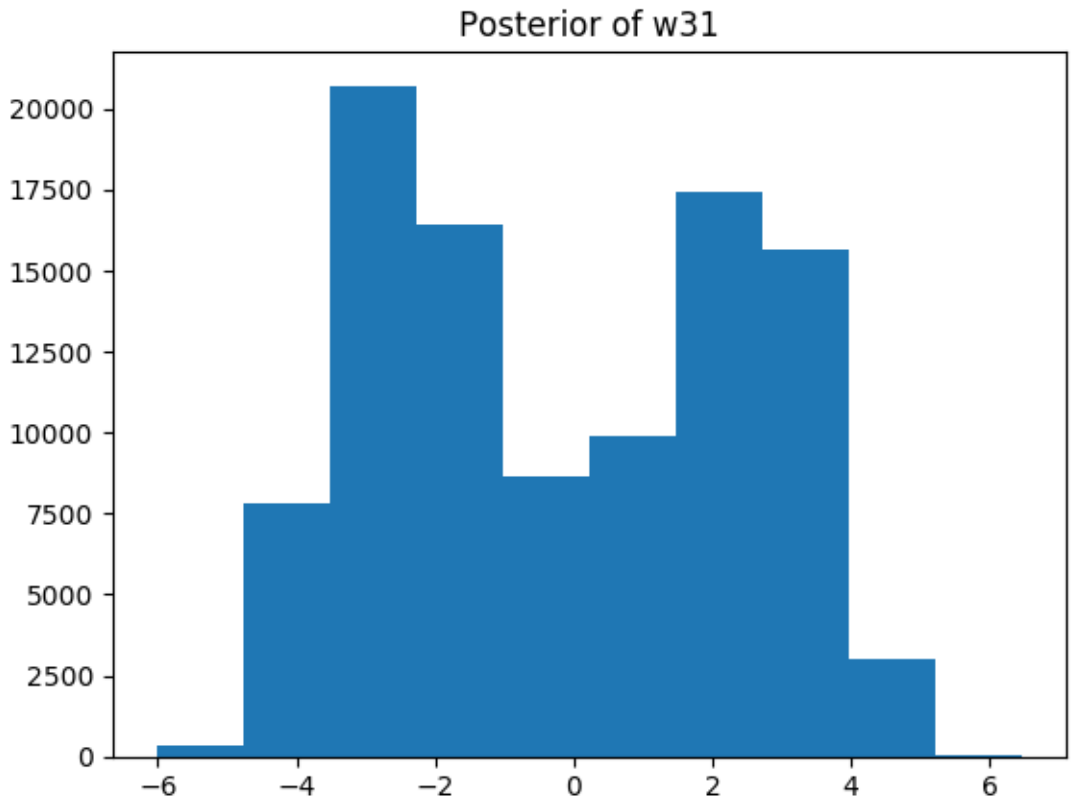


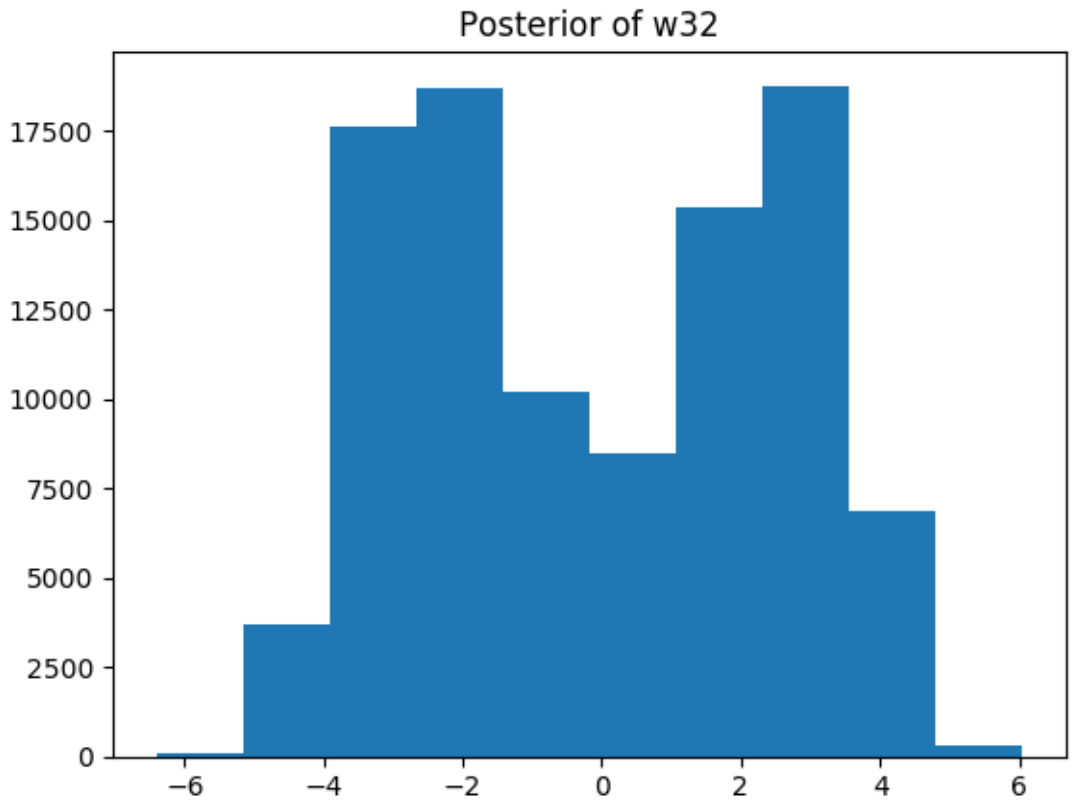
Sanity Check



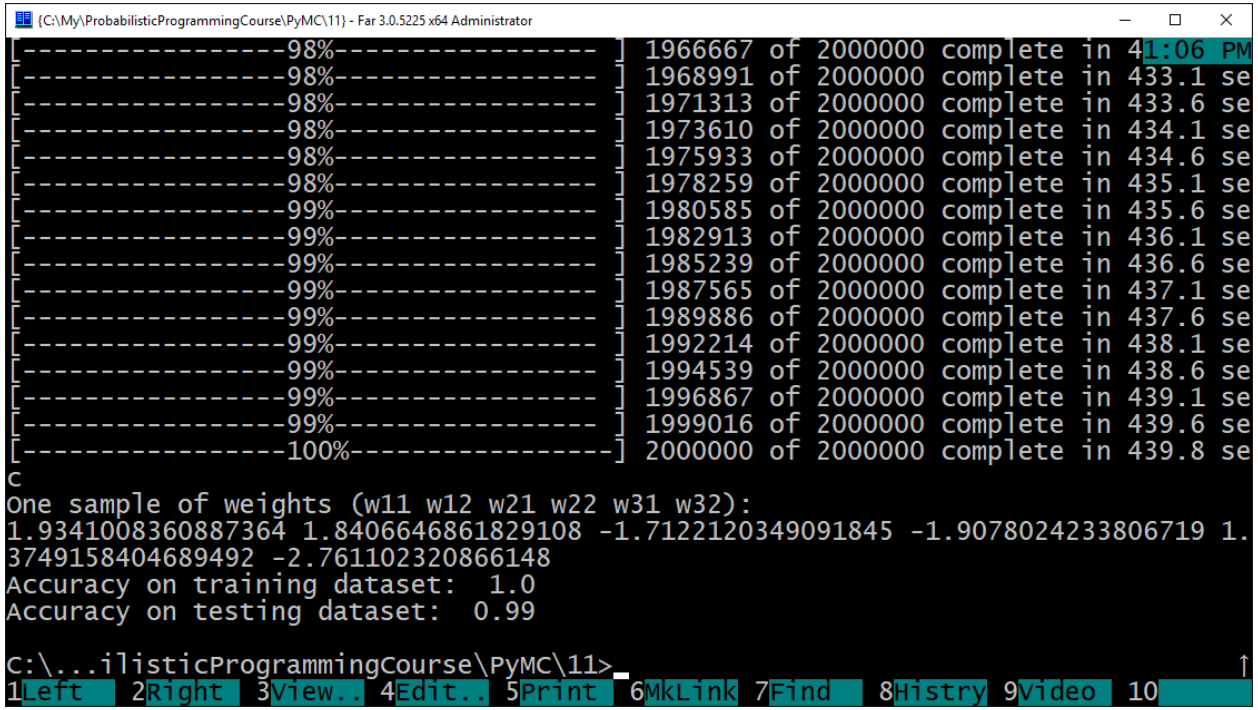


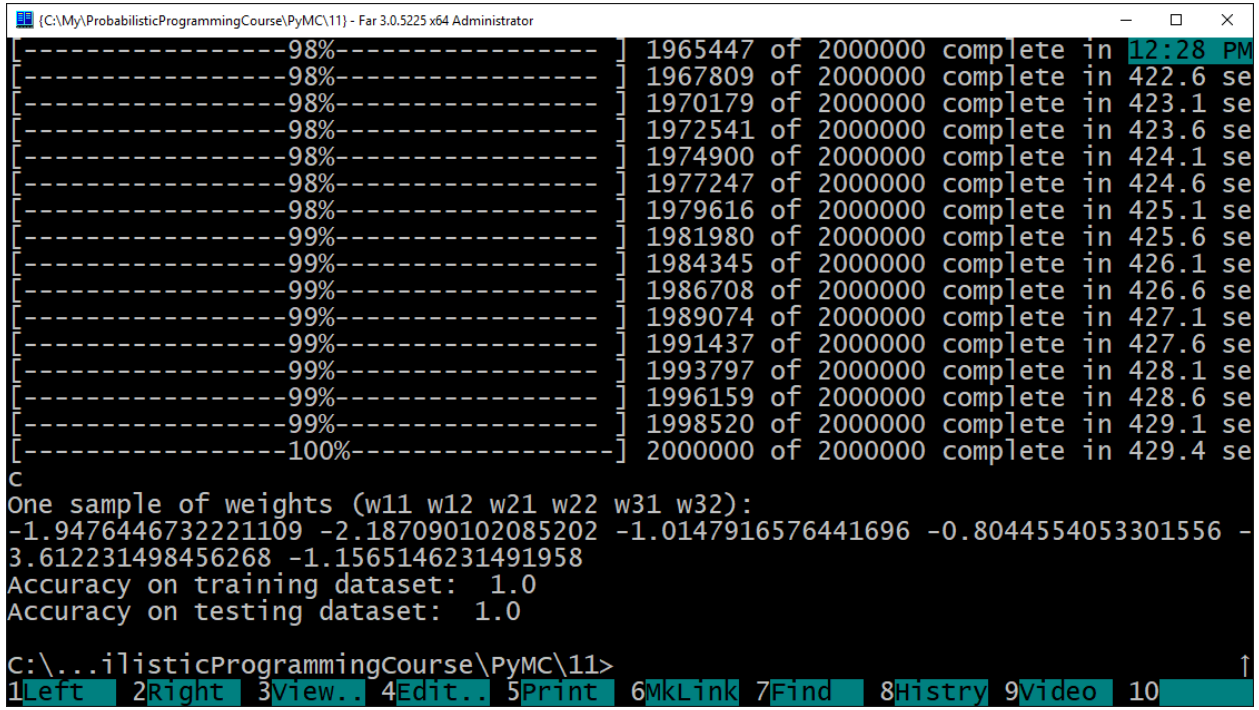
Sanity Check



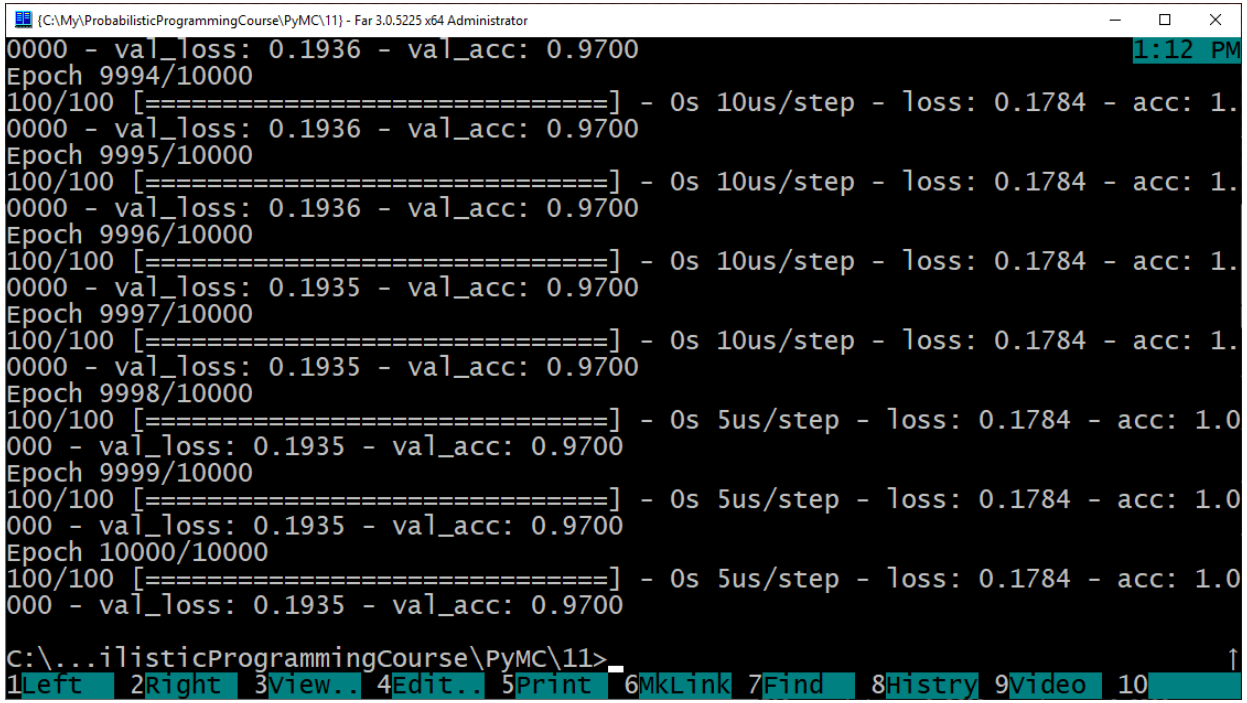


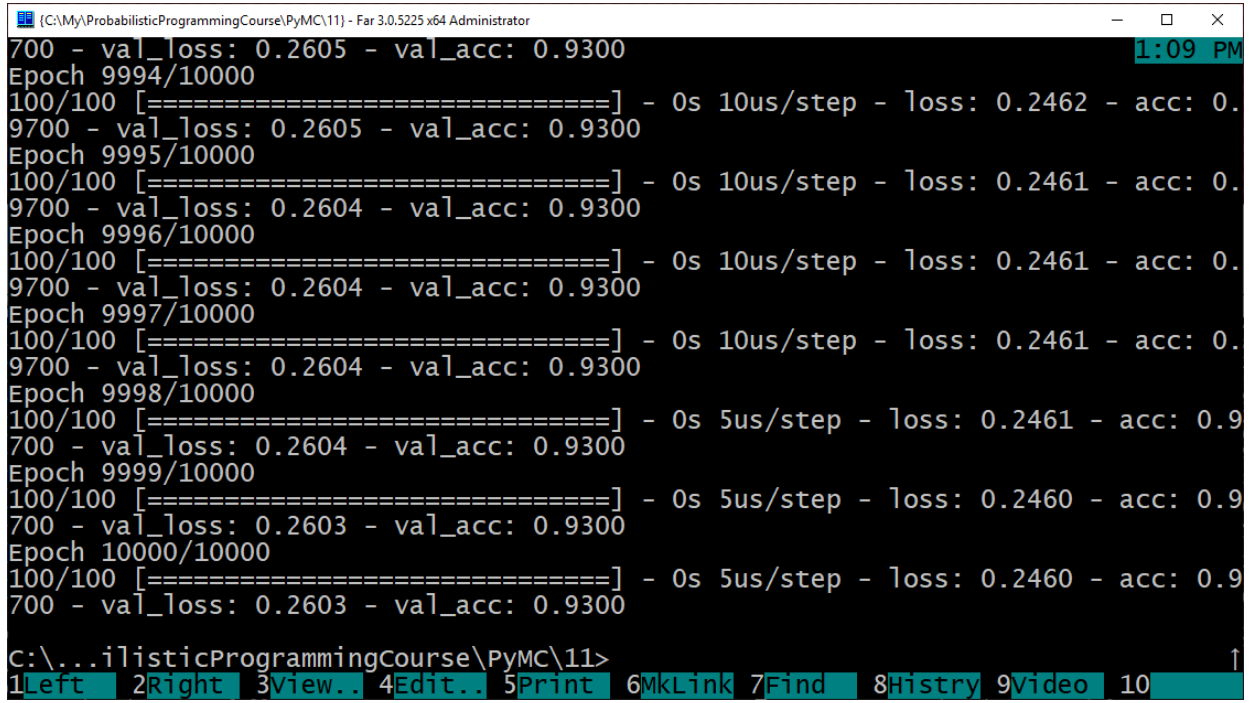
Sanity Check



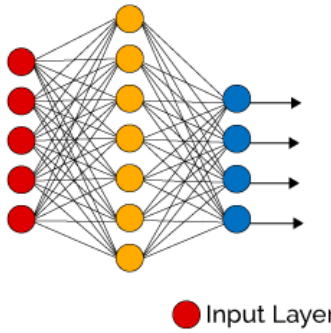


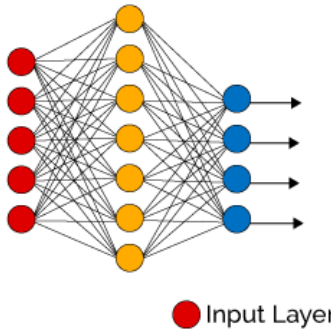
Comparison to “classic” neural networks





Overparametrization May Help Optimization : Folklore Experiment





Generate labeled data by feeding random input vectors Into depth 2 net with hidden layer of size n

Still no theorem explaining this…

Difficult to train a new net using this labeled data with same # of hidden nodes

Much easier to train a new net with bigger hidden layer!

Adapted from Sanjeev Arora's ICML 2018 tutorial "Toward Theoretical Understanding of Deep Learning" (http://unsupervised.cs.princeton.edu/deeplearningtutorial.html)

Other small datasets

"Two Moons" dataset: toy dataset for a simple binary classification problem that's not linearly separable.

```
sklearn.datasets.make_moons(noise=0.2, random_state=0, n_samples=1000)
```

Extras

- Extensions (up to 1.5 points):
  - Priors over architectures
  - Weight sharing (CNN, Siamese Networks)
  - for more ideas see chapter 2 from “Bayesian Learning for Neural Networks”
  - …