

# SCALABLE BAYESIAN DEEP LEARNING: SUBSPACE INFERENCE

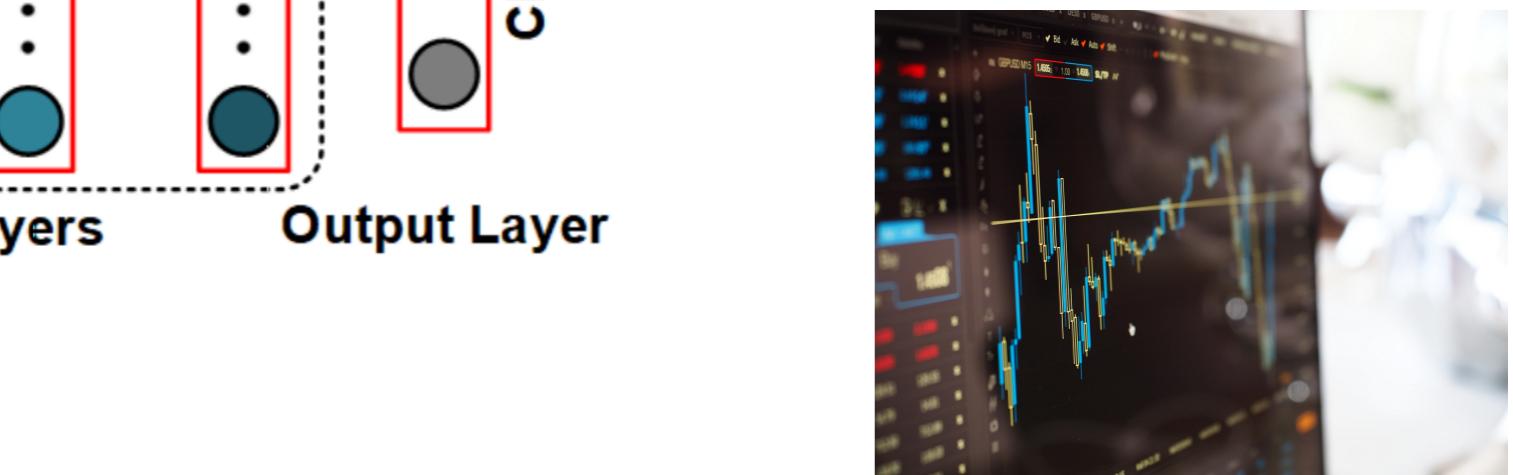
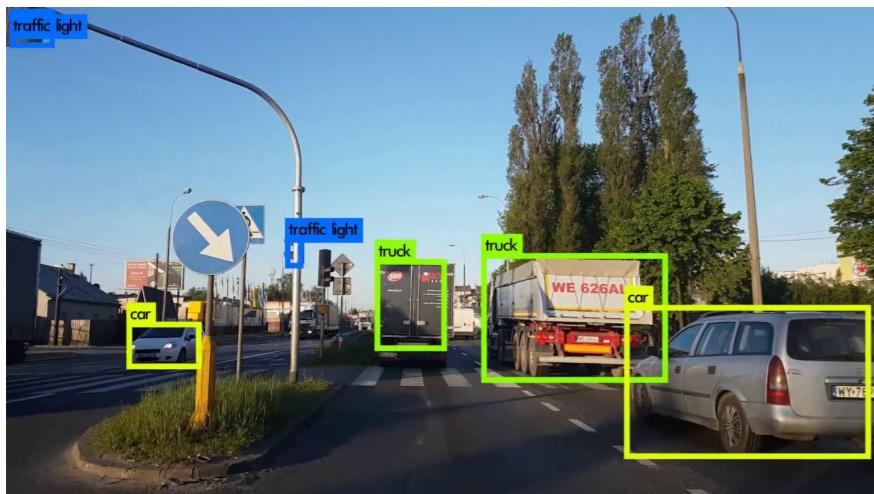
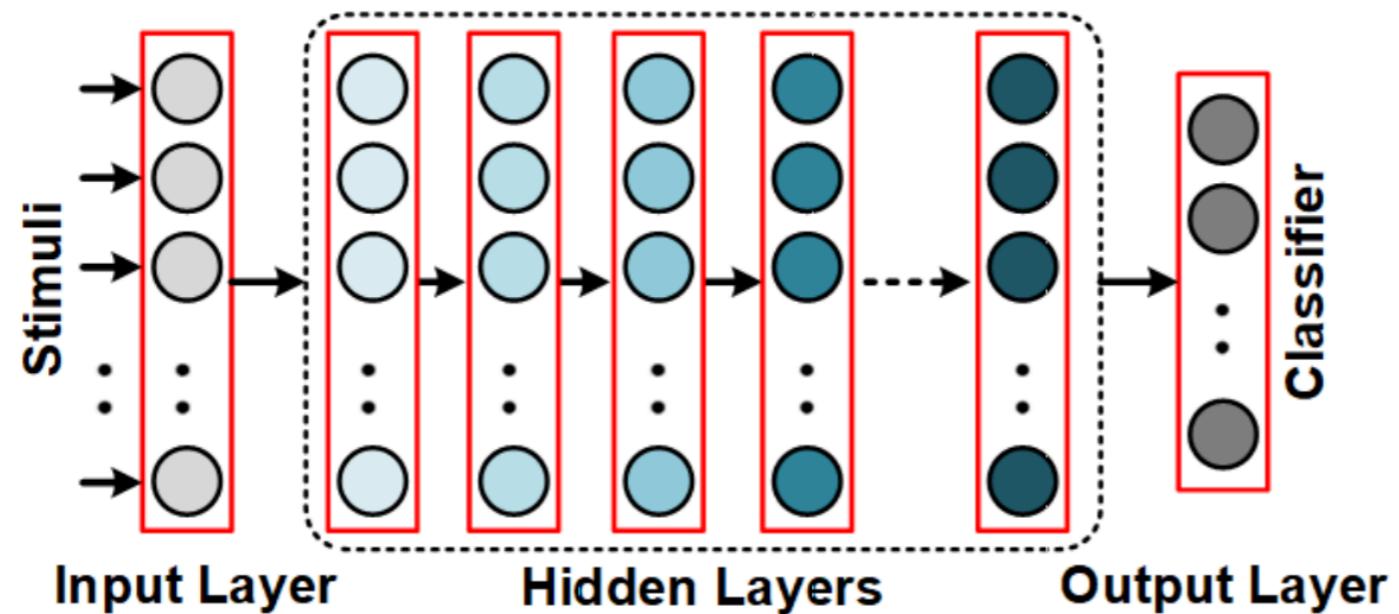
---

Polina Kirichenko

New York University



# DEEP LEARNING



Google Translate

Text Documents

ENGLISH - DETECTED ENGLISH SPANISH FRENCH FRENCH ENGLISH SPANISH

city ville

'site'

Definitions of city

**Noun**

- a large town.  
"But we do not accept this fate with the torpor of other city dwellers."
- a place or situation characterized by a specified attribute.  
"panic city"
- the financial and commercial district of London, England.  
"Reaction in the City was on the cool side, as it also tended to be in Europe."

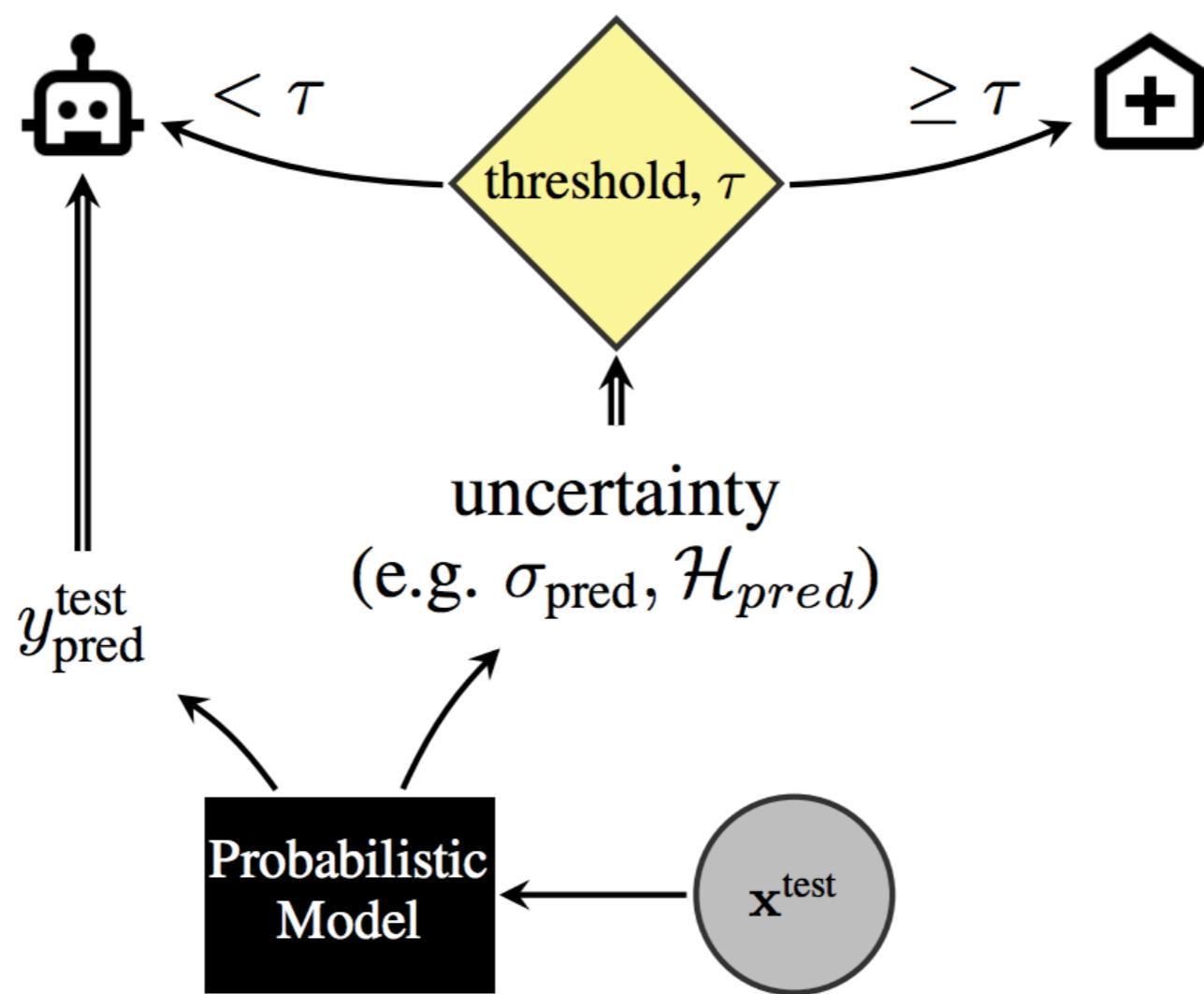
Translations of city

**Noun**

la ville city, town, place, burgh

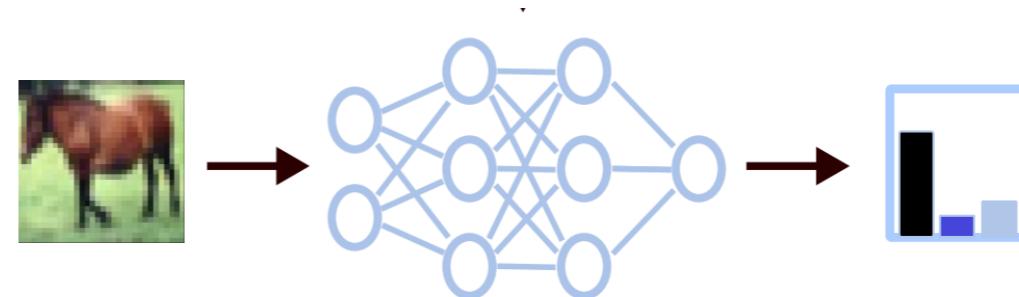
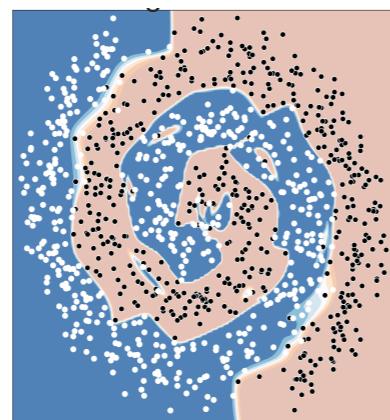
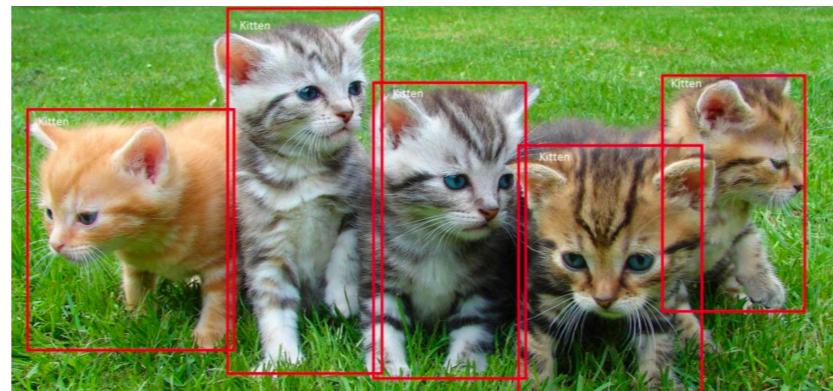
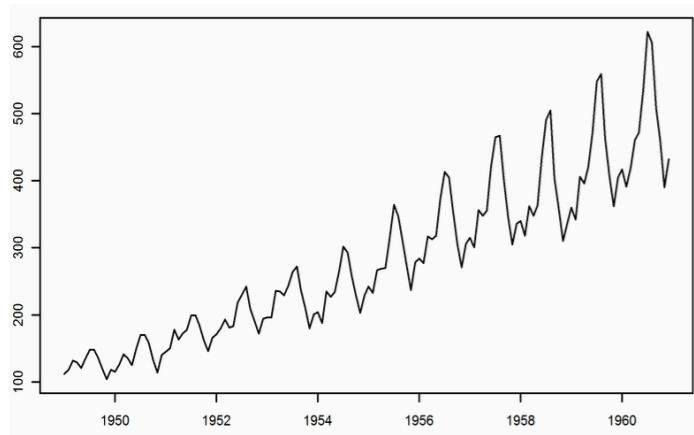
# UNCERTAINTY IN DEEP LEARNING

Automated diagnosis: human-in-the-loop



# DEEP LEARNING

- ▶ Supervised learning: for  $\{X_i, y_i\}_{i=1}^N$  estimate  $p_\theta(y|x)$  where  $X, y \sim p^*(x, y)$



# DEEP LEARNING

- ▶ Supervised learning: for  $\{X_i, y_i\}_{i=1}^N$  estimate  $p_\theta(y|x)$  where  $X, y \sim p^*(x, y)$
- ▶ A neural network  $f_\theta(x)$  will define a  $p_\theta(y|x)$
- ▶ The network is trained by minimizing loss  $\mathcal{L}(f_\theta(x), y)$  for training examples

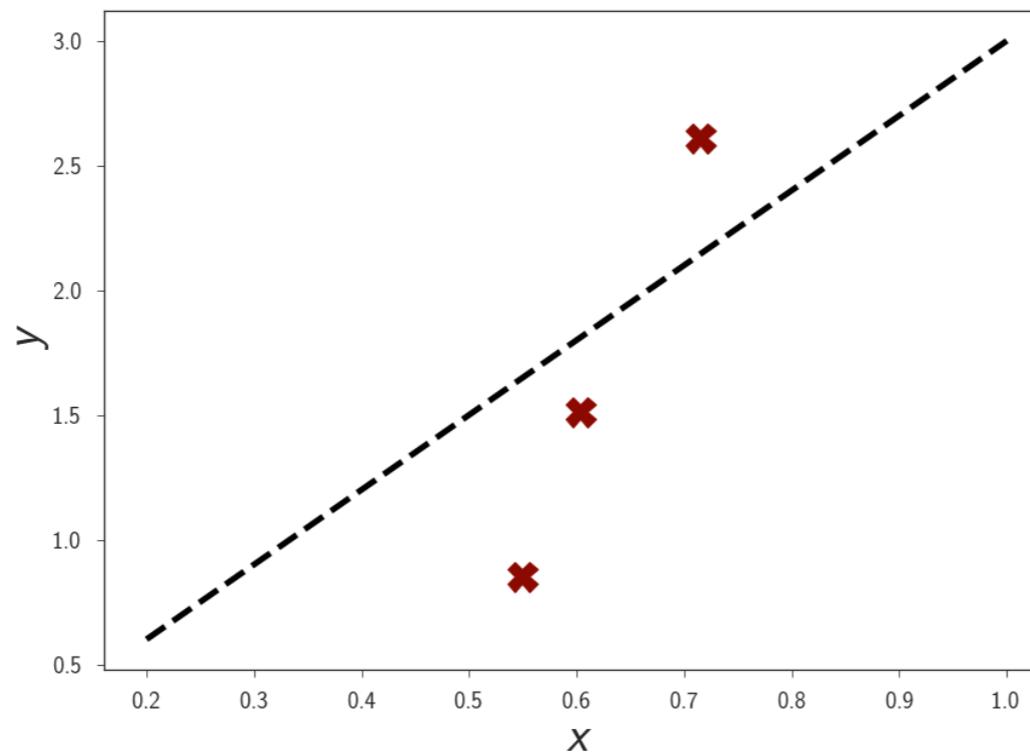
# DEEP LEARNING

- ▶ Supervised learning: for  $\{X_i, y_i\}_{i=1}^N$  estimate  $p_\theta(y|x)$  where  $X, y \sim p^*(x, y)$
- ▶ A neural network  $f_\theta(x)$  will define a  $p_\theta(x)$
- ▶ The network is trained by minimizing loss  $\mathcal{L}(f_\theta(x), y)$  for training examples
- ▶ Equivalent to maximizing **likelihood** function  $\log p_\theta(y|x)$

# BAYESIAN MACHINE LEARNING

Consider a simple linear regression problem:

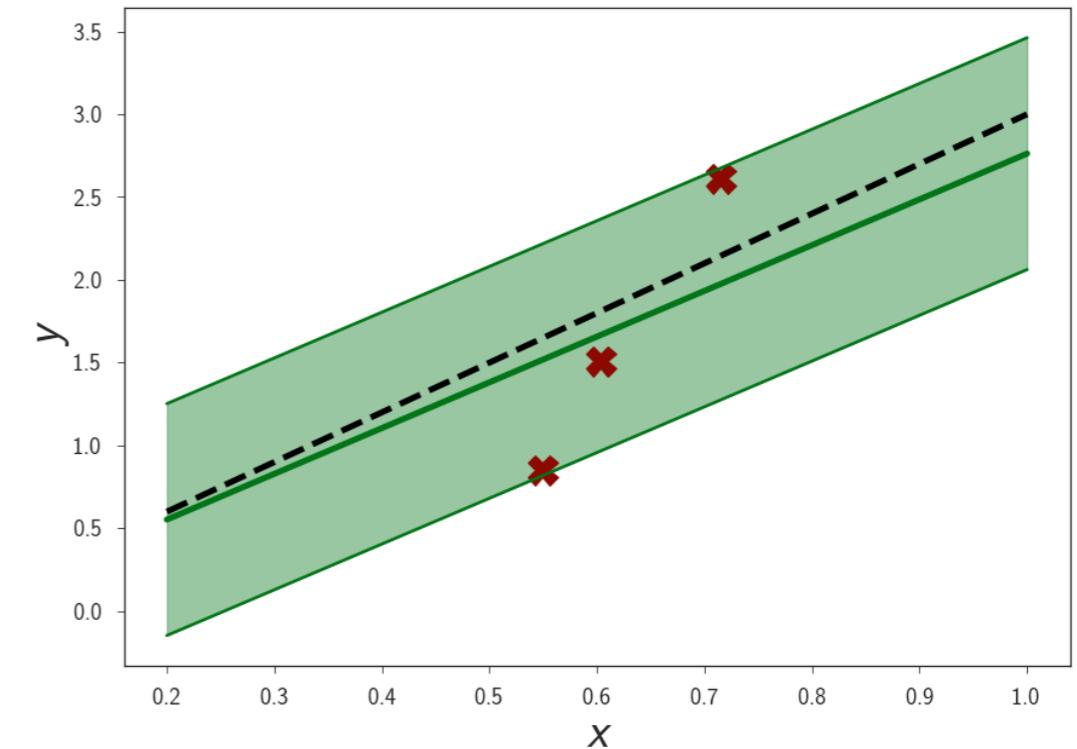
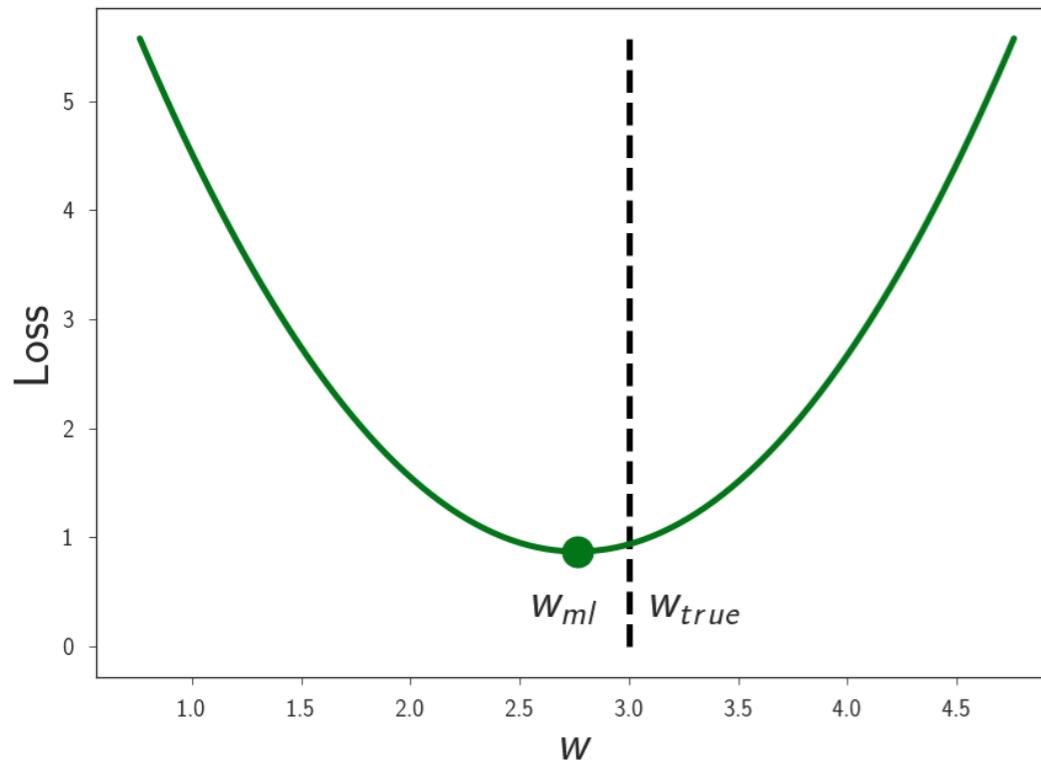
$$y = wx + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$



# BAYESIAN MACHINE LEARNING

Standard linear regression:

$$\max_w \sum_{i=1}^N \log \mathcal{N}(y_i | wx_i, \sigma^2) \iff \min_w \frac{1}{N} \sum_{i=1}^N (y_i - wx_i)^2$$

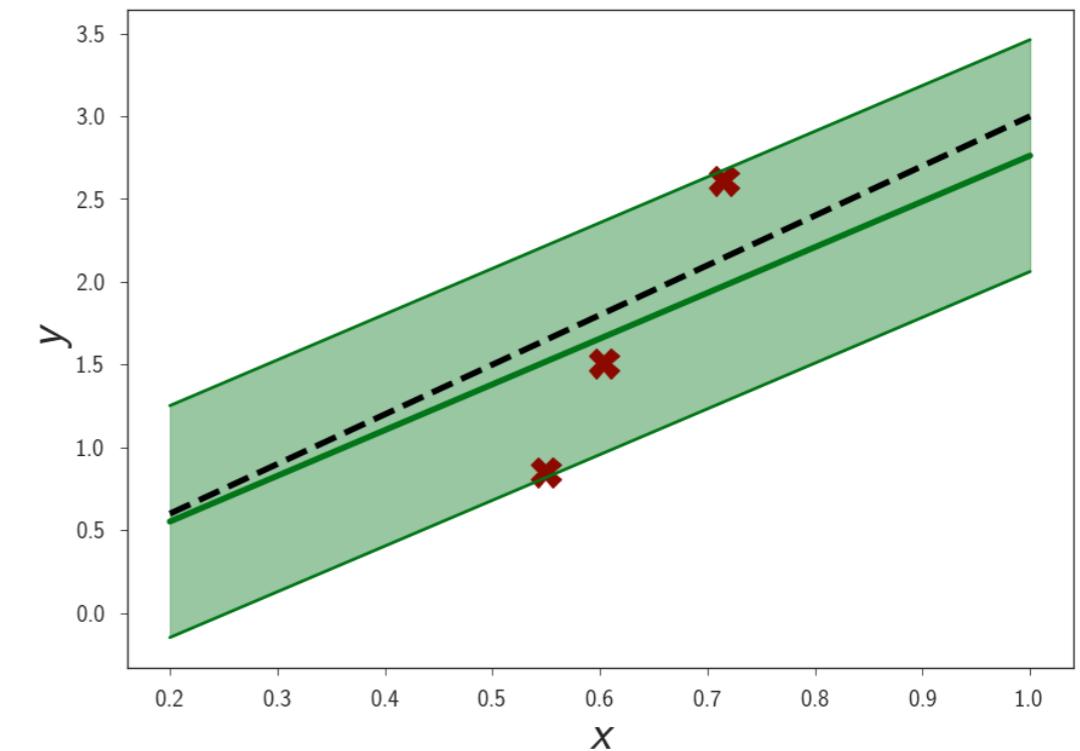
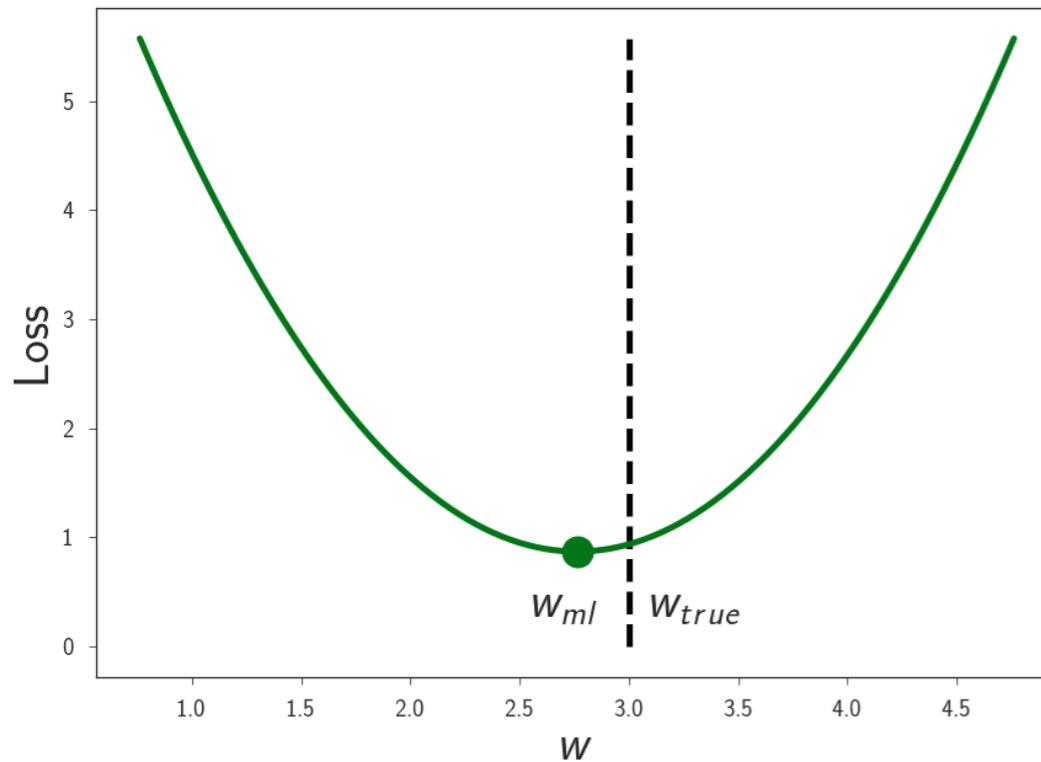


# BAYESIAN MACHINE LEARNING

Standard linear regression:

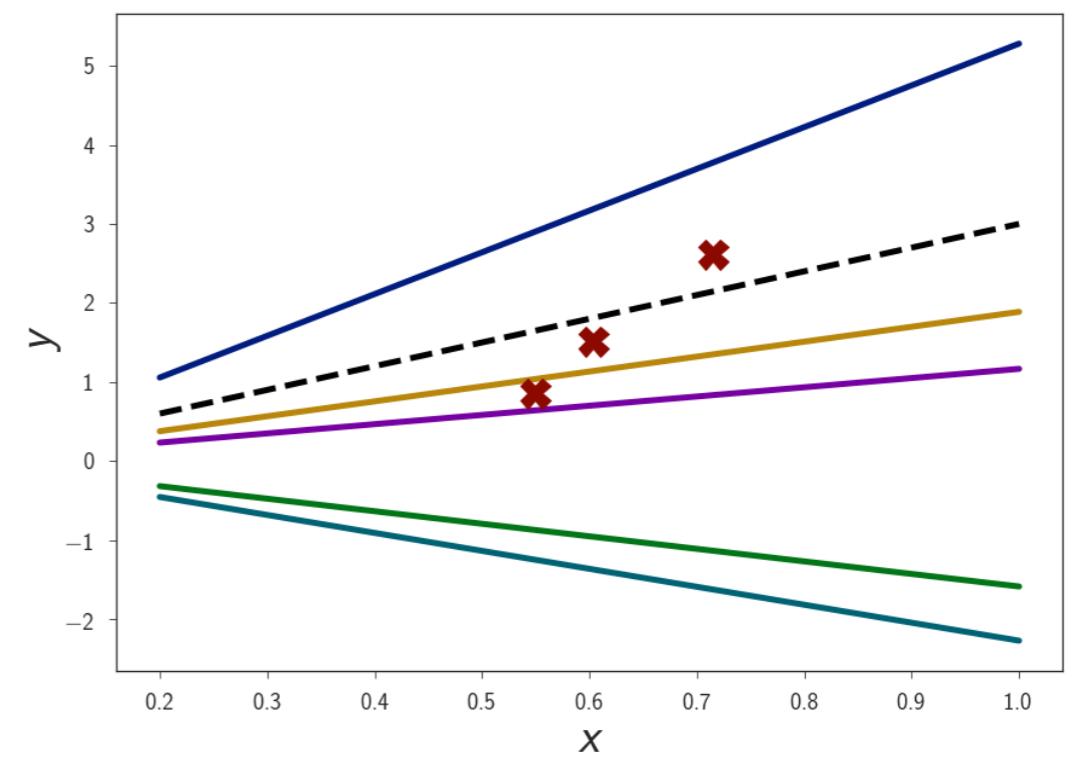
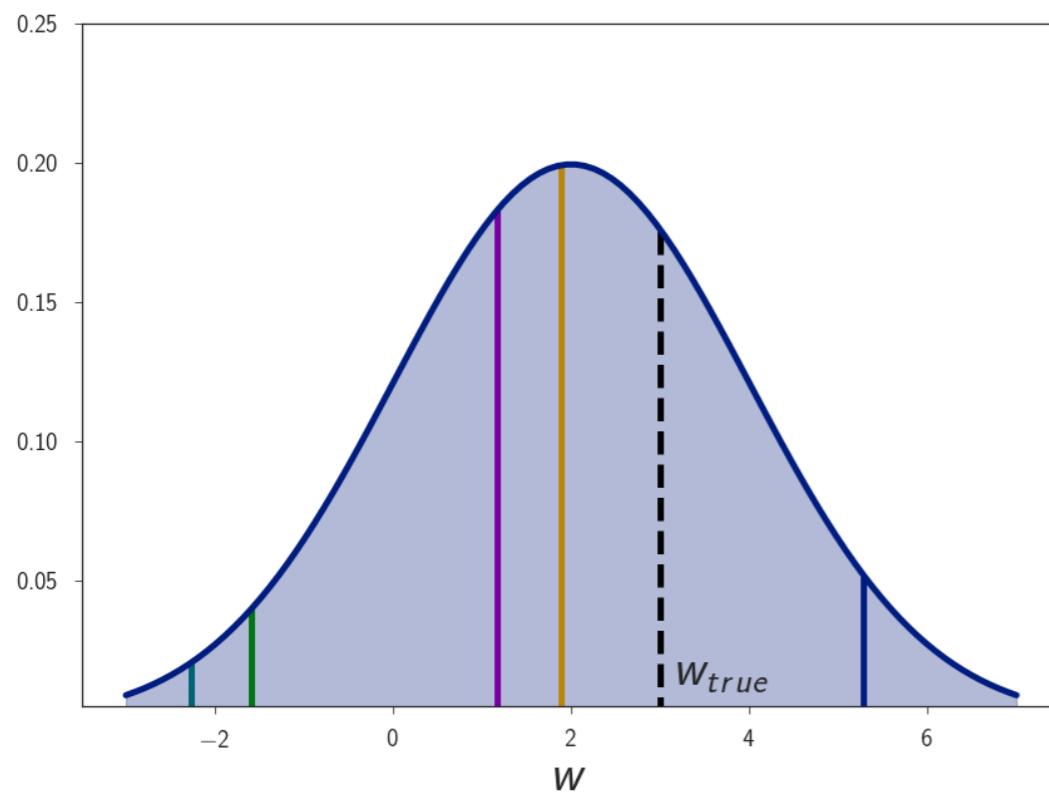
$$\max_w \sum_{i=1}^N \log \mathcal{N}(y_i | wx_i, \sigma^2) \iff \min_w \frac{1}{N} \sum_{i=1}^N (y_i - wx_i)^2$$

*We want to model uncertainty over parameters of the model*



# BAYESIAN MACHINE LEARNING

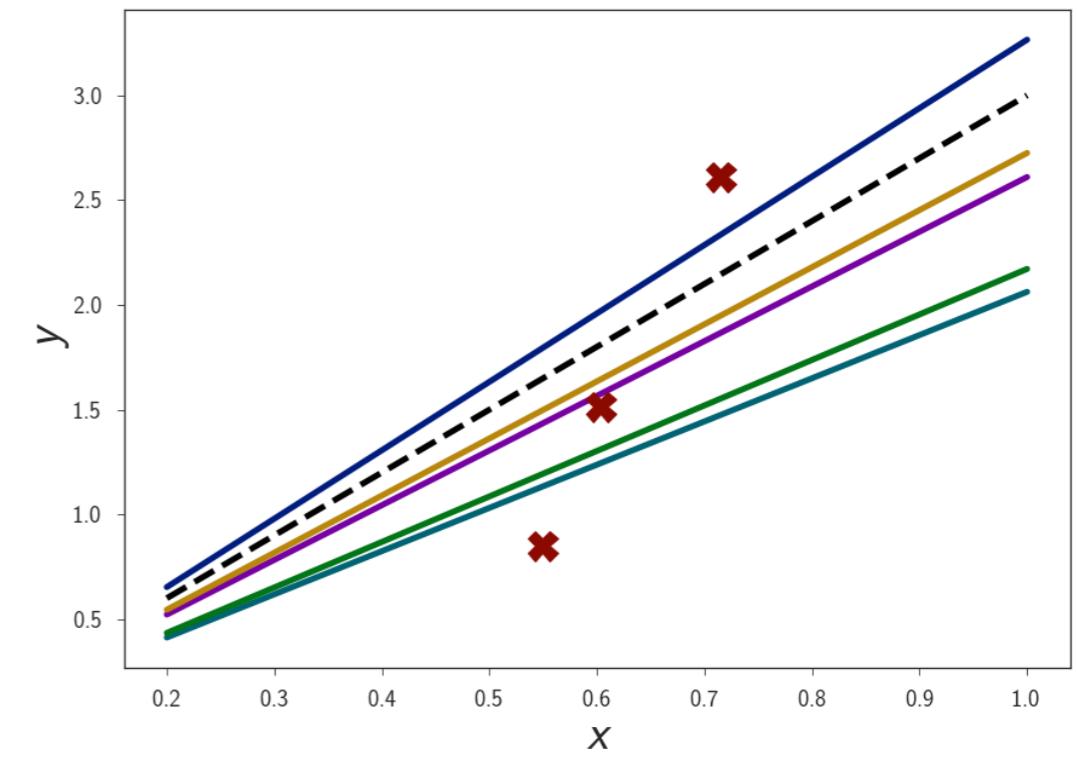
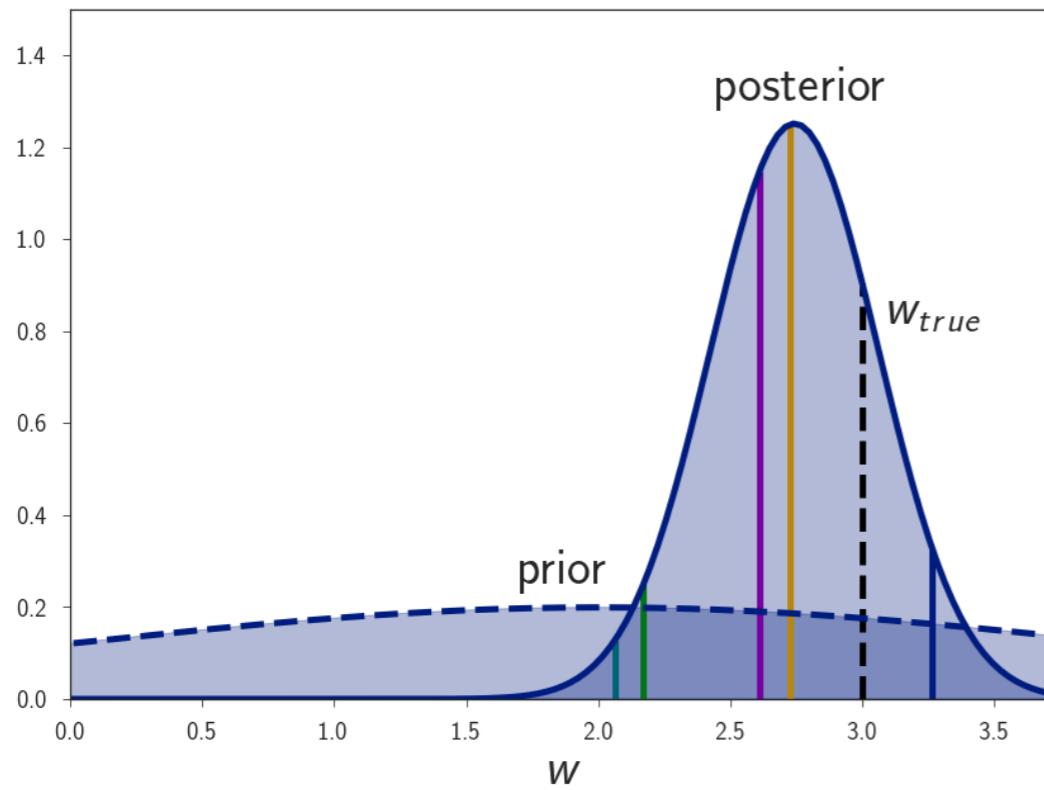
Step 1: introduce a prior distribution  $p(w)$  over parameters



# BAYESIAN MACHINE LEARNING

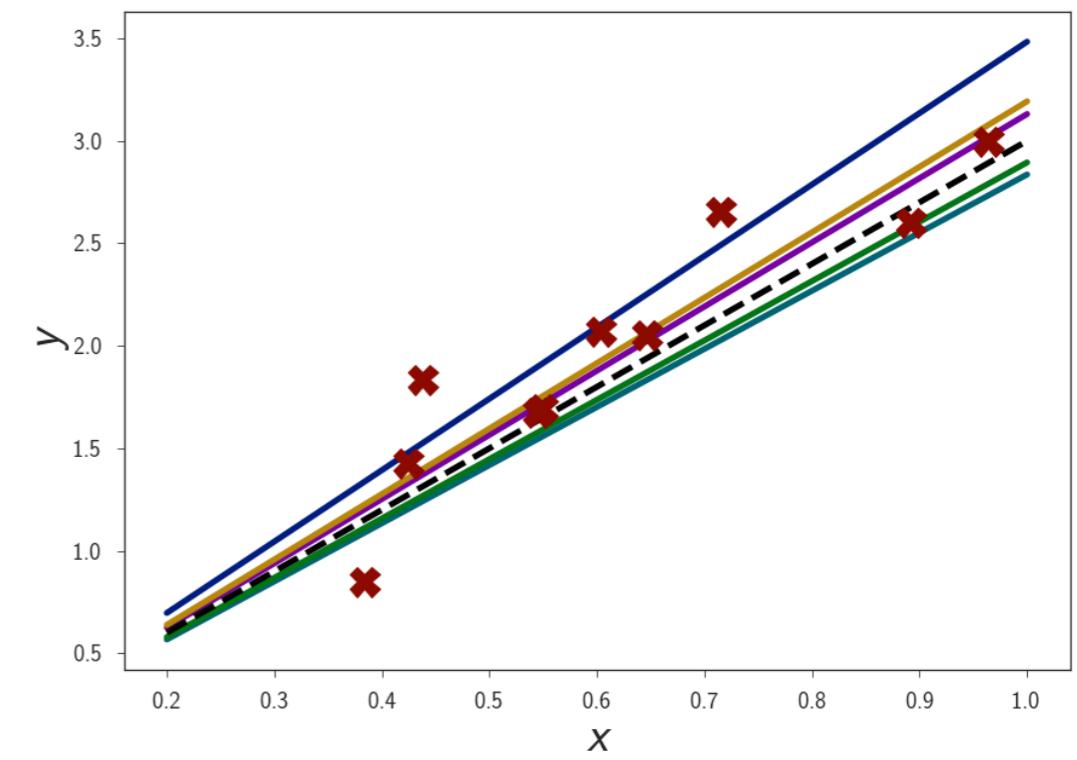
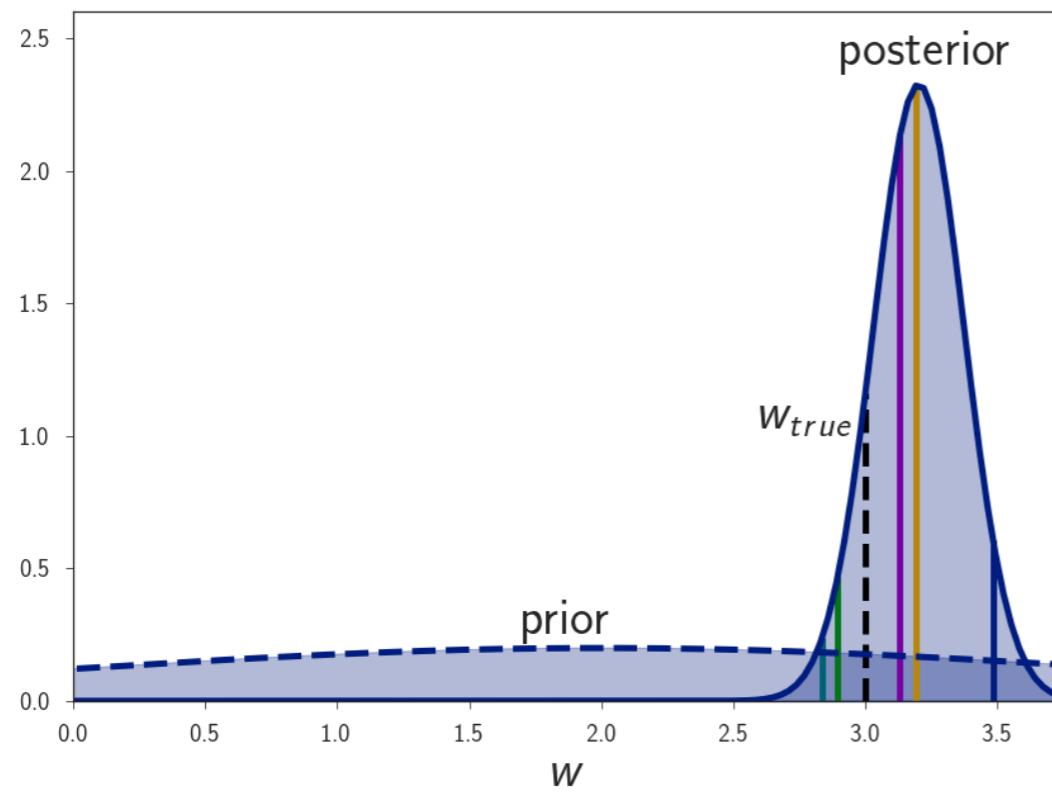
Step 2: Compute posterior  $p(w|D)$  using Bayes rule

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$



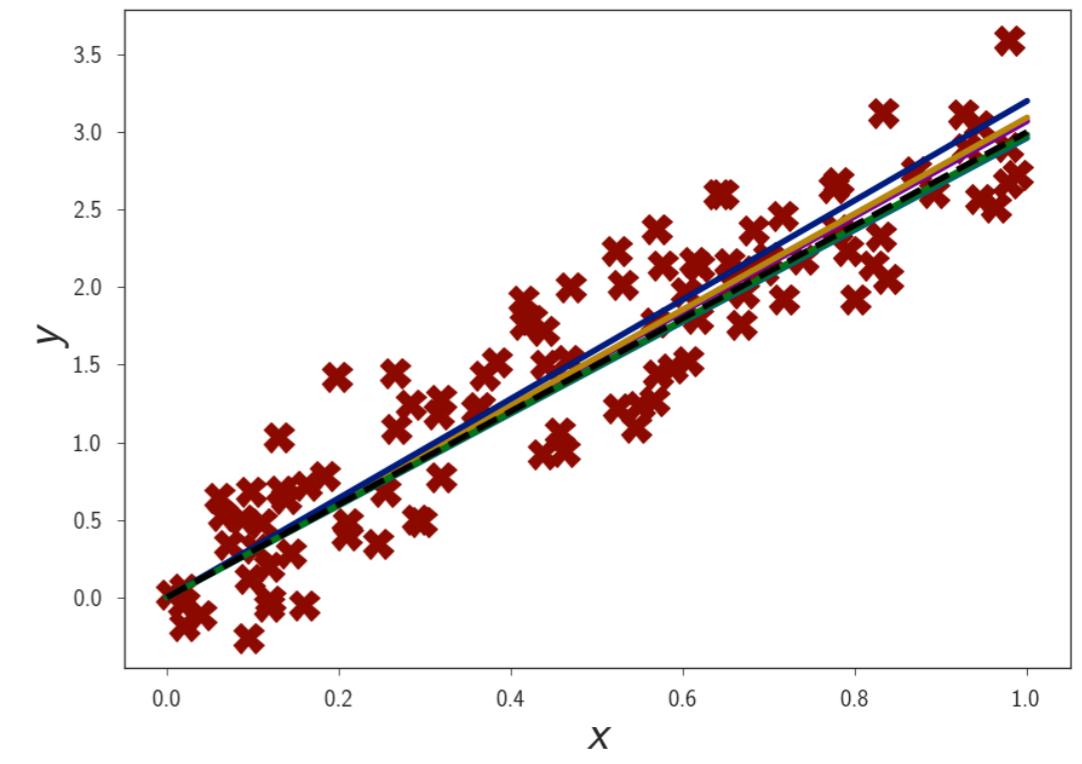
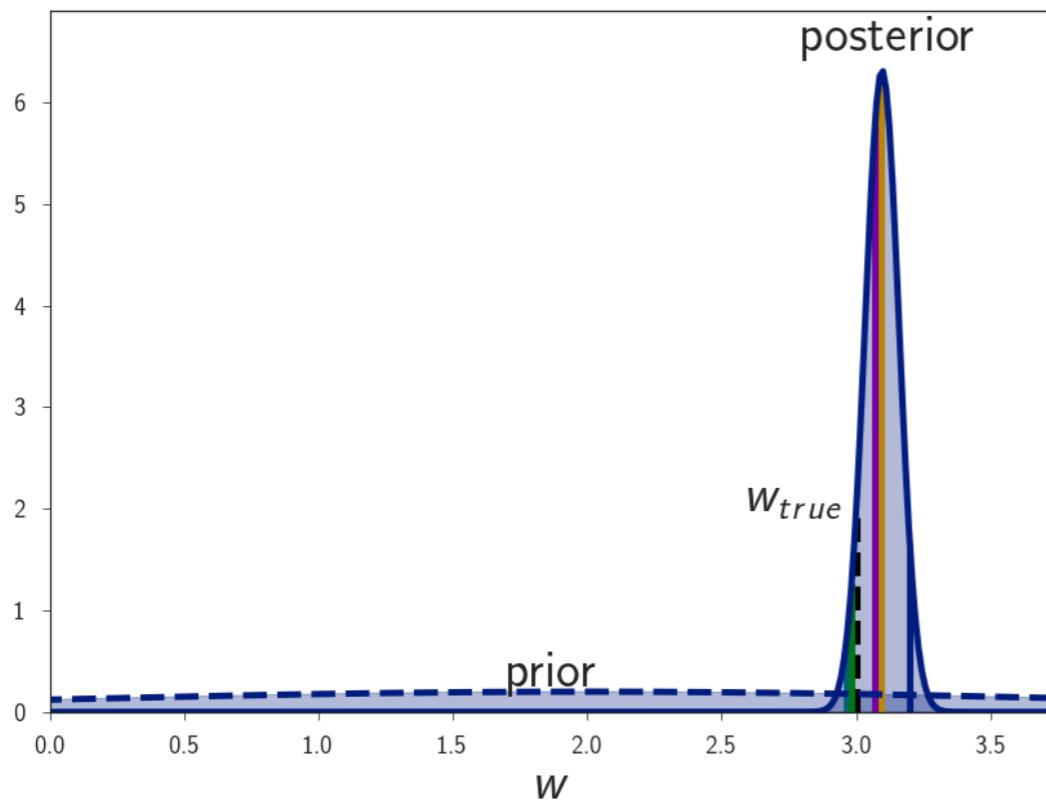
# BAYESIAN MACHINE LEARNING: POSTERIOR CONTRACTION

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$



# BAYESIAN MACHINE LEARNING: POSTERIOR CONTRACTION

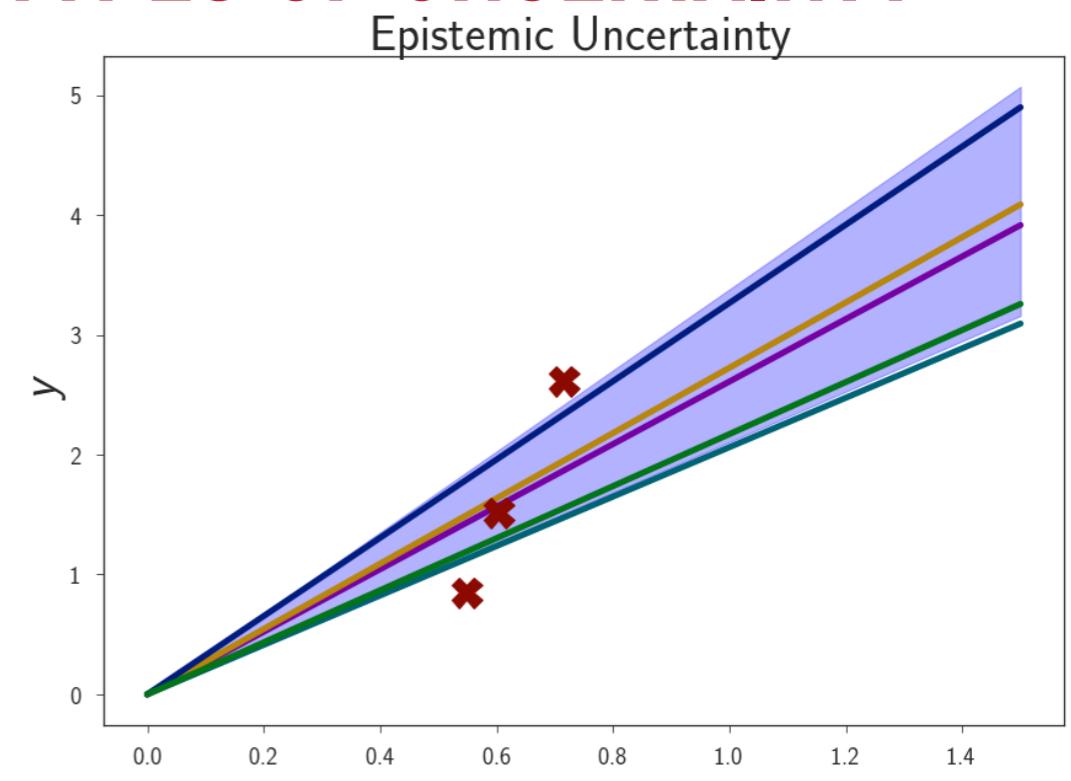
$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$



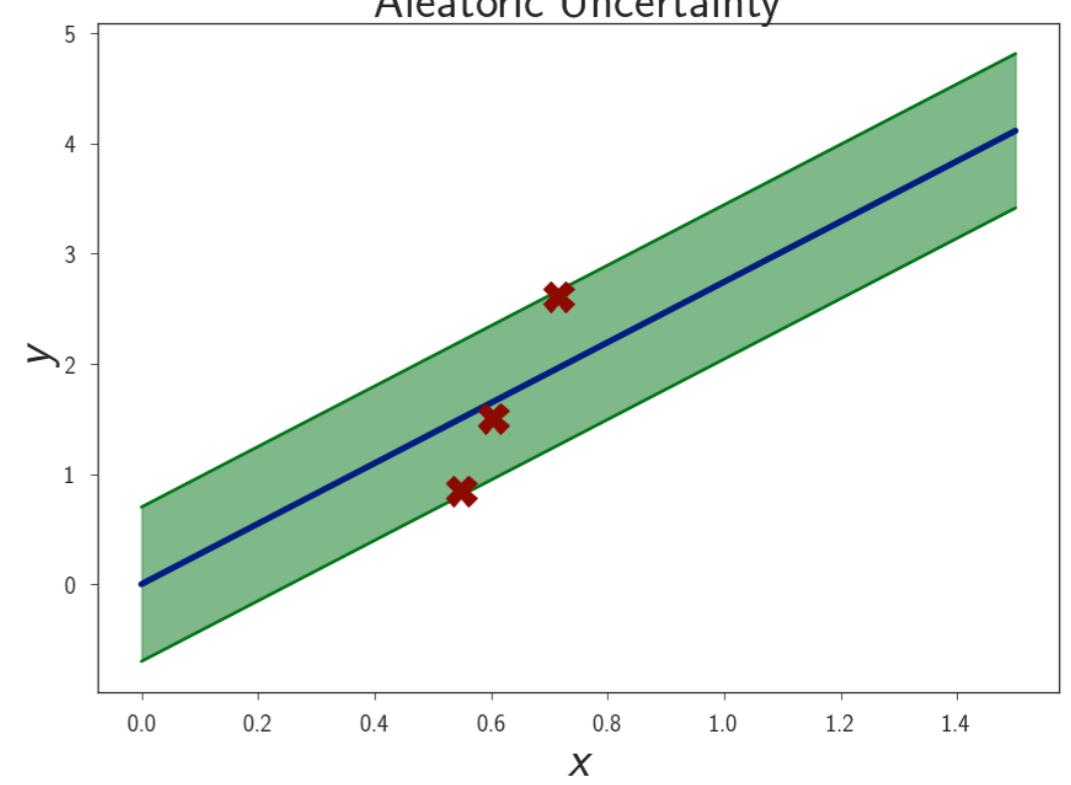
# BAYESIAN MACHINE LEARNING: TWO TYPES OF UNCERTAINTY

Epistemic uncertainty is our uncertainty over the model

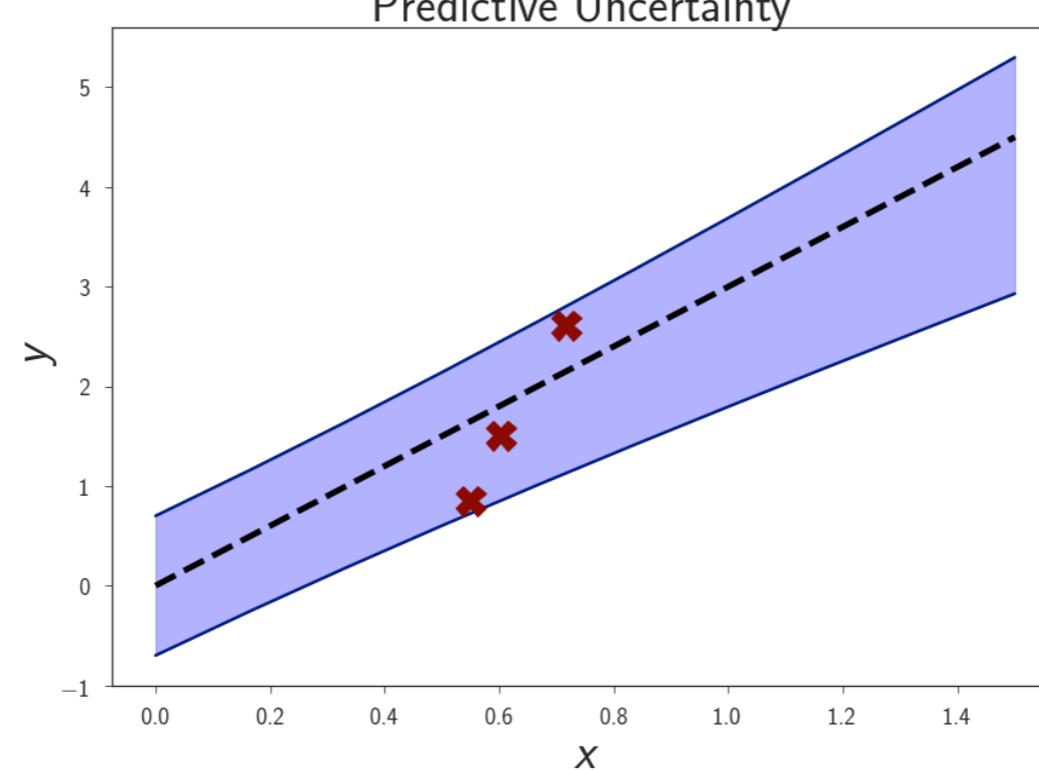
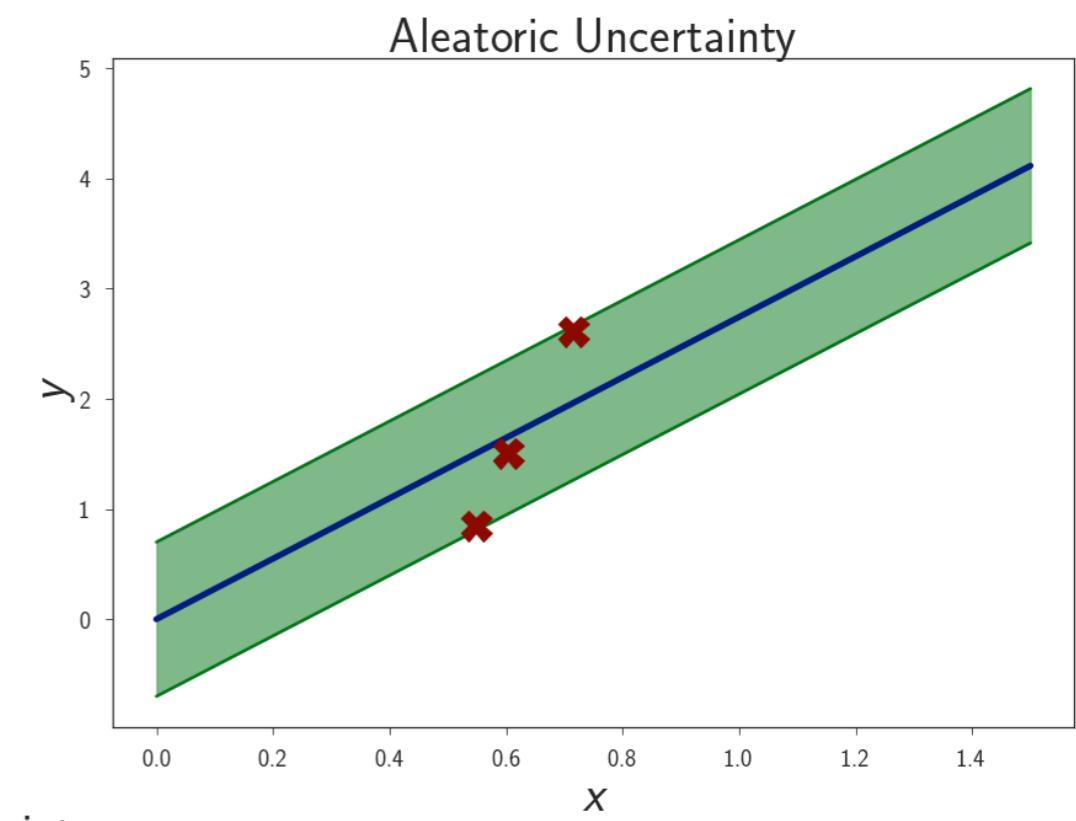
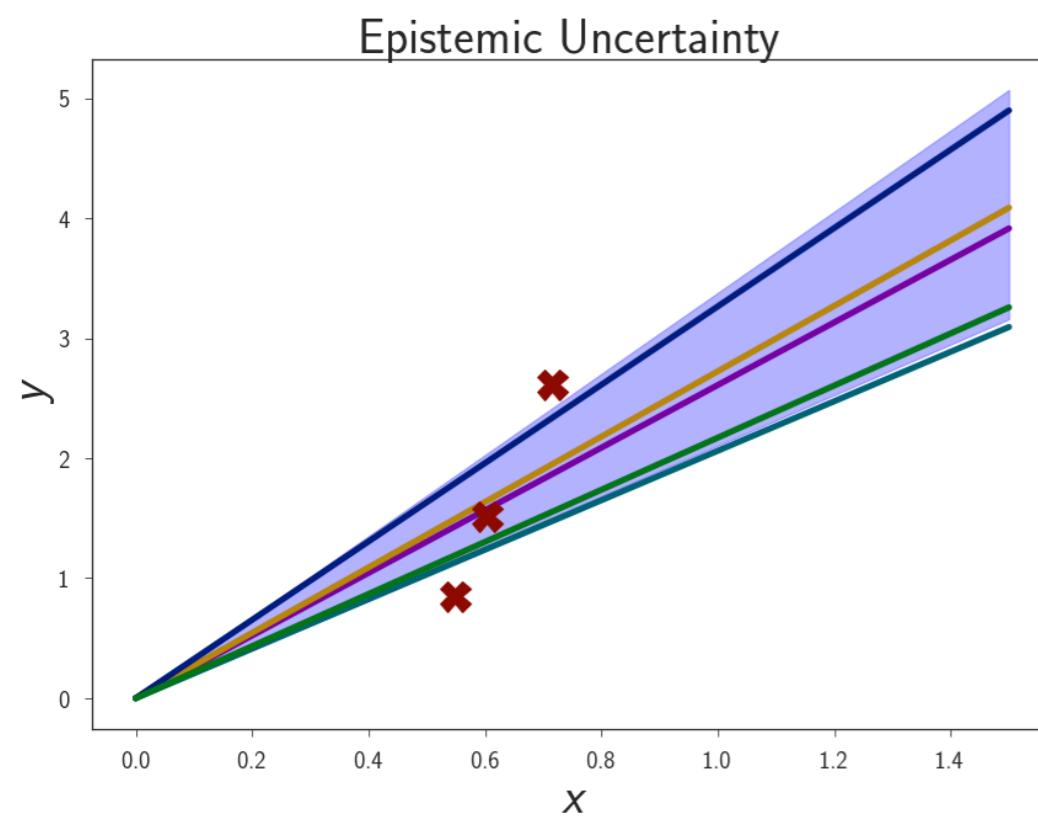
- ▶ Grows with  $x$  because uncertainty in  $w$  is multiplied by  $x$



Aleatoric uncertainty is our uncertainty over the data for a fixed model, e.g. noise.



# BAYESIAN MACHINE LEARNING: BAYESIAN MODEL AVERAGING



## BAYESIAN MACHINE LEARNING: BAYESIAN MODEL AVERAGING

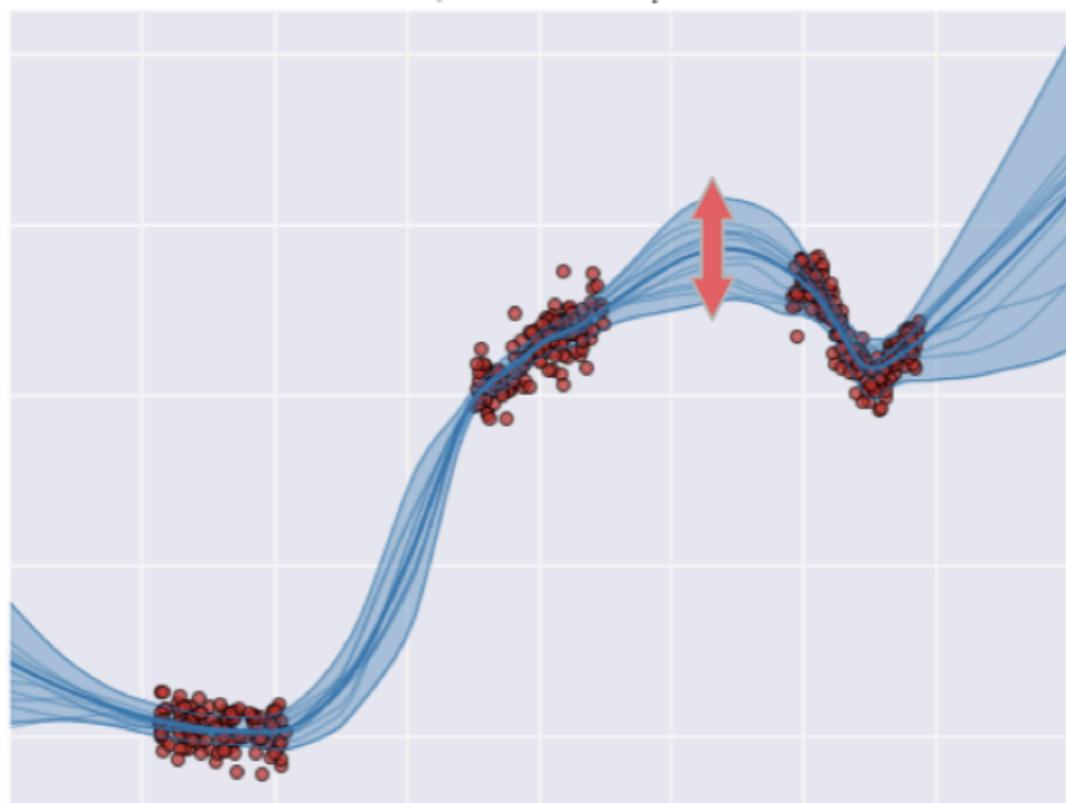
- ▶ We combine aleatoric and epistemic uncertainties via BMA:

$$p(y^*|x^*, D) = \int_w p(y^*|x^*, w)p(w|D)dw$$

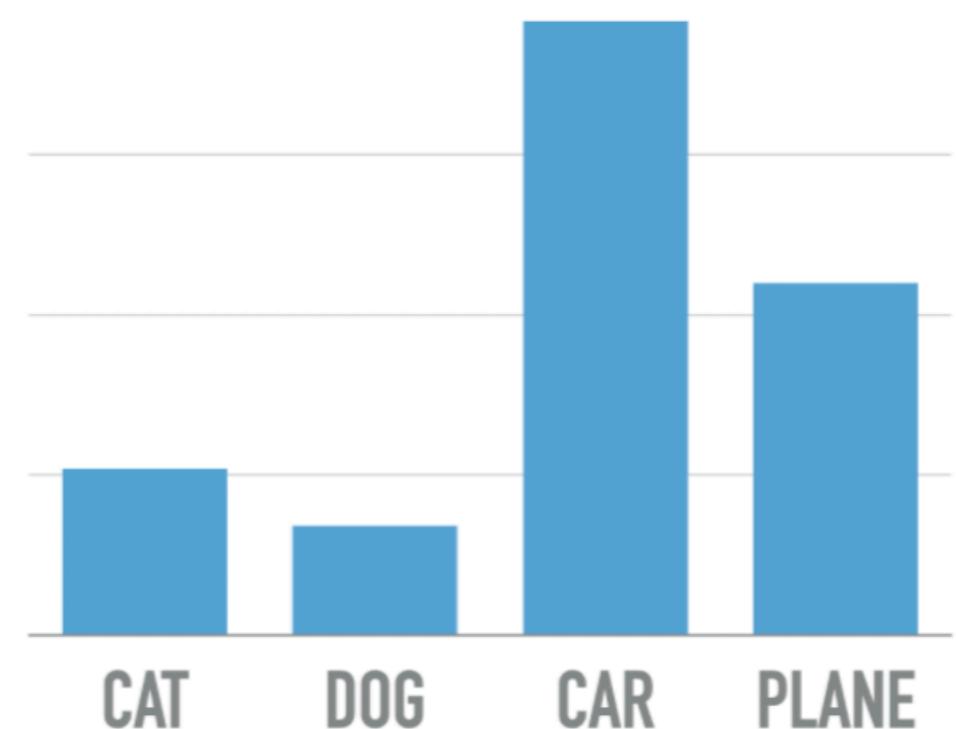
- ▶ Ignoring the uncertainty in the posterior over  $w$  leads to overconfident predictions

# UNCERTAINTY: HOW TO MEASURE?

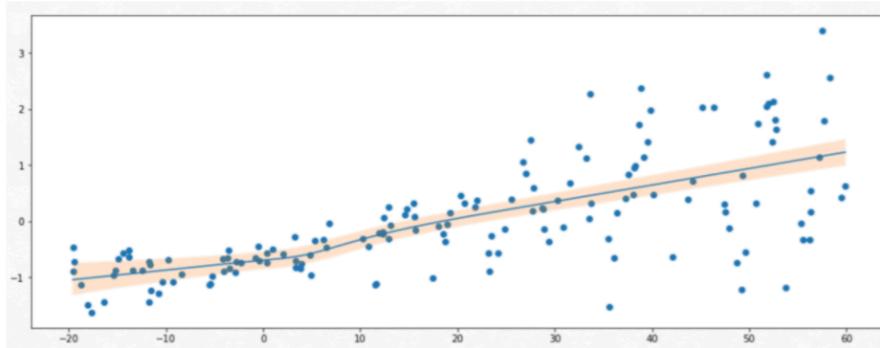
Regression: predictive variance



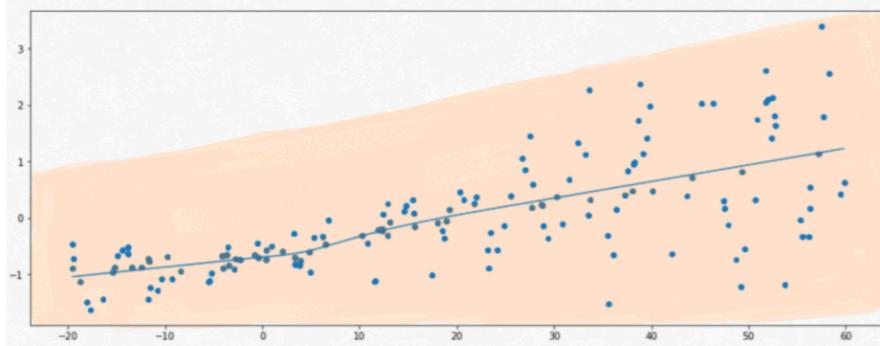
Classification: softmax probabilities



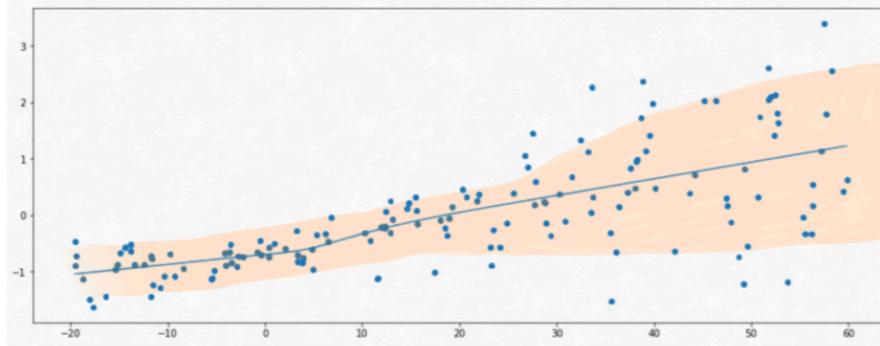
# REGRESSION UNCERTAINTY



**Under-calibrated:** the 95% predictive interval covers **less** than 95% of the data

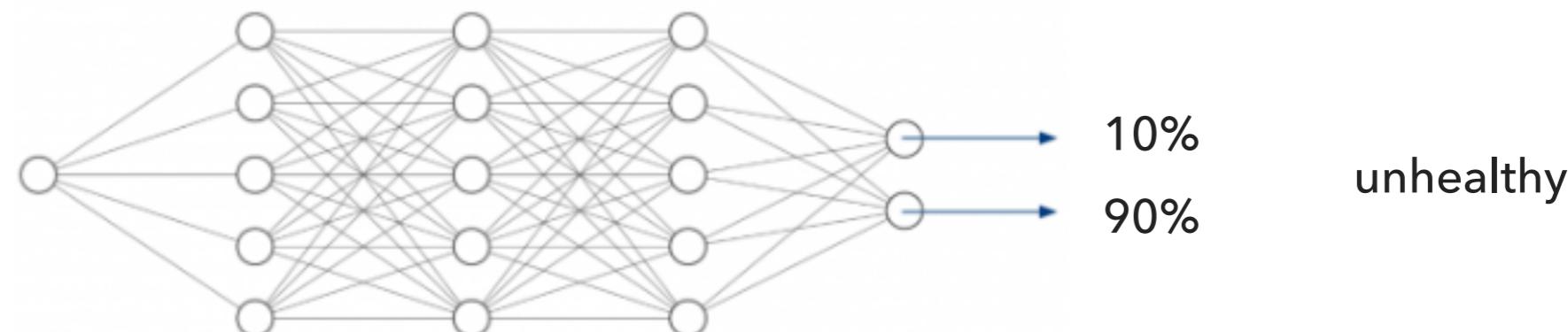
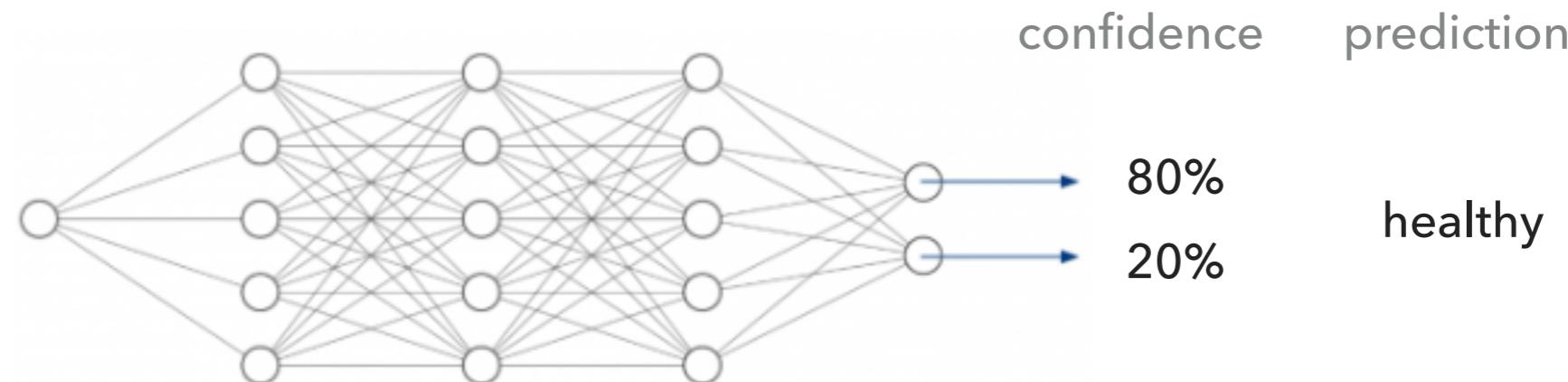
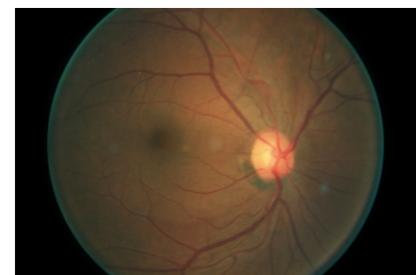


**Over-calibrated:** the 95% predictive interval covers **more** than 95% of the data

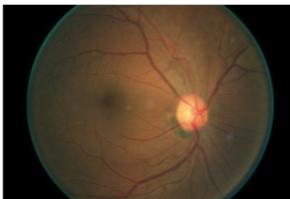
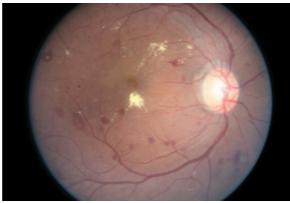


**Well-calibrated:** the 95% predictive interval covers **about** 95% of the data

# CALIBRATION

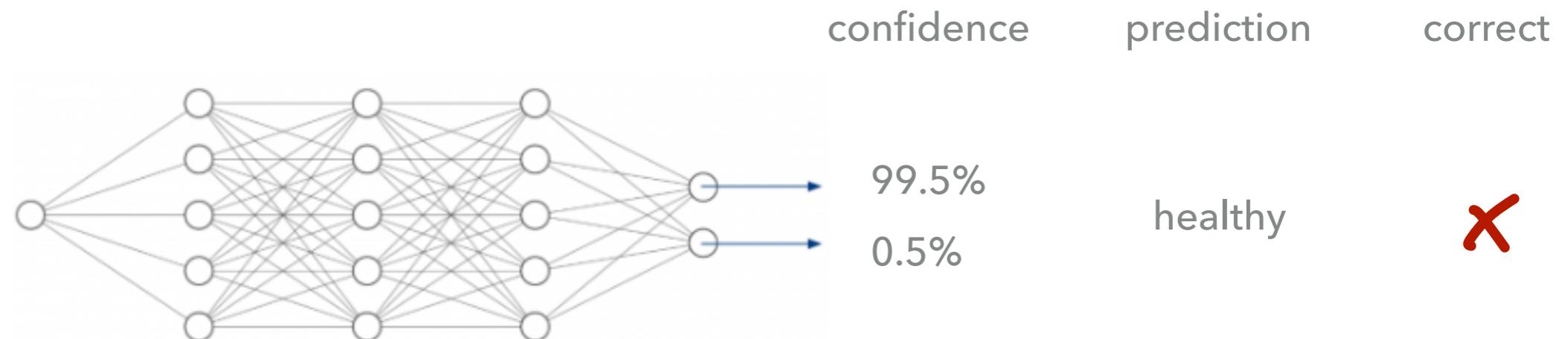


# CALIBRATION

	confidence	prediction	correct
	<input type="radio"/> 80% <input checked="" type="radio"/> 20%	healthy	✓
	<input type="radio"/> 80% <input checked="" type="radio"/> 20%	healthy	✓
	<input type="radio"/> 80% <input checked="" type="radio"/> 20%	healthy	✓
	<input type="radio"/> 80% <input checked="" type="radio"/> 20%	healthy	✗
	<input type="radio"/> 80% <input checked="" type="radio"/> 20%	healthy	✓

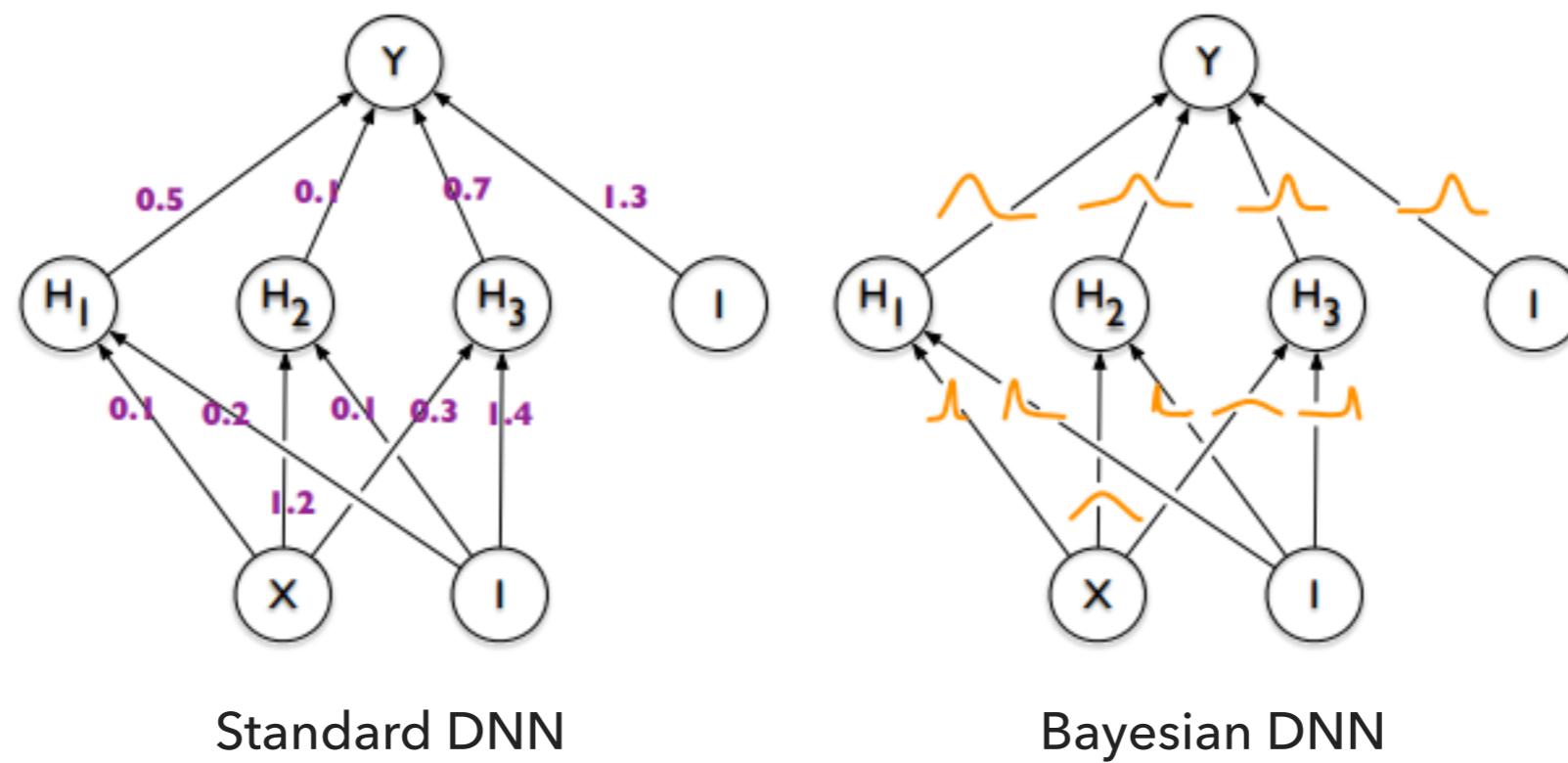
# UNCERTAINTY: OVERCONFIDENCE IN NEURAL NETWORKS

- ▶  $p(y|x)$  should represent probabilities of belonging to a class
- ▶ Neural networks are often over-confident in their predictions



# BAYESIAN DEEP LEARNING

- ▶ In Bayesian deep learning we model posterior distribution over the weights of neural networks
- ▶ In theory, leads to better predictions and well-calibrated uncertainty



# BAYESIAN DEEP LEARNING: CHALLENGES

Bayesian inference for deep neural networks is challenging

- ▶ Posterior is intractable
- ▶ Millions of parameters
- ▶ Large datasets
- ▶ Unclear which priors to use

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)} = \frac{p(D|w)p(w)}{\int_{w'} p(D|w')p(w')dw'}$$

# DEEP LEARNING: BAYESIAN MODEL AVERAGING

- ▶ We cannot compute the exact integral for BMA:

$$p(y^*|x^*, D) = \int_w p(y^*|x^*, w)p(w|D)dw$$

- ▶ Monte Carlo approximation:

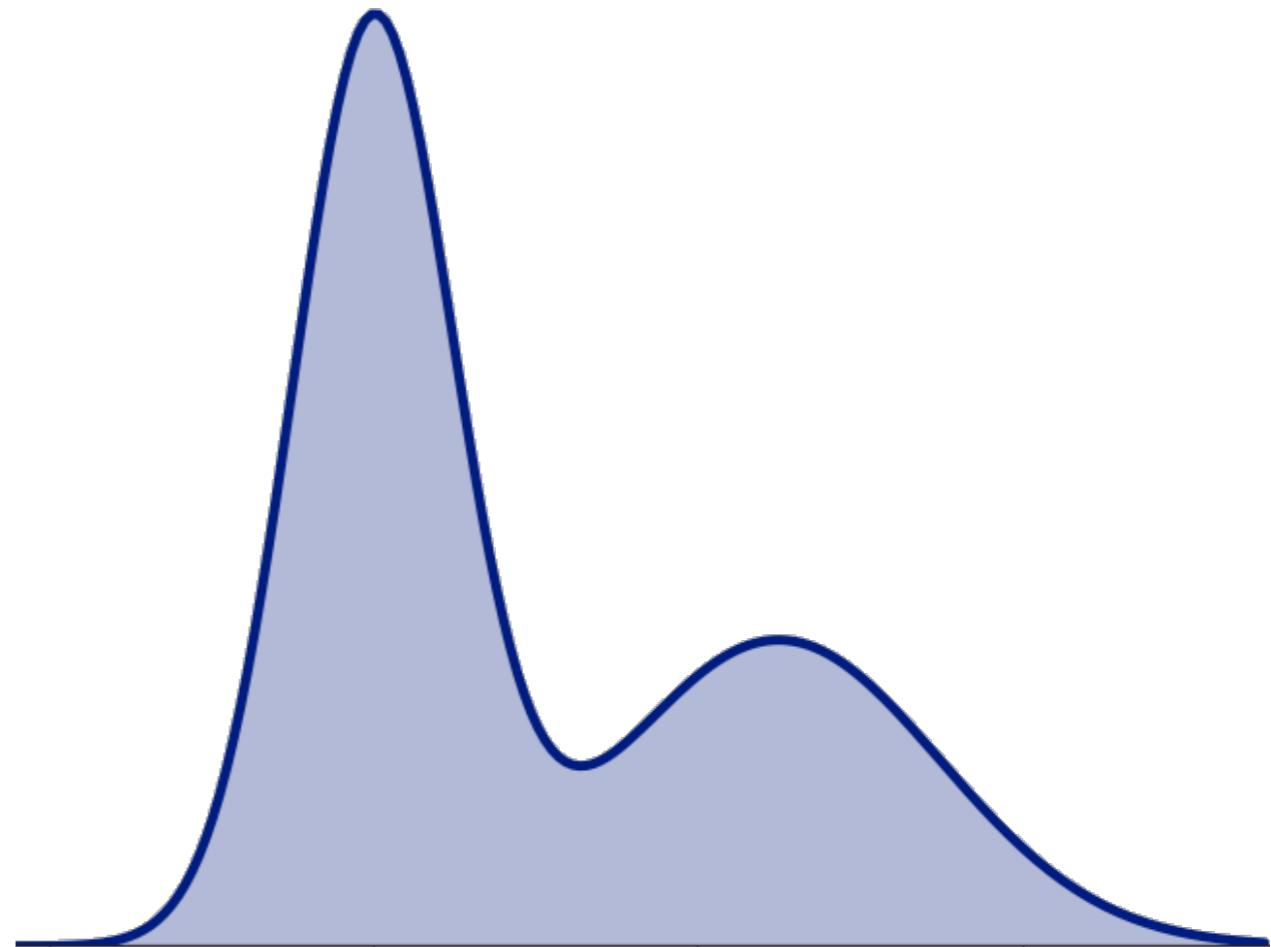
$$p(y^*|x^*, w) \approx \frac{1}{K} \sum_{i=1}^K p(y^*|x^*, \hat{w}) \quad \hat{w} \sim p(w|D)$$

$$p(y^*|x^*, w) \approx \frac{1}{K} \sum_{i=1}^K p(y^*|x^*, \hat{w}) \quad \hat{w} \sim q(w), \quad q(w) \approx p(w|D)$$

# HOW CAN WE DO APPROXIMATE BAYESIAN INFERENCE?

Posterior Approximation:

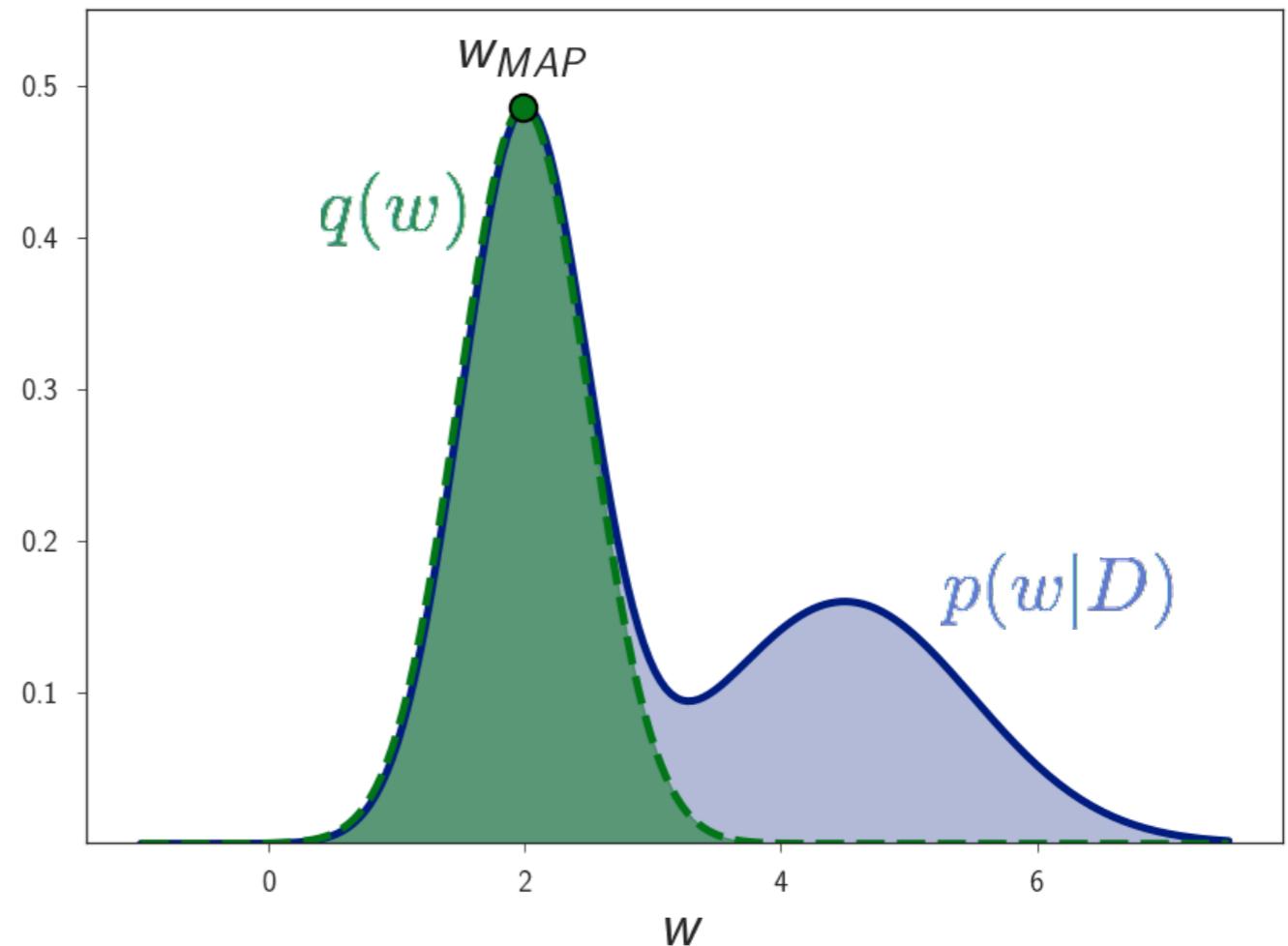
- ▶ Laplace Approximation
- ▶ Variational Inference
- ▶ Markov Chain Monte Carlo
- ▶ **Geometrically Inspired Methods**



## LAPLACE APPROXIMATION

Approximate posterior with a Gaussian  $\mathcal{N}(w|\mu, A^{-1})$

- ▶  $w = w_{MAP}$  mode (local maximum) of  $p(w|D)$
- ▶  $A = -\nabla\nabla \log[p(D|w)p(w)]$
- ▶ Only captures a single mode

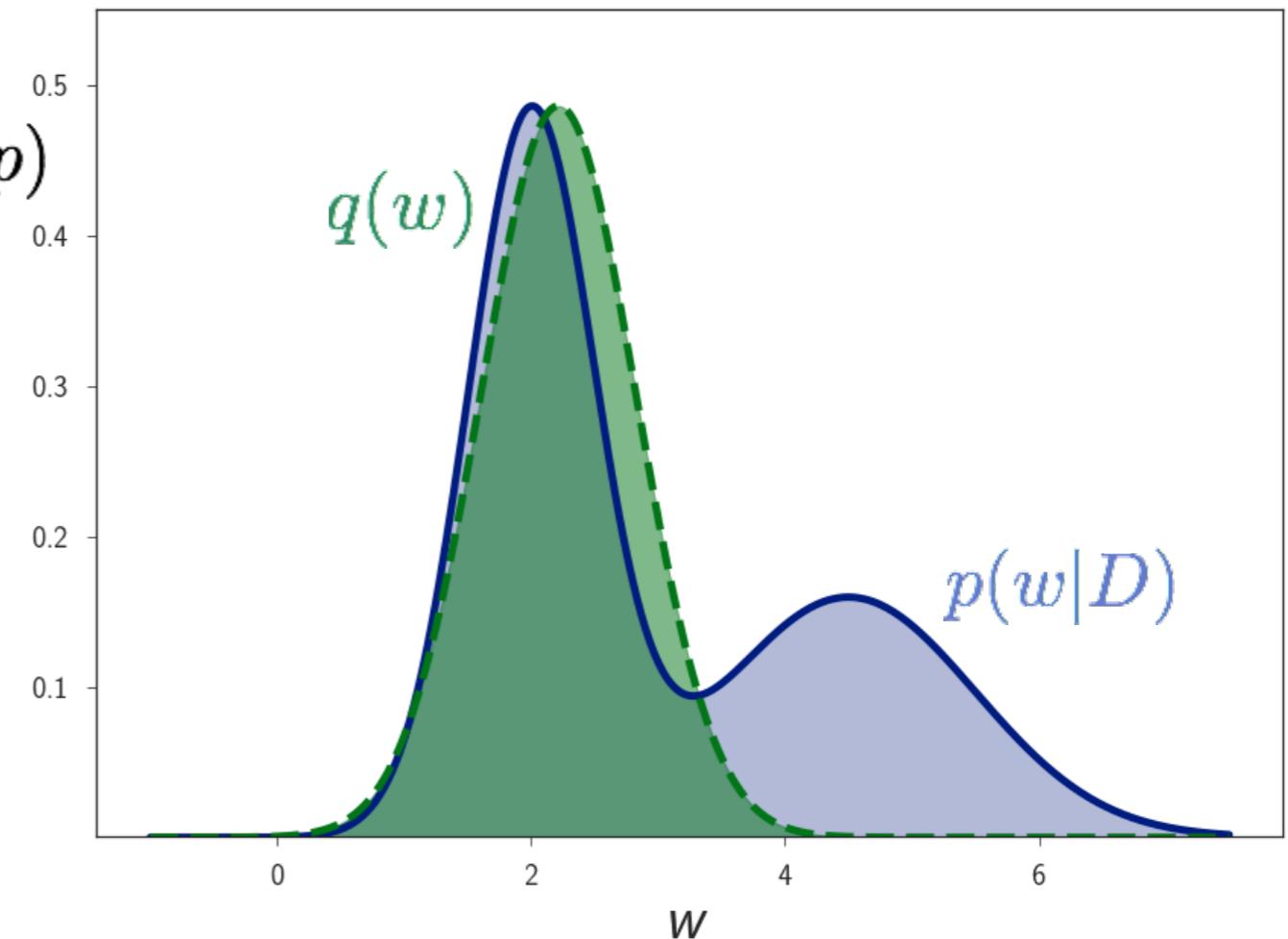
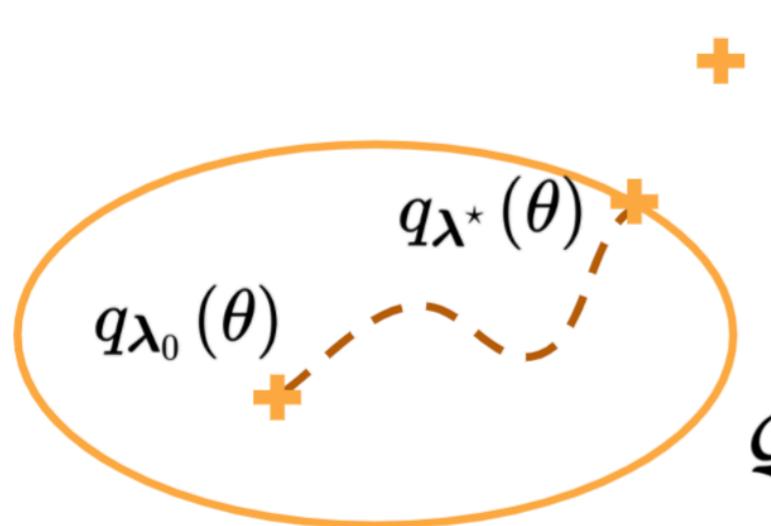


# VARIATIONAL INFERENCE

We can find the best approximating distribution within a given family with respect to KL-divergence

- $KL(q||p) = \int_w q(w) \log \frac{q(w)}{p(w|D)} dw$

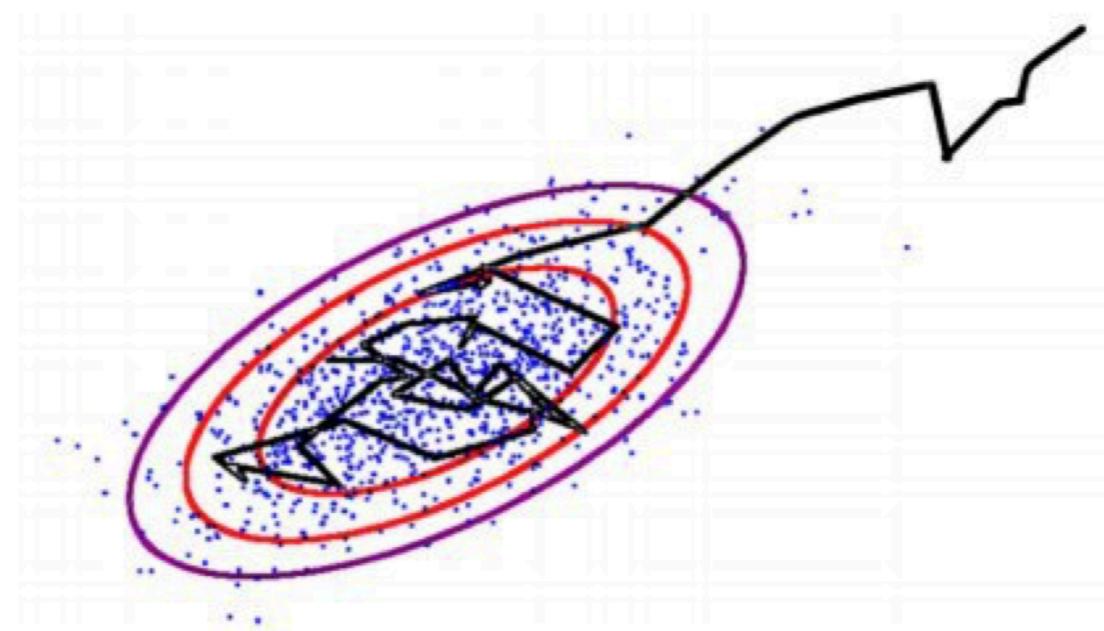
- If  $q = \mathcal{N}(\mu, \Sigma)$ , then  $\min_{\mu, \Sigma} KL(q||p)$



## MARKOV CHAIN MONTE CARLO: METROPOLIS HASTINGS

1. Sample proposal  $\theta'$  from a "proposal" distribution  $\mathcal{N}(\theta' | \theta, \sigma^2)$
2. Accept sample with probability  $\min(1, p^*(\theta')/p^*(\theta))$
3. If rejected, the next sample is the same as the previous  $\theta' = \theta$

$$\theta^{(1)} \rightarrow \theta^{(2)} \rightarrow \dots \rightarrow \theta^{(t-1)} \rightarrow \theta^{(t)}$$

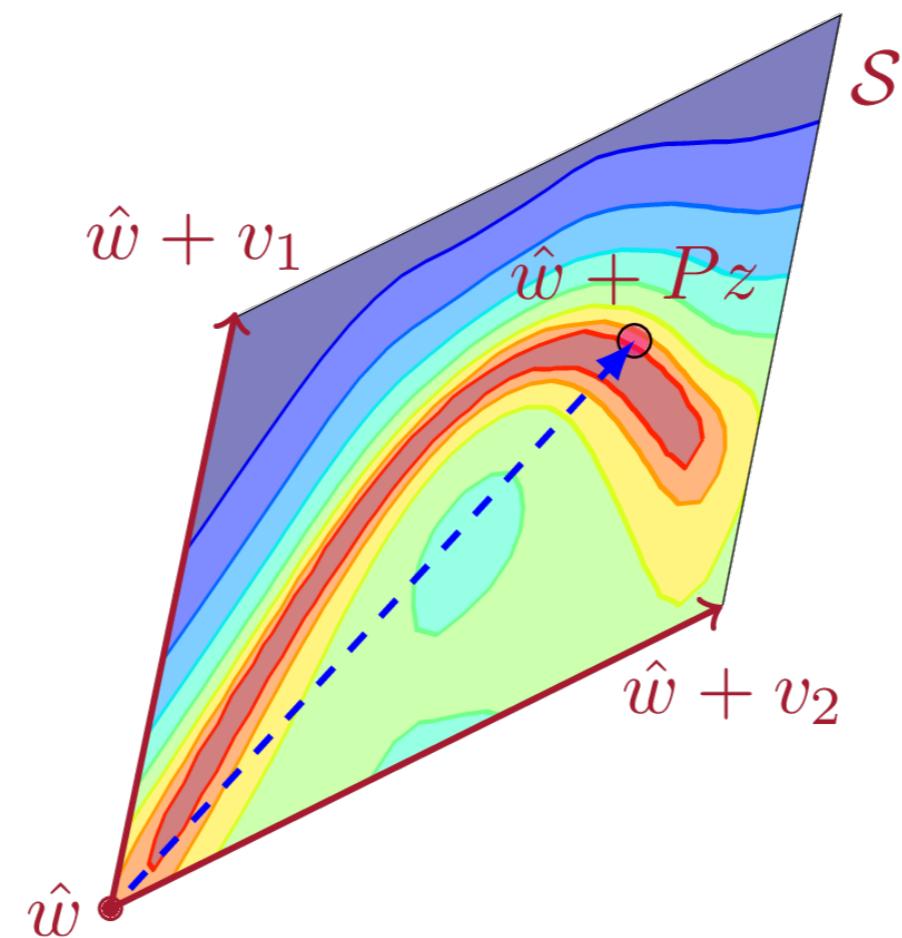


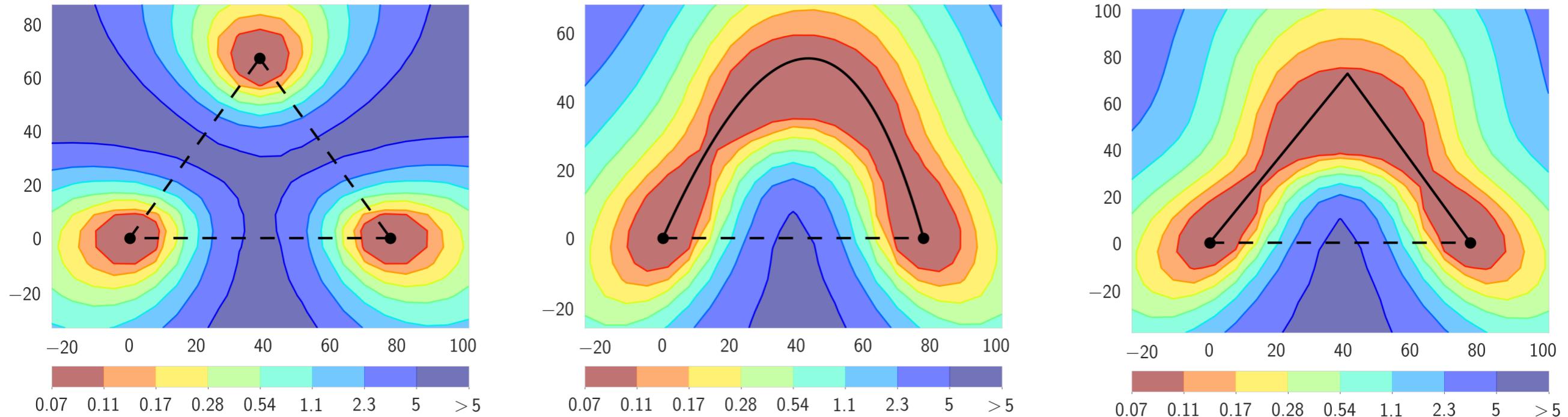
## GEOMETRICALLY INSPIRED APPROXIMATE INFERENCE

- ▶ Learn low-dimensional representations  $z$  for model's weights  $w$
- ▶ Perform inference over  $z$
- ▶ Bayesian Model Averaging by sampling  $z$ , and transforming  $z \rightarrow w$

## SUBSPACE INFERENCE

- Motivation: capture *geometry* of the posterior in *interesting* low-dimensional subspaces of the parameter space



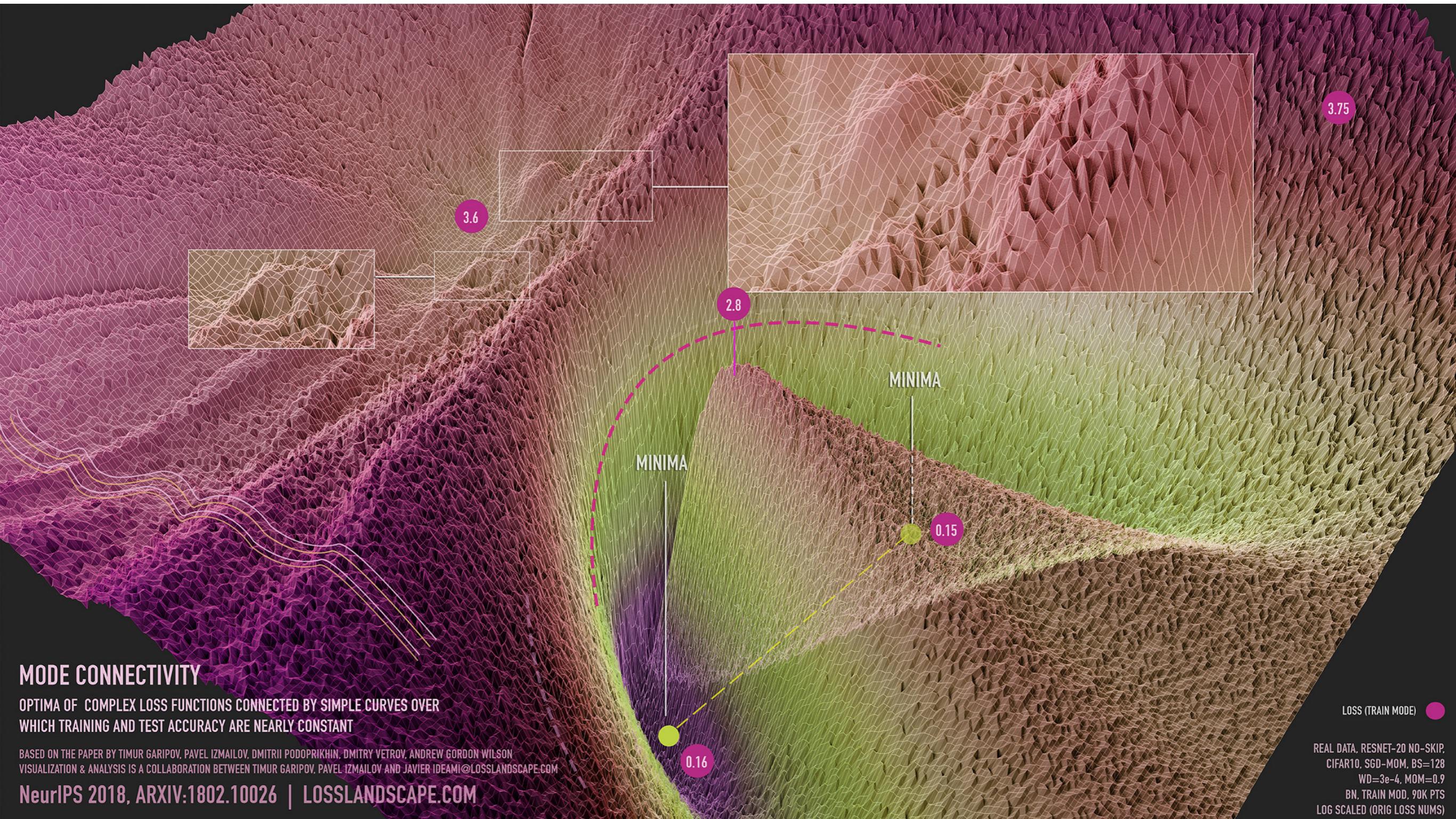


- Weights of pretrained networks:  $\hat{w}_1, \hat{w}_2 \in \mathbb{R}^{|net|}$
- Define parametric curve:  $\phi_\theta(\cdot) : [0, 1] \rightarrow \mathbb{R}^{|net|}$   

$$\phi_\theta(0) = \hat{w}_1, \quad \phi_\theta(1) = \hat{w}_2$$
- DNN loss function:  $\mathcal{L}(w)$
- Minimize averaged loss w.r.t.  $\theta$

$$\underset{\theta}{\text{minimize}} \quad \ell(\theta) = \int_0^1 \mathcal{L}(\phi_\theta(t)) dt = \mathbb{E}_{t \sim U(0,1)} \mathcal{L}(\phi_\theta(t))$$

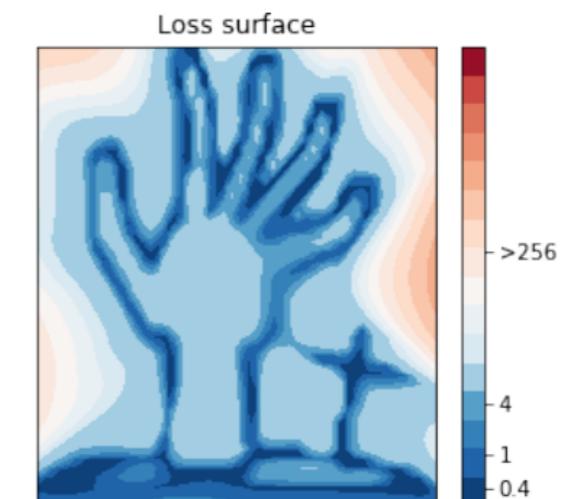
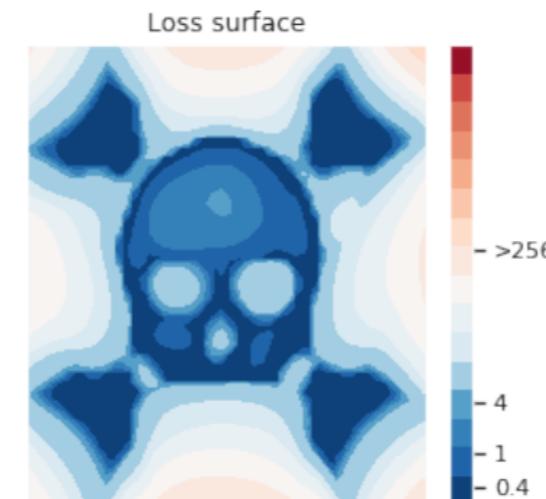
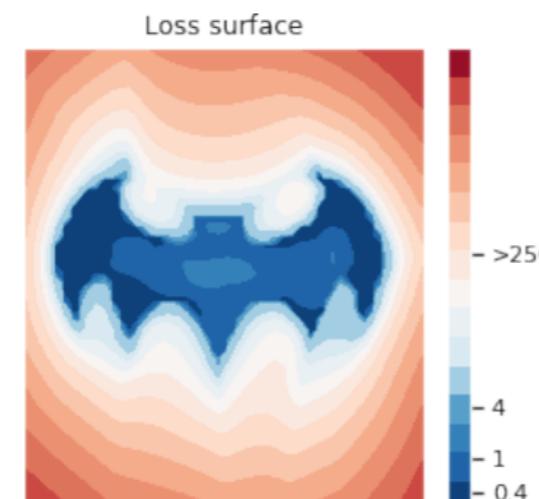
"Loss Surfaces, Mode Connectivity, and Fast Ensembling  
of DNNs" by Timur Garipov, Pavel Izmailov, Dmitrii  
Podoprikhin, Dmitry P Vetrov, Andrew Gordon Wilson



# LOSS SURFACES

Loss Surfaces of Neural Networks are extremely complex:

- ▶ Live in million-dimensional parameter spaces
- ▶ Highly Non-convex
- ▶ Very Multimodal



# SUBSPACE INFERENCE

A modular approach:

- ▶ Design subspace
- ▶ Approximate posterior over parameters in the subspace
- ▶ Sample from approximate posterior for Bayesian model averaging

# SUBSPACE INFERENCE

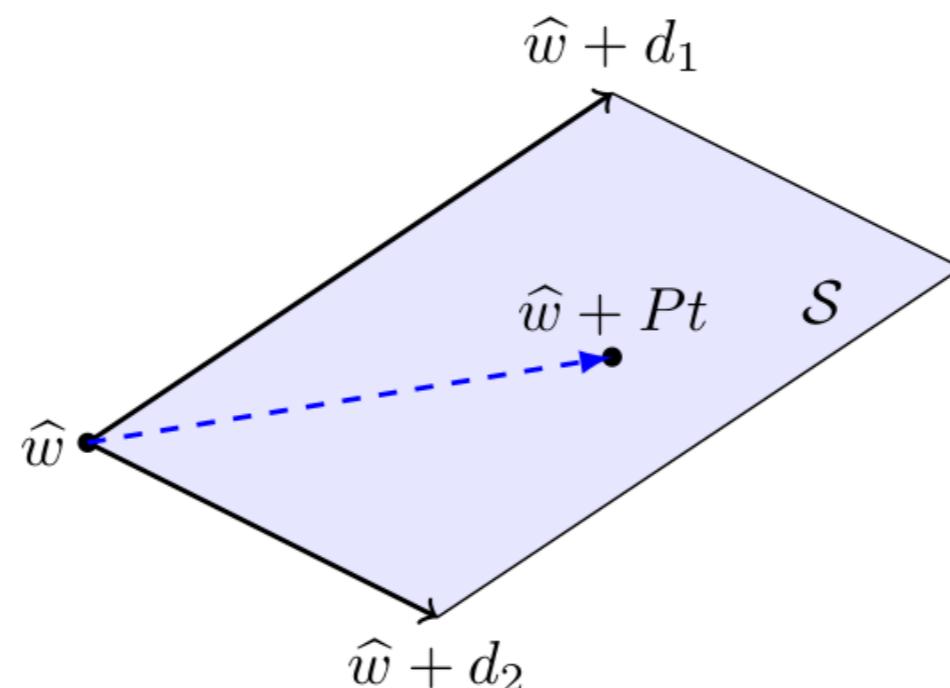
A modular approach:

- ▶ Design subspace
- ▶ Approximate posterior over parameters in the subspace
- ▶ Sample from approximate posterior for Bayesian model averaging

**We can approximate posterior of 36 million dimensional WideResNet in 5D subspace and get state-of-the-art results!**

## SUBSPACE

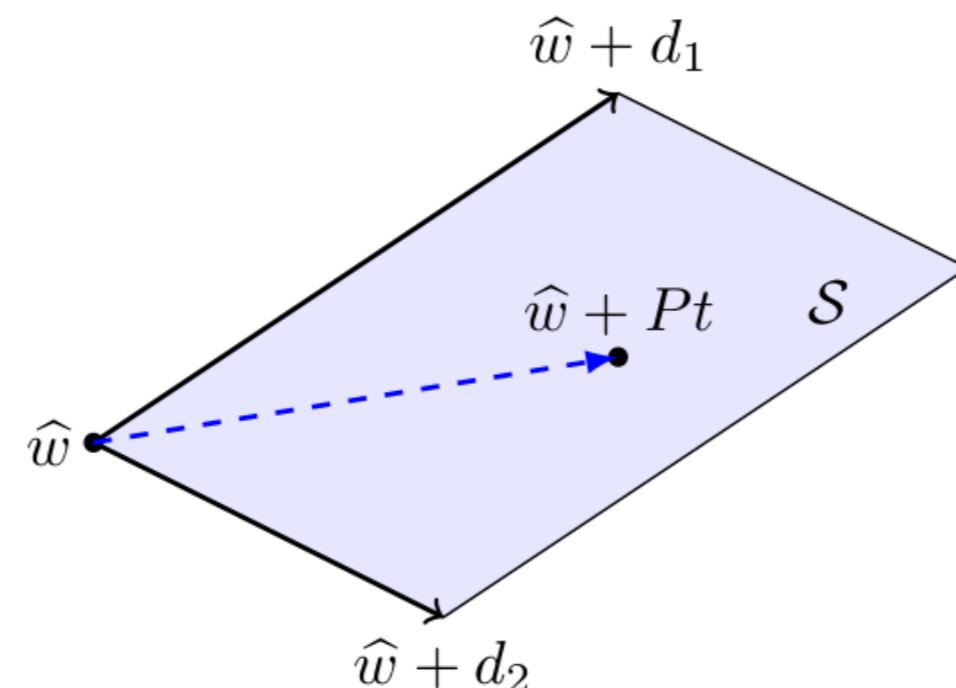
- ▶ Choose shift  $\hat{w}$  and basis vectors  $\{d_1, \dots, d_K\}$
- ▶ Define subspace  $S = \{w \mid w = \underbrace{\hat{w} + t_1 d_1 + \dots + t_k d_K}_{Pt}\}$
- ▶ Likelihood  $p(D \mid t) = p_M(D \mid w = \hat{w} + Pt)$ .



# INFERENCE

- ▶ Approximate inference over parameters  $t$
- ▶ Bayesian model averaging at test time:

$$p(D^* | D) = \frac{1}{J} \sum_{i=1}^J p_M(D^* | \tilde{w} = \hat{w} + P\tilde{t}_i), \quad \tilde{t}_i \sim q(t | D)$$



# INFERENCE

In the subspace, we are able to apply inference methods which struggle in the full parameter space.

- ▶ Variational Inference
  - ▶ Mean Field Gaussian family
  - ▶ RealNVP normalizing flow family
- ▶ Monte Carlo
  - ▶ No-U-Turn Sampler
  - ▶ Elliptical Slice Sampling

## TEMPERING POSTERIOR

- ▶ In the subspace model # parameters << # data points
  - ▶ ~5-10 parameters, ~50K data points
- ▶ Posterior over  $t$  is extremely concentrated
- ▶ To address this issue, we utilize the tempered posterior:

$$p_T(t|D) \propto \underbrace{p(D|t)^{1/T}}_{\text{likelihood}} \underbrace{p(t)}_{\text{prior}}$$

- ▶  $T$  can be learned by cross-validation
- ▶ Heuristic:  $T = \frac{\# \text{ data points}}{\# \text{ parameters}}$

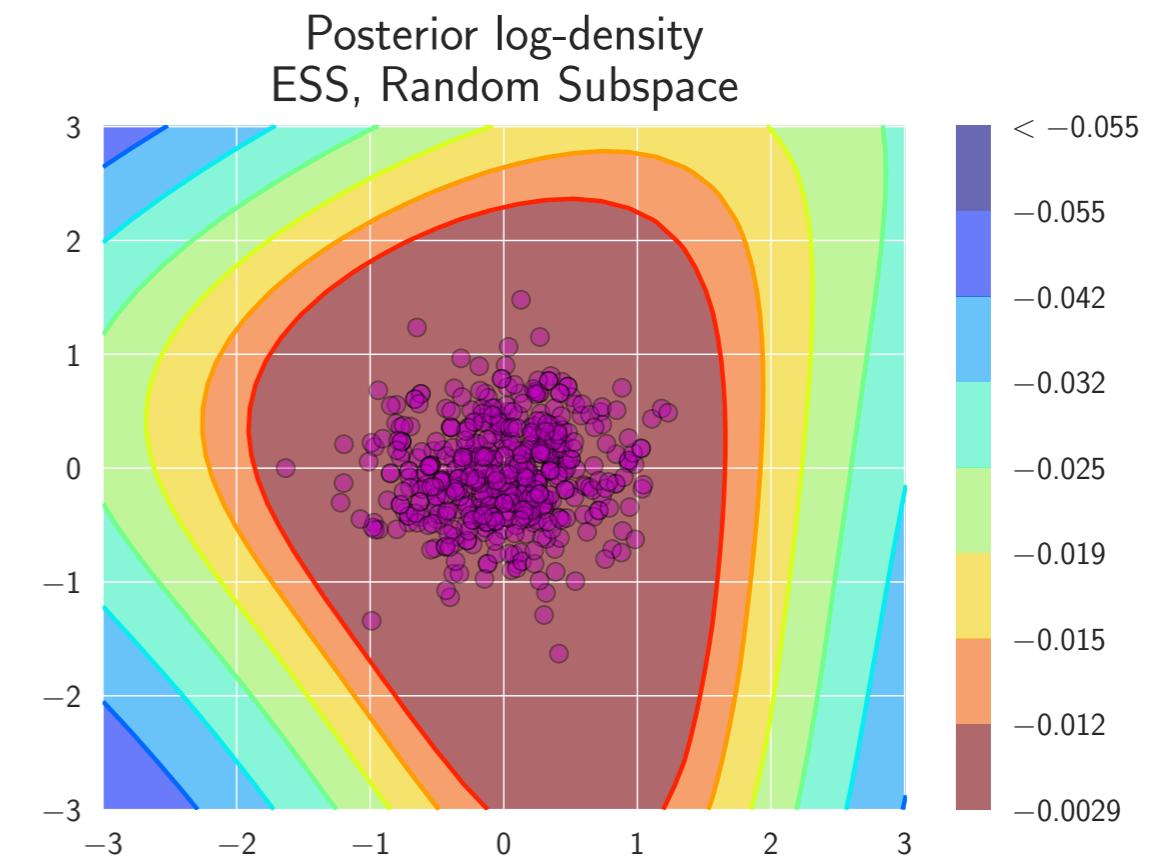
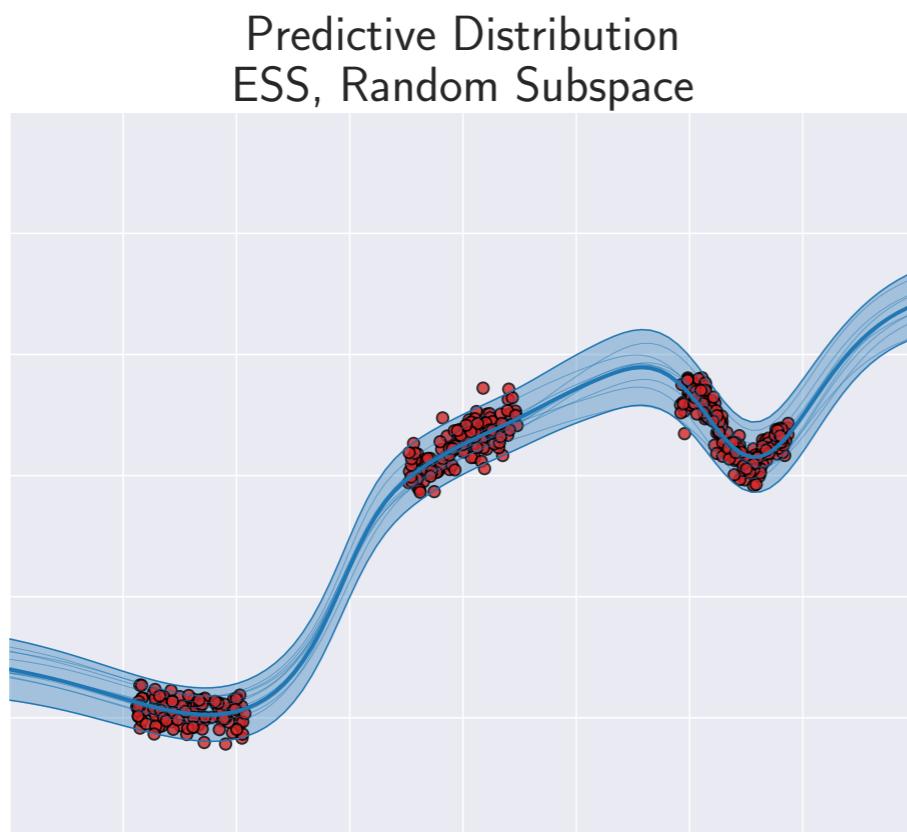
## SUBSPACE CHOICE

We want a subspace that

- ▶ Contains **diverse** models
- ▶ **Cheap** to construct

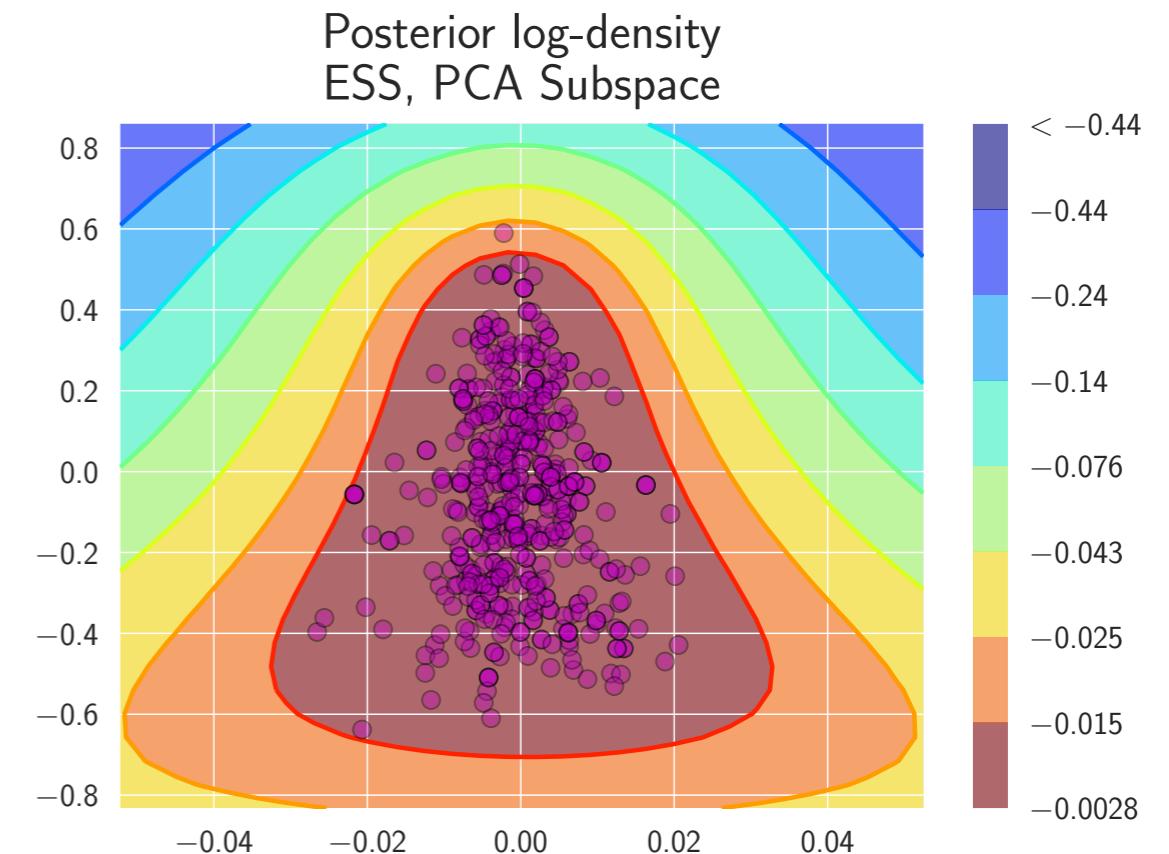
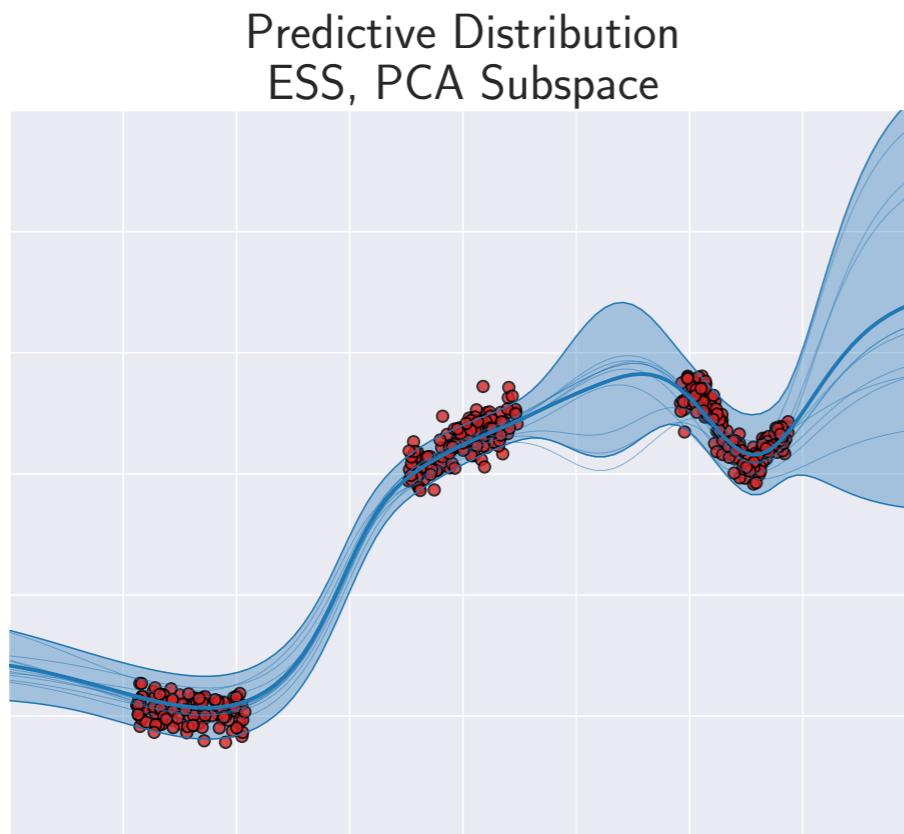
## RANDOM SUBSPACE

- ▶ Directions  $d_1, \dots, d_K \sim N(0, I_p)$
- ▶ Use pre-trained solution as shift  $\hat{w}$
- ▶ Subspace  $S = \{w \mid w = \hat{w} + Pt\}$



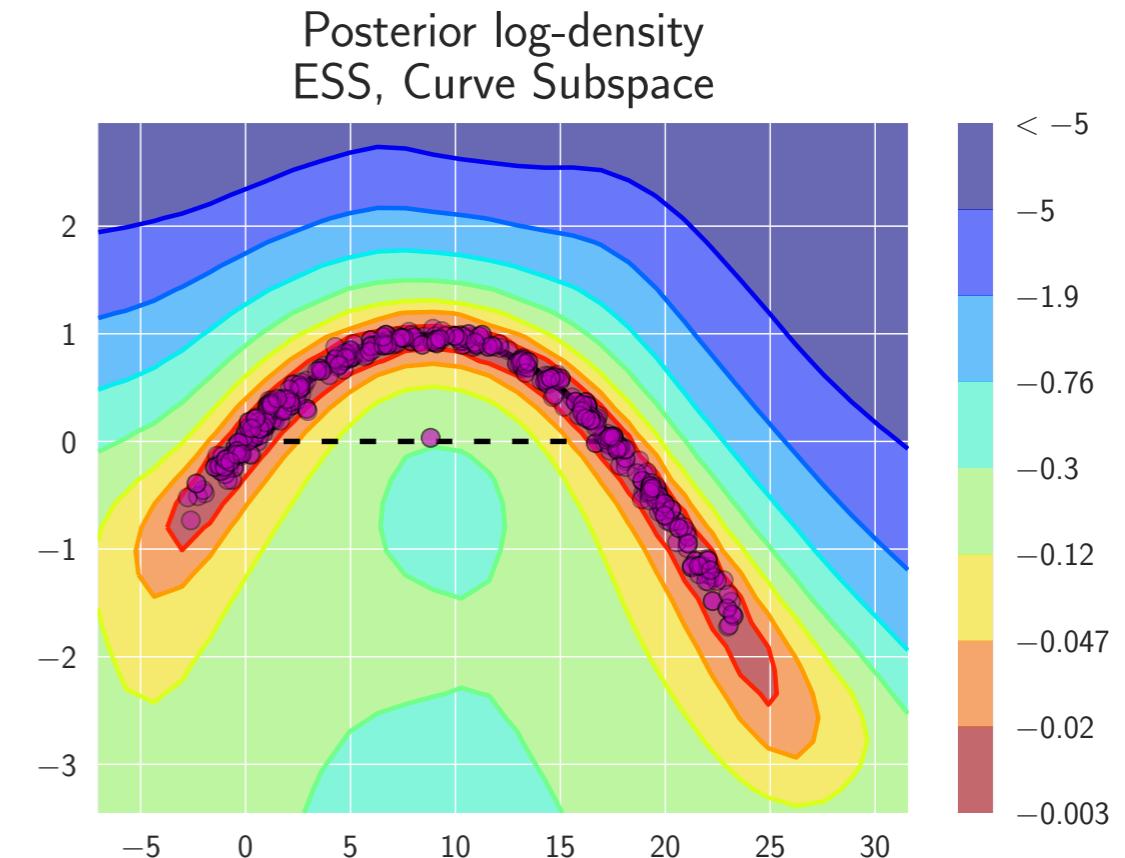
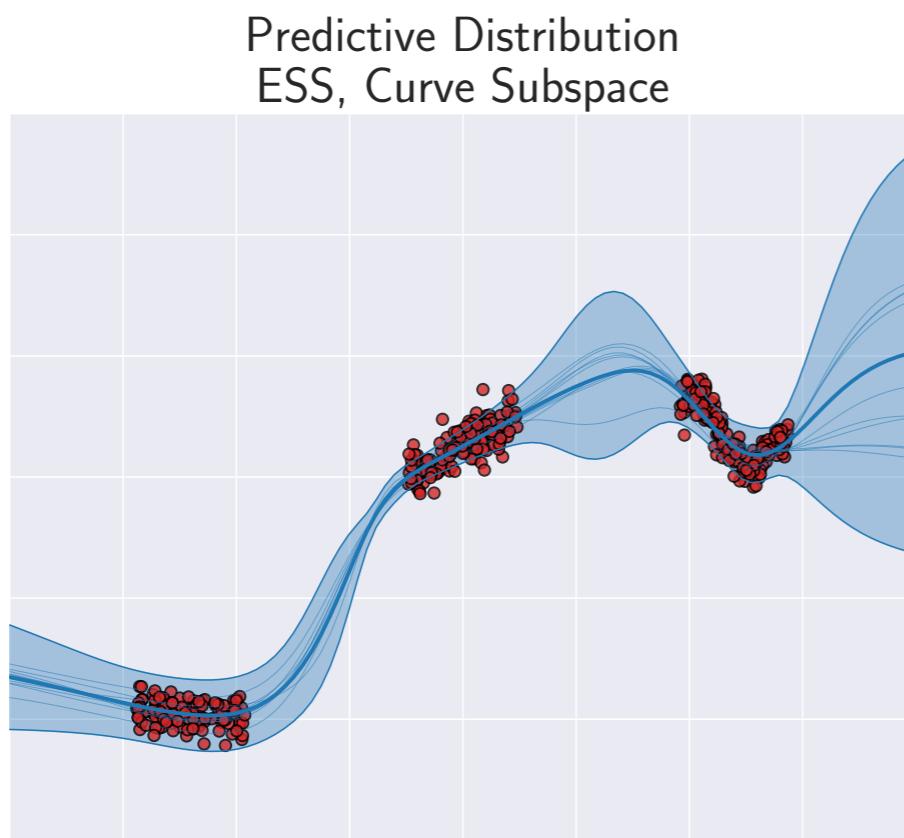
## PCA OF THE SGD TRAJECTORY

- ▶ Run SGD with high constant learning rate from a pre-trained solution
- ▶ Collect snapshots of weights  $w_i$
- ▶ Use SWA solution as shift  $\hat{w} = \frac{1}{T} \sum_i w_i$  [Izmailov et al, 2018]
- ▶  $\{d_1, \dots, d_K\}$  – first  $K$  PCA components of vectors  $\hat{w} - w_i$

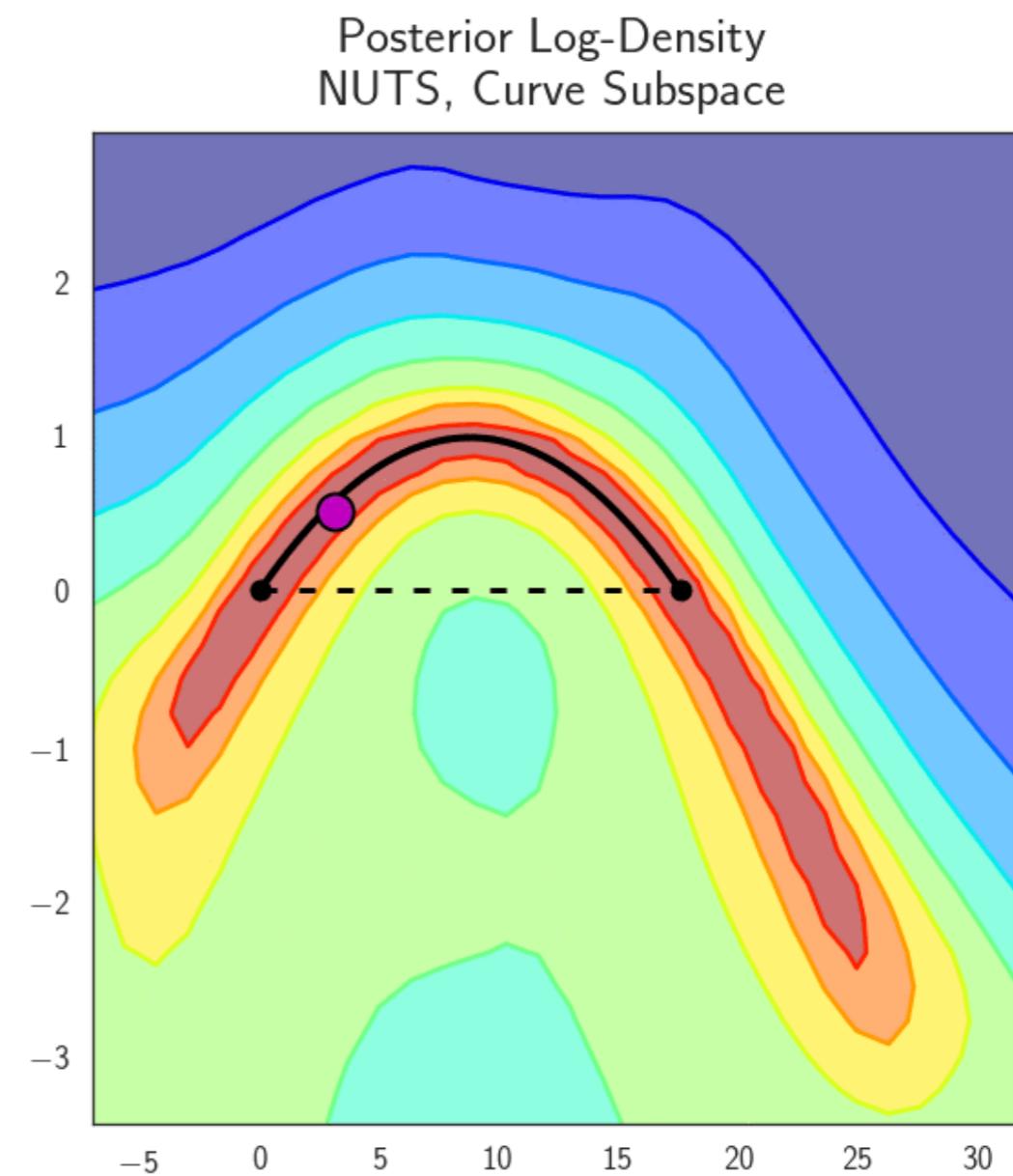
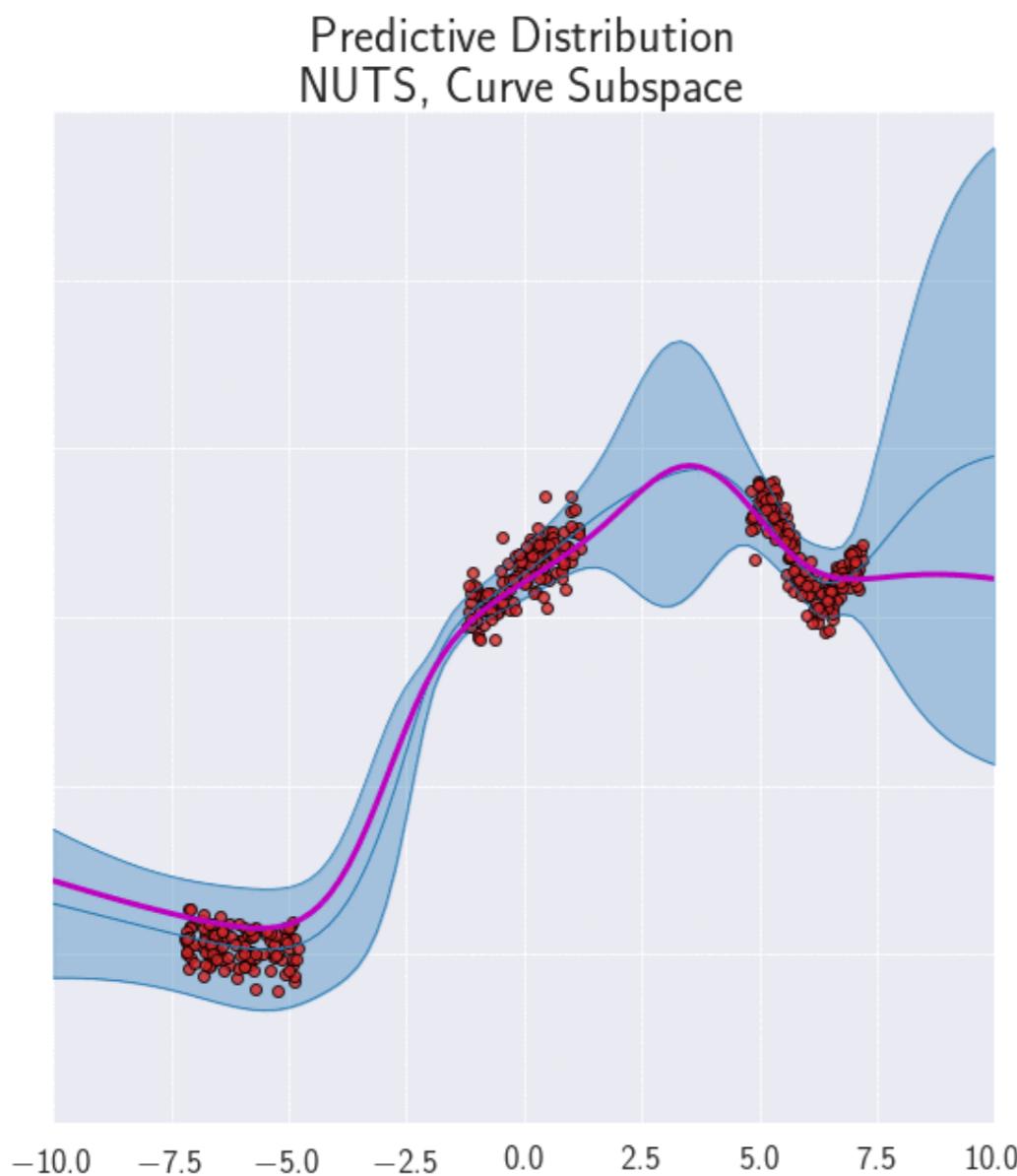


## CURVE SUBSPACE

- Garipov et al. 2018 proposed a method to find 2D subspaces containing a path of low loss between weights of two independently trained neural networks

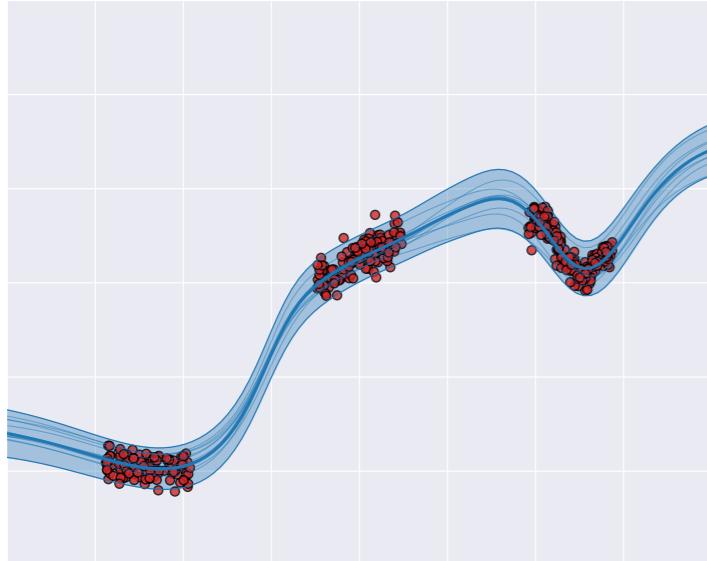


## SUBSPACE INFERENCE ILLUSTRATION

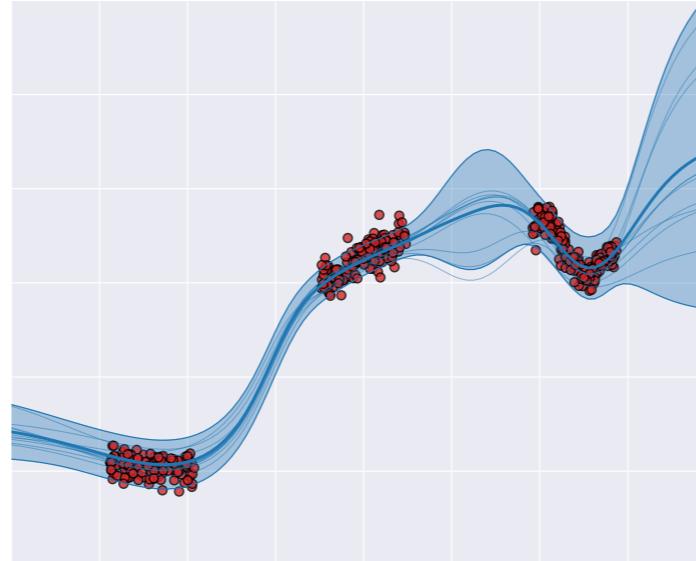


# SUBSPACE COMPARISON

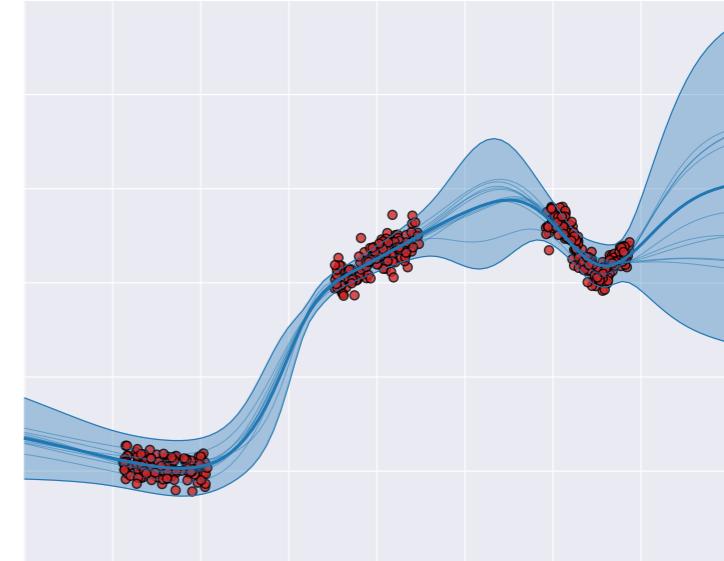
Predictive Distribution  
ESS, Random Subspace



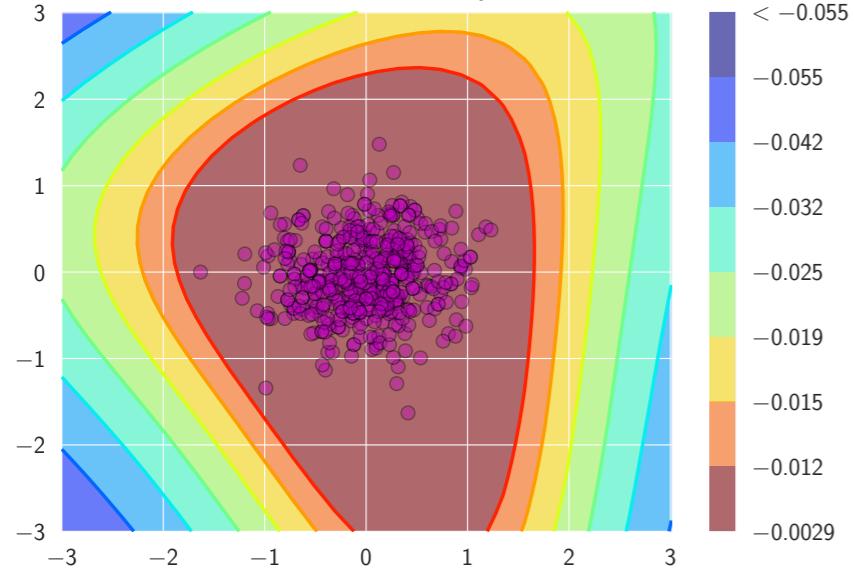
Predictive Distribution  
ESS, PCA Subspace



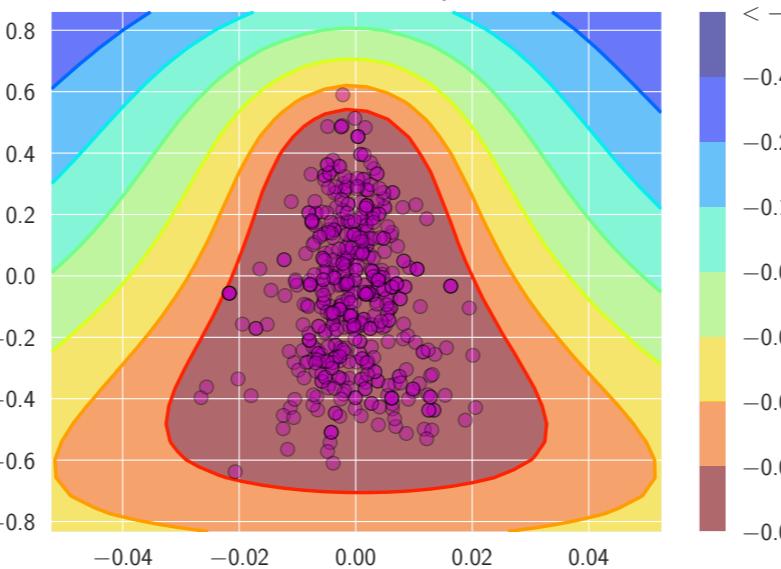
Predictive Distribution  
ESS, Curve Subspace



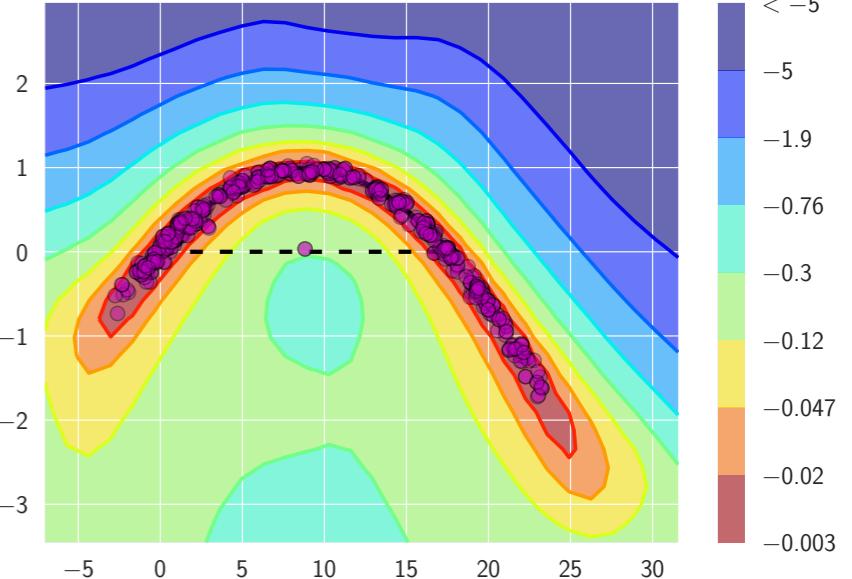
Posterior log-density  
ESS, Random Subspace



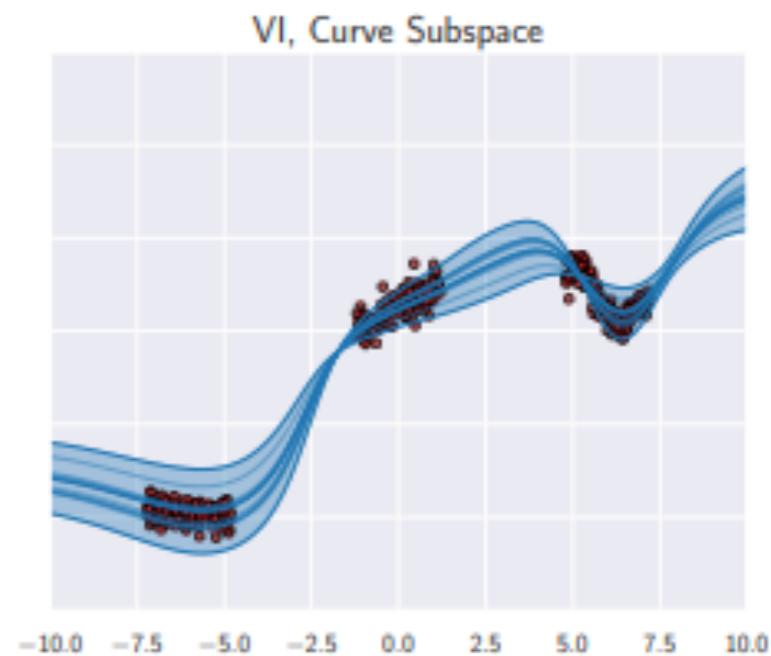
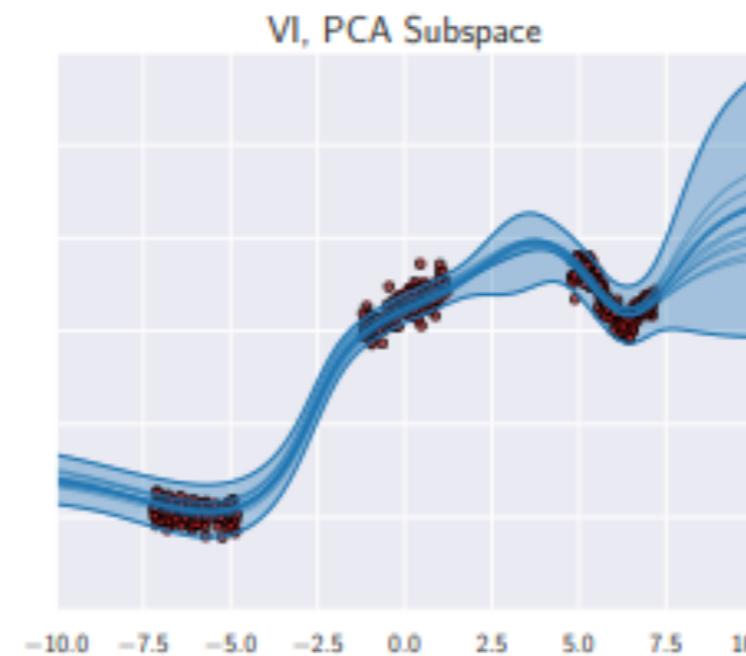
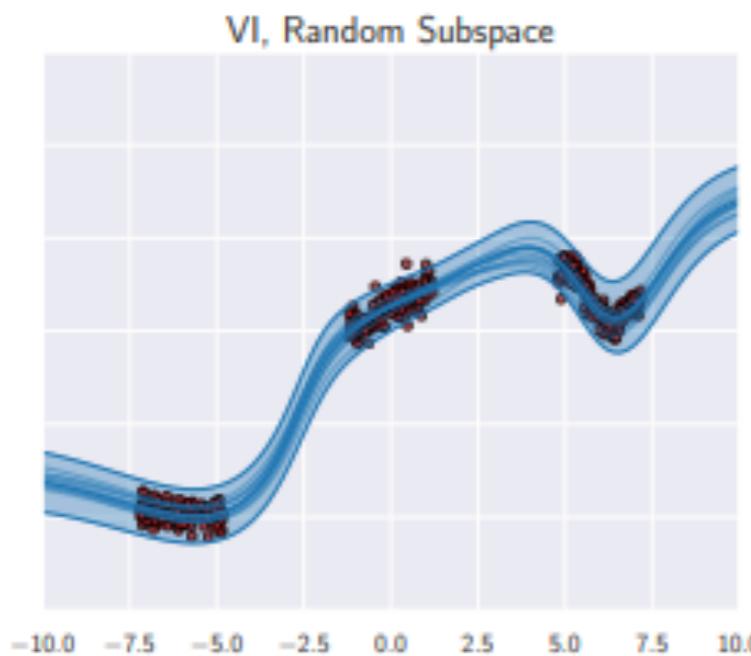
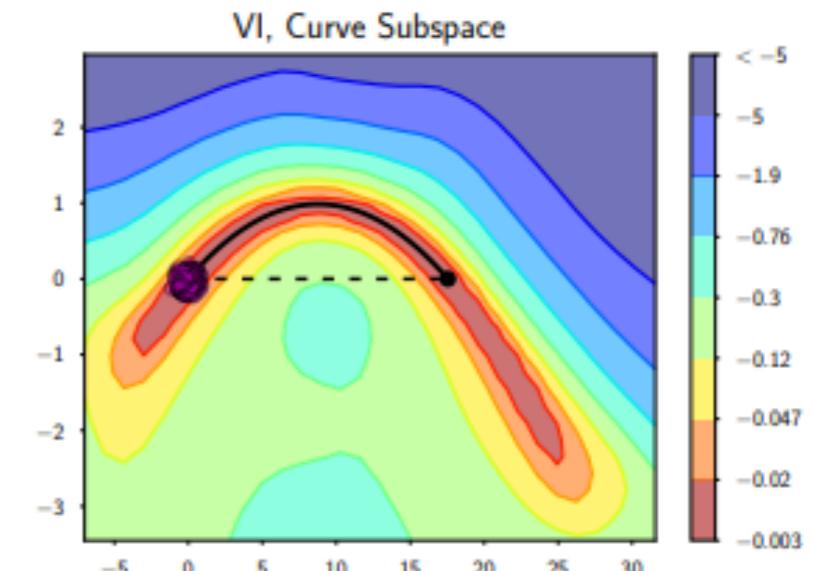
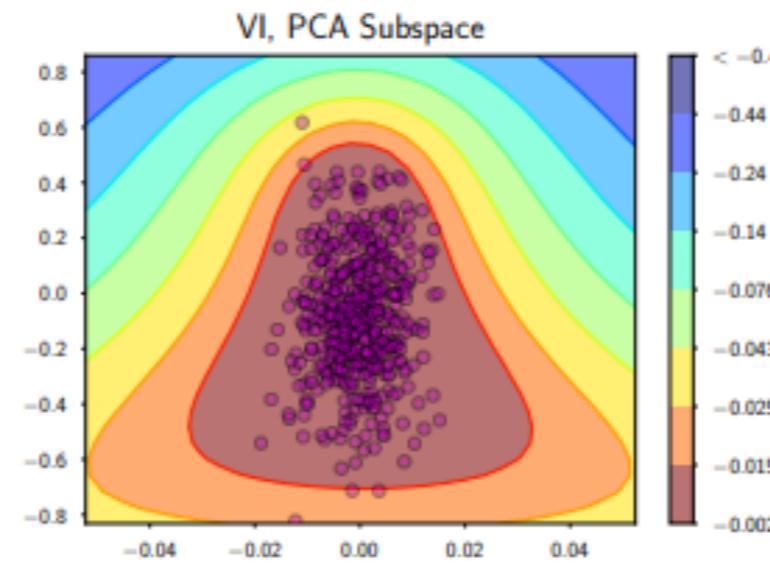
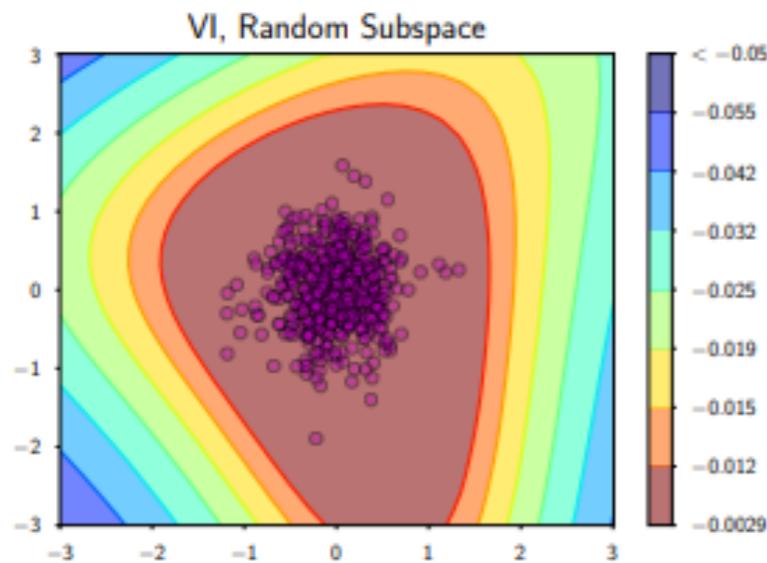
Posterior log-density  
ESS, PCA Subspace



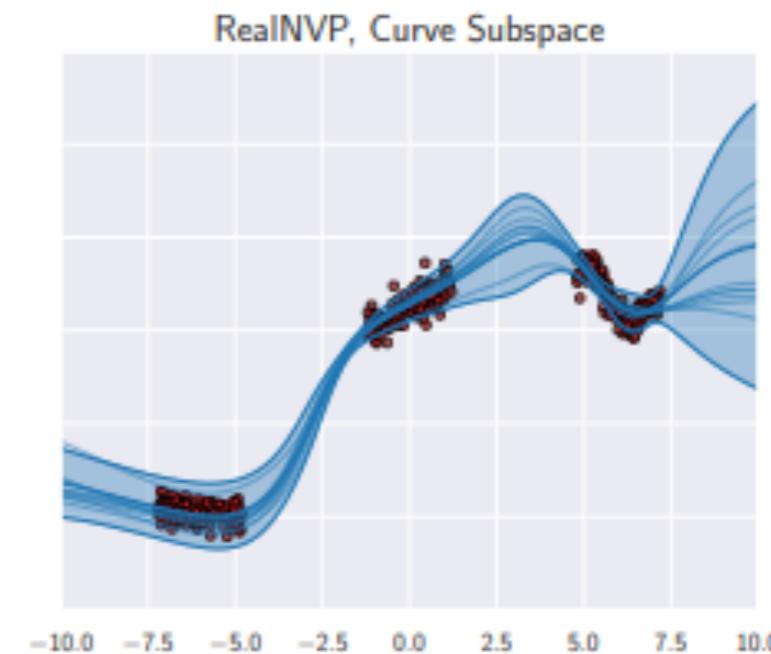
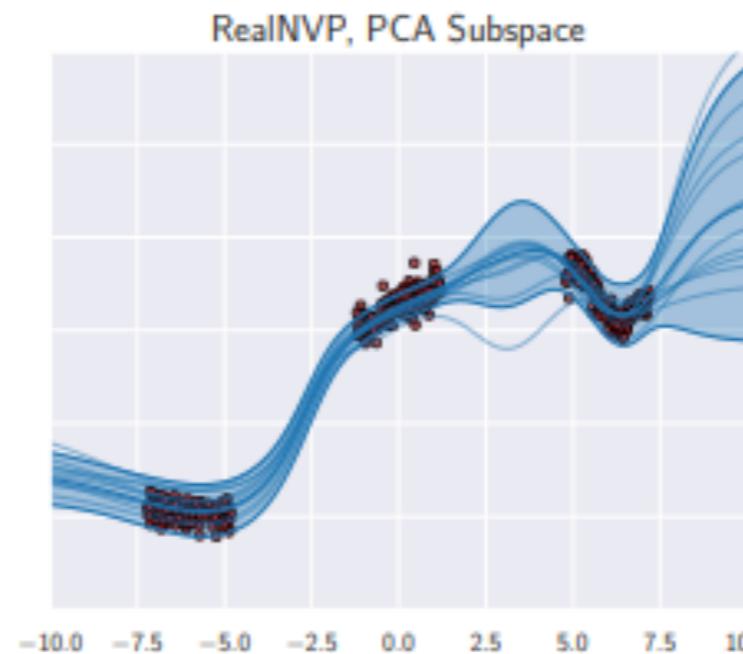
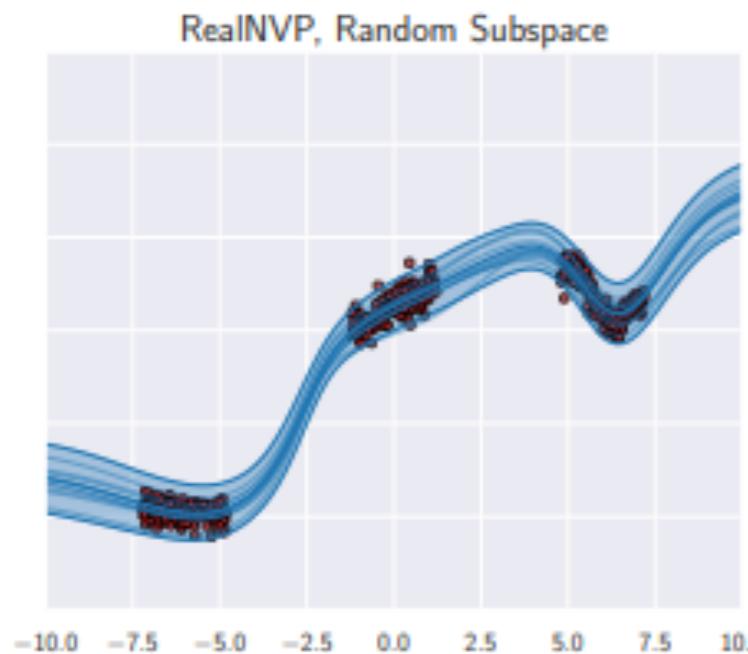
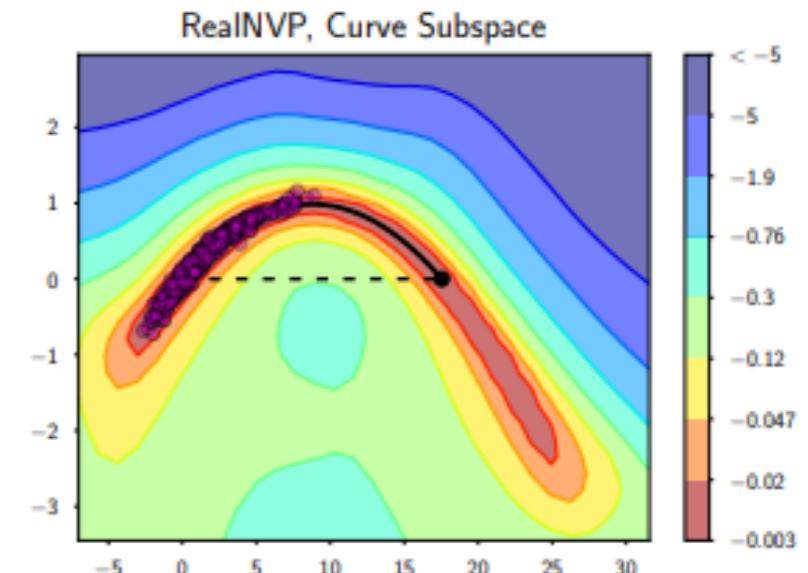
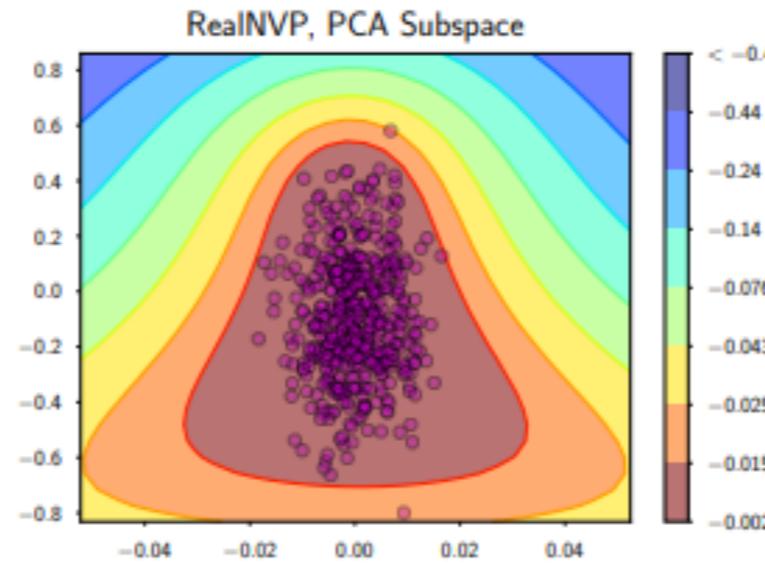
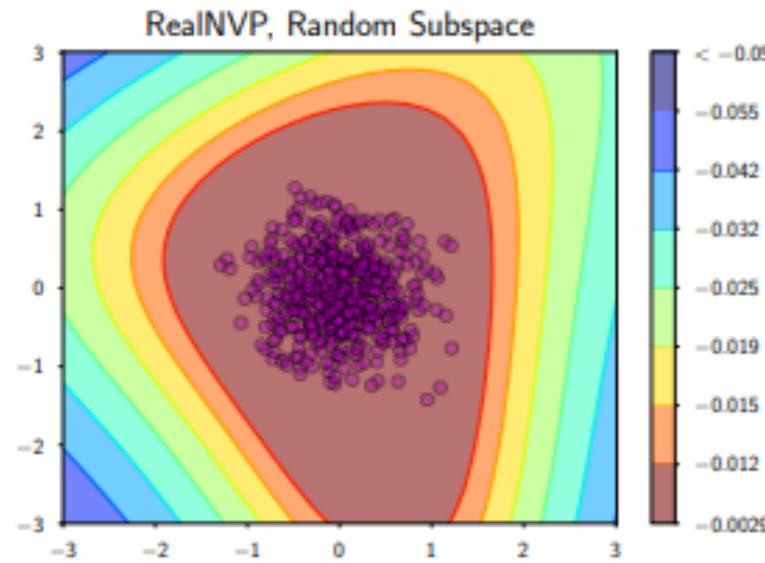
Posterior log-density  
ESS, Curve Subspace



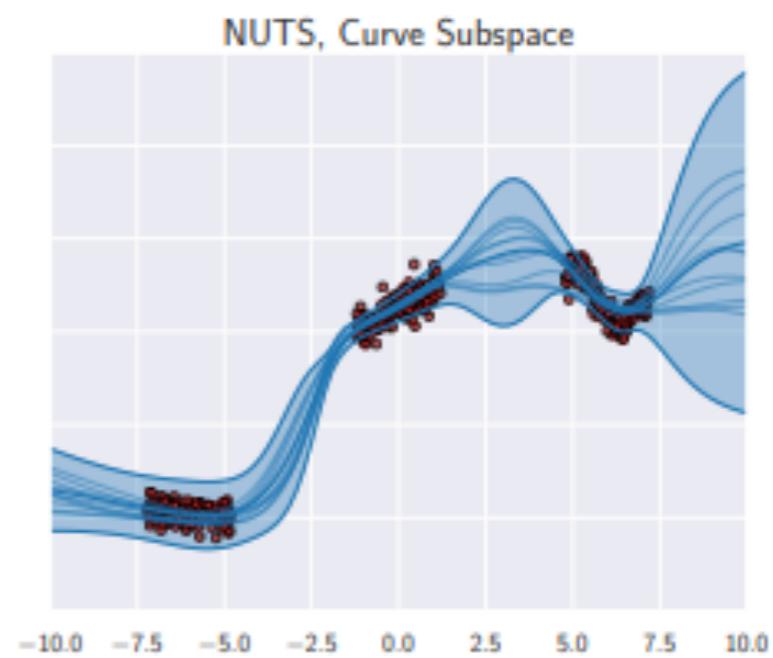
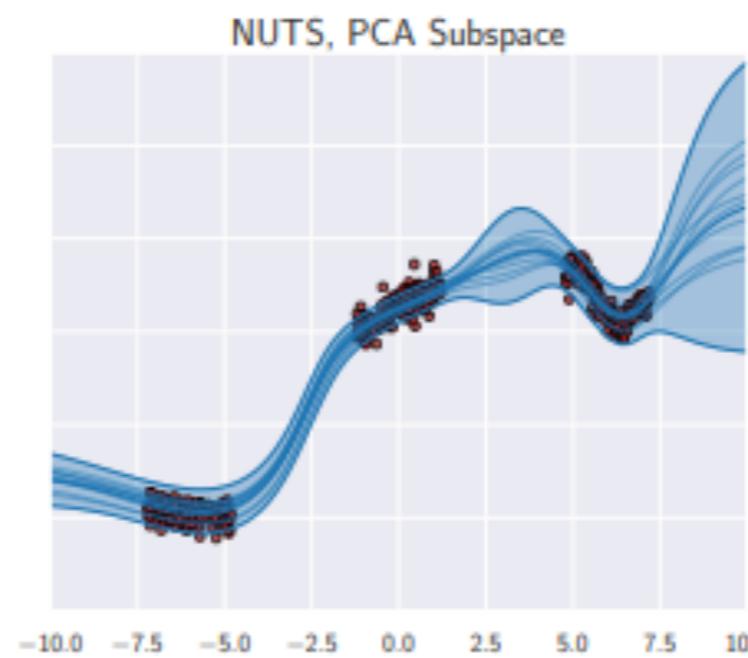
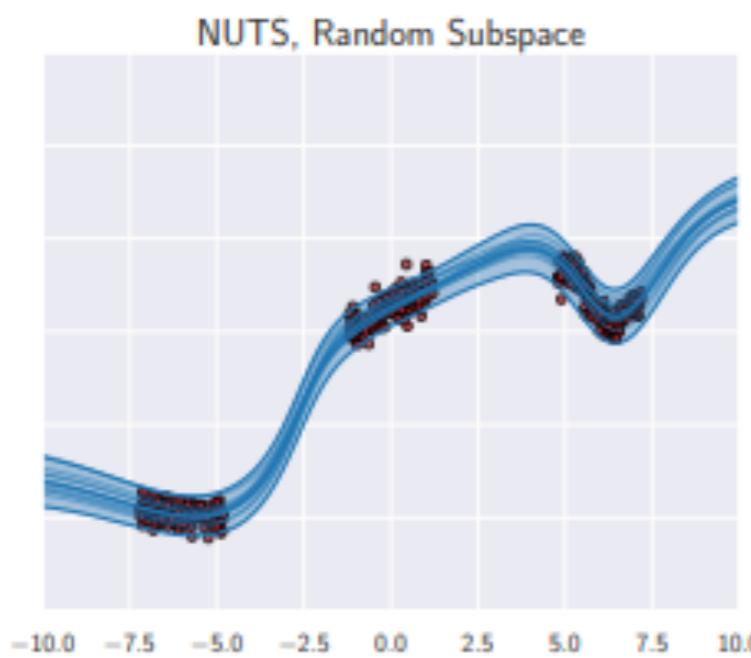
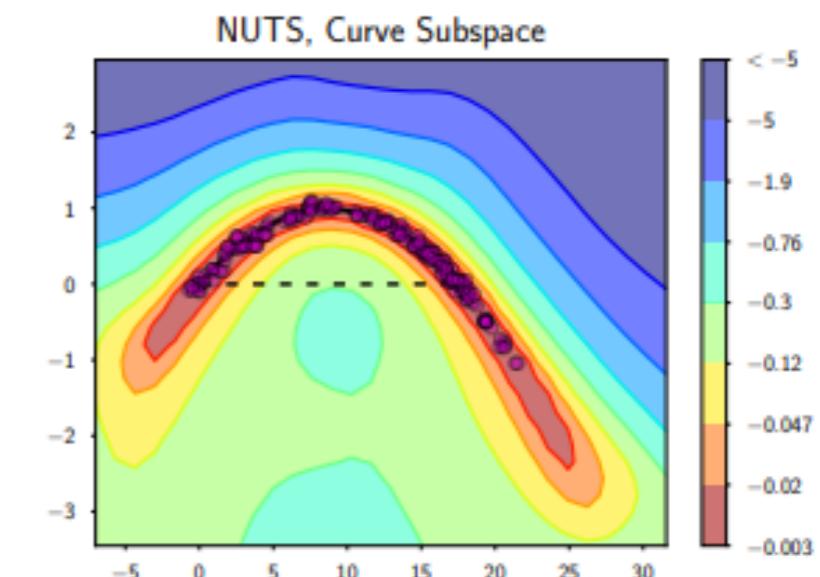
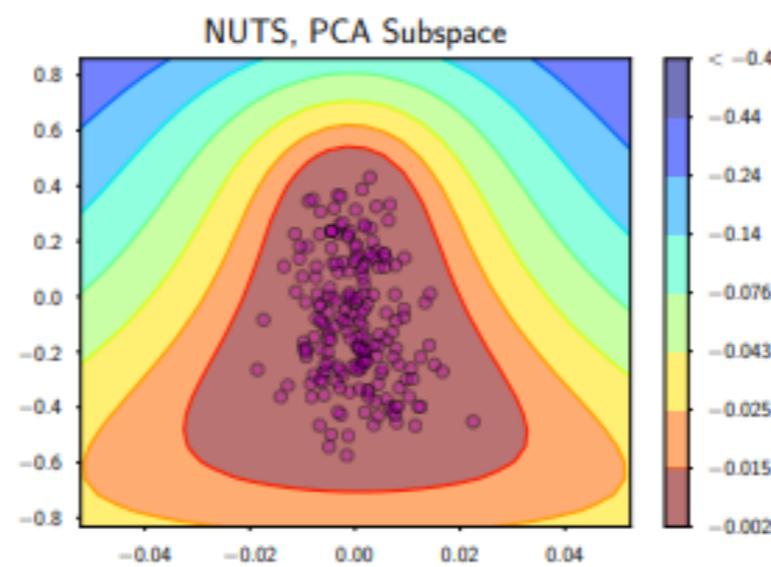
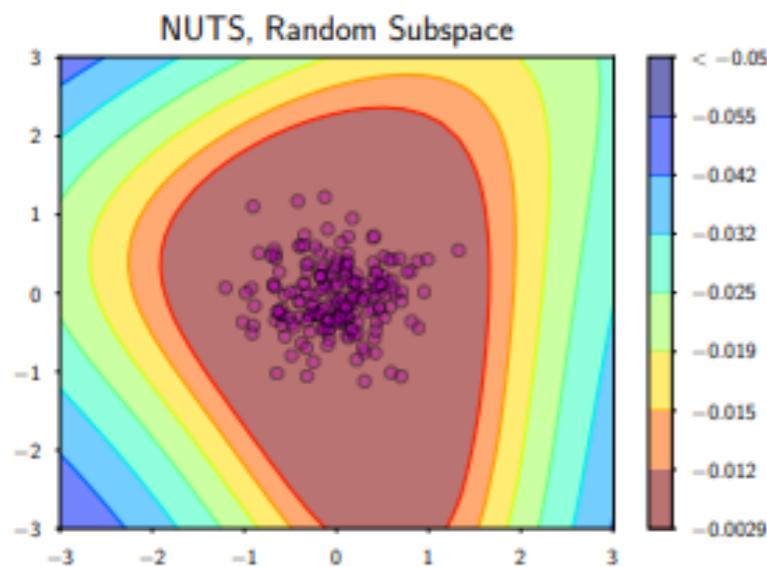
## INFERENCE METHOD COMPARISON



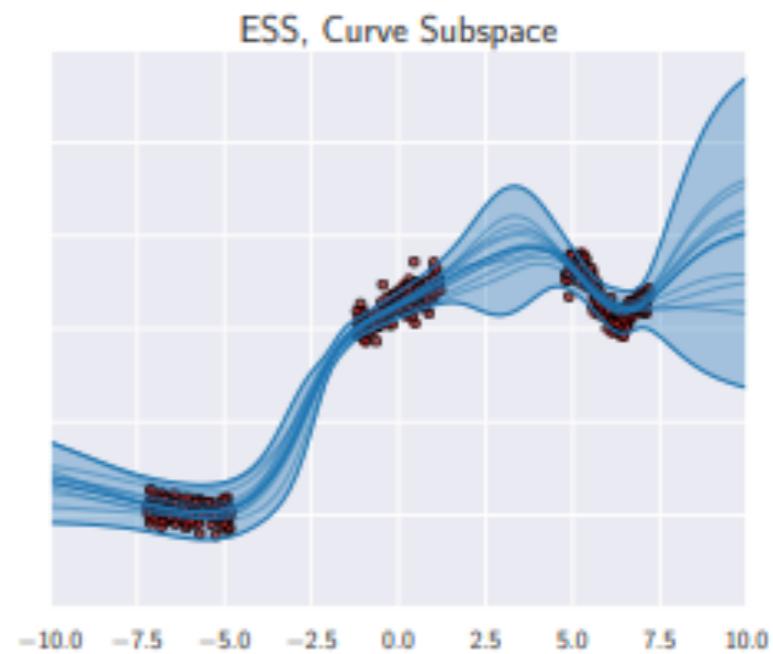
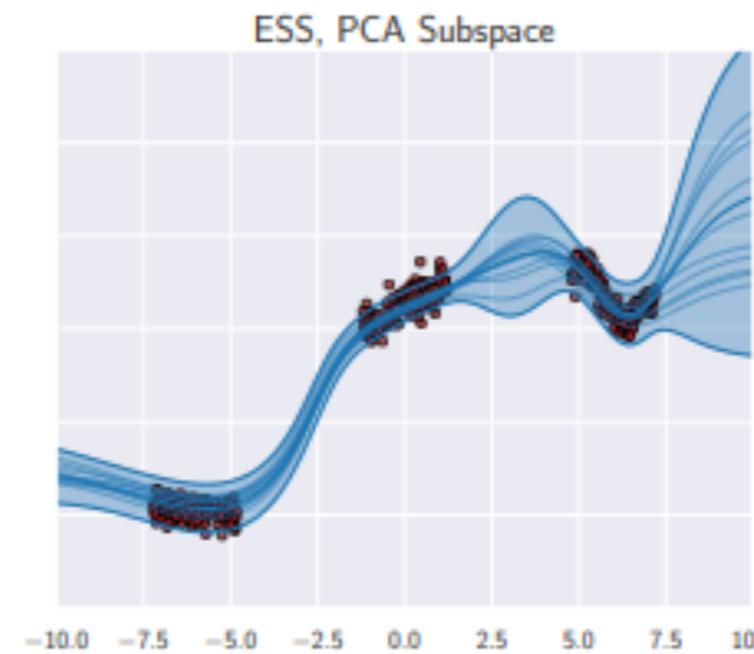
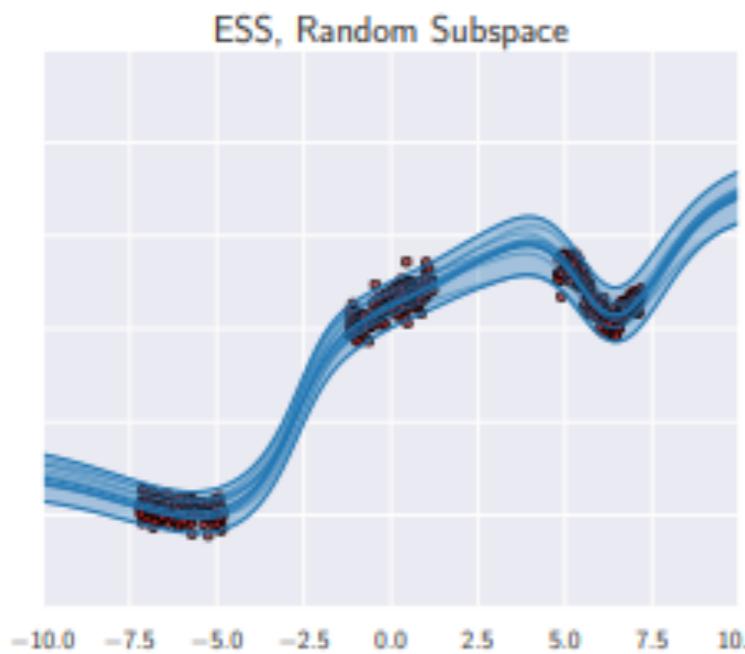
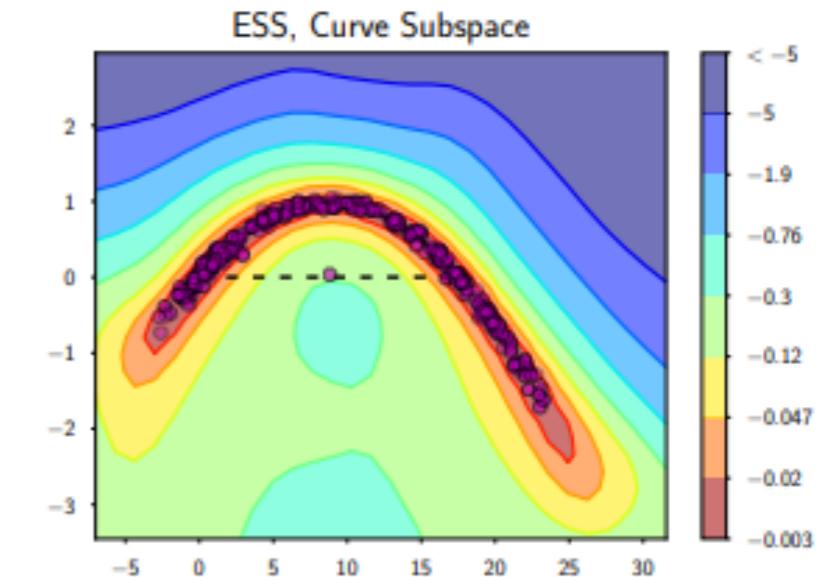
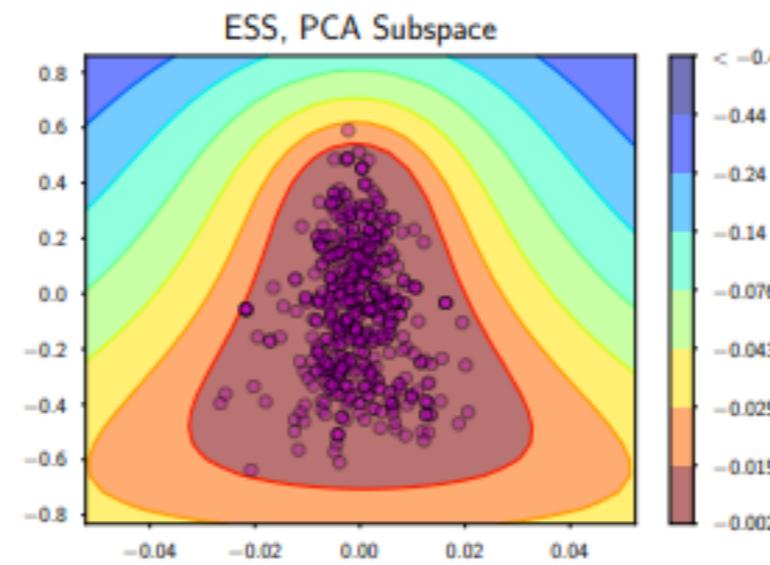
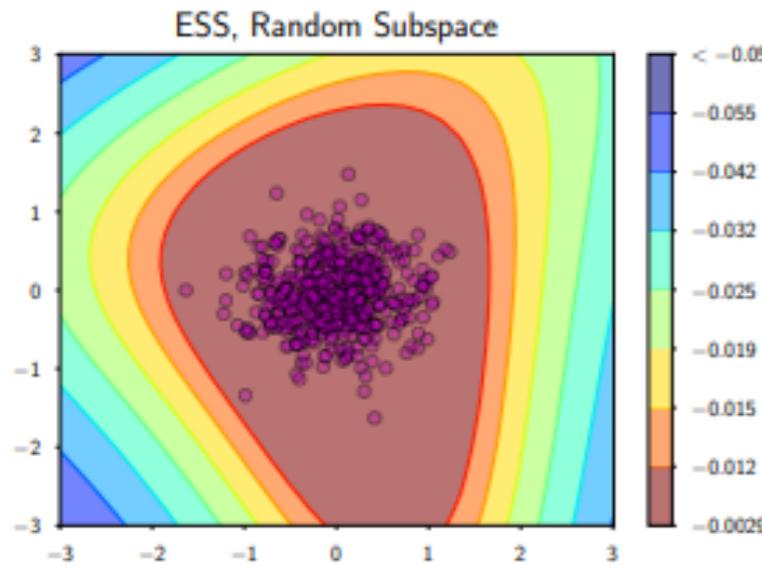
## INFERENCE METHOD COMPARISON



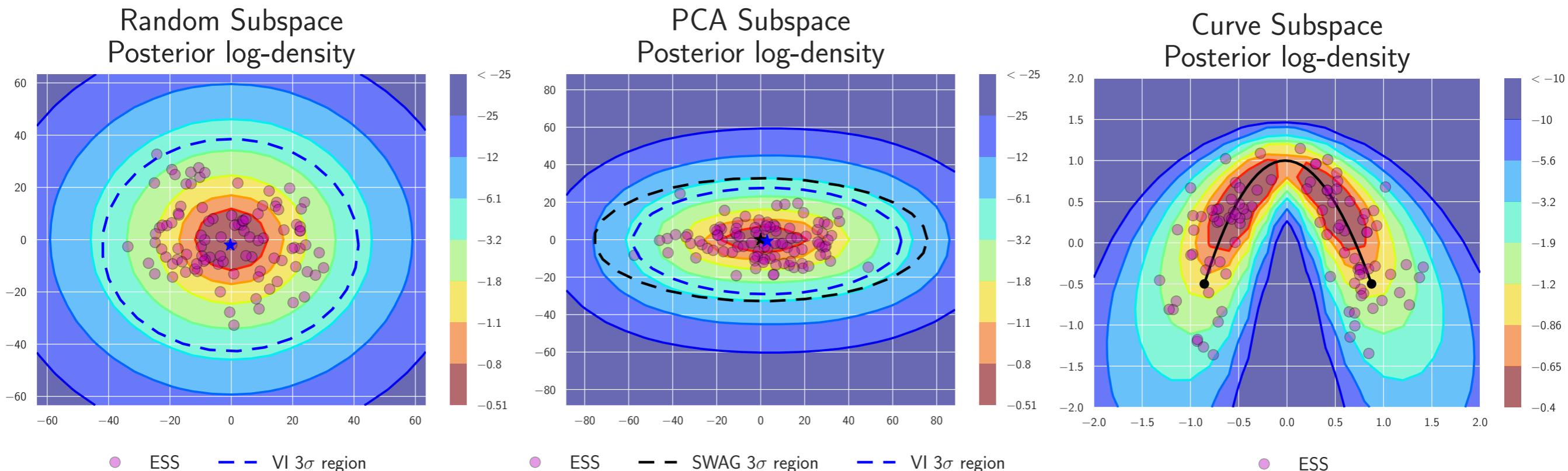
## INFERENCE METHOD COMPARISON



## INFERENCE METHOD COMPARISON

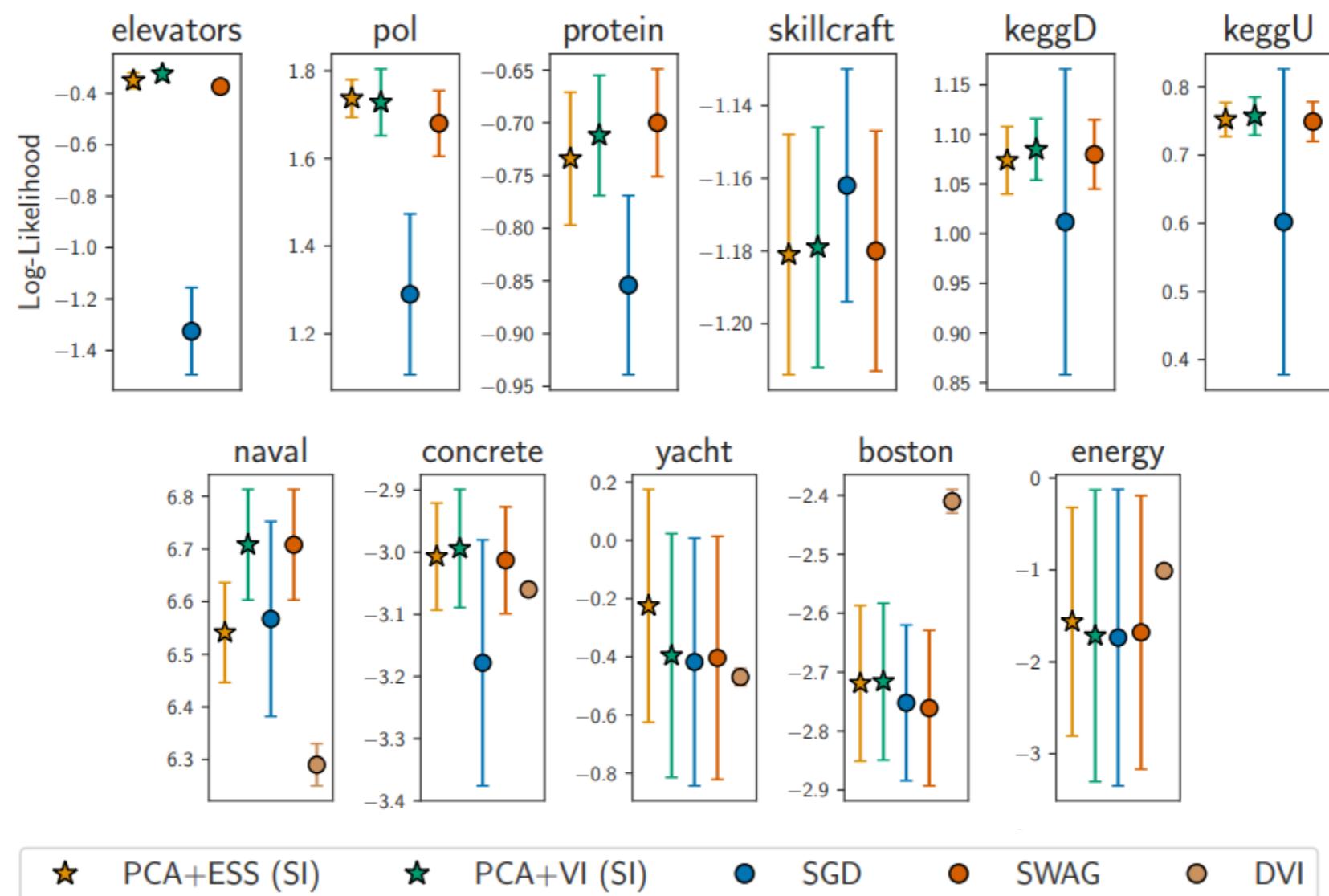


# SUBSPACE COMPARISON ON PRERESNET-164, CIFAR-100



	SGD	Random	PCA	Curve
NLL	$0.946 \pm 0.001$	$0.686 \pm 0.005$	$0.665 \pm 0.004$	$0.646$
Accuracy (%)	$78.50 \pm 0.32$	$80.17 \pm 0.03$	$80.54 \pm 0.13$	$81.28$

## UCI EXPERIMENTS

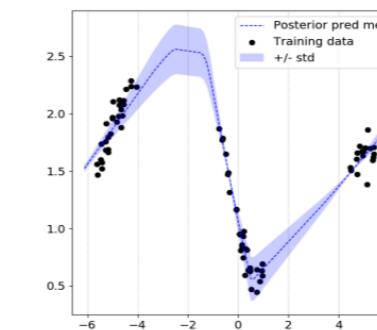
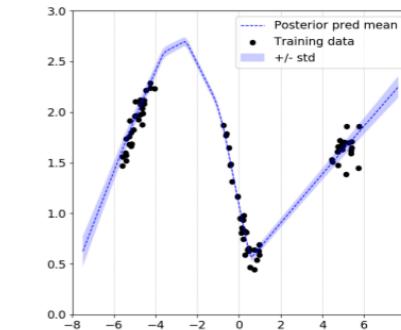


## SUBSPACE INFERENCE TAKEAWAYS

- ▶ We can apply standard approximate inference methods in subspaces of parameter space
- ▶ More diverse subspaces => better performance:  
Curve Subspace > **PCA Subspace** > Random Subspace
- ▶ Subspace Inference in the PCA subspace is competitive with SWAG (Maddox et al., 2019), MC-Dropout (Gal & Ghahramani, 2016) and Temperature Scaling (Guo et al., 2017) on image classification and UCI regression

# SCALABLE POSTERIOR APPROXIMATIONS SINCE 2019

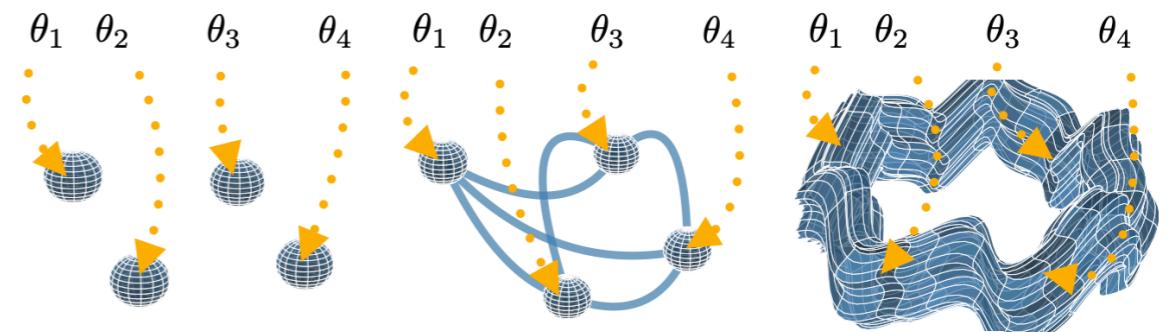
- ▶ Pradier et al, Projected BNNs: Avoiding weight-space pathologies by learning latent representations of neural network weights.
- ▶ Dusenberry et al, Efficient and scalable Bayesian Neural Nets with rank-1 factors.
- ▶ Benton et al, Loss Surface Simplexes for Mode Connecting Volumes and Fast Ensembling
- ▶ Franchi et al, Encoding the latent posterior of Bayesian Neural Networks for uncertainty quantification (**one of the upcoming UQSay seminars!**)
- ▶ ...

(a) Proj-BNN ( $D_z = 2$ )

(b) BbB

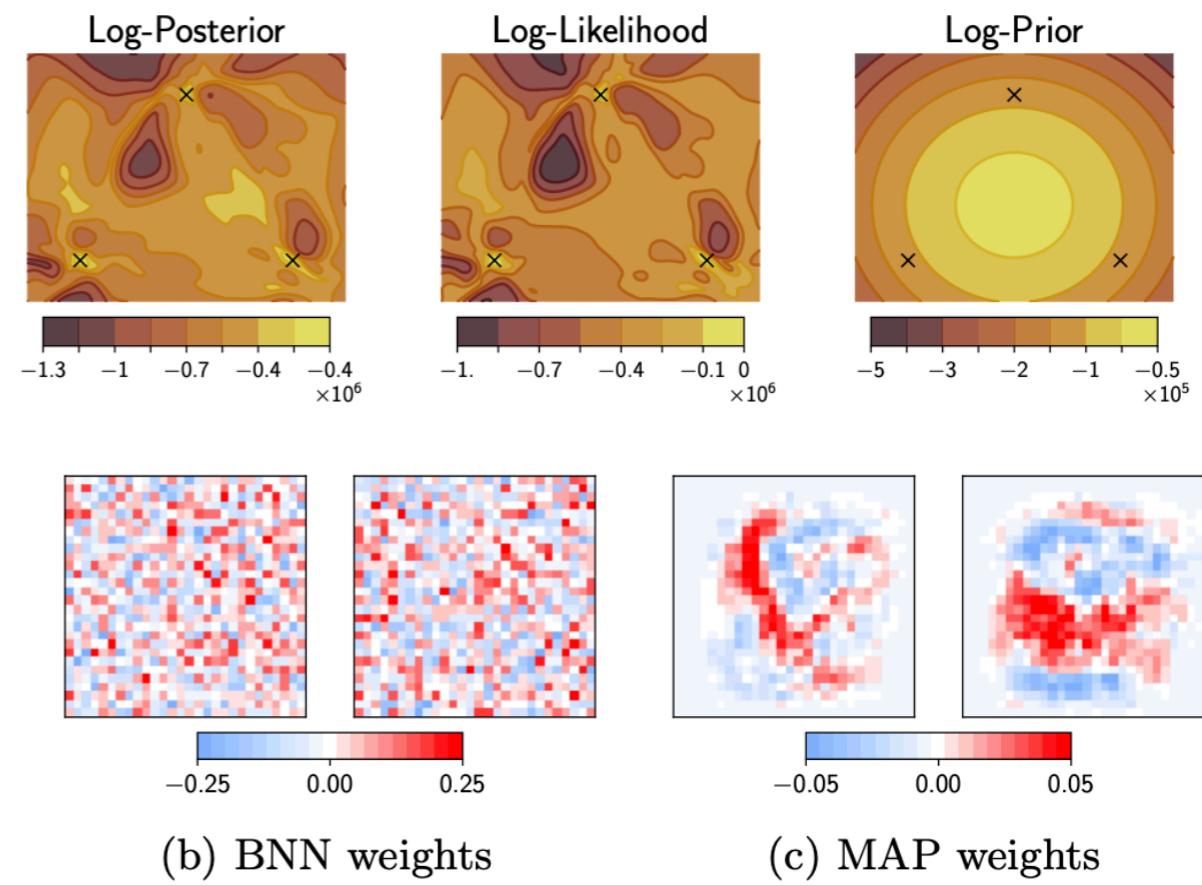
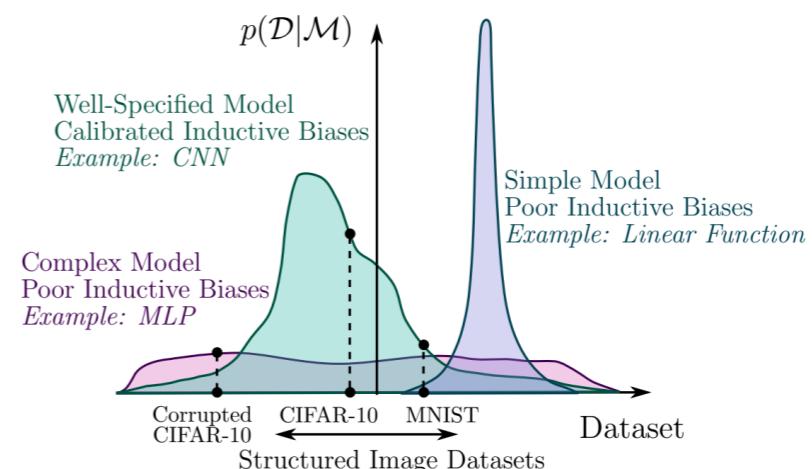
$$\begin{array}{ccc}
 \text{W} & \xrightarrow{\text{O}} & \begin{matrix} \text{r}_1 & \text{r}_1\text{s}_1^T \\ \text{r}_2 & \text{r}_2\text{s}_2^T \end{matrix} = \begin{matrix} \text{s}_1^T \\ \text{s}_2^T \end{matrix} & W_1 = W \circ \text{r}_1\text{s}_1^T \\
 & & \text{W} & W_2 = W \circ \text{r}_2\text{s}_2^T
 \end{array}$$

The diagram illustrates the decomposition of a weight matrix  $\text{W}$  into two rank-1 components. It shows a color bar ranging from -1 (red) to 1 (blue). The first row shows  $\text{W}$  being multiplied by  $\text{r}_1$  (indicated by an arrow) to produce  $\text{r}_1\text{s}_1^T$ , which is then equated to  $\text{s}_1^T$ . The second row shows  $\text{W}$  being multiplied by  $\text{r}_2$  (indicated by an arrow) to produce  $\text{r}_2\text{s}_2^T$ , which is then equated to  $\text{s}_2^T$ .

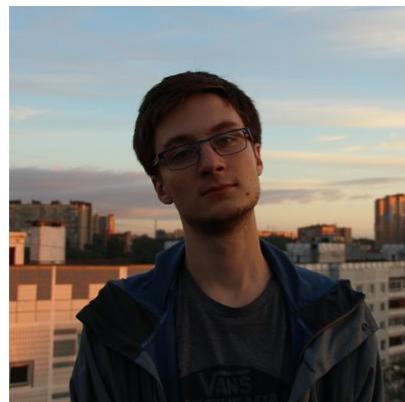


# RECENT PAPER ON BAYESIAN DEEP LEARNING

- ▶ Wilson et al, Bayesian Deep Learning and a probabilistic perspective of generalization
- ▶ Izmailov et al, What Are Bayesian Neural Network Posteriors Really Like?
- ▶ Izmailov et al, Dangers of Bayesian Model Averaging under Covariate Shift
- ▶ ...



# ACKNOWLEDGEMENTS



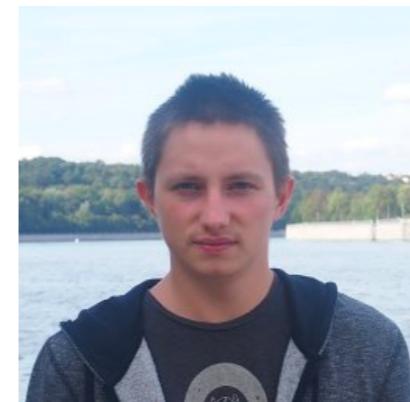
**Pavel Izmailov**

NYU



**Wesley Maddox**

NYU



**Timur Garipov**

MIT



**Andrew Wilson**

NYU

- ▶ Paper: <https://arxiv.org/abs/1907.07504> (UAI 2019)
- ▶ Code: <https://github.com/wjmaddox/drbayes>