Week 9 Presentation

PHY 496

BRADEN KRONHEIM

MARCH 22, 2019

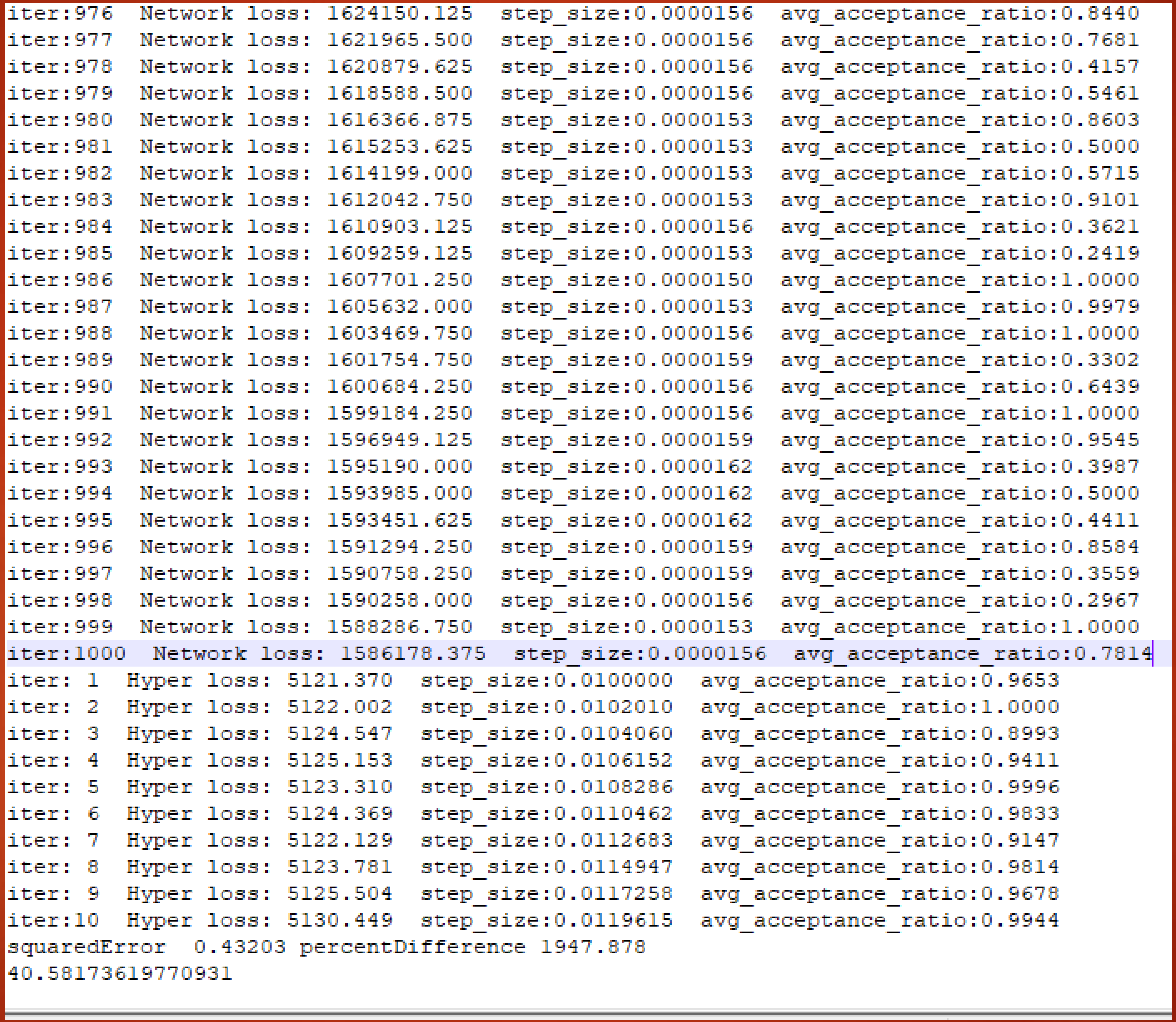
Summary

- Cleaned code to make the Bayesian Neural Networks more easily implemented for general regression problems
- Started experimenting with the SUSY data
    - One run was very effective but I can’t reproduce it

Implementation

- Create a Network object with a datatype and training and validation data
- Add layers with variable number of input and output neurons
  - Layers currently default to a relu activation function except the final, which is a linear activation
  - Layers also currently have set priors which should become customizable later
- Setup the MCMC with a step size and number of leapfrog steps for the general training and the hyper parameter training
- Train the network by specififying number of epochs, how often to update hypers, for how many epochs the hyper training should be run, when to start collecting network results, and how often to do this

Output



```
iter:976  Network loss: 1624150.125  step_size:0.0000156  avg_acceptance_ratio:0.8440
iter:977  Network loss: 1621965.500  step_size:0.0000156  avg_acceptance_ratio:0.7681
iter:978  Network loss: 1620879.625  step_size:0.0000156  avg_acceptance_ratio:0.4157
iter:979  Network loss: 1618588.500  step_size:0.0000156  avg_acceptance_ratio:0.5461
iter:980  Network loss: 1616366.875  step_size:0.0000153  avg_acceptance_ratio:0.8603
iter:981  Network loss: 1615253.625  step_size:0.0000153  avg_acceptance_ratio:0.5000
iter:982  Network loss: 1614199.000  step_size:0.0000153  avg_acceptance_ratio:0.5715
iter:983  Network loss: 1612042.750  step_size:0.0000153  avg_acceptance_ratio:0.9101
iter:984  Network loss: 1610903.125  step_size:0.0000156  avg_acceptance_ratio:0.3621
iter:985  Network loss: 1609259.125  step_size:0.0000153  avg_acceptance_ratio:0.2419
iter:986  Network loss: 1607701.250  step_size:0.0000150  avg_acceptance_ratio:1.0000
iter:987  Network loss: 1605632.000  step_size:0.0000153  avg_acceptance_ratio:0.9979
iter:988  Network loss: 1603469.750  step_size:0.0000156  avg_acceptance_ratio:1.0000
iter:989  Network loss: 1601754.750  step_size:0.0000159  avg_acceptance_ratio:0.3302
iter:990  Network loss: 1600684.250  step_size:0.0000156  avg_acceptance_ratio:0.6439
iter:991  Network loss: 1599184.250  step_size:0.0000156  avg_acceptance_ratio:1.0000
iter:992  Network loss: 1596949.125  step_size:0.0000159  avg_acceptance_ratio:0.9545
iter:993  Network loss: 1595190.000  step_size:0.0000162  avg_acceptance_ratio:0.3987
iter:994  Network loss: 1593985.000  step_size:0.0000162  avg_acceptance_ratio:0.5000
iter:995  Network loss: 1593451.625  step_size:0.0000162  avg_acceptance_ratio:0.4411
iter:996  Network loss: 1591294.250  step_size:0.0000159  avg_acceptance_ratio:0.8584
iter:997  Network loss: 1590758.250  step_size:0.0000159  avg_acceptance_ratio:0.3559
iter:998  Network loss: 1590258.000  step_size:0.0000156  avg_acceptance_ratio:0.2967
iter:999  Network loss: 1588286.750  step_size:0.0000153  avg_acceptance_ratio:1.0000
iter:1000  Network loss: 1586178.375  step_size:0.0000156  avg_acceptance_ratio:0.7814
iter: 1  Hyper loss: 5121.370  step_size:0.0100000  avg_acceptance_ratio:0.9653
iter: 2  Hyper loss: 5122.002  step_size:0.0102010  avg_acceptance_ratio:1.0000
iter: 3  Hyper loss: 5124.547  step_size:0.0104060  avg_acceptance_ratio:0.8993
iter: 4  Hyper loss: 5125.153  step_size:0.0106152  avg_acceptance_ratio:0.9411
iter: 5  Hyper loss: 5123.310  step_size:0.0108286  avg_acceptance_ratio:0.9996
iter: 6  Hyper loss: 5124.369  step_size:0.0110462  avg_acceptance_ratio:0.9833
iter: 7  Hyper loss: 5122.129  step_size:0.0112683  avg_acceptance_ratio:0.9147
iter: 8  Hyper loss: 5123.781  step_size:0.0114947  avg_acceptance_ratio:0.9814
iter: 9  Hyper loss: 5125.504  step_size:0.0117258  avg_acceptance_ratio:0.9678
iter:10  Hyper loss: 5130.449  step_size:0.0119615  avg_acceptance_ratio:0.9944
squaredError  0.43203 percentDifference 1947.878
40.58173619770931
```

SUSY Results

- Initial results from yesterday indicated that a network with 2 layers and 50 neurons would converge to good networks
  - Due to a bug I lost the test results from this network and I haven't been able to reproduce it
  - Most of the time, the acceptance rate for new states rapidly declines after a period of time and the network gets stuck in a bad form

Bayesian Network from HMC

- All weights and biases in a layer are pulled from a weight distribution and a bias distribution which control all the weights and biases from a specific layer.
- The mean and standard deviation of these distributions are drawn from another set of 2 distributions.
- Starting weight and bias values are chosen at random from their distributions.
- HMC is then run on the weights and bias values where the probability of a state is measured by the probability of each weight and bias value being chosen from their distributions, and the probability of the output value given a distribution with standard deviation 0.1
- The last values from HMC is taken to be the new values for the weights and biases.

Goals for next week

- Add customizable activation functions for the BNNs
- Find out why the networks are getting stuck early in the training period
- Try starting the networks with biases and weight values from one of the Flipout Networks