MIT-IBM Watson AI Lab

IBM Research

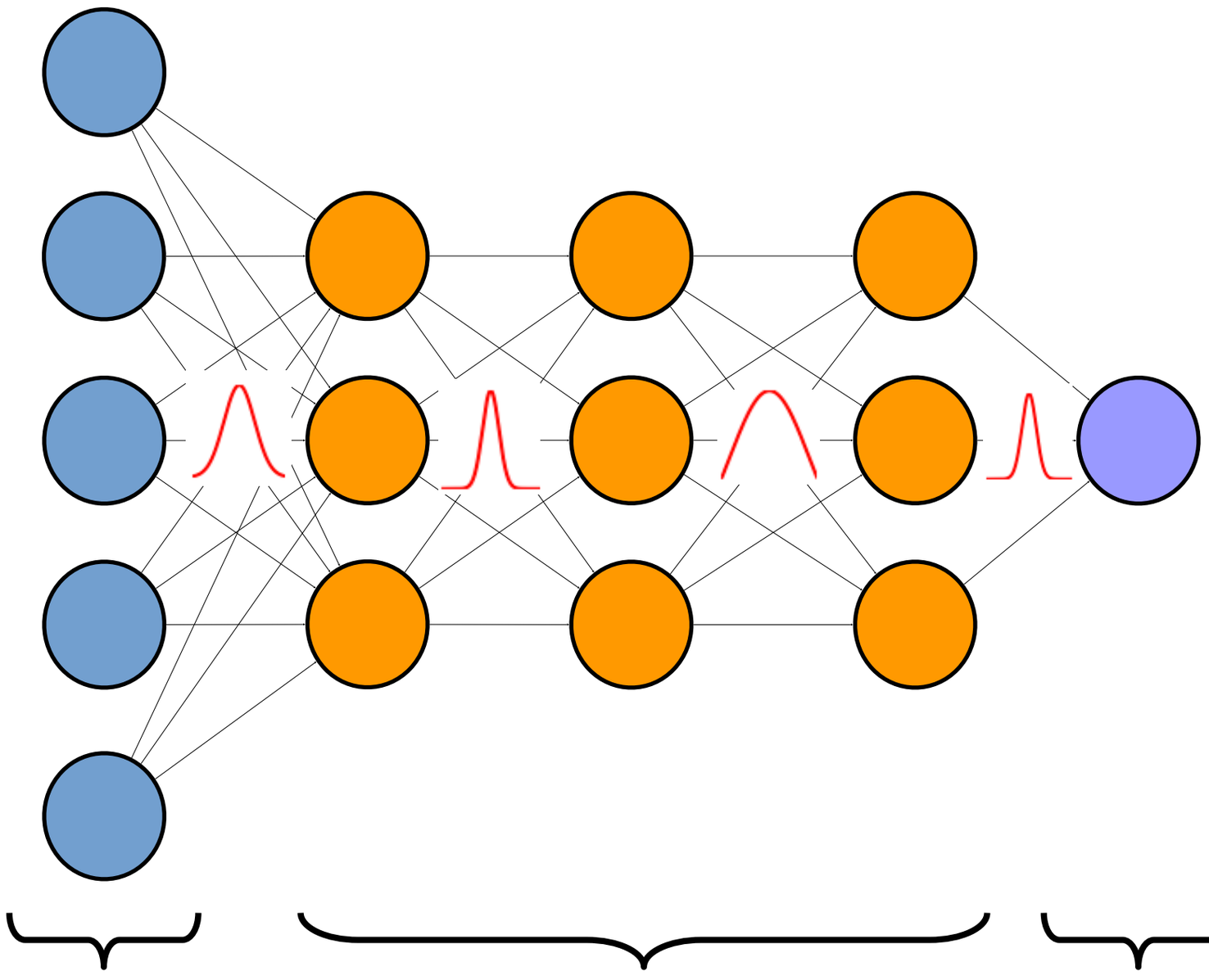
Horseshoe Priors for Bayesian Neural Networks

Soumya Ghosh
IBM Research
MIT-IBM Watson AI lab

Jiayu Yao
Harvard

Finale Doshi-Velez
Harvard

Neural Networks



Inputs x

Layers, parameterized by weights W

Outputs y

$$y = h_{\mathcal{W}}(x) + noise$$

Being Bayesian:

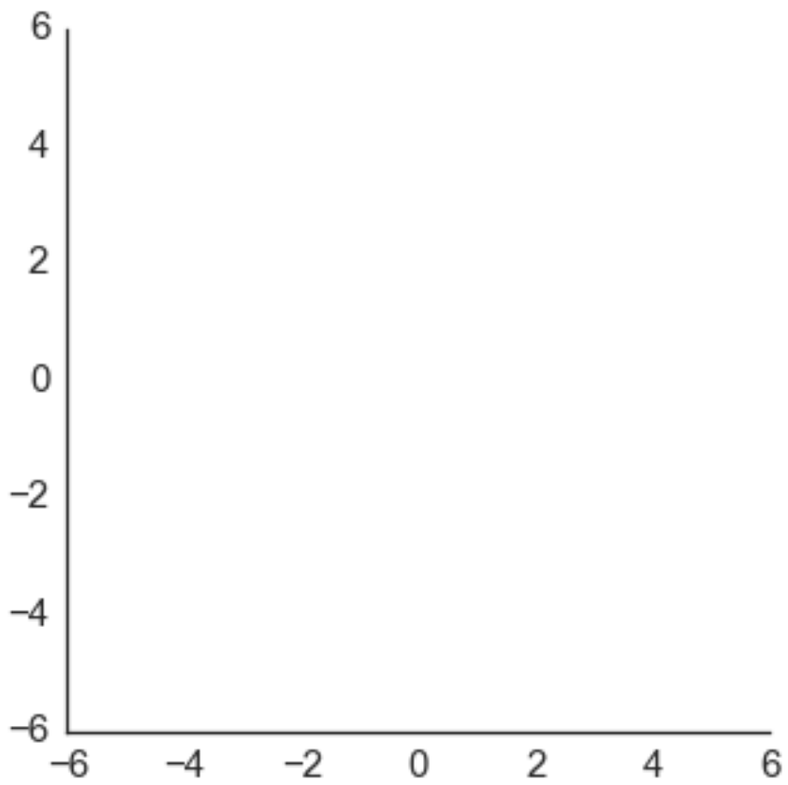
$$p(\mathcal{W} \mid \lambda) \rightarrow p(\mathcal{W} \mid y, x, \lambda)$$

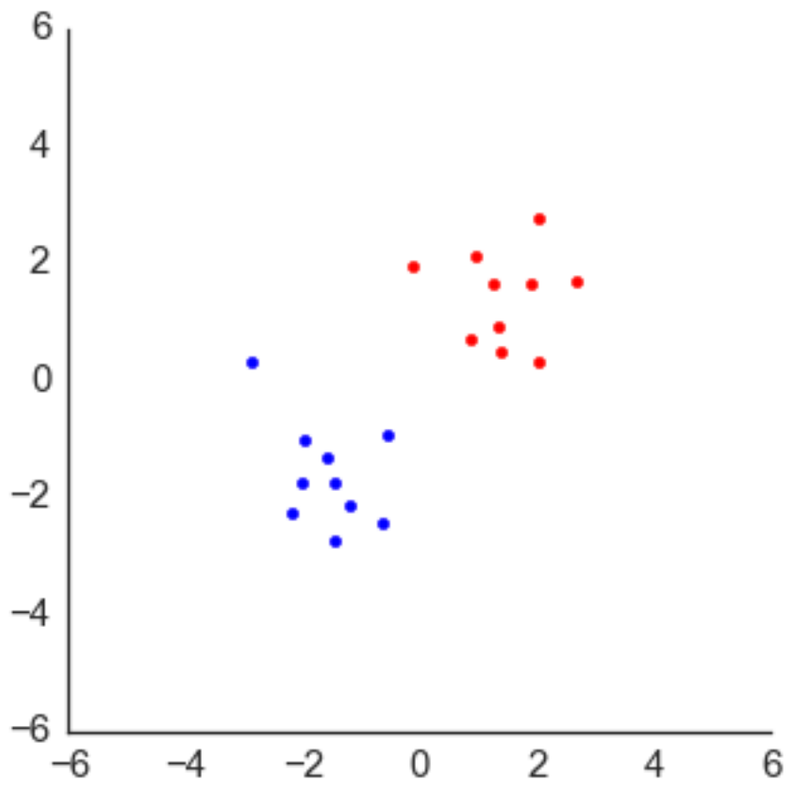
$$\downarrow$$

$$p(y_* \mid x_*, y, x, \lambda)$$

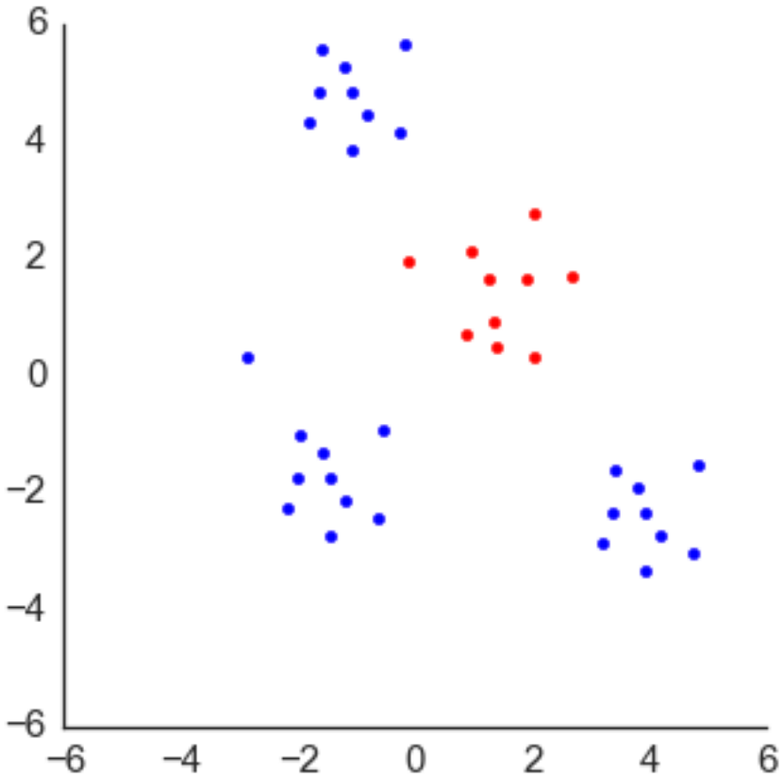
Why Bother?

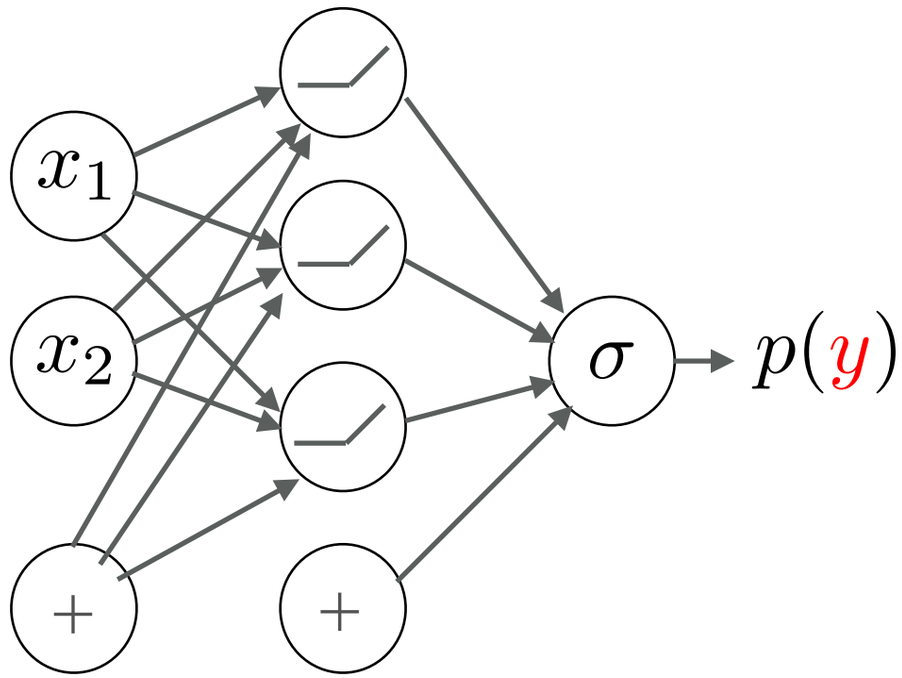
- Need to guard against unintended consequences.
- Need to know when the model doesn't know.

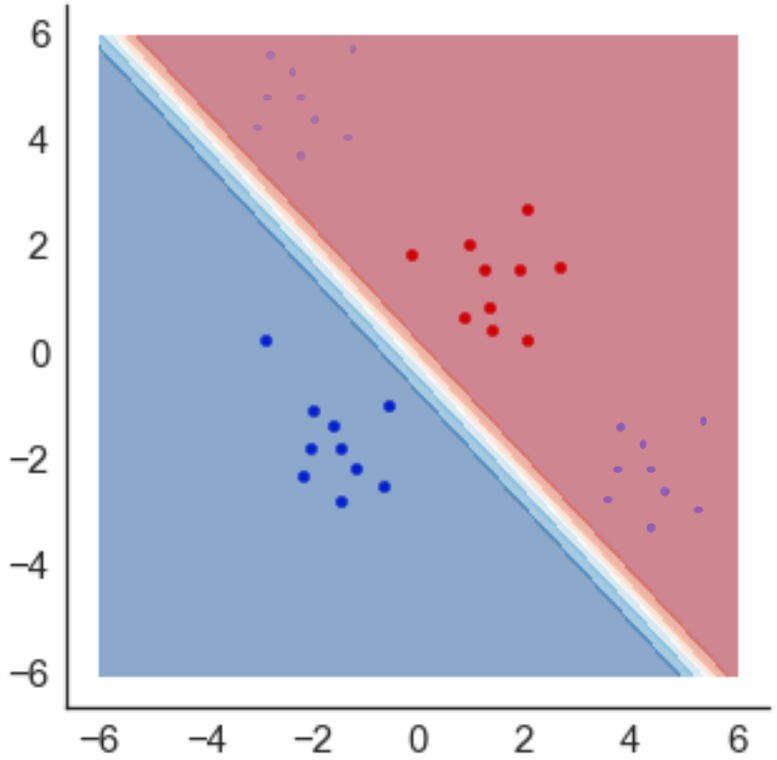




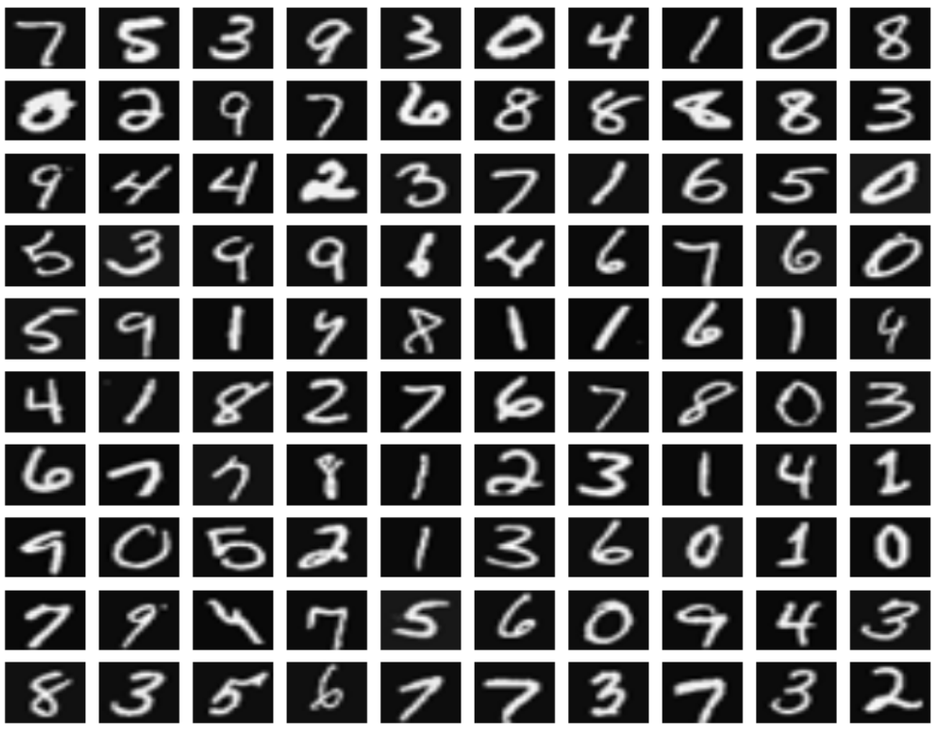
Predictive Uncertainty



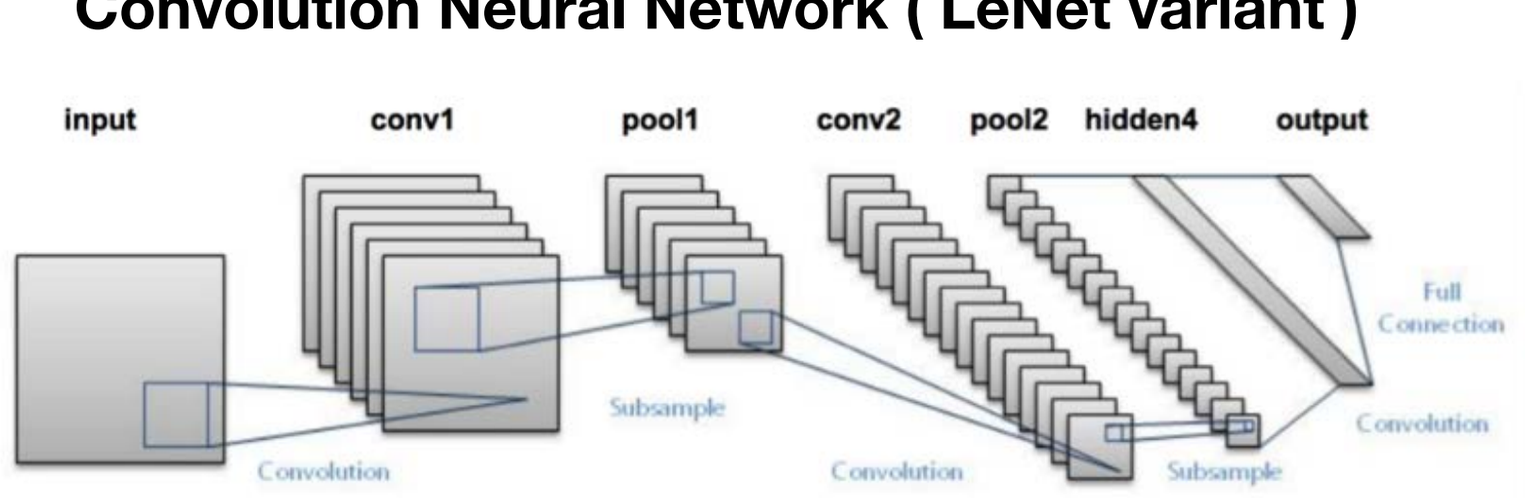




Larger Data and Modern Architectures

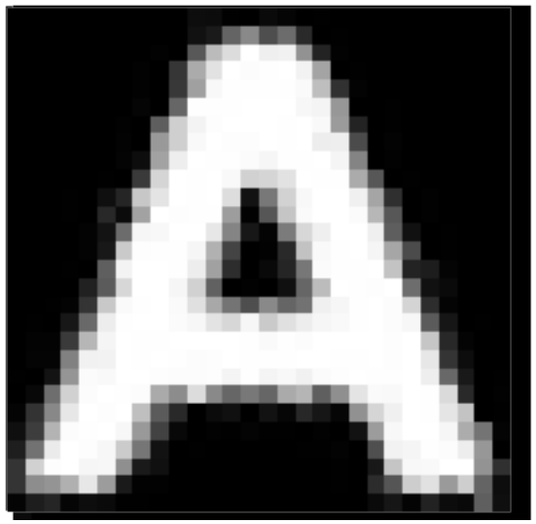


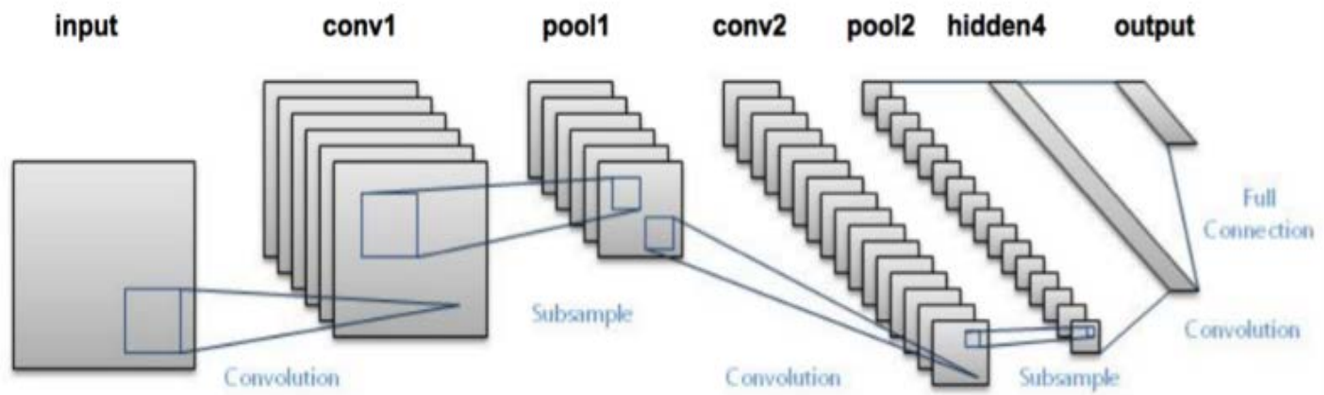
Train: 60,000 Handwritten digits
Test: 10,000 heldout digits

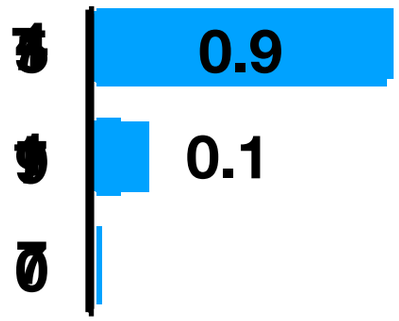


Test Error ~ 1%

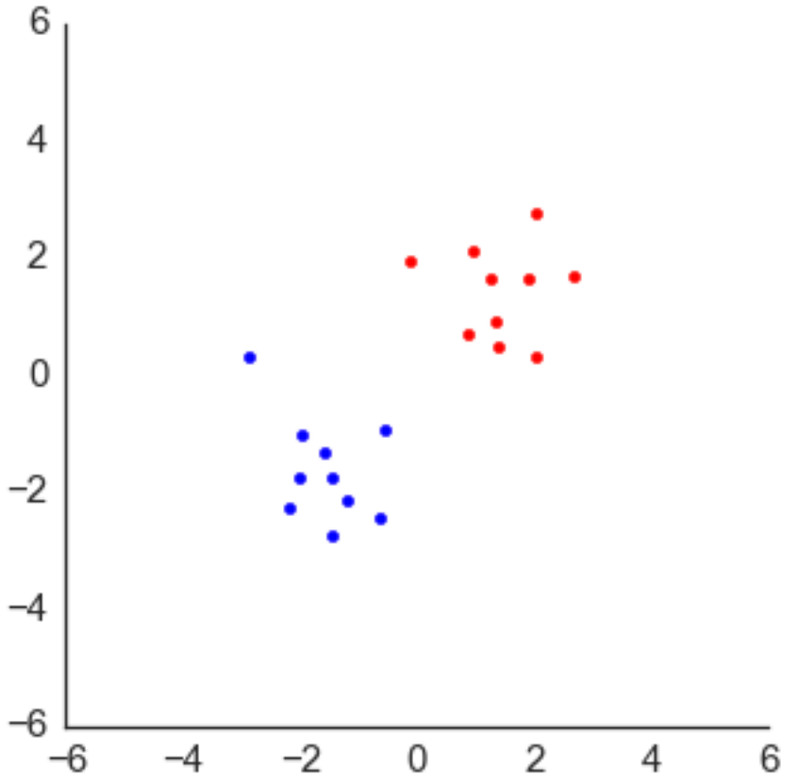
Are the predictions robust?

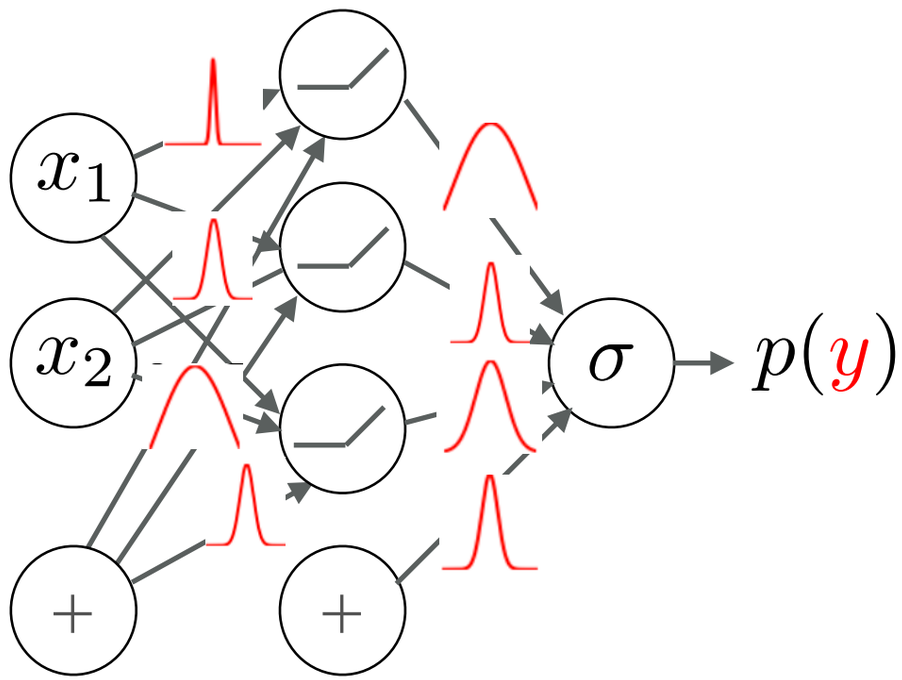


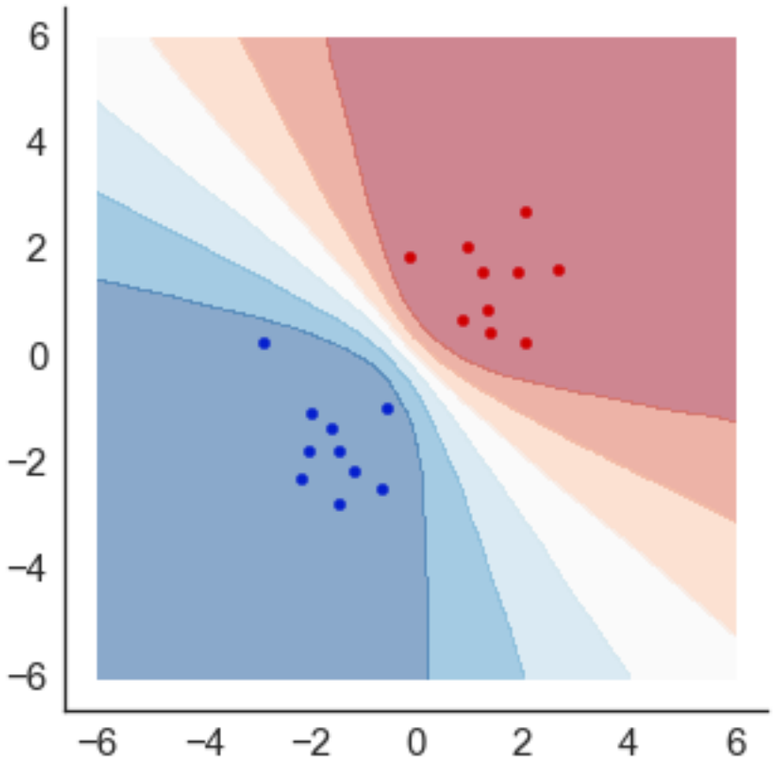




Bayesian Neural Networks





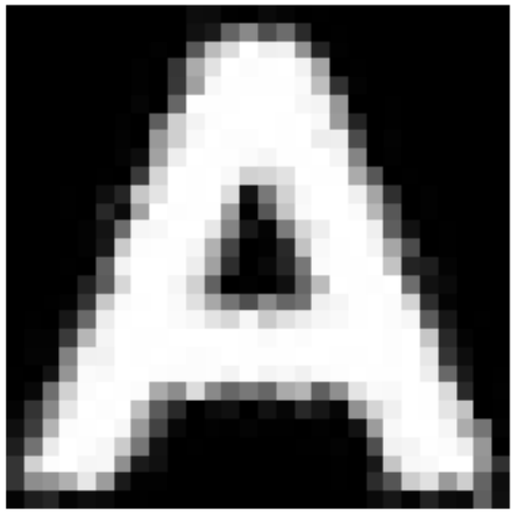


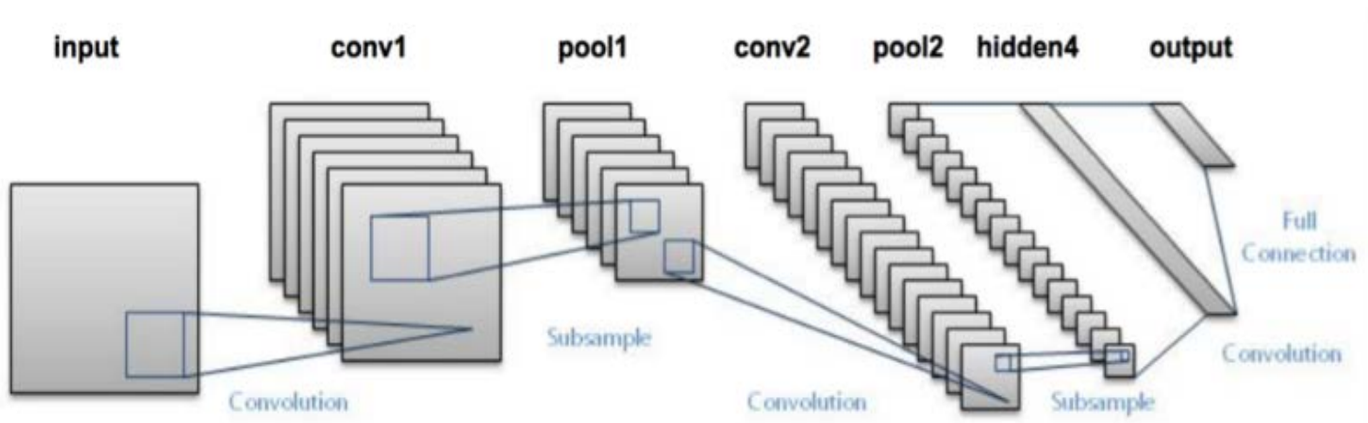
Distribution on weights: $p(\mathcal{W})$

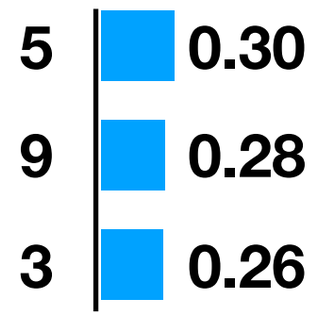
$$\int p(y_* \mid \mathcal{W}, x_*) p(\mathcal{W} \mid y_{\text{train}}, x_{\text{train}}) d\mathcal{W}$$

source: Ghosh et al., AAAI 2016
Balan et al., NIPS 2016

Bayesian Neural Networks

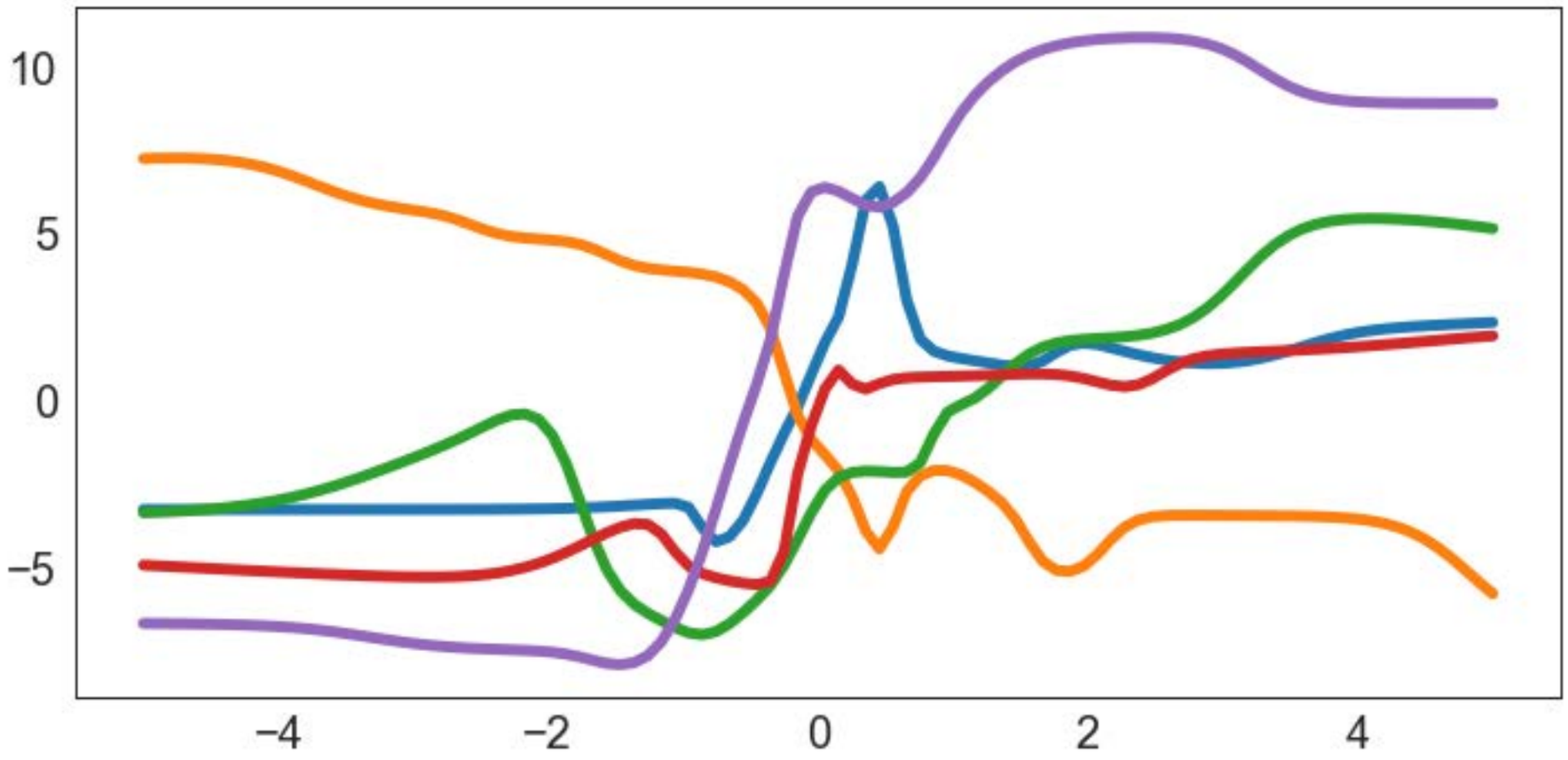






experiment inspired by: Gal et al., 2015

Distribution over Functions

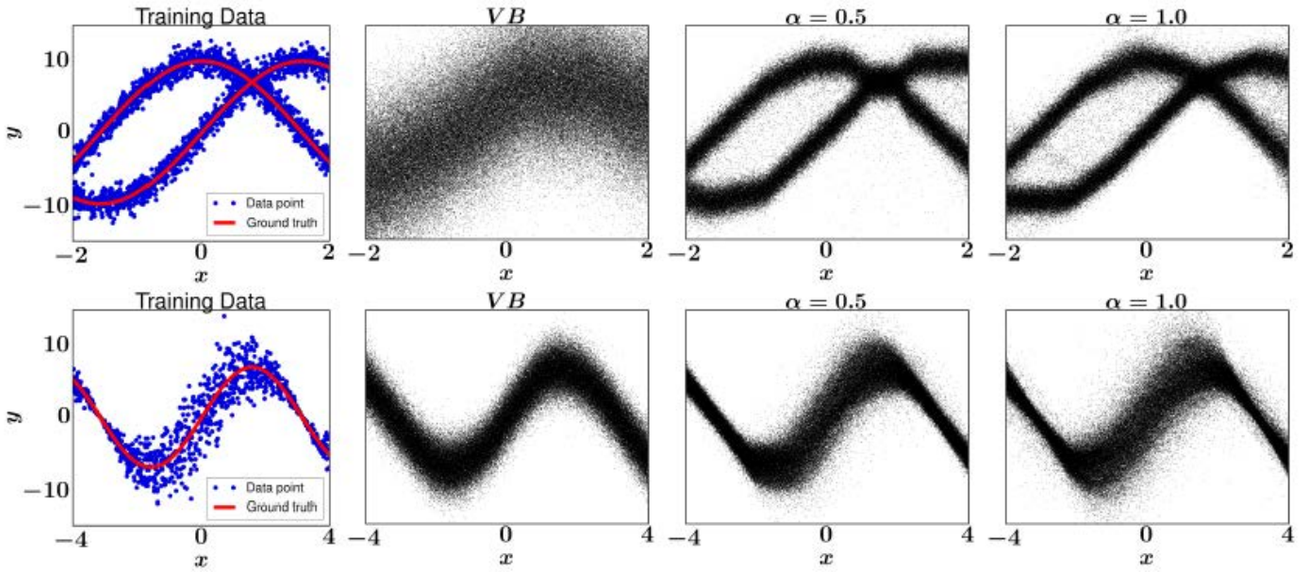


$$f(x) = h_{\mathcal{W}}(x)$$

random variable

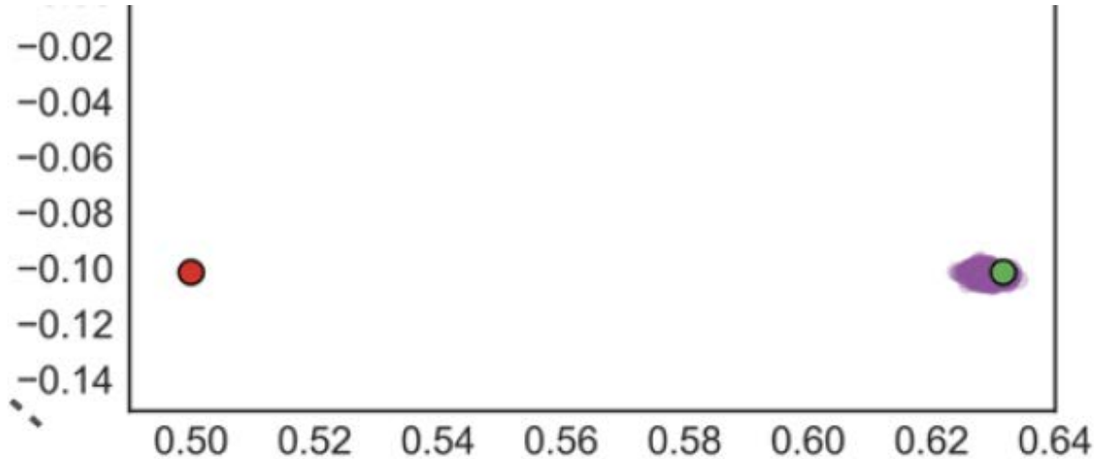
random variable

BNNs - applications



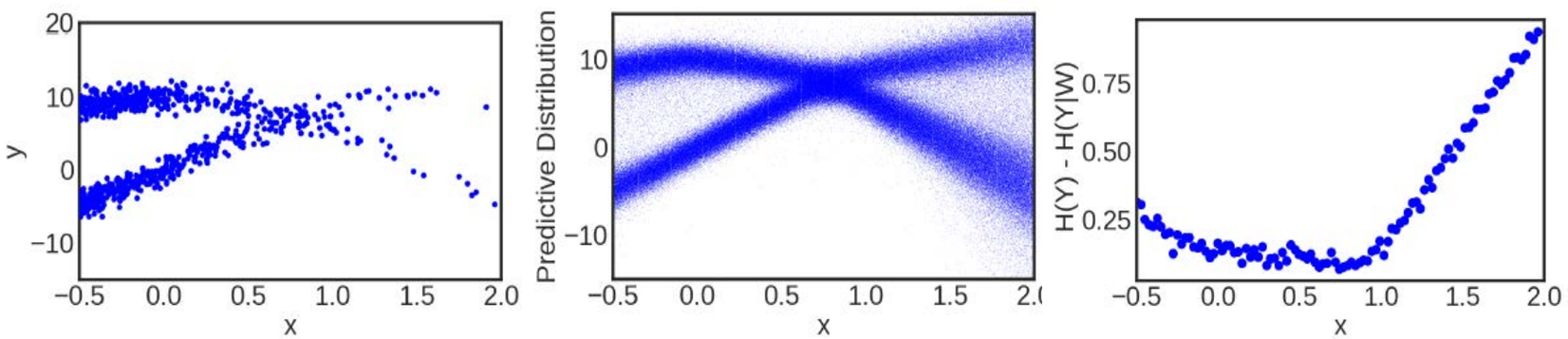
Model stochastic functions

Depweg et al., ICLR 2017



Model uncertainty in deterministic functions

Killian et al., NIPS 2017



Predictive uncertainties for active learning, sequential decision making

Hernández-Lobato et al., ICML 2015, Gal et al., ICML 2017, Joshi et al., CVPR 2017, Zhang et al., AISTATS 2018, Depweg et al., ICML 2018, Riquelme et al., ICLR 2018

Alternate distribution over functions

Gaussian Processes

$$f(x) \sim \mathrm{GP}(m(x), K(x, x'))$$

Bayesian Neural Networks

$$f(x) \sim h_{\mathcal{W}}(x)$$

Rest of this talk, Noisy data model:

$$y(x) \mid f(x) \sim \mathcal{N}(f(x), \sigma^2)$$

Gaussian likelihoods

Gaussian Processes

$$f(x) \sim \mathrm{GP}(m(x), K(x, x'))$$

- Exact Inference*
  * Only for Gaussian Likelihoods
- Scales Poorly with n
- Well calibrated uncertainties

$$f \sim GP(.,.); f \in C$$

- Constraining the space of functions can be difficult

Bayesian Neural Networks

$$f(x) \sim h_{\mathcal{W}}(x)$$

- Approximate Inference
- Scales Well
- Predictive uncertainties can be poor
- Some are easy; depends on C

Gaussian Processes

$$f(x) \sim \mathrm{GP}(m(x), K(x, x'))$$

Completely specified by:

$$m(x) \quad K(x, x')$$

Intuitive, well understood parameterization

Bayesian Neural Networks

$$f(x) \sim h_{\mathcal{W}}(x)$$

Need to specify:

h

architecture *non-linearity*

$$p(\mathcal{W} \mid \lambda)$$

prior on weights

Implied distribution on functions is poorly understood

Predictive Uncertainties?

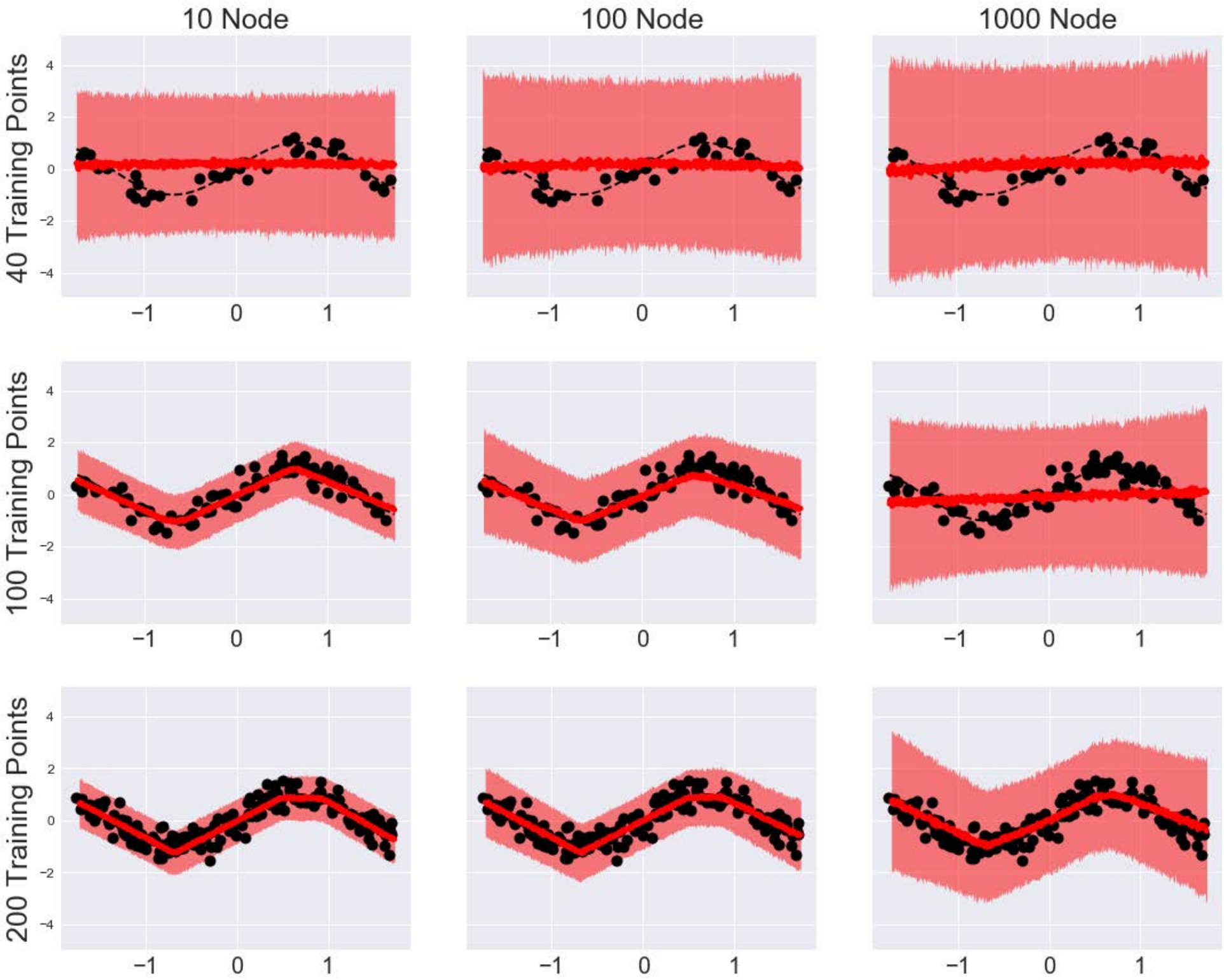
Single layer network, with prior:

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

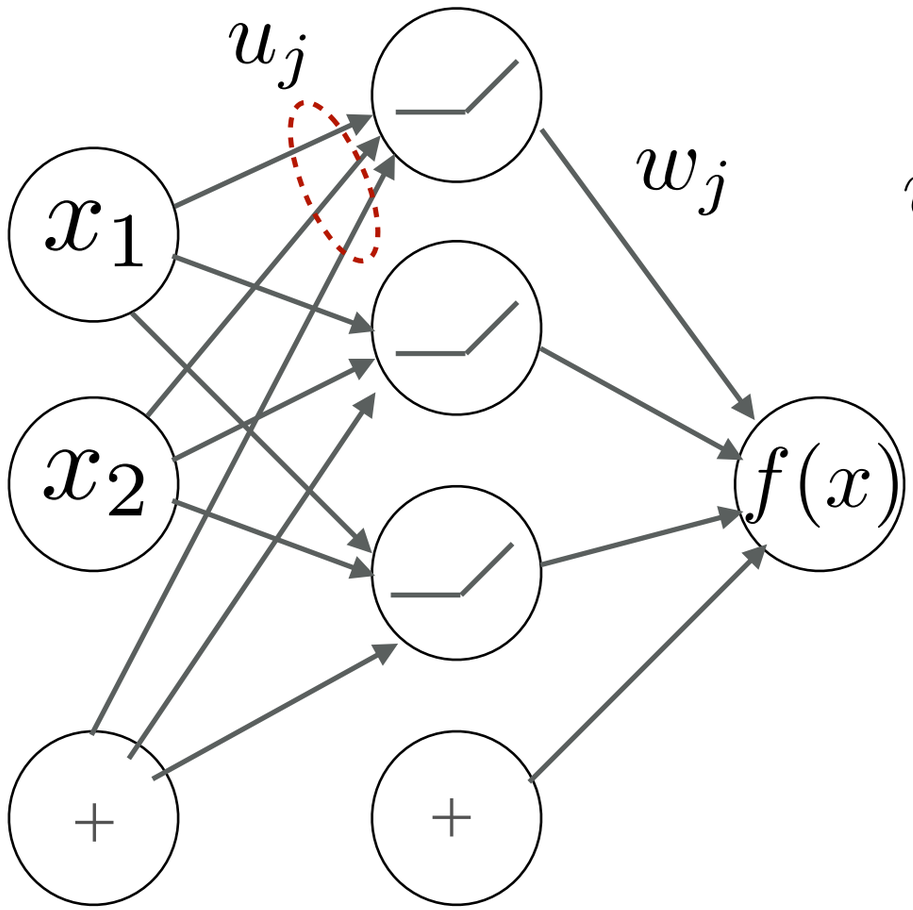
$$\mathcal{N}(y \mid f(x; \mathcal{W}), \gamma^{-1})$$

(Same results across many initialization strategies)

What is happening?



Prior uncertainty



$$w_j \sim \mathcal{N}(0, \sigma_w^2); b \sim \mathcal{N}(0, \sigma_b^2)$$

$$f(x) = b + \sum_{j=1}^{J} w_j \phi(x; u_j)$$

$$E_w[f(x)] = 0$$

$$E_w[f(x)f(x')] = \sigma_b^2 + J\sigma_w^2 E_u[\phi(x; u_j)\phi(x'; u_j)]$$

Computing with Infinite Networks, C. Williams, NIPS 1997
Bayesian leaning for Neural Networks, R. Neal, LNS, 1996

Predictive Uncertainties?

Single layer network, with prior:

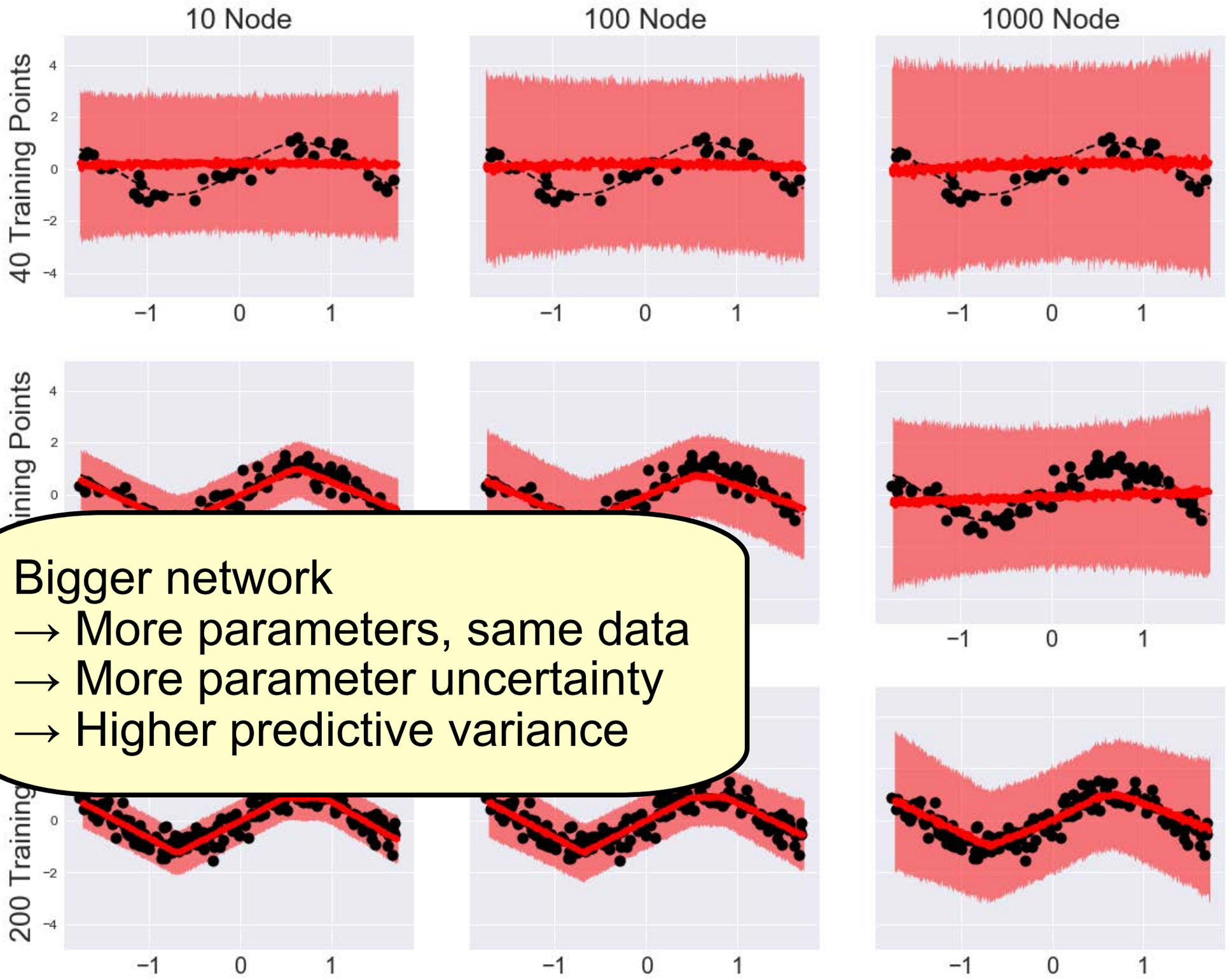
$$\mathcal{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\mathcal{N}(y \mid f(x; \mathcal{W}), \gamma^{-1})$$

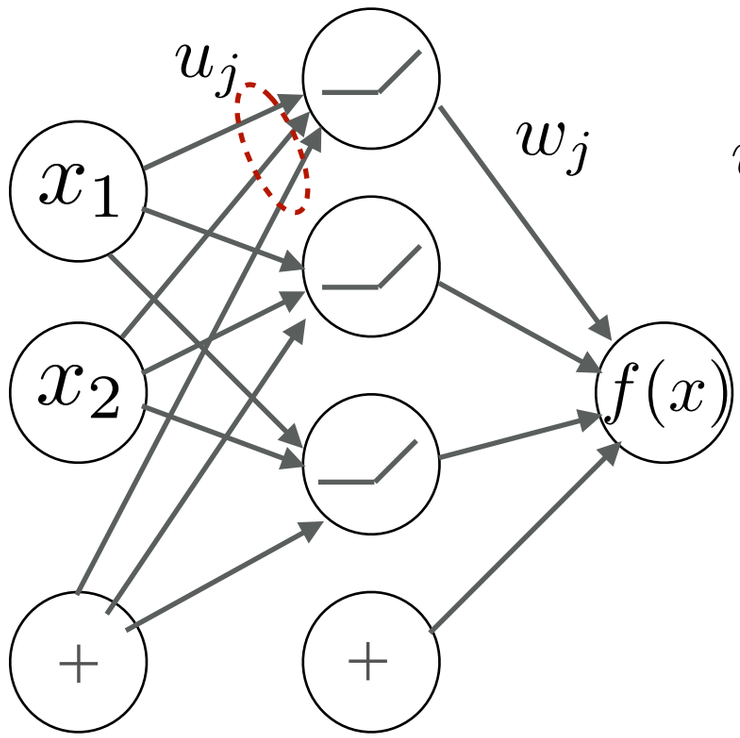
(Same results across many initialization strategies)

What is happening?

Bigger network
→ More parameters, same data
→ More parameter uncertainty
→ Higher predictive variance



Bounding Prior variance



$$w_j \sim \mathcal{N}(0, \sigma_w^2); b \sim \mathcal{N}(0, \sigma_b^2)$$

$$f(x) = b + \sum_{j=1}^{J} w_j \phi(x; u_j)$$

$$E_w[f(x)f(x')] = \sigma_b^2 + J\sigma_w^2 E_u[\phi(x; u_j)\phi(x'; u_j)]$$

Could scale by J $\longrightarrow$ $\sigma_w^2 = \frac{a}{J}$ *C. Williams, NIPS 1997, R. Neal, LNS, 1996*

or

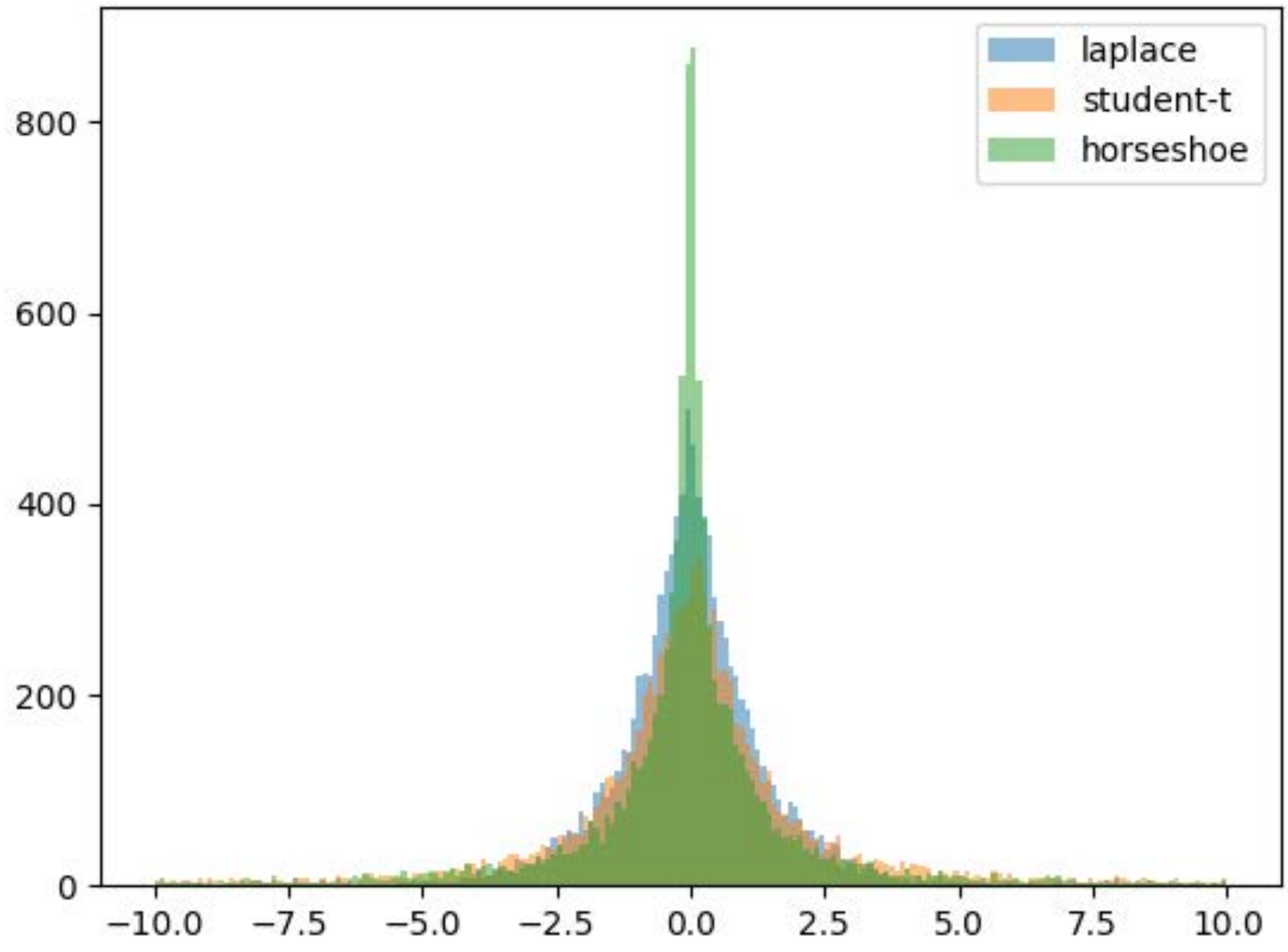
Force J to be small, by turning units off.

Horseshoe Priors for Model Selection

The horseshoe prior is a scale mixture of normals:

$$w_k \sim \mathcal{N}(0, \tau_k^2 v^2)$$

$$\tau_k \sim C^+(0, 1)$$



Group Horseshoe Priors for BNNs

- Horseshoe BNN:

For each layer l, draw a global scale: $v_l \sim C^+(0, b_g)$

For node k in layer l:

- Draw a local scale for the node: $\tau_{kl} \sim C^+(0, b_0)$
- For each incident weight: $w_{kk',l} \sim \mathcal{N}(0, \tau_{kl}^2 v_l^2)$

- Inference:

Stochastic gradient variational Bayes / BBVI + reparameterized gradients

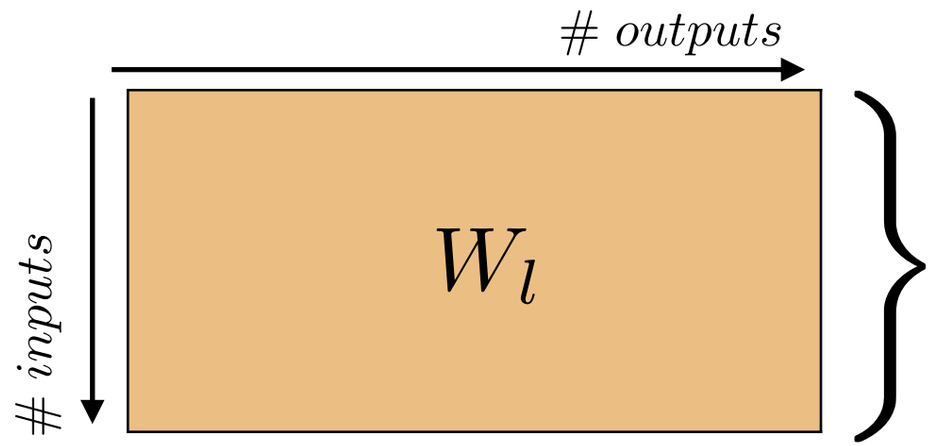
$$\mathcal{L}(\phi) = E_{q(\mathcal{W},\tau,v;\phi)}[\underbrace{\ln\, p(y \mid \mathcal{W}, x))}_{\text{NN; intractable expectation}} + \ln\, p(\mathcal{W}, \tau, v)] + H[q(\mathcal{W}, \tau, v; \phi)]$$

Model Selection in Bayesian Neural Networks via Horseshoe Priors; Ghosh & Doshi-Velez, 2017
Bayesian deep compression; Louizos et. al., 2017

Local Reparameterization

- Continuous *weights* and *variances* —> reparameterization trick

$$\nabla_{\mu,\sigma}\mathbb{E}_{q_w}[g(w)] \Leftrightarrow \nabla_{\mu,\sigma}\mathbb{E}_{\mathcal{N}(\epsilon|0,1)}[g(\mu+\sigma\epsilon)] \approx \frac{1}{S}\sum_s \nabla_{\mu,\sigma} g(\mu+\sigma\epsilon^{(s)})$$

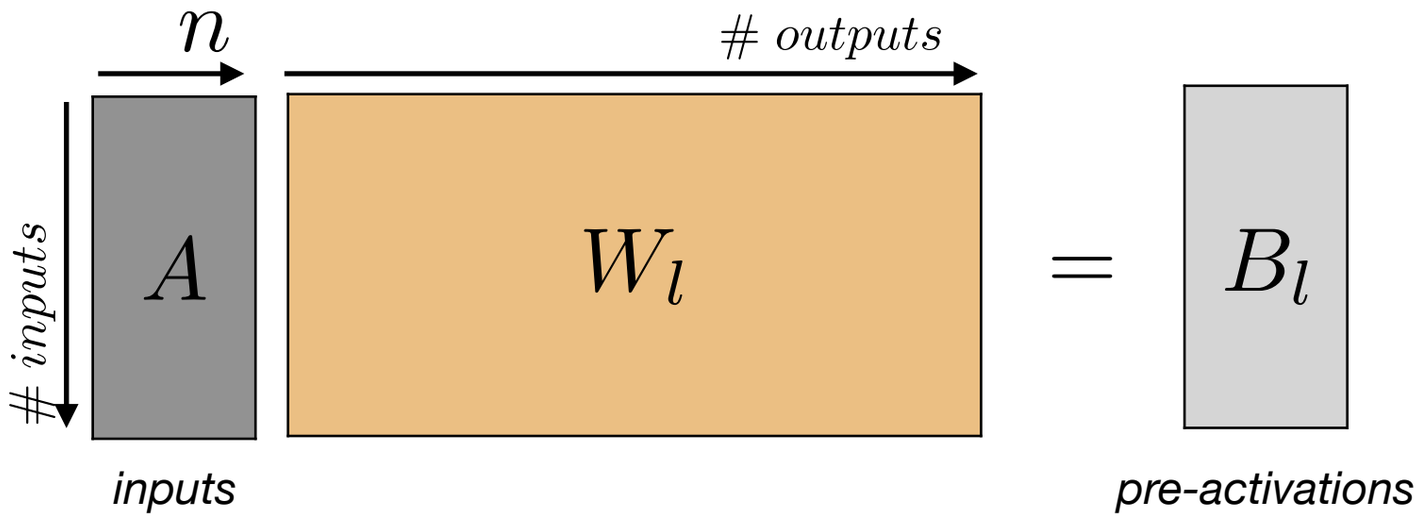


weights

$$W_l^{(s)} \sim q(W)$$

naive application

- Local reparameterization — provably lower variance



$$B_l^{(s)} \sim q(B)$$

For certain q(W), form of implied q(B = AW) is known

Variational dropout and the local reparameterization trick, Kingma' 2015

Variational Family

- Fully factorized variational approximation

$$q(\mathcal{W}, \tau, v; \phi) = \prod_{i,j,l} \mathcal{N}(w_{i,j,l} \mid \mu_{i,j,l}, \sigma^2_{i,j,l}) \prod_{k,l} q(\tau_{k,l} \mid \phi_{\tau_{k,l}}) \prod_{l} q(v_l \mid \phi_{v_l})$$

Louizos et. al., 2017
Ghosh & Doshi-Velez 2017

- But Horseshoe shrinkage stems from coupling between weights and scales

- Retaining this structure is important for strong shrinkage!

Group Horseshoe Priors for BNNs

- ~~Horseshoe BNN:~~ Regularized Horseshoe BNN

  For each layer l, draw a global scale: $v_l \sim C^+(0, b_g)$

  For node k in layer l:

  - Draw a local scale for the node: $\tau_{kl} \sim C^+(0, b_0)$
  - For each incident weight: $w_{kk',l} \sim \mathcal{N}(0, \tau_{kl}^2 v_l^2)$

- Inference:

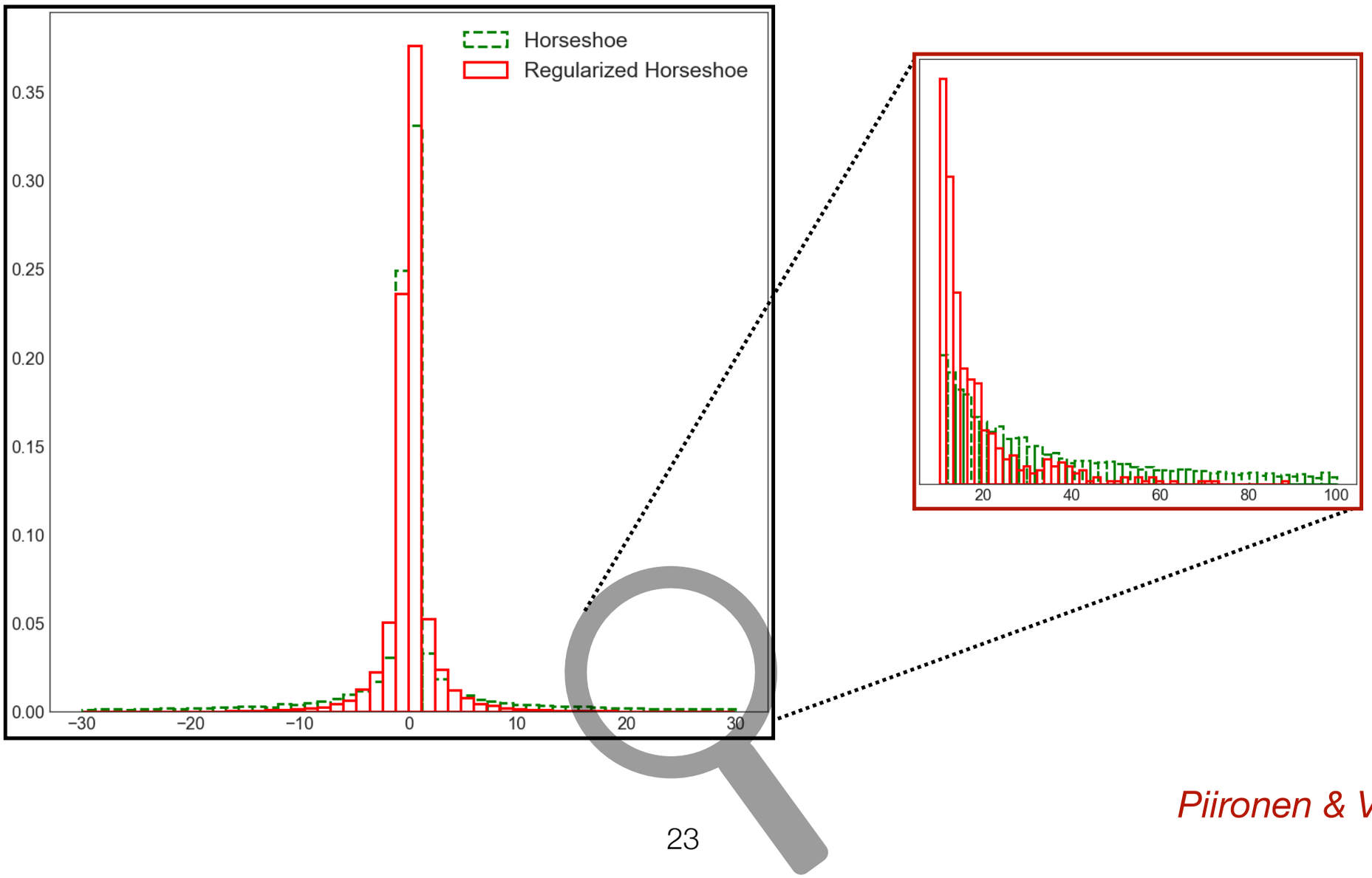
  Stochastic gradient variational Bayes with ~~naive~~ *structured* ~~*fully factorized*~~ variational approximations.

Regularized Horseshoe

$$p(w_{kk',l} \mid \tau_{kl}, v_l, \ c\ ) \propto \mathcal{N}(w_{kk',l} \mid 0, \tau_{kl}^2 v_l^2)\ \mathcal{N}(w_{kk',l} \mid 0, c^2)$$

Equivalently,

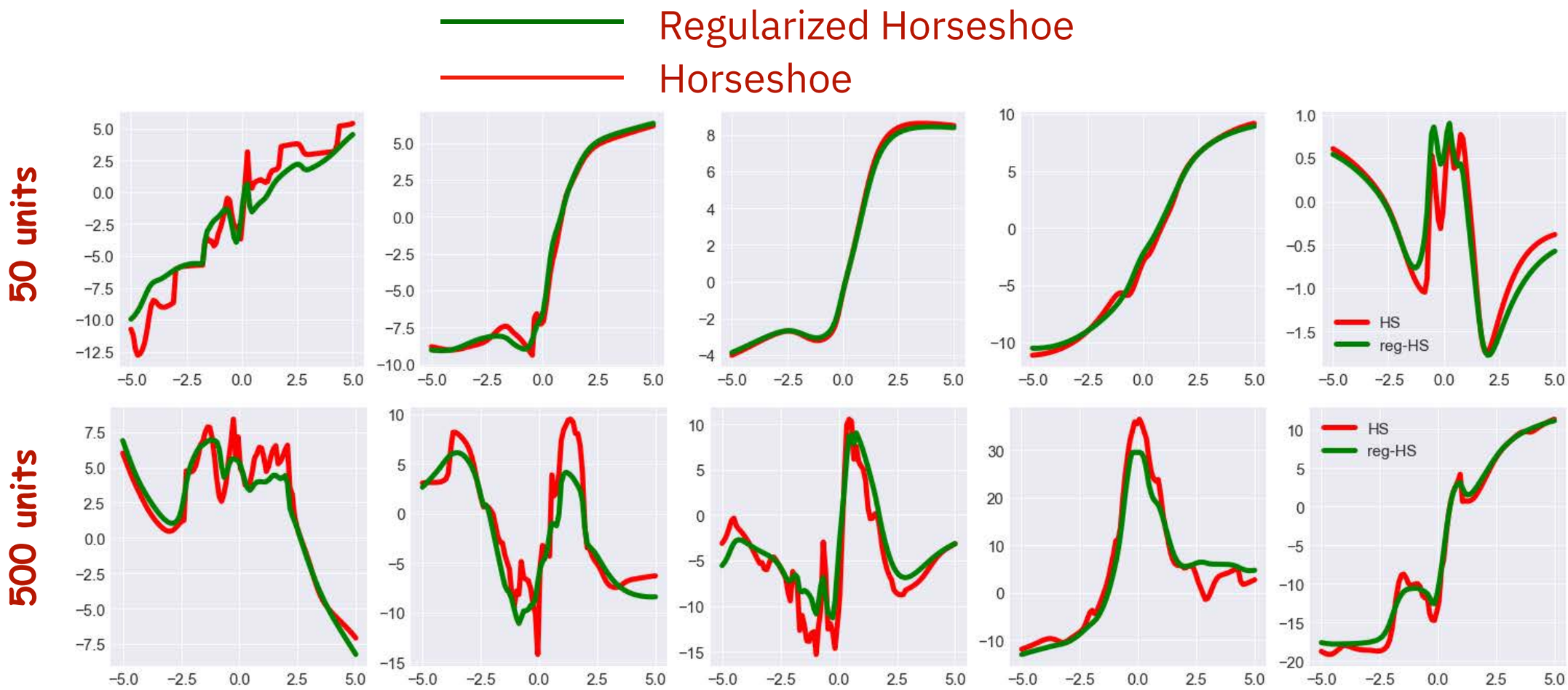
$$w_{kk',l} \mid c, \tau_{kl}, v_l \sim \mathcal{N}(w_{kl} \mid 0, \tilde{\tau}_{kl}^2 v_l^2); \qquad \frac{1}{\tilde{\tau}_{kl}^2 v_l^2} = \frac{1}{c^2} + \frac{1}{\tau_{kl}^2 v_l^2}$$



Piironen & Vehtari, 2017

Regularized Horseshoe BNNs

$$w_{kl} \mid \tau_{kl}, v_l, c \sim \mathcal{N}(0, (\tilde{\tau}_{kl}^2 v_l^2)\mathbb{I}), \quad \frac{1}{\tilde{\tau}_{kl}^2 v_l^2} = \frac{1}{c^2} + \frac{1}{\tau_{kl}^2 v_l^2}$$

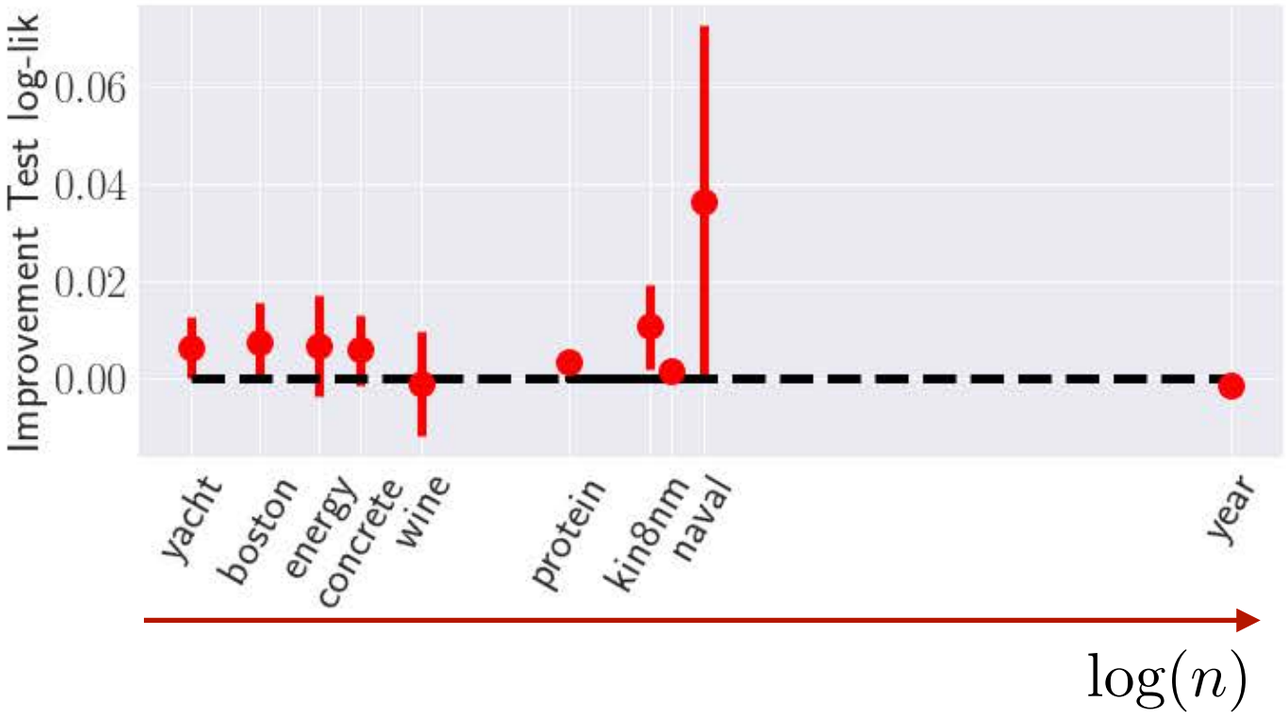


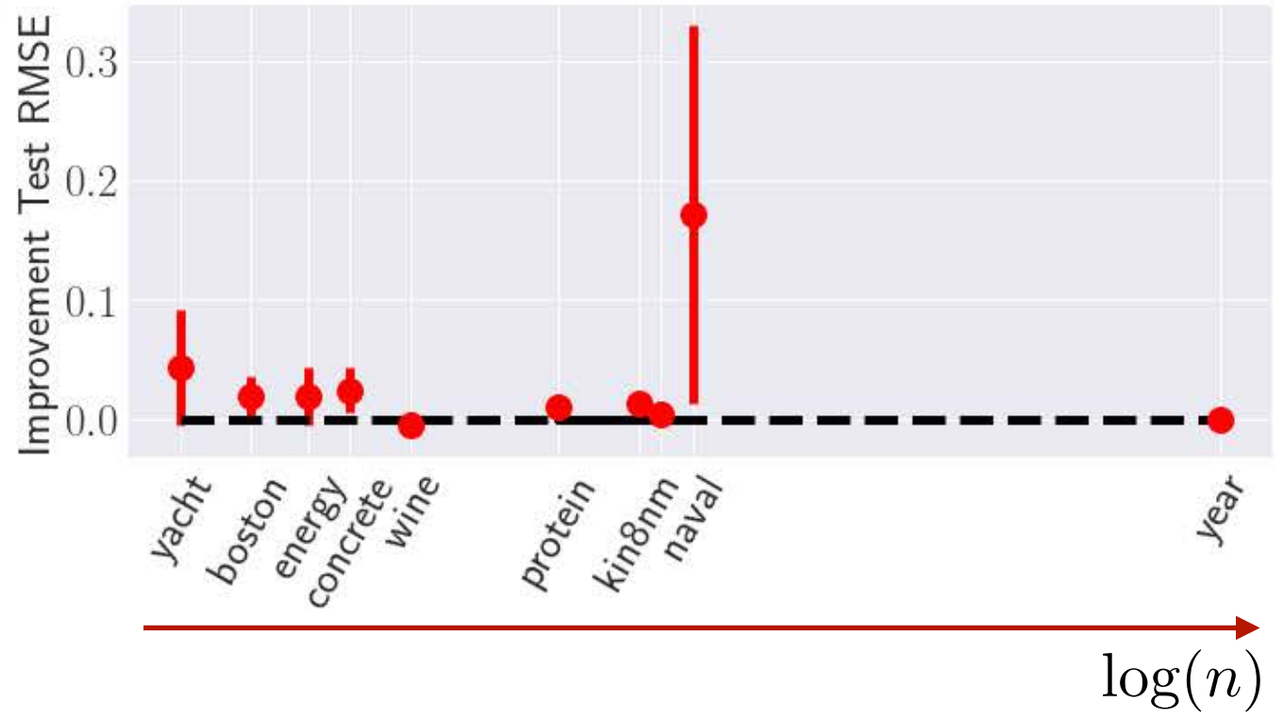
Random functions from single hidden layer (tanh) network with HS and reg-HS priors

Regularized Horseshoe BNNs

$$w_{kl} \mid \tau_{kl}, \upsilon_l, c \sim \mathcal{N}(0, (\tilde{\tau}_{kl}^2 \upsilon_l^2)\mathbb{I}), \quad \frac{1}{\tilde{\tau}_{kl}^2 \upsilon_l^2} = \frac{1}{c^2} + \frac{1}{\tau_{kl}^2 \upsilon_l^2}$$

UCI Regression Benchmarks *(Hernández-Lobato and Adams' 2015)*



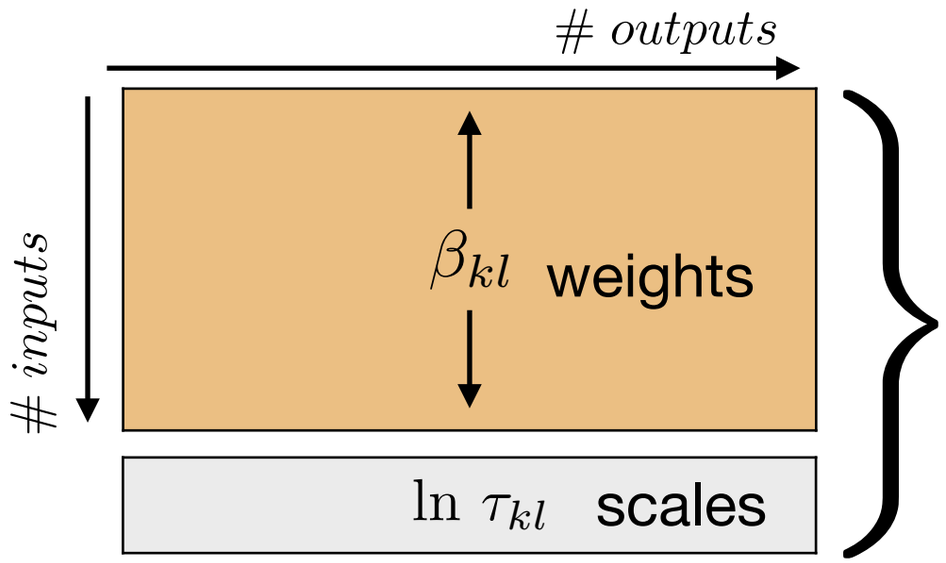


reg-HS BNNs improves predictive performance over HS BNNs for smaller datasets.

Relative improvement: (x - y)/ max(|x|, |y|)

Structured Variational Approximation

- Weights incident on a unit: $w_{kl} \mid \tau_{kl}, v_l, c \sim \mathcal{N}(0, (\tilde{\tau}_{kl}^2 v_l^2)\mathbb{I})$
- Non-centered Parameterization: $\beta_{kl} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ $w_{kl} = \tau_{kl} v_l \beta_{kl}$
- Layer specific structured variational approximations:



$$q(B) = \text{Matrix-Normal}(M, U, V)$$

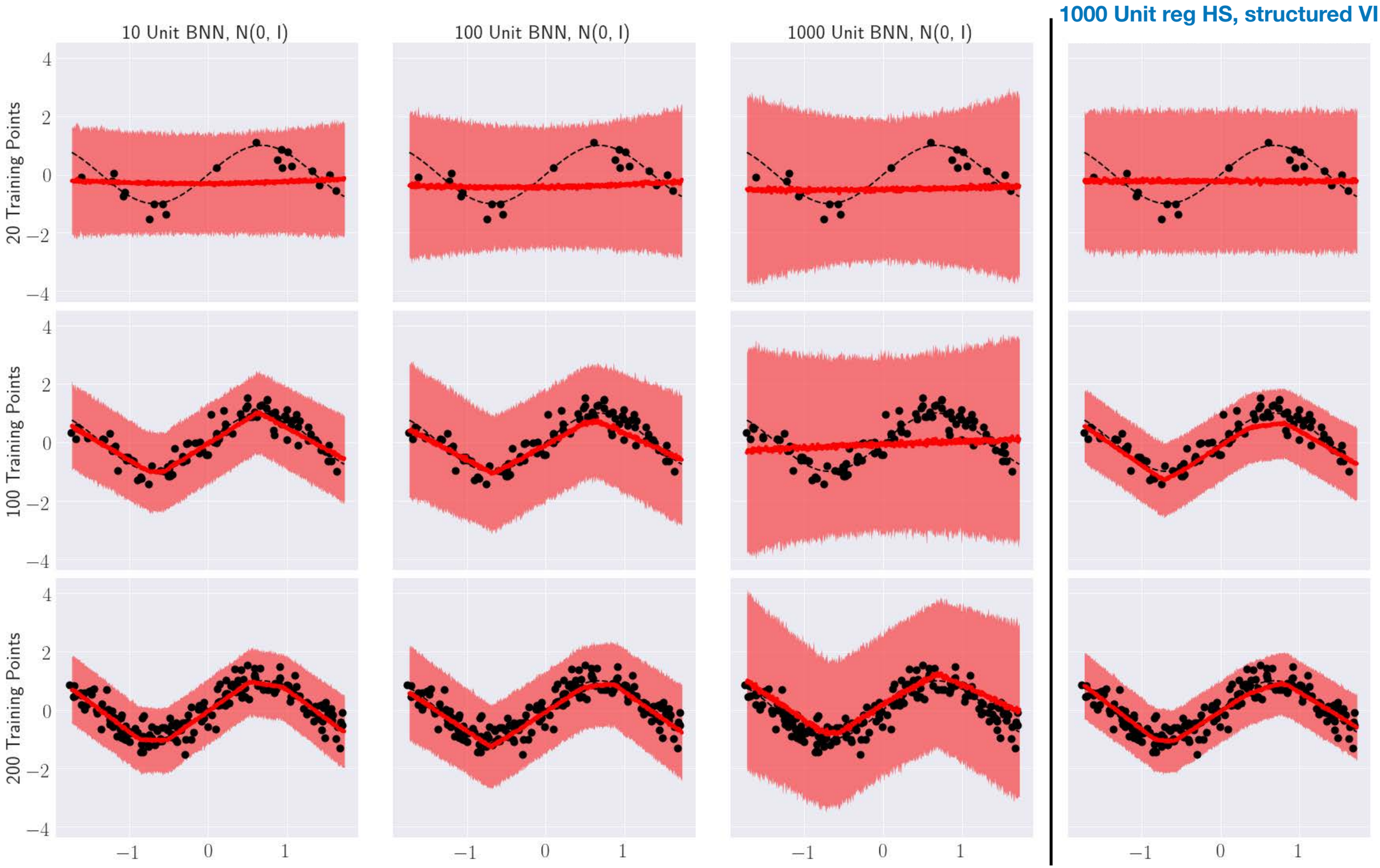
$$U = hh' + \Psi$$

Low dimensional covariance maintains posterior structure between **weights** and **scales**.

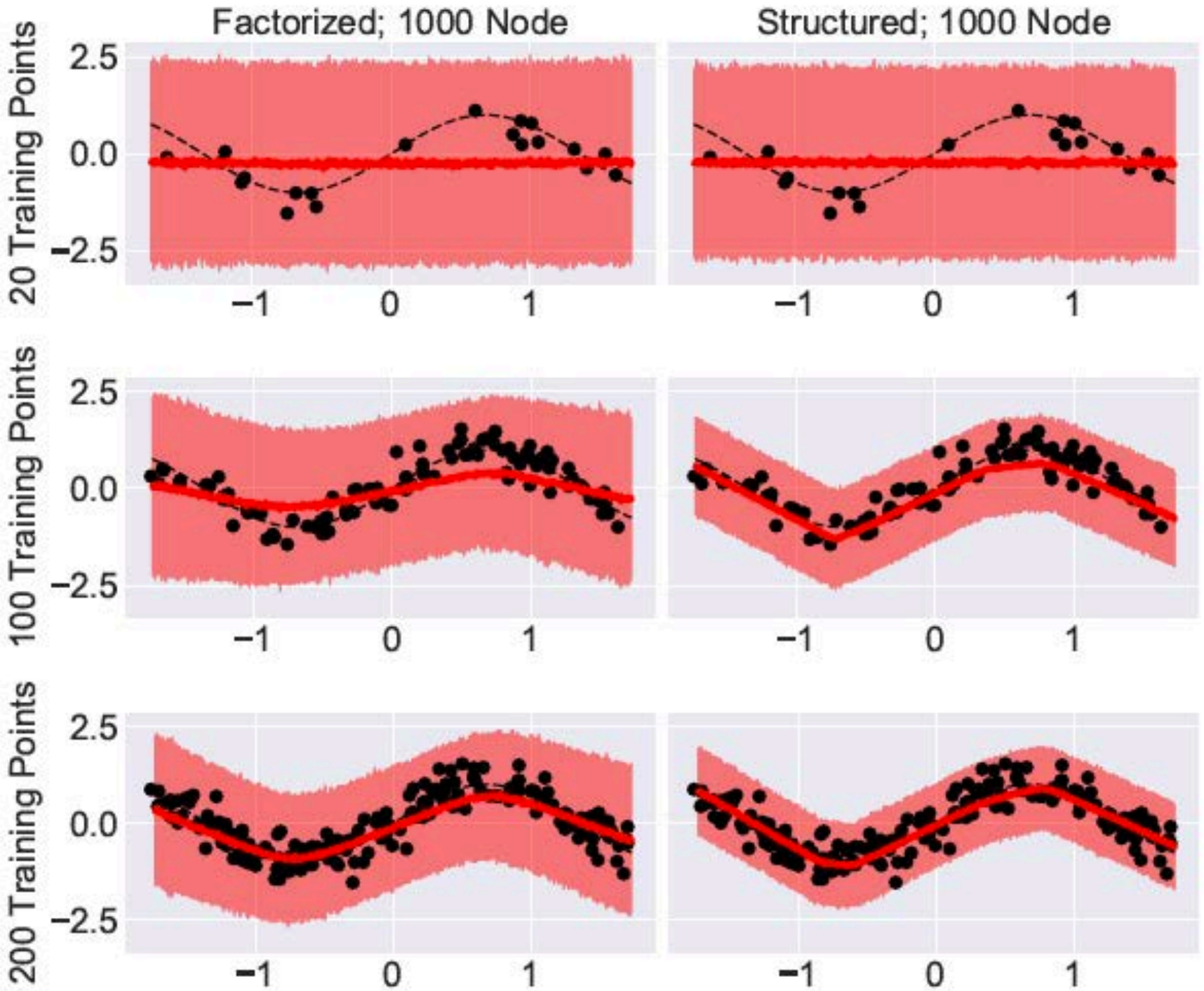
- Local re-parameterization:

$$q\Big(\boxed{\beta_{kl}} \;\Big|\; \boxed{\ln \tau_{kl}}\Big) = \text{Matrix-Normal}(M_{\beta|\tau}, U_{\beta|\tau}, V_{\beta|\tau})$$

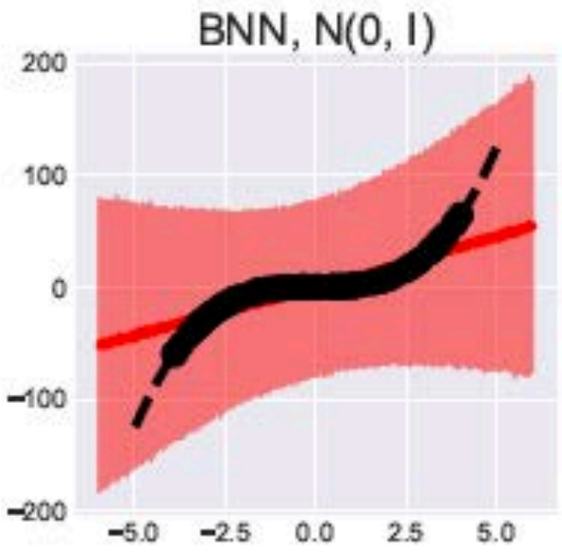
Synthetic Data: Better Fits

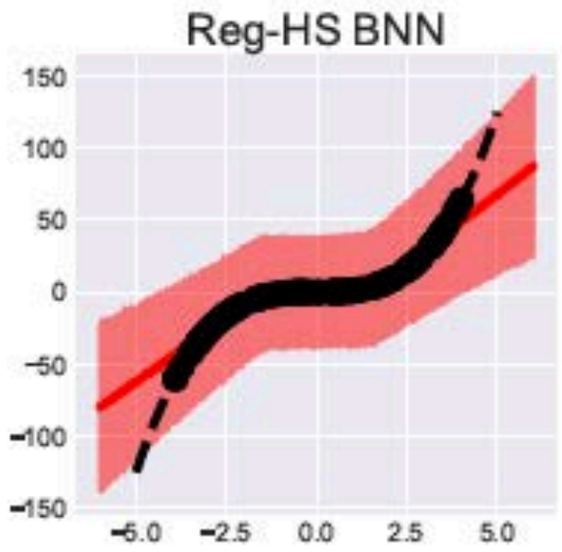


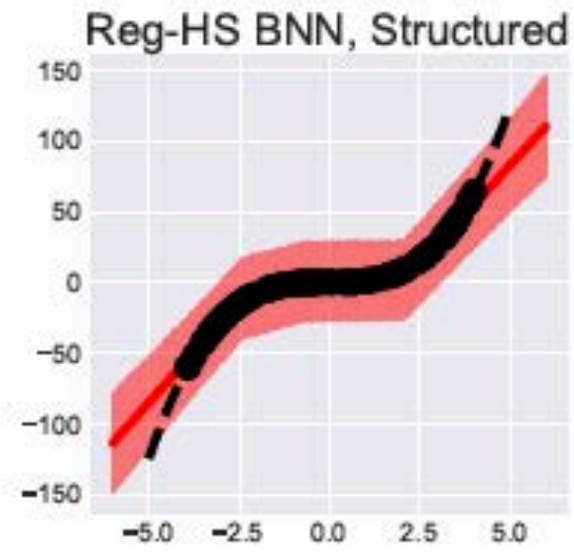
Structured vs Factorized

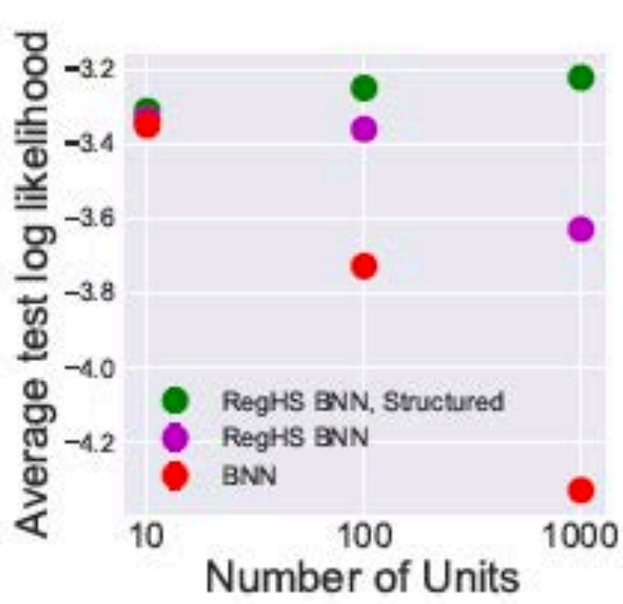


Structured vs Factorized







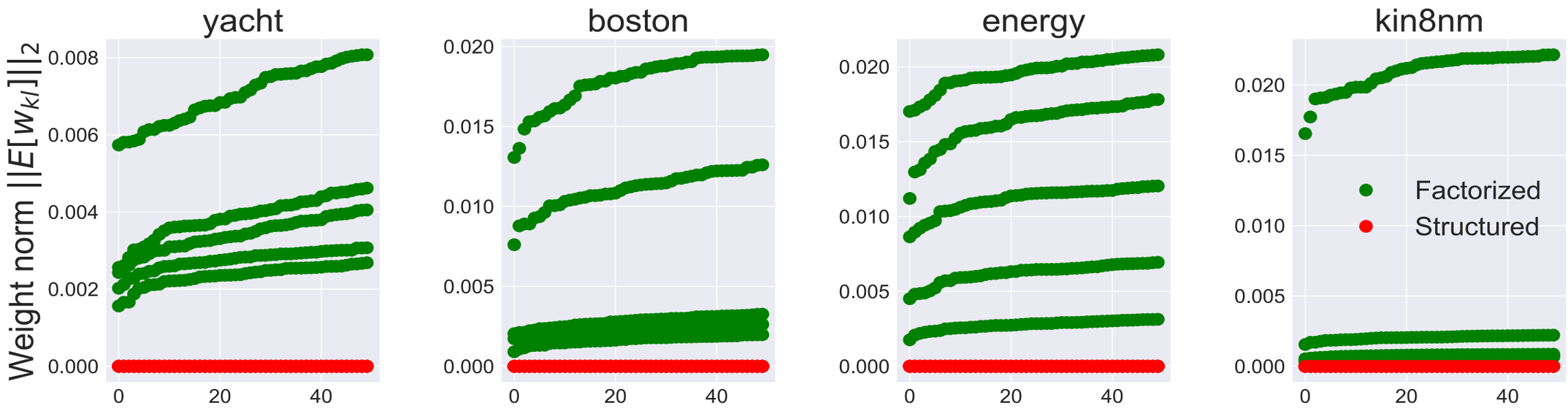


$$y_n = x_n^3 + \epsilon_n,\ \epsilon_n \sim \mathcal{N}(0, 9)$$

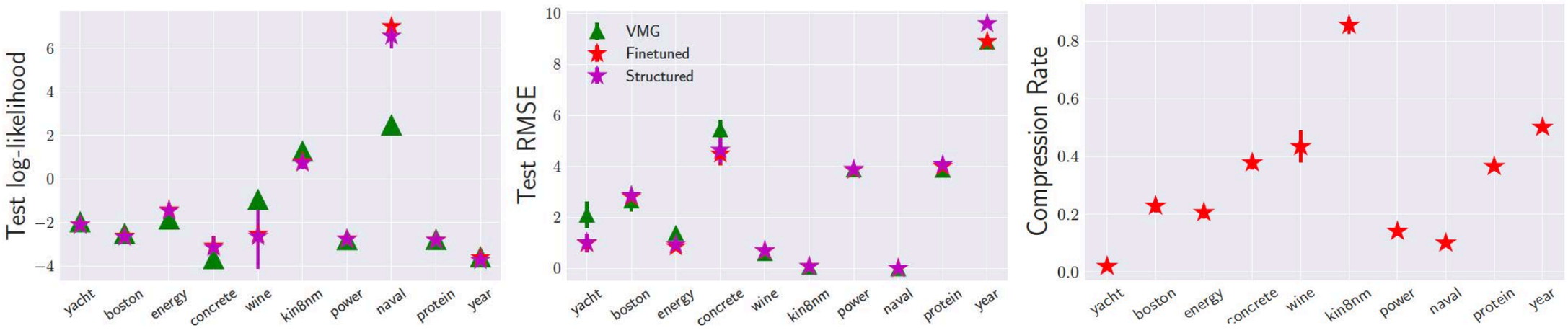
Five hundred training points

UCI Regression Tasks

- Structured variational approximation -> stronger shrinkage, similar predictive performance



- Predictive Performance:



Comparisons with Variational matrix Gaussian (Louizos & Welling, ICML 2016)

$$q(\tau_{kl} v_l < \delta) > p_0$$

Pruning rule uses the variational posterior

Summary

- (Regularized) Horseshoe Priors for BNNs can assist with model-selection
  - Recover small networks with similar performance to larger networks.

- Careful modeling of posterior structure between weights and scales is essential for reliable shrinkage.

We are hiring!

http://mitibmwatsonailab.mit.edu/careers/
http://www.research.ibm.com/labs/cambridge/

75 Binney Street, Cambridge, MA 02142