Week 8 Presentation

PHY 496

BRADEN KRONHEIM

MARCH 15, 2019

Summary

- Implemented a Bayesian Neural Network through using the Hamiltonian Monte Carlo Algorithm to create a Markov Chain
  - Code has been run a simple training set generated from sin(2x)/2cos(5x)/2 with excellent results

Hamiltonian Monte Carlo

- Each collection of values in the parameter space is treated as a position q and given a corresponding potential energy E where the probability density P(q) is proportional to exp(-E(q)) at that point.
  - This makes E(q) = -log(P(q))
- The values are also given momentum p so that the total energy at that point is E(q, p) = exp(-H(q, p))
  - H(q, p) is the Hamiltonian and equals E(q) + K(p) where K(p) is the kinetic energy due to the momentum p with mass 1
  - The momentum p is chosen at random at the start of each iteration from the probability distribution exp(-K(p)) or exp(-0.5*p^2)

HMC dynamics

- The Hamiltonian has an associated set of differential equations which show how the system evolved over time which can be approximated using the leapfrog differential equations algorithm.
  - As the momentum resets every iteration this will be able to explore the whole parameter space.
  - As the leapfrog algorithm introduces some error a new point is accepted with the probability min(1, exp(-(H(q*, p*)-H(q, p))) where q* and p* are the new points proposed by the leapfrog algorithm
  - This will eventually converge to an area of low potential energy which has a probability close to 1
  - If the probability measures the fit of network parameters to previously chosen priors and the quality of the network predictions, this area of high probability will correspond to the area of good possible networks

HMC Advantages

- Due to the use of the Hamiltonian dynamics, if a sufficiently large number of leapfrog steps is used the algorithm will not be a random walk and will converge much faster than algorithms like the Metropolis algorithm
  - Error tends to oscillate along the leap frog paths, allowing a large number of small steps to travel a significant distance in the solution space, which contributes to it not being a random walk and converging quickly

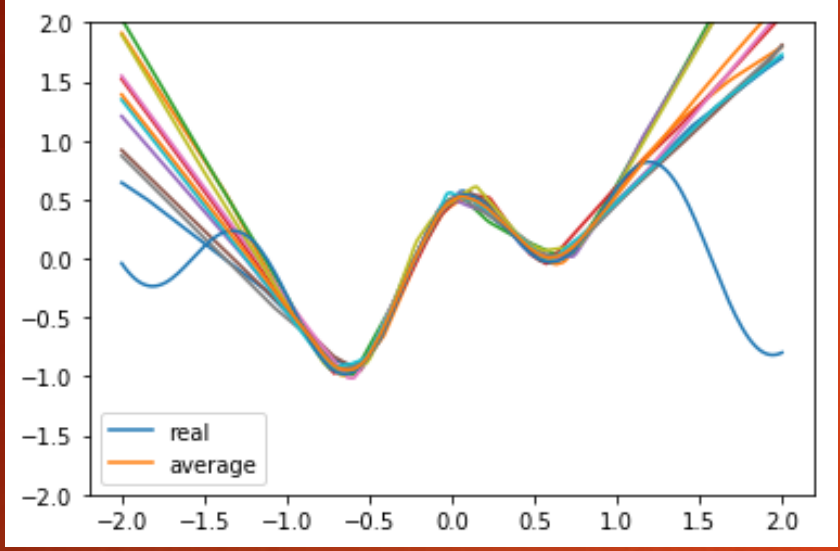
Bayesian Network from HMC

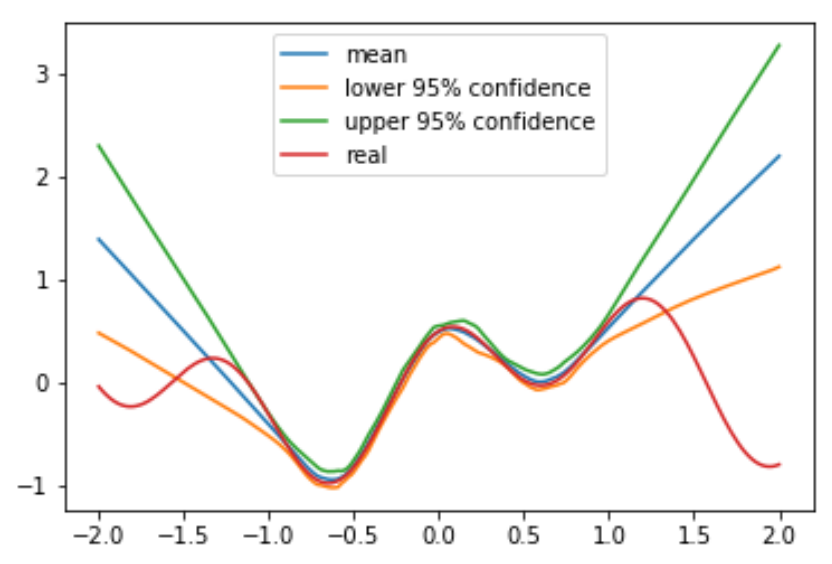
- All weights and biases in a layer are pulled from a weight distribution and a bias distribution which control all the weights and biases from a specific layer.
- The mean and standard deviation of these distributions are drawn from another set of 2 distributions.
- Starting weight and bias values are 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.

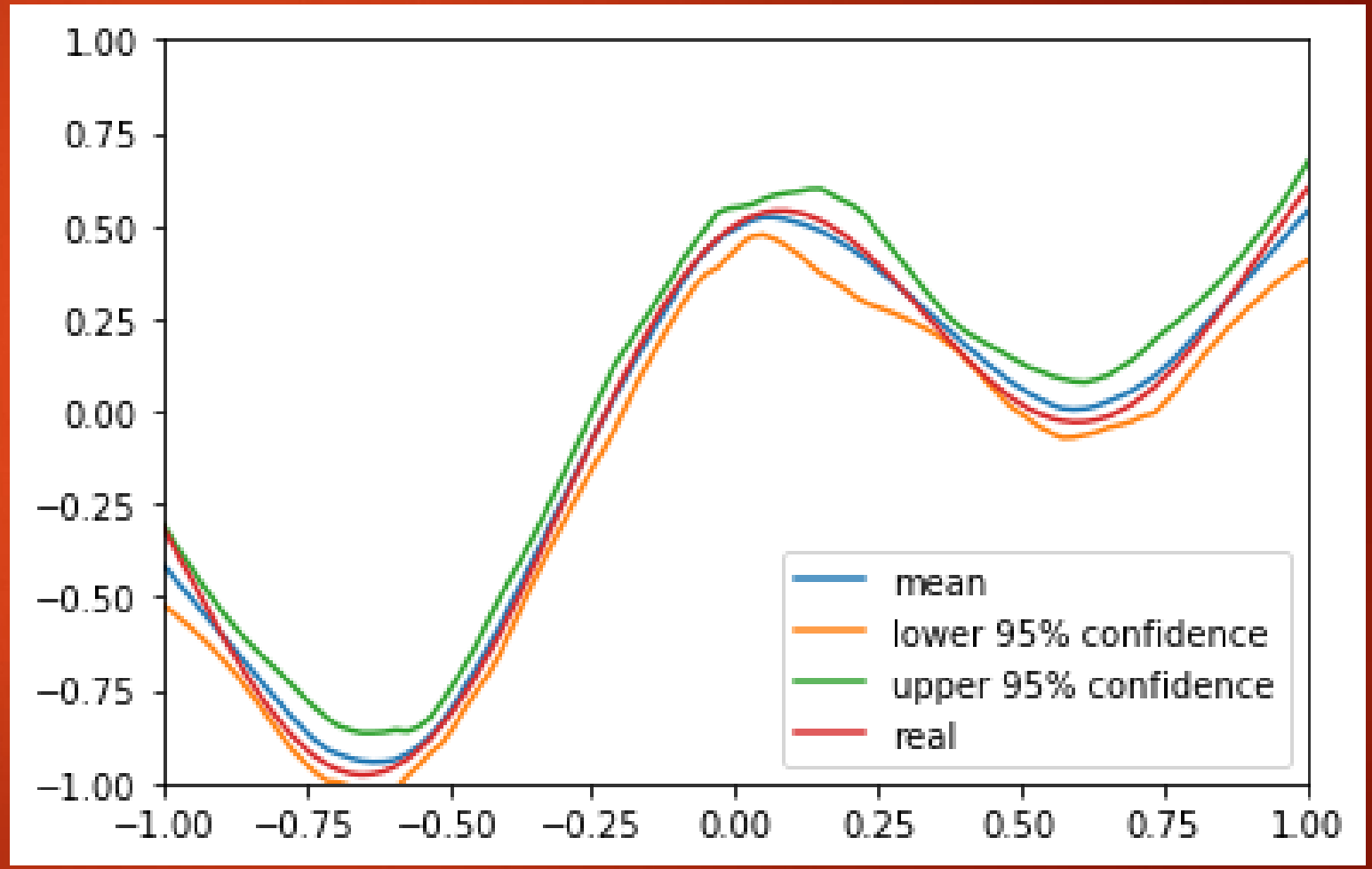
Bayesian Network from HMC (cont.)

- After a certain number of iterations, this process is repeated for the weight and bias distribution means and standard deviations.
  - The probability here is the probability of the current biases and weights given their distributions plus the probability of the distribution means and standard deviations given their distributions
  - The end values from HMC are then the new means and standard deviations for the weight and bias distributions
- After training for a certain number of iterations, HMC will converge to the area with the best networks. Making predictions from a certain number of these networks represents the output of the trained network

Sample Output

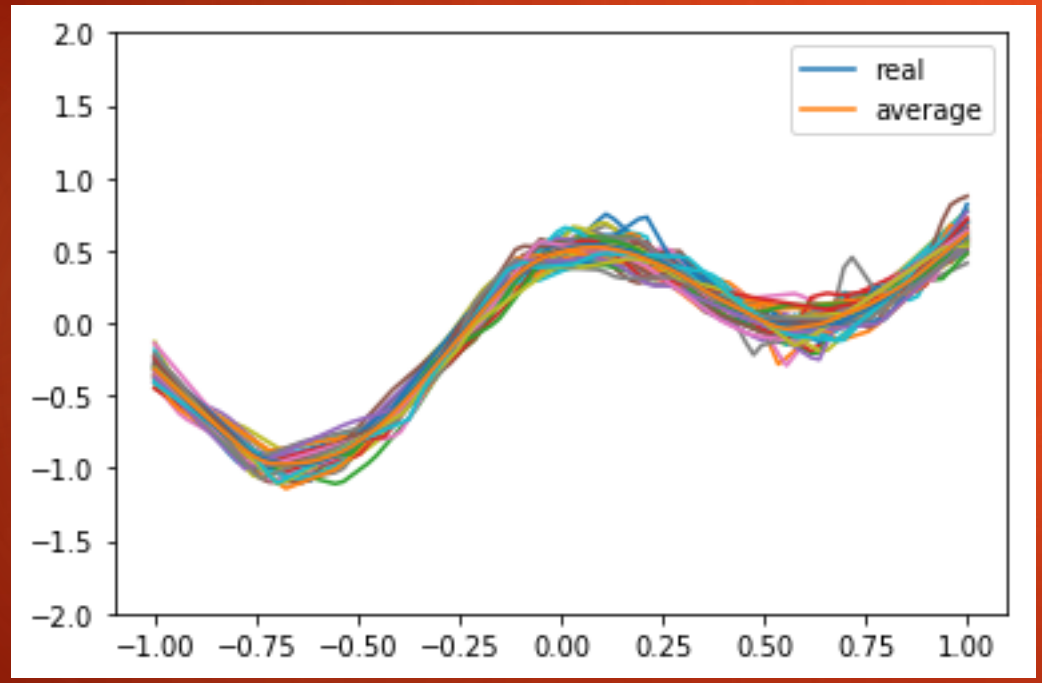


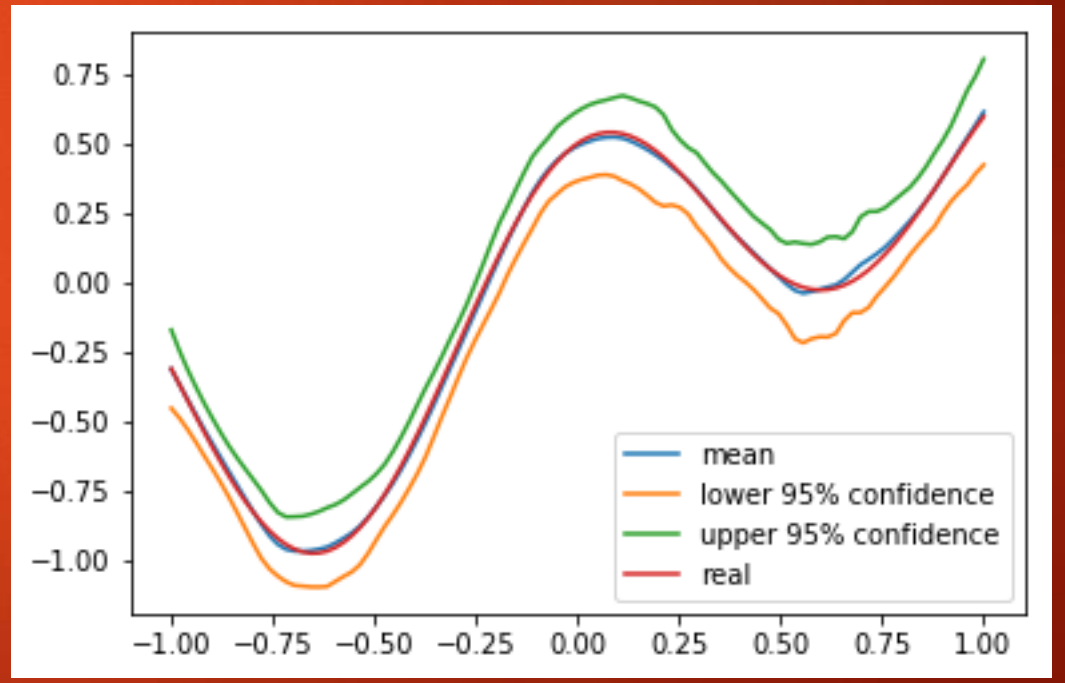




Variable Training Data

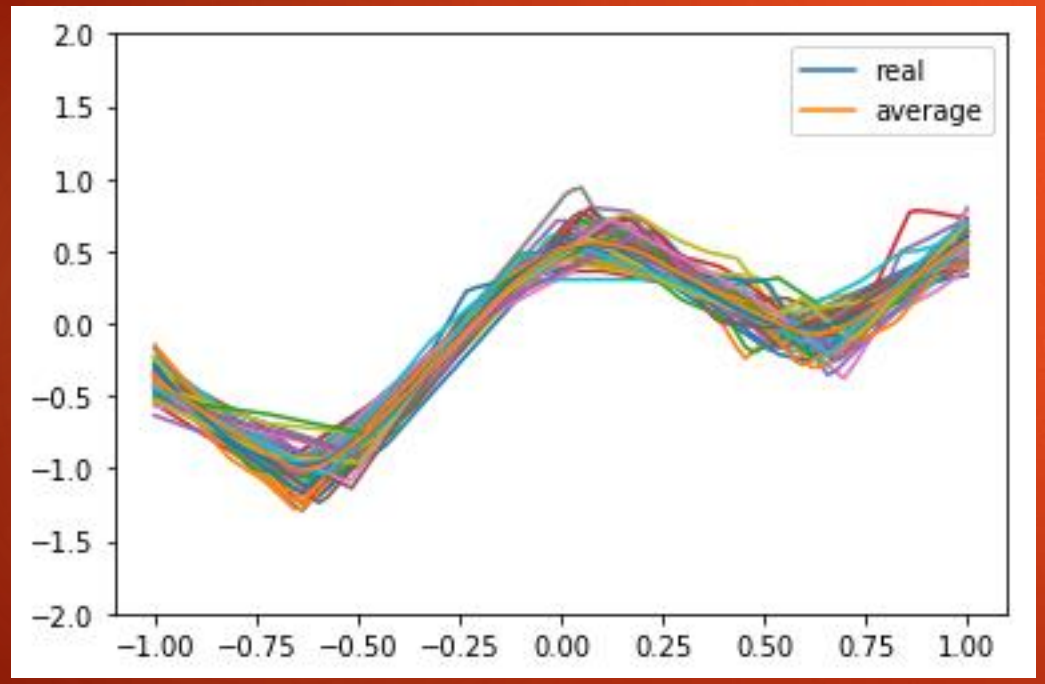
- 25 training points, 100 validation points,100 networks 0.66% Error

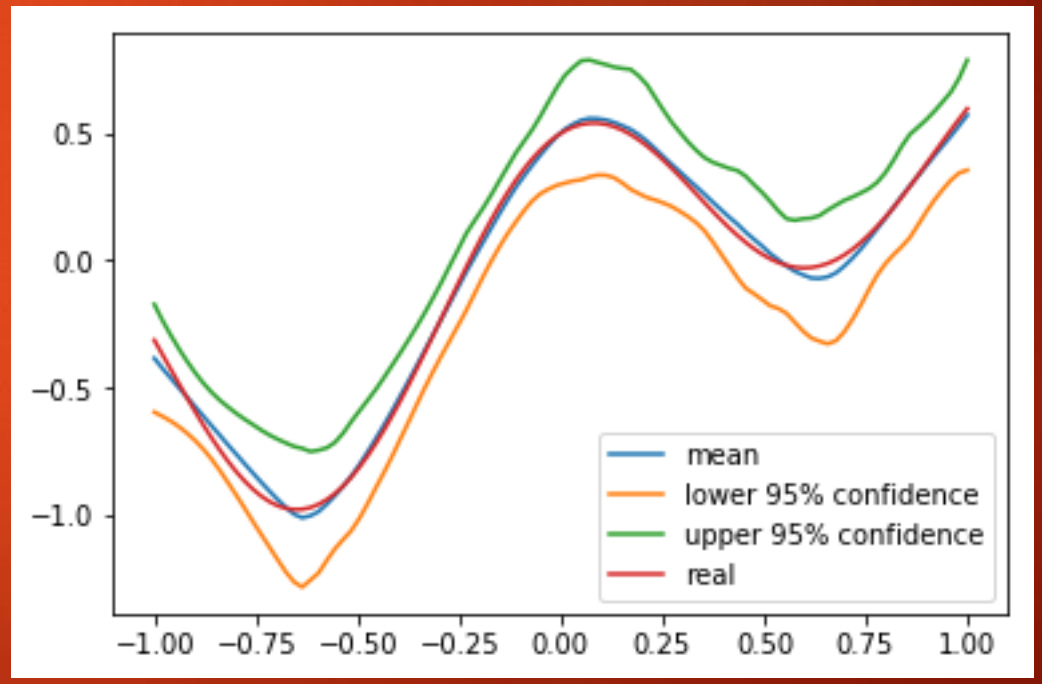




Variable Training Data

- 10 training points, 100 validation points,100 networks 1.48% Error





Goals for next week

- Finish cleaning the HMC Bayesian Neural Network code
- See how it fairs the pMSSM data compared to the network built off of the TensorFlow Probability layers.