Modelling Uncertainty with Bayesian Neural Netwowkrs

Ellis Brown, Melanie Manko, Ethan Matlin

Columbia University

EECSE6699

September 21, 2020

Motivation

- Neural Networks are really good at predicting stuff
- But what about when they're wrong?
- We want to know something about the certainty of a NN's prediction.
- A natural framework for modelling uncertainty is the Bayesian paradigm.

Experiment 1: Why Uncertainty Matters

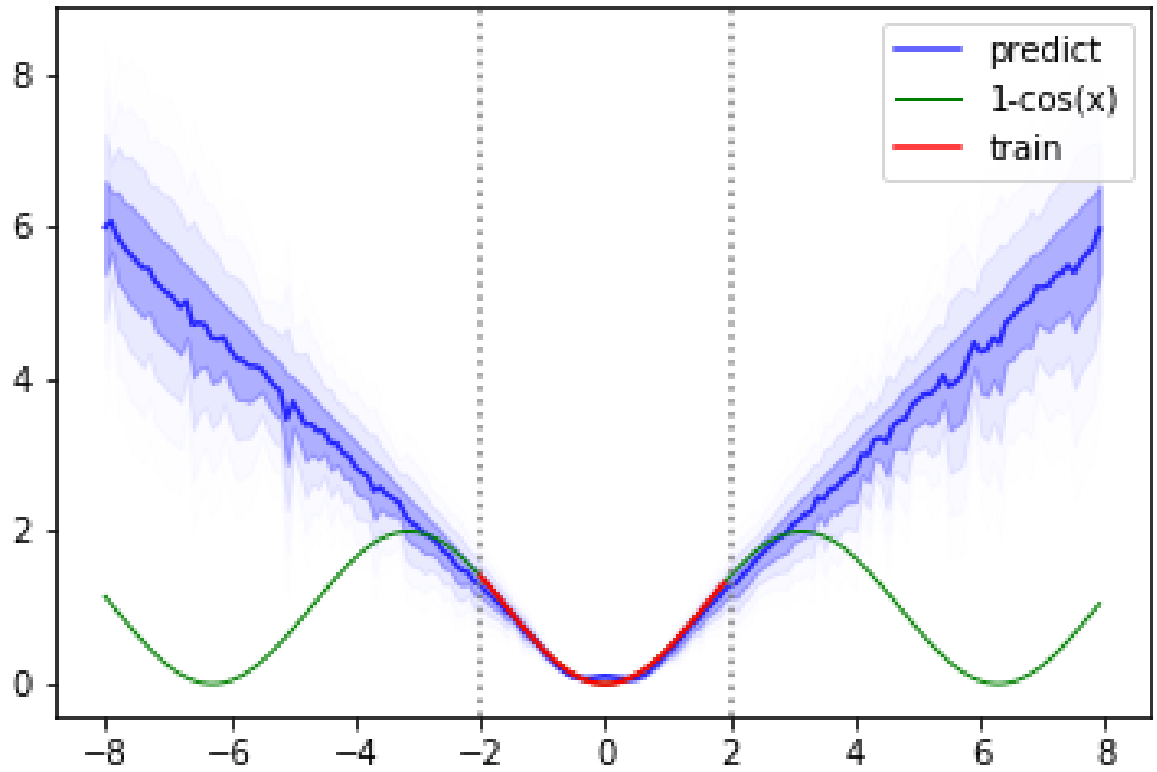


Figure: ReLU Network with 5 Hidden Layers and 1024 neurons per layer

But you just fabricated an example to fit your story...

Uncertainty following the Great Recession: Gross Domestic Product

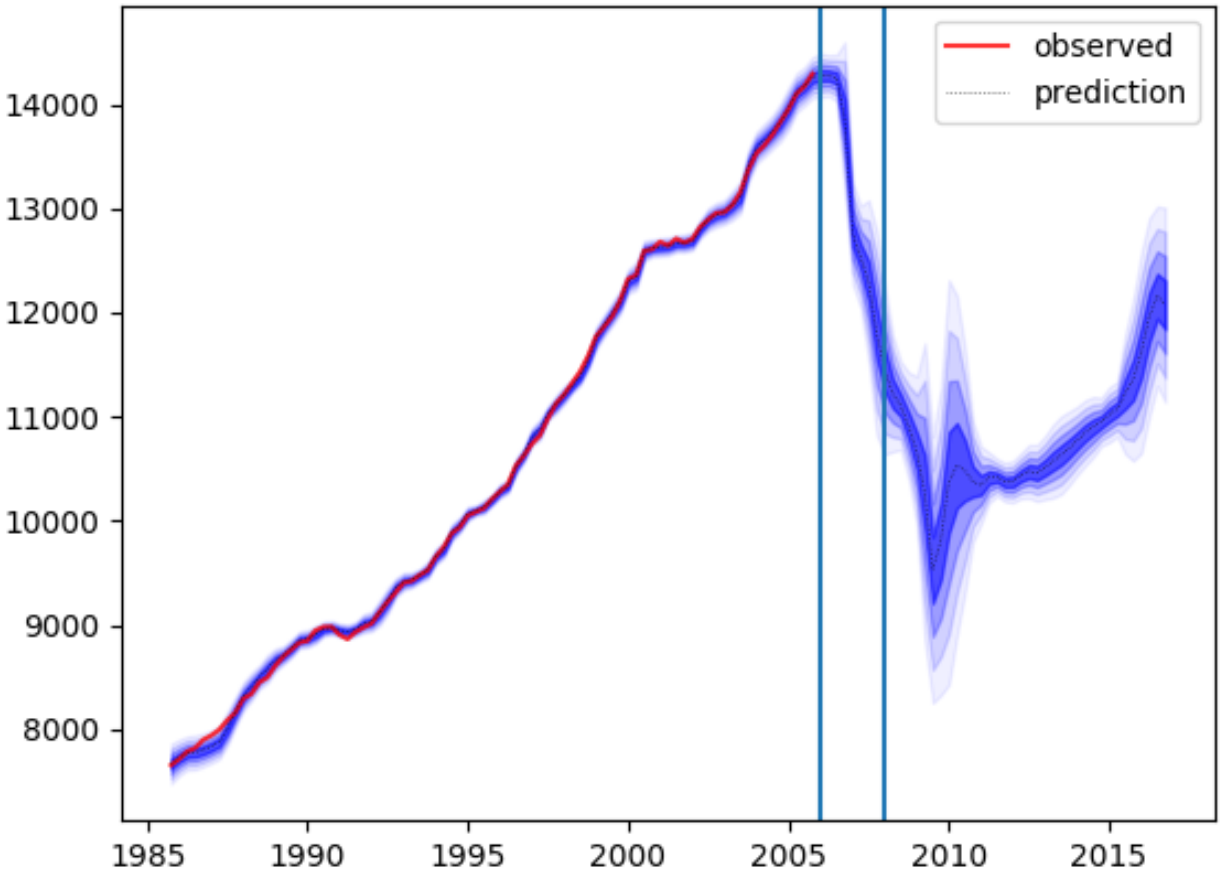
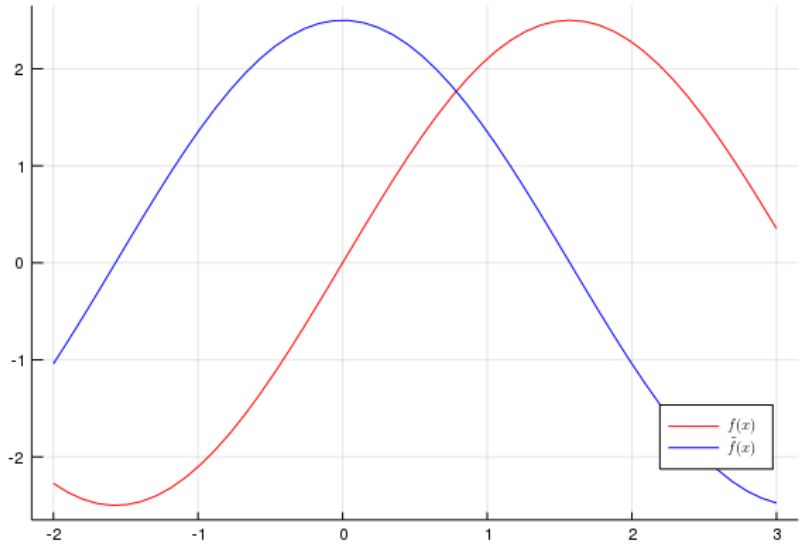
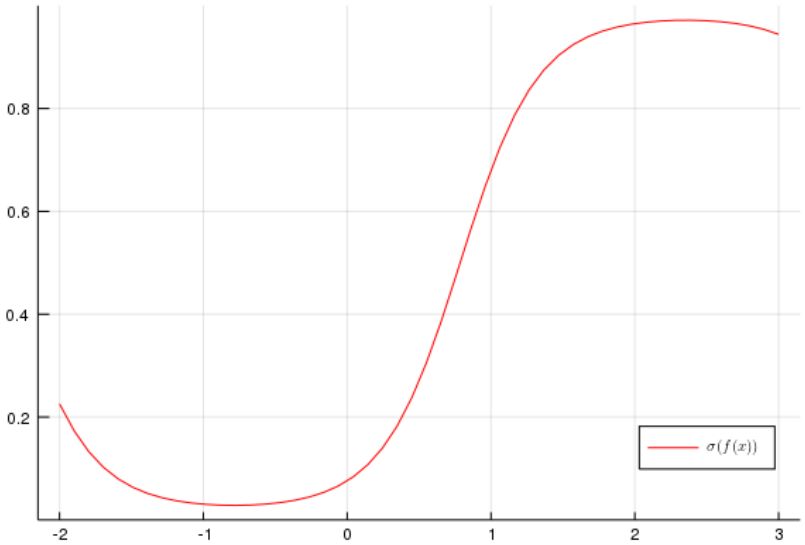


Figure: ReLU Network with 5 Hidden Layers and 1024 neurons per layer

Experiment 2: But what about Softmax? Doesn't that give uncertainty?

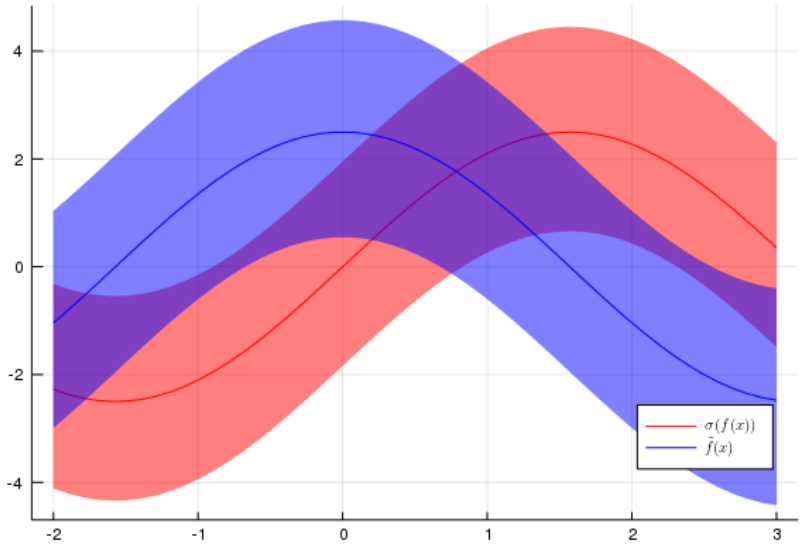


$f(x)$

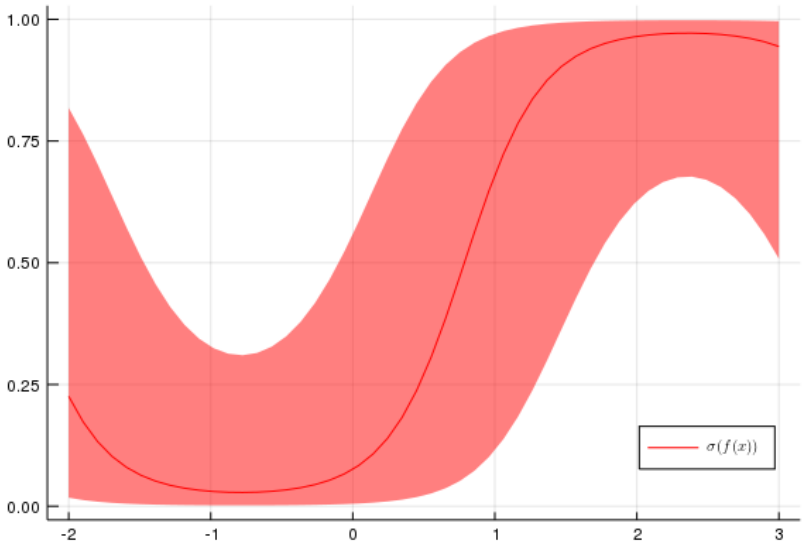


$\sigma(f(x))$

Experiment 2: But what about Softmax? Doesn't that give uncertainty?



$\tilde{f}(x)$



$\sigma(\tilde{f}(x))$

Outline

1. Neural Networks Need Better Notions of Uncertainty ✓
2. Methods of Reasoning about Uncertainty
   - Infinite Bayesian NN $\iff$ Gaussian Process
   - Finite Bayesian NN: Use Numerical Techniques (MCMC, Variational Inference, **Dropout**)
3. How do priors on the weights translate to priors on functions?
4. Activation Functions and the Posterior **Distribution**
5. Shortfalls of the Approach

Bayesian Neural Networks and Gaussian Processes

- Infinite neural network $\iff$ Gaussian Process (Neal [1995], Williams [1998])
  - 1 hidden layer
  - Bounded nonlinearites
  - Weights (u_{ij} and ν_{jk}) are i.i.d with zero mean and finite variance

$$f_k(x) = \sum_{j=1}^{H} \nu_{jk} \sigma(\sum_{i=1}^{I} u_{ij} x_i) \tag{1}$$

- Central Limit Theorem $\implies$ $f_k(x) \sim \mathcal{N}\left(0, \sigma_b^2 + H\sigma_\nu^2 V(x)\right)$
- $V(x)$ is the covariance function of the Gaussian Process corresponding to the network.

Specific Covariance Functions (Williams [1998])

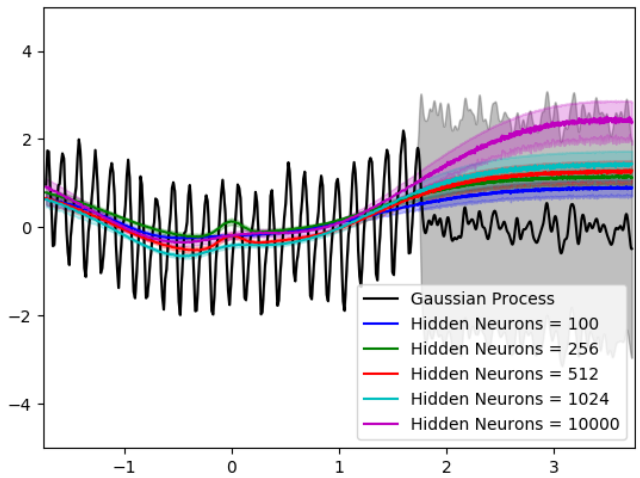
Gaussian Nonlinearity in NN $\iff$ Scaled Squared Exponential in GP

$$V_G(x, x') = \left(\frac{\sigma_e}{\sigma_u}\right)^d \exp\left(-\frac{x^T x}{2\sigma_m^2}\right) \exp\left(-\frac{(x - x')^T (x - x')}{2\sigma_s^2}\right) \exp\left(-\frac{x'^T x'}{2\sigma_m^2}\right)$$

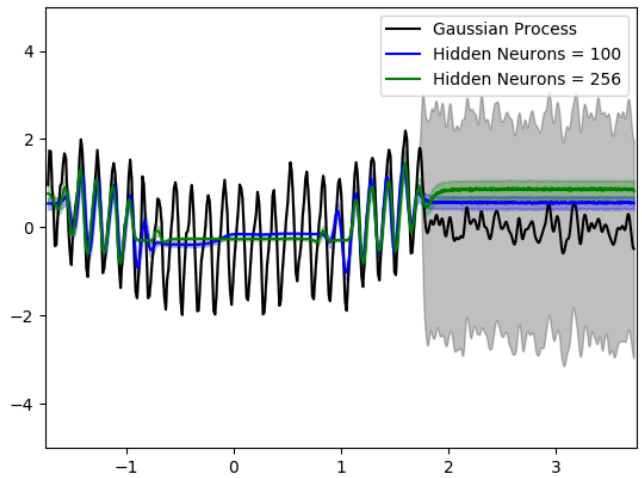
Sigmoid Nonlinearity in NN $\iff$ ArcSin Covariance Function in GP

$$V_{\text{erf}}(x, x') = \frac{2}{\pi} sin^{-1} \frac{2x^T \Sigma x'}{\sqrt{(1 + 2x^T \Sigma x)(1 + 2x'^T \Sigma x')}}$$

Experiment 3: In practice though we have only finite Networks



(a) **NN** (Gaussian Nonlinearity, 1 Hidden Layer);
GP (Squared Exponential Covariance)



(b) **NN** (Gaussian Nonlinearity, 5 Hidden Layers);
GP (Squared Exponential Covariance)

Dropout

- How to evaluate a finite NN?
  - Monte Carlo techniques: HMC, RWMH, SMC, etc. Neal [1995]
  - Variational Inference: Barber & Bishop [1998], Graves [2011], Blundell et al. [2015]
- **Dropout is equivalent to Variational Inference in Gaussian Processes** (Gal [2016], Gal & Ghahramani [2015b], Gal & Ghahramani [2016a], Gal & Ghahramani [2016b], Gal & Ghahramani [2015a])
  - Used as a regularization technique [Hinton et al. [2012], Srivastava et al. [2014]]
  - Intuitively, makes sense as a way to get a distribution with uncertainty.

Theory: Dropout $\iff$ Variational Inference

NN Variational Inference Objective Function:

$$\hat{\mathcal{L}} = -\frac{N}{M}\sum_{i \in S} \log p(y_i | f^{g(\lambda,\epsilon)}) + KL(q_\lambda(\theta)||p(\theta)) \tag{2}$$

NN Loss Function:

$$\mathcal{L} = -\log p(y | f^{g(\theta,\epsilon_i)}) + \text{const.} + \lambda_1||W_1|| + \lambda_2||W_2|| + \lambda_3||b|| \tag{3}$$

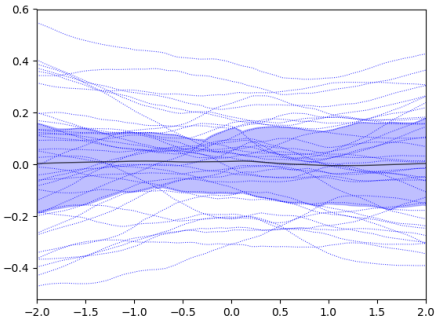
KL condition:

$$\frac{\partial}{\partial\theta} KL(Q_\lambda(\theta)||p(\theta)) = \frac{\partial}{\partial\theta}\lambda_1||W_1|| + \lambda_2||W_2 + \lambda_3||b|| \tag{4}$$

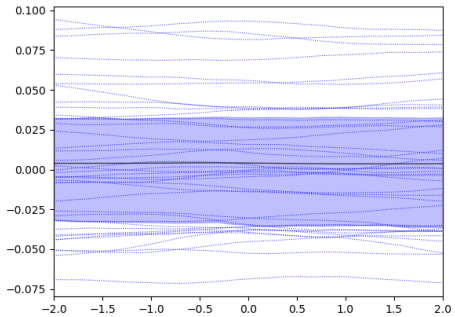
$\implies$ the two are equivalent.

Experiment 4: What prior should I choose? How do weight priors correspond to function priors?

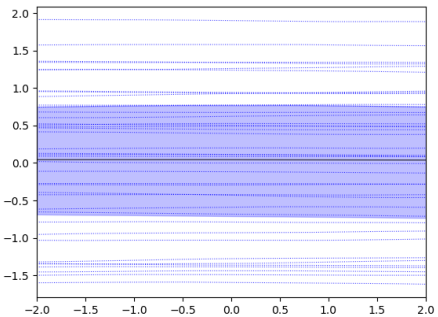
$W \sim \mathcal{N}(0, 0.05)$



$W \sim \text{Unif}[-0.05, 0.05]$



$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{N_{in}+N_{out}}}\right)$



$W \sim \text{Unif}\left[-\sqrt{\frac{6}{N_{in}}}, \sqrt{\frac{6}{N_{in}}}\right]$

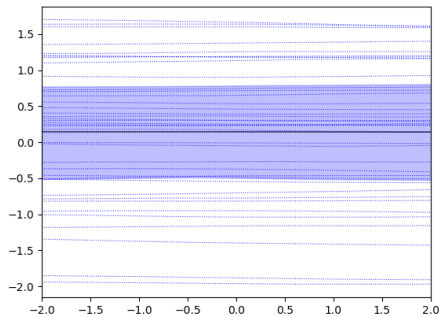


Figure: ReLU (Untrained), 5 Hidden Layers, 1024 neurons per layer

Experiment 5: Activation Functions and Uncertainty Estimates

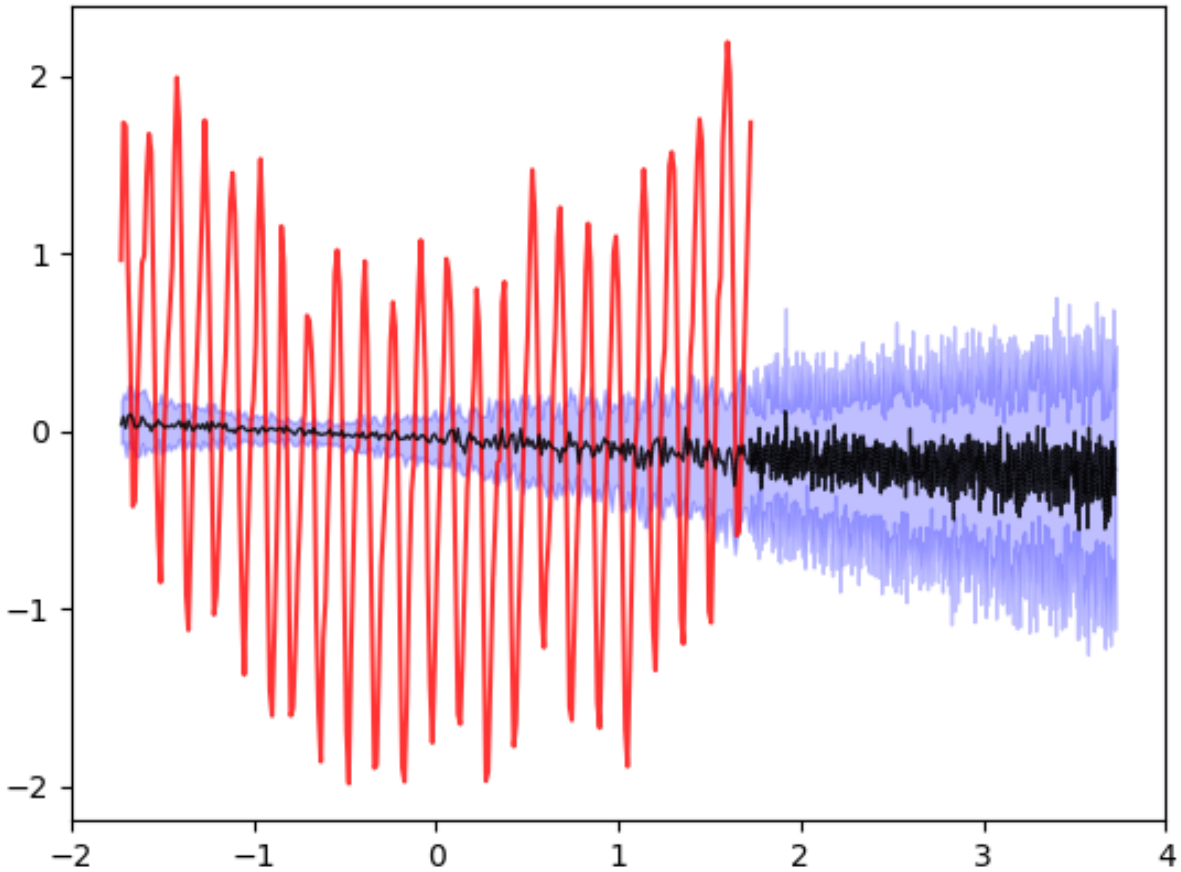


Figure: Linear; 5 Hidden Layers, 1024 Neurons per Layer

Experiment 5: Activation Functions and Uncertainty Estimates

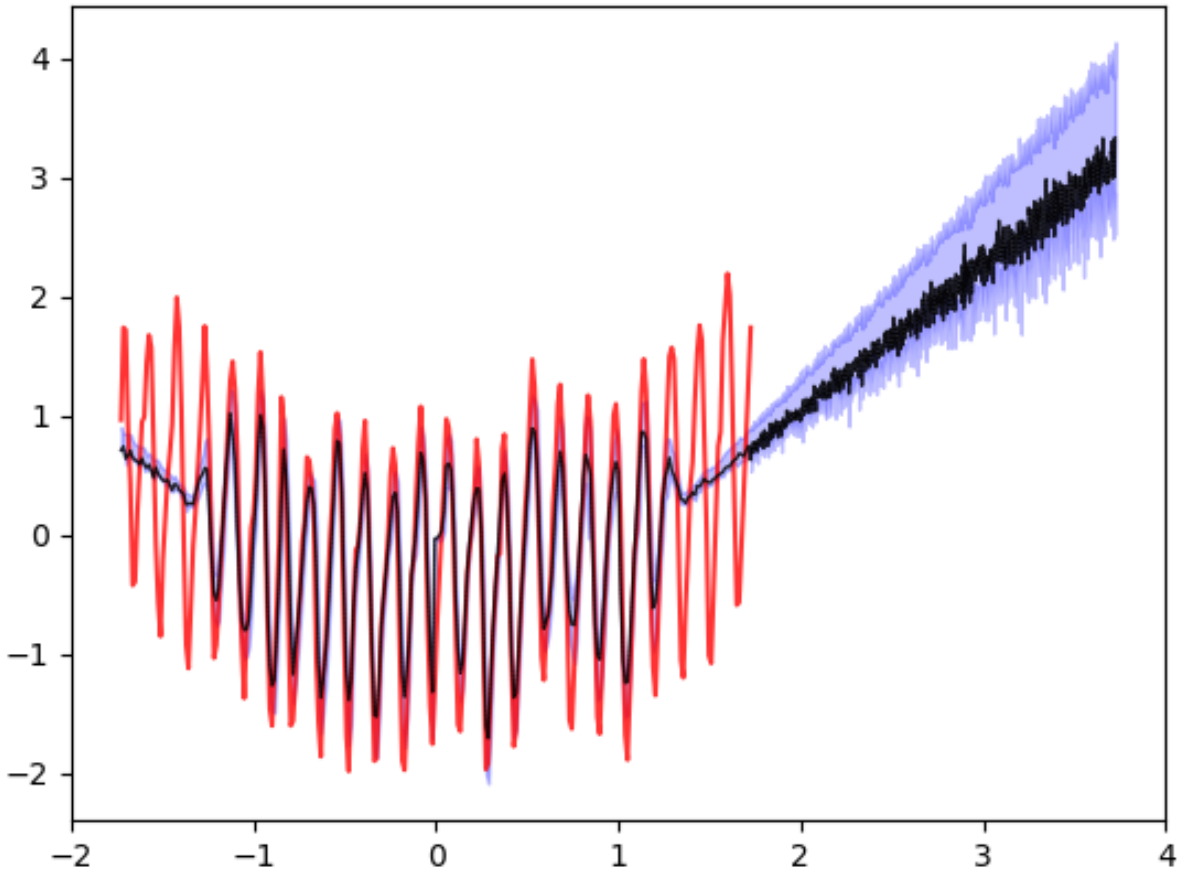


Figure: ReLu; 5 Hidden Layers; 1024 Neurons per Layer

Experiment 5: What Activation Function?

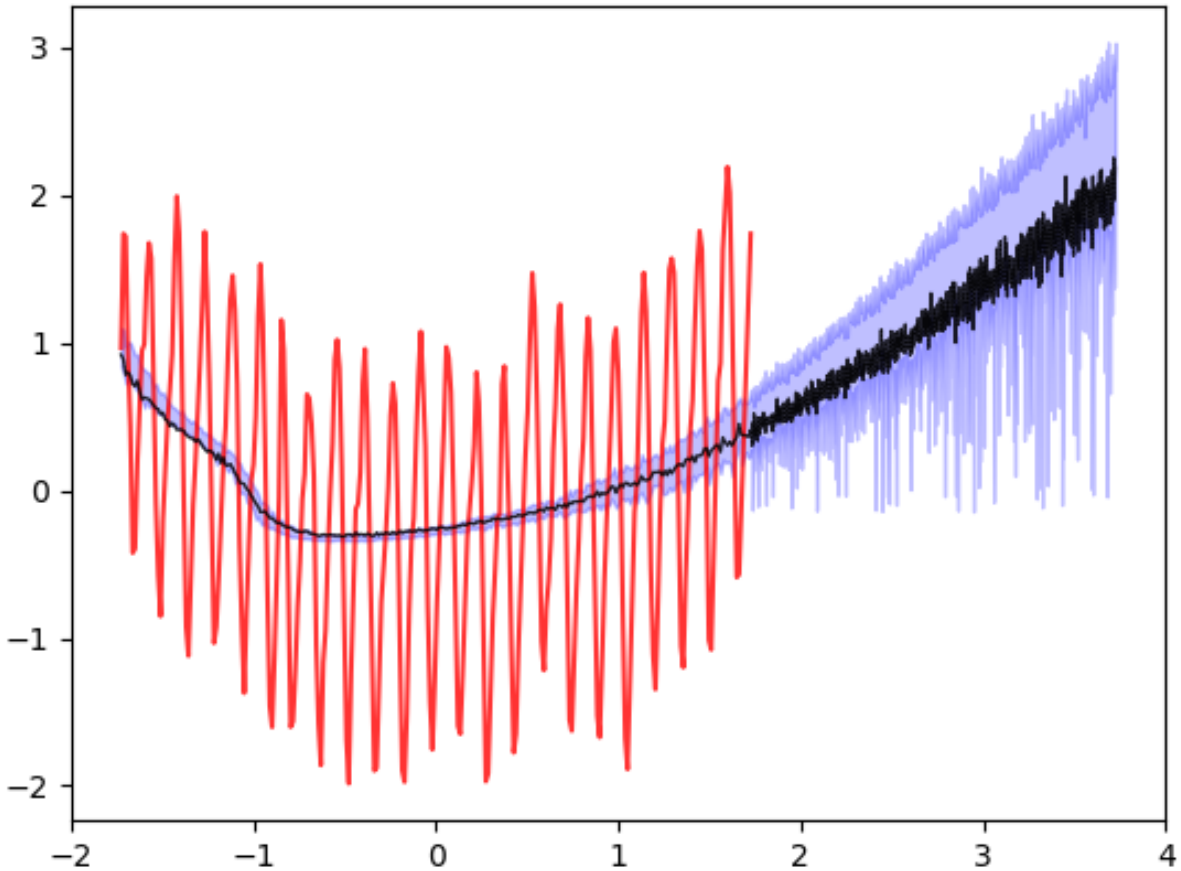


Figure: Softplus; 5 Hidden Layers; 1024 Neurons per Layer

Experiment 5: What Activation Function?

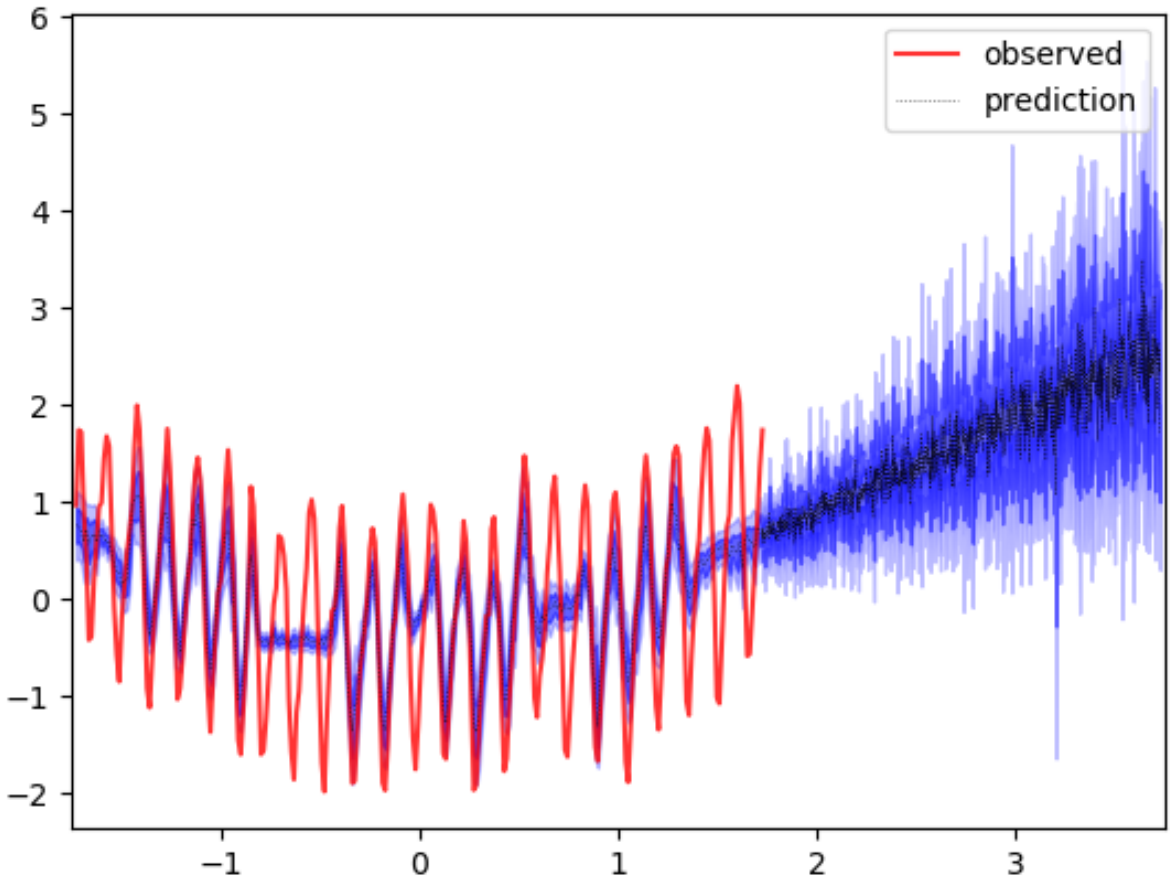


Figure: Softplus; 5 Hidden Layers; 1024 Neurons per Layer

Experiment 6: Is the Network Really Learning Uncertainty as it Trains?