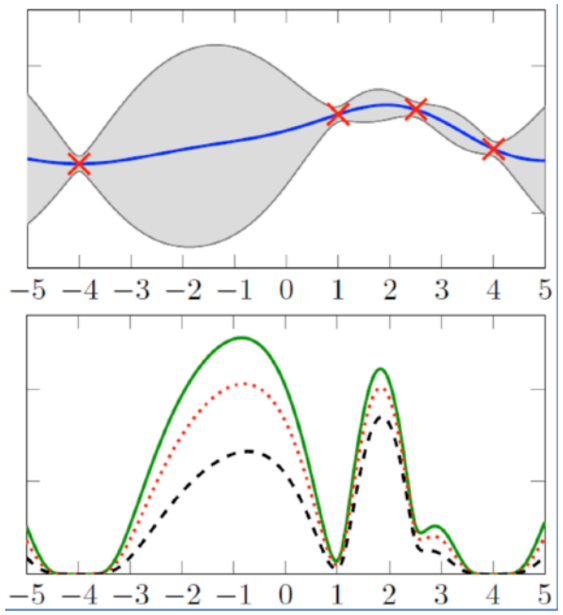
Function Space Priors in Bayesian Deep Learning

Roger Grosse

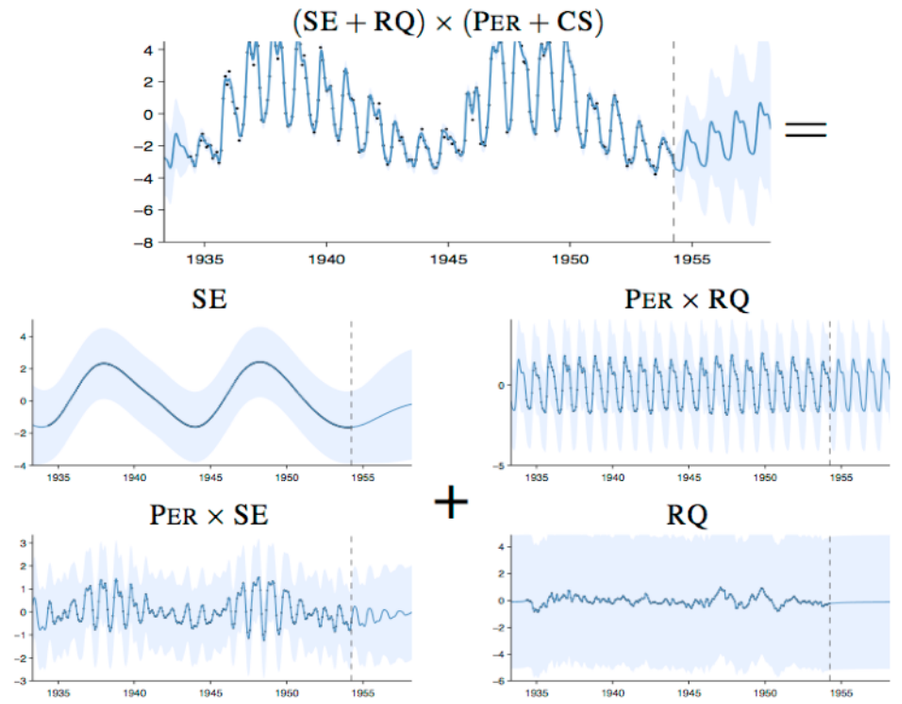
Motivation

- Today Bayesian deep learning is most often tested on
  - regularization (Bayesian Occam's Razor, description length regularization)
  - smoothing the predictions
  - calibration and confidence intervals
  - novelty and out-of-distribution detection
  - noise to encourage exploration in RL
- But these all have non-Bayesian approaches that are competitive

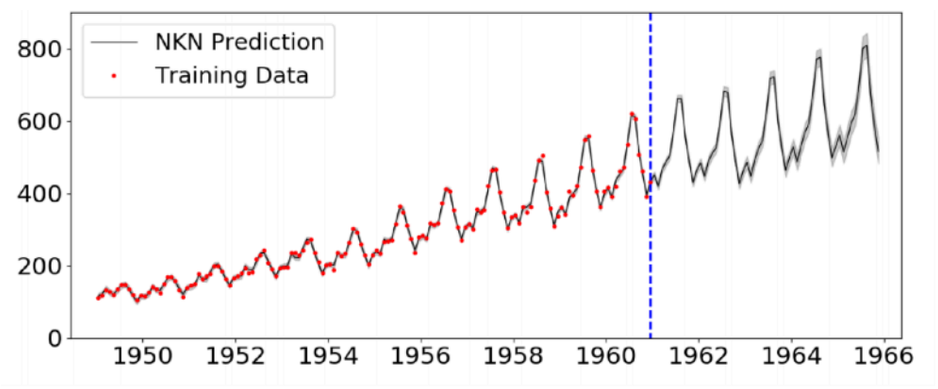
The Three X's



Exploration



Explanation

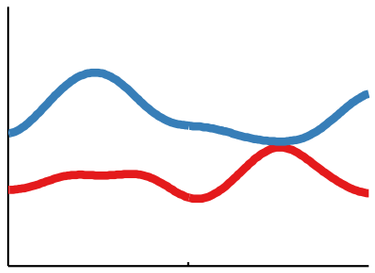


Extrapolation

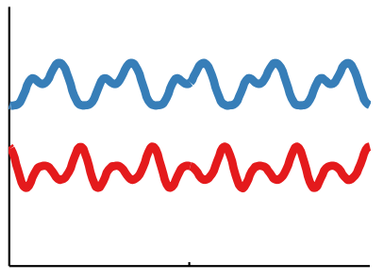
Compositional GP Kernels

Gaussian processes are distributions over functions, specified by kernels.

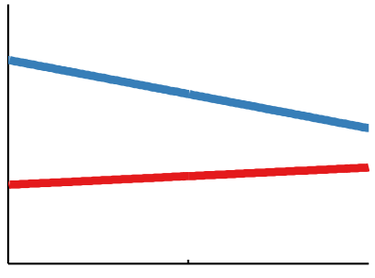
Primitive kernels:



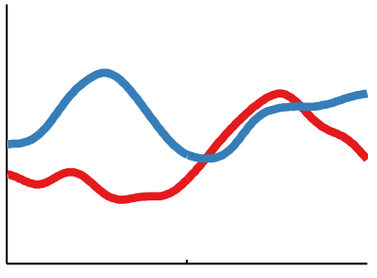
SE



Per

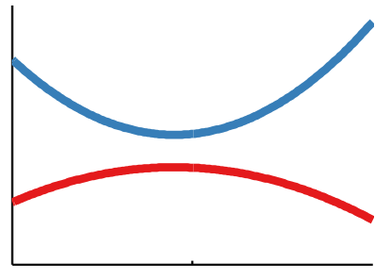


Lin

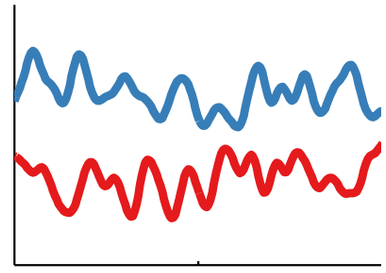


RQ

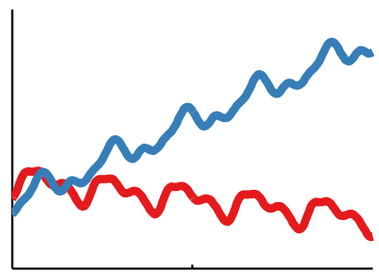
Composite kernels:



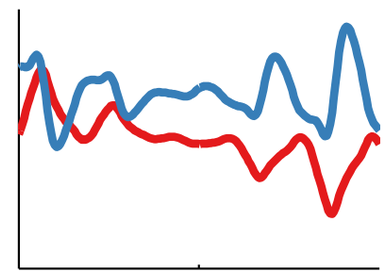
Lin × Lin



SE × Per

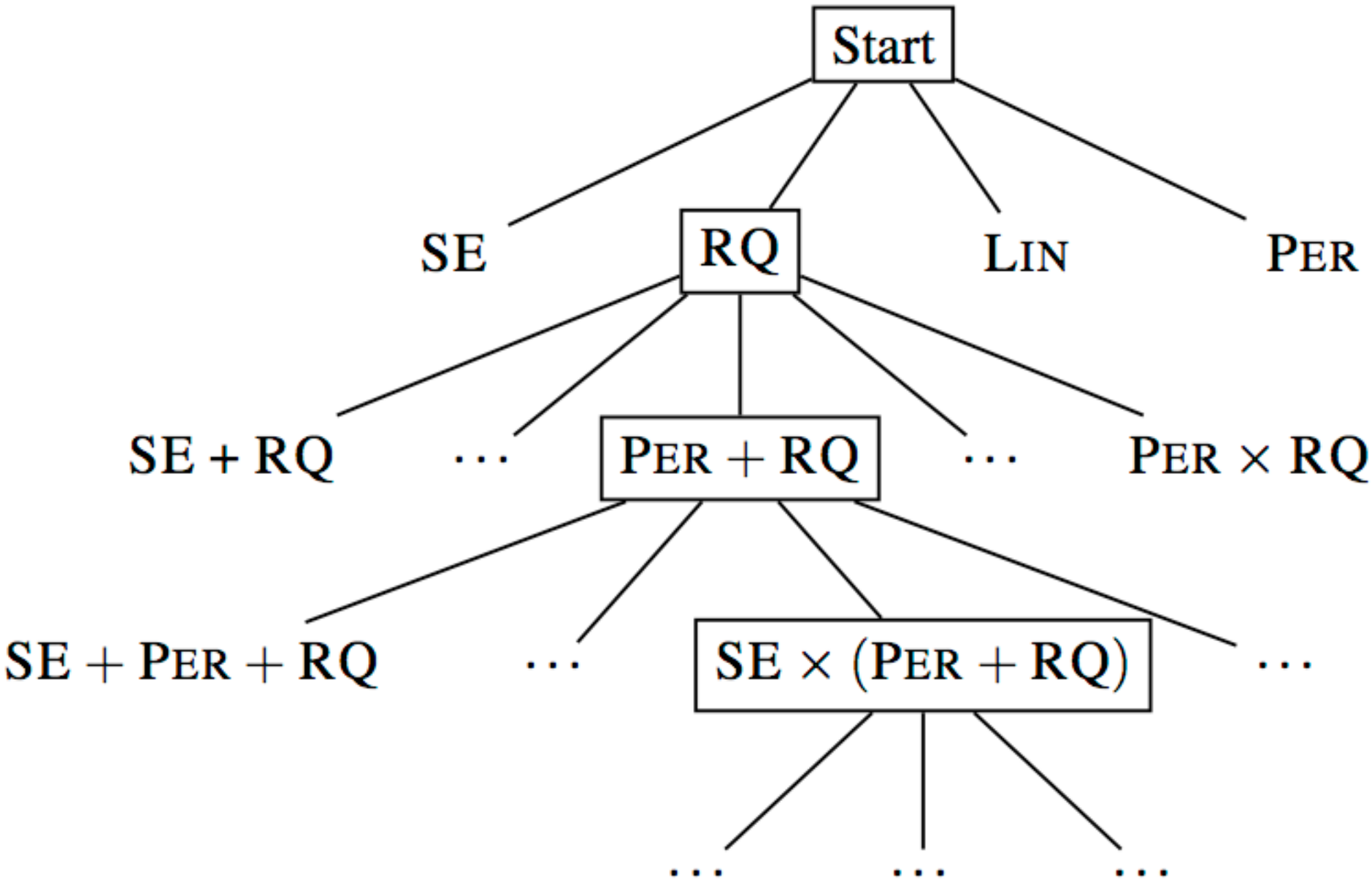


Lin + Per



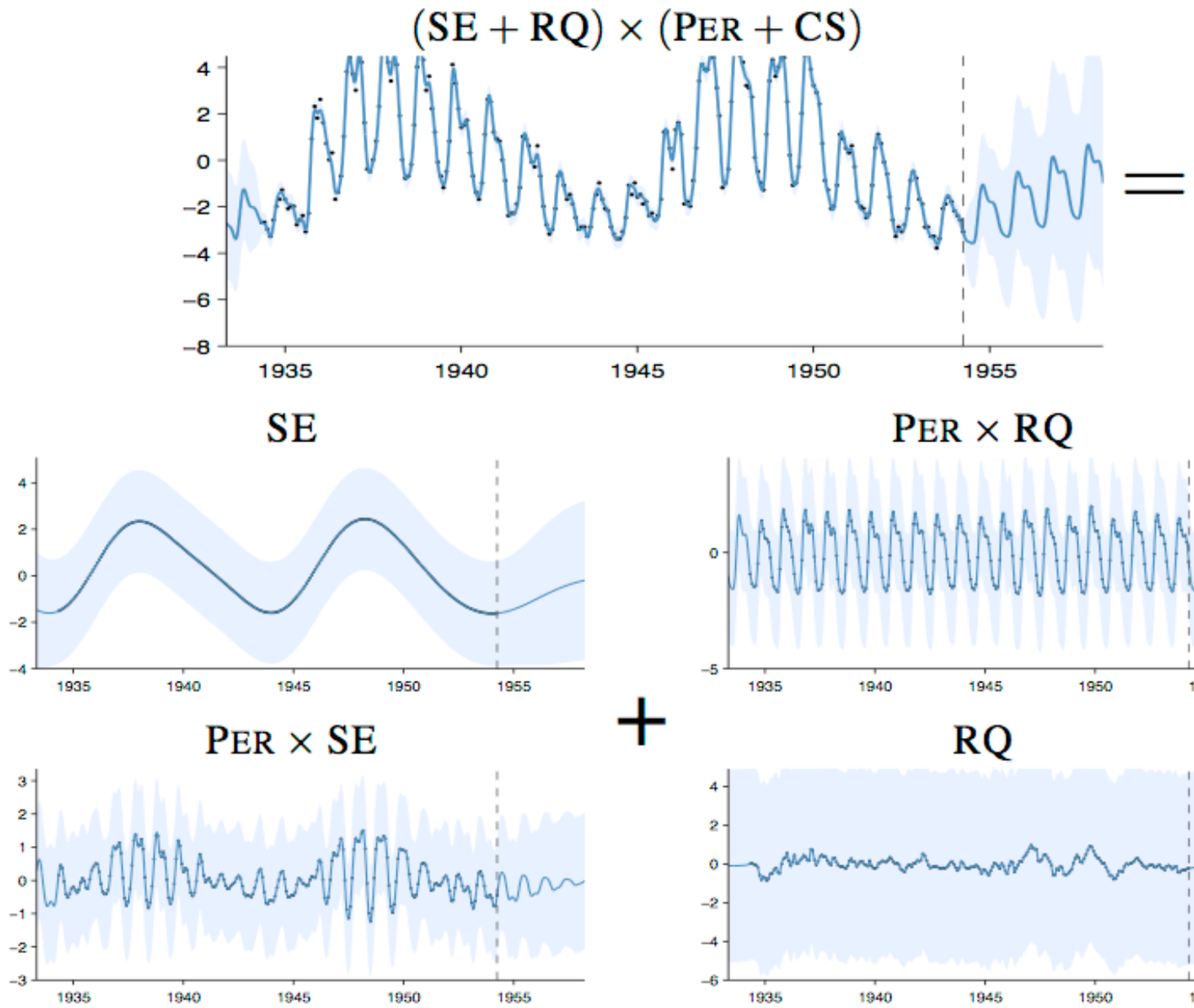
Lin × Per

Automatic Statistician



- Duvenaud et al., 2013, “Structure discovery in nonparametric regression through compositional kernel search”

Automatic Statistician



- Duvenaud et al., 2013, “Structure discovery in nonparametric regression through compositional kernel search”

An automatic report for the dataset : 01-airline

The Automatic Statistician

Abstract

This report was produced by the Automatic Bayesian Covariance Discovery (ABCD) algorithm.

1 Executive summary

The raw data and full model posterior with extrapolations are shown in figure 1.

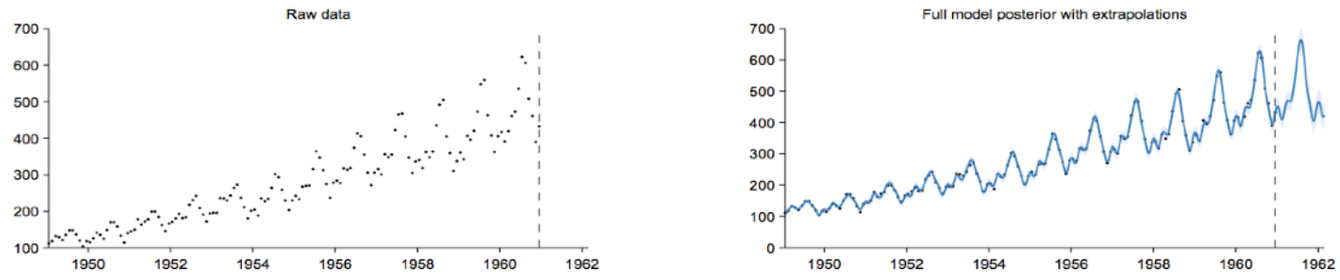


Figure 1: Raw data (left) and model posterior with extrapolation (right)

The structure search algorithm has identified four additive components in the data. The first 2 additive components explain 98.5% of the variation in the data as shown by the coefficient of determination (R^2) values in table 1. The first 3 additive components explain 99.8% of the variation in the data. After the first 3 components the cross validated mean absolute error (MAE) does not decrease by more than 0.1%. This suggests that subsequent terms are modelling very short term trends, uncorrelated noise or are artefacts of the model or search procedure. Short summaries of the additive components are as follows:

- Lloyd et al., 2014, “Automatic construction and natural-language description of nonparametric regression models”

Structured Priors and Deep Learning

- This demonstrates the power and flexibility of function space priors.
- Problems
  - Requires a discrete search over the space of kernel structures (tries thousands to analyze a dataset)
  - Need to re-fit the kernel hyperparameters for each candidate structure
- Can Bayesian deep learning discover and exploit structured function space priors?
  - **Discover:** Neural Kernel Network learns compositional kernels
  - **Exploit:** functional variational BNN performs variational inference in function space
- Caveat: we haven't yet figured out how to do both simultaneously

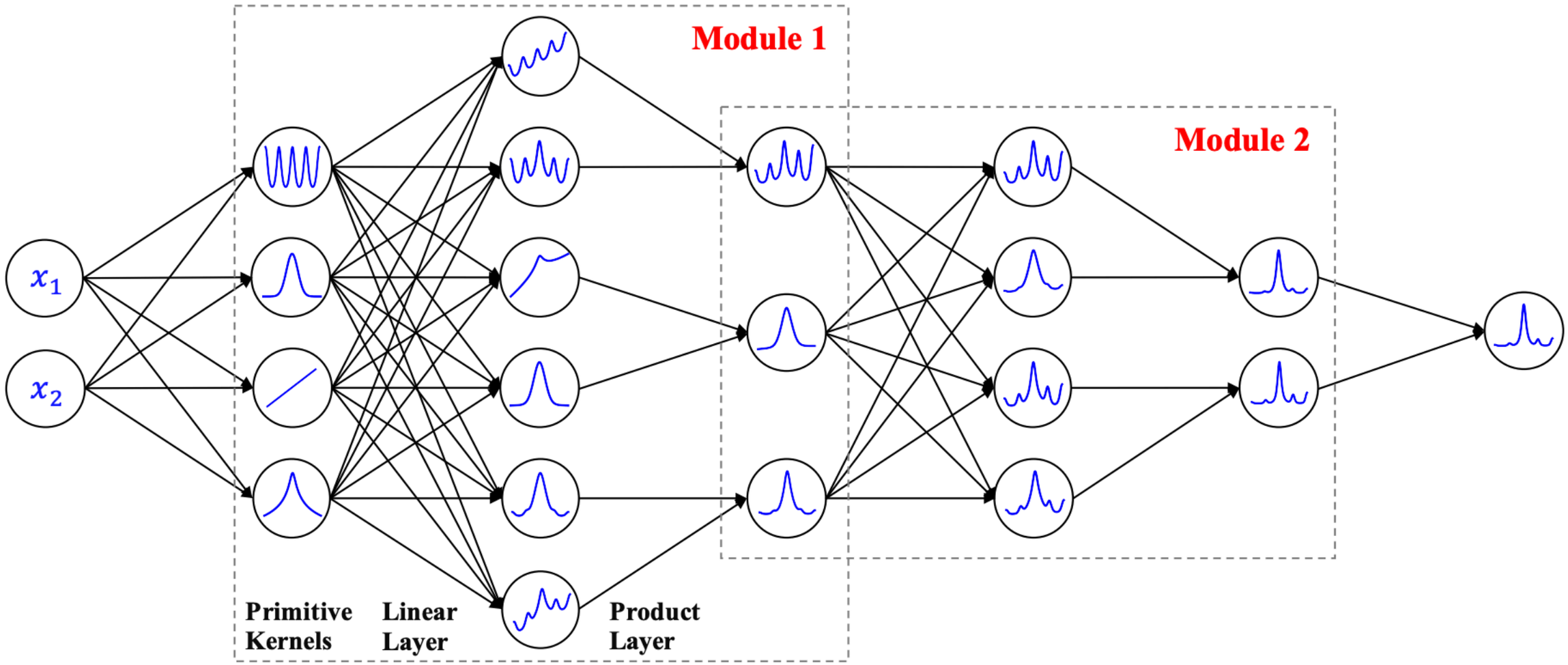
Differentiable Compositional Kernel Learning for Gaussian Processes

Shengyang Sun, Guodong Zhang, Chaoqi Wang,
Wenyuan Zeng, Jiaman Li

ICML 2018

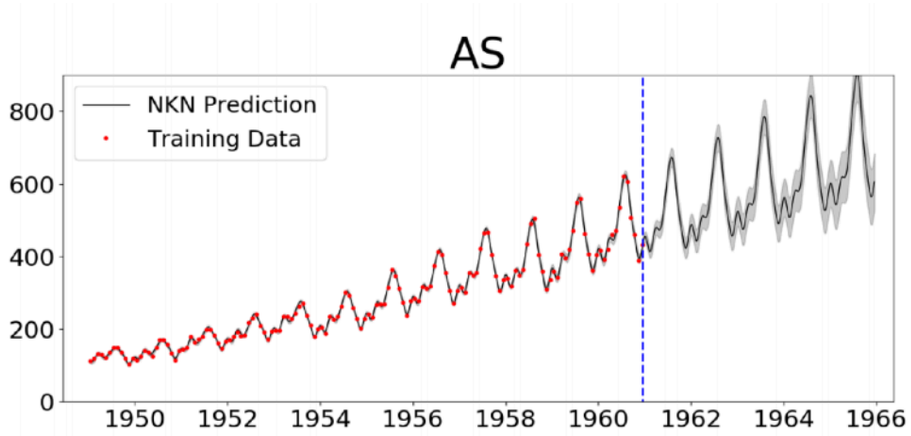
Neural Kernel Network

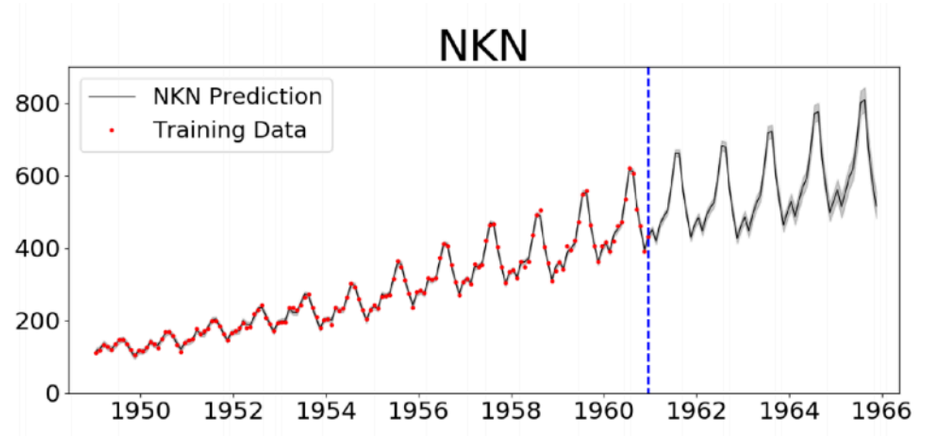
- **Neural Kernel Network:** a neural net architecture that takes two input locations and computes the kernel between them
- Layers are defined using the composition rules, so every unit corresponds to a valid kernel.
- Good at representing the same compositional structures as the Automatic Statistician, but is end-to-end differentiable



Learning Flexible GP Kernels

- Extrapolates time series datasets similarly to the Automatic Statistician



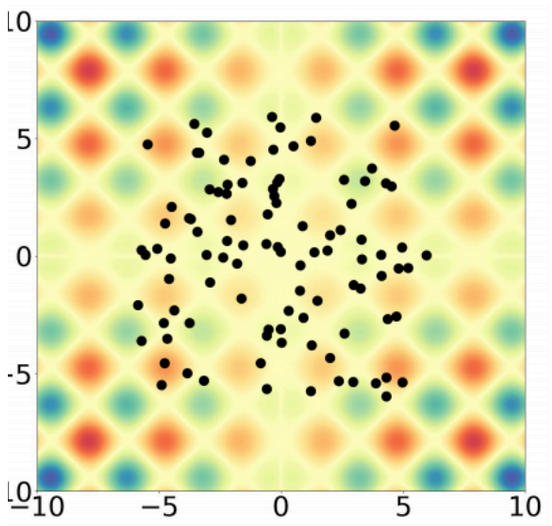


- Runs in minutes rather than hours:

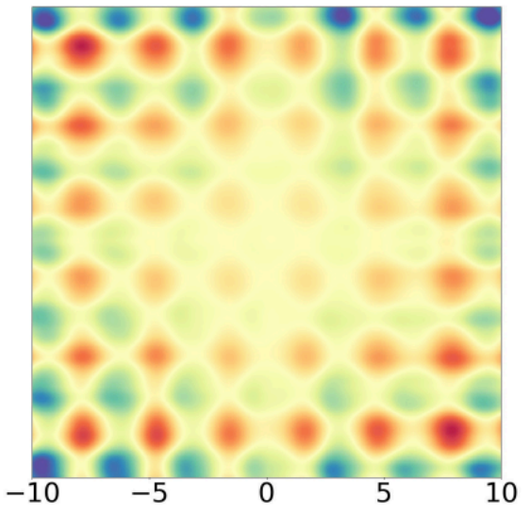
	Airline	Mauna	Solar
Automatic Statistician	6147	51065	37716
NKN	201	576	962

Learning Flexible GP Kernels

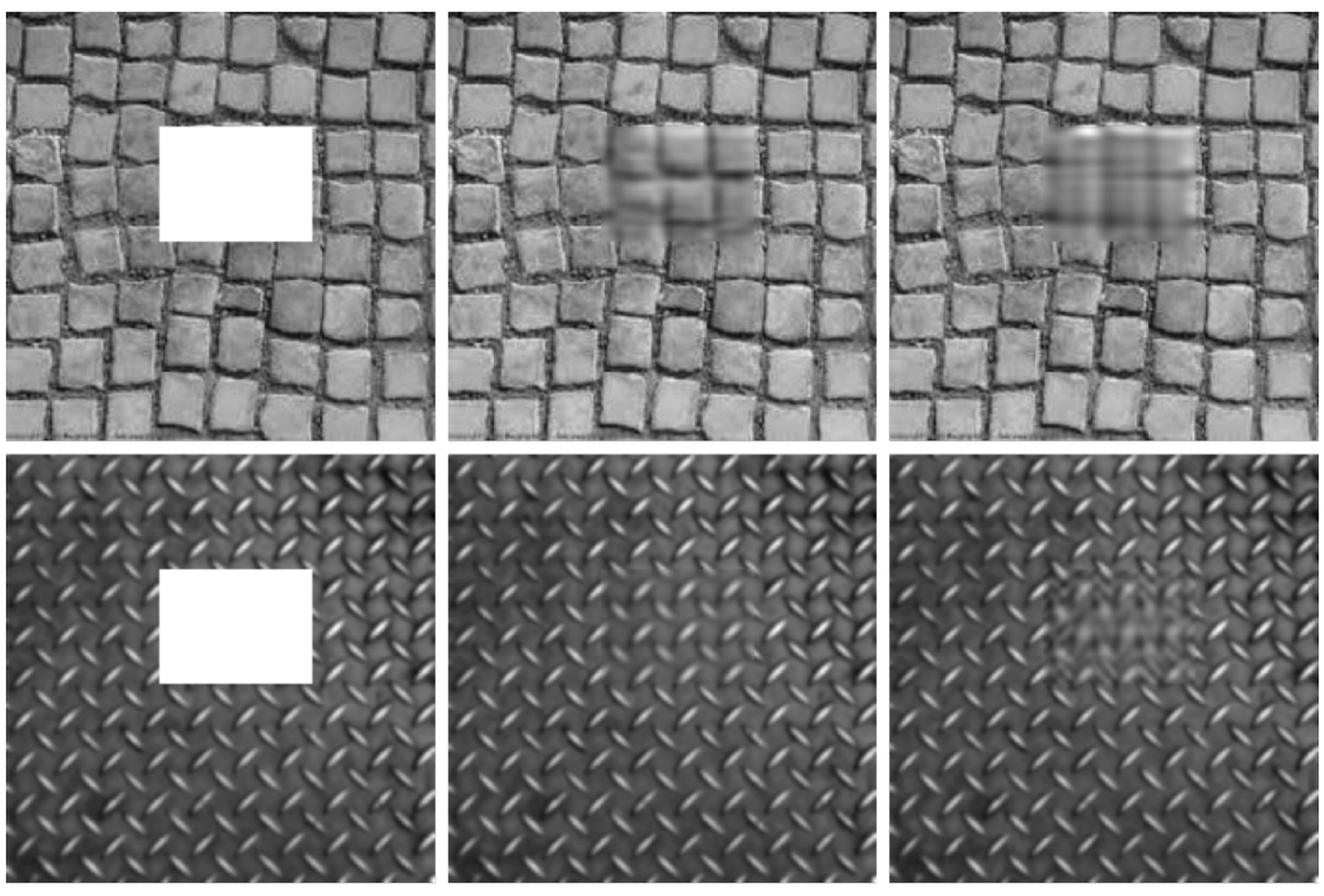
- Extrapolating 2-D patterns



Ground truth



NKN prediction



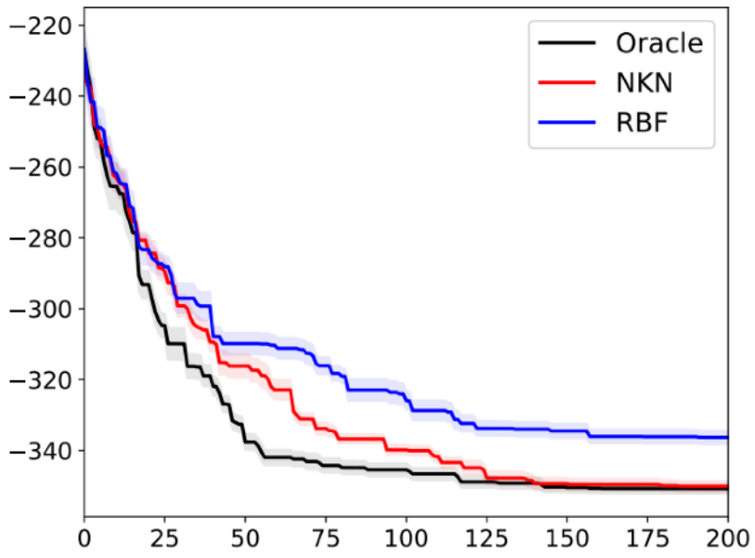
Observation

NKN

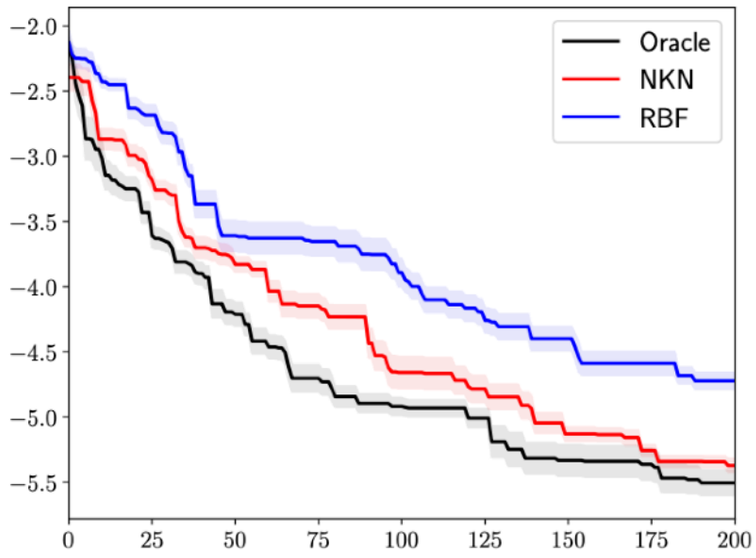
Spectral Mixture
(10 components)

Structured Kernels for Bayes Opt

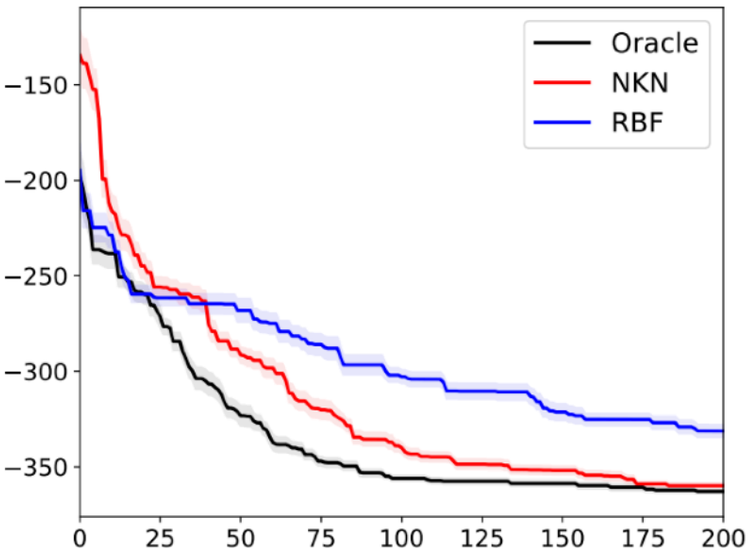
- Structured kernels can help BayesOpt search much faster.
- E.g., if a function is additive, i.e. $f(x_1, \ldots, x_N) = f(x_1) + \cdots + f(x_N)$, then the search is linear rather than exponential. (e.g. Kandasamy et al., 2015)
- BayesOpt with an NKN kernel can learn to make use of additive structure when it exists.



(a) Stybtang



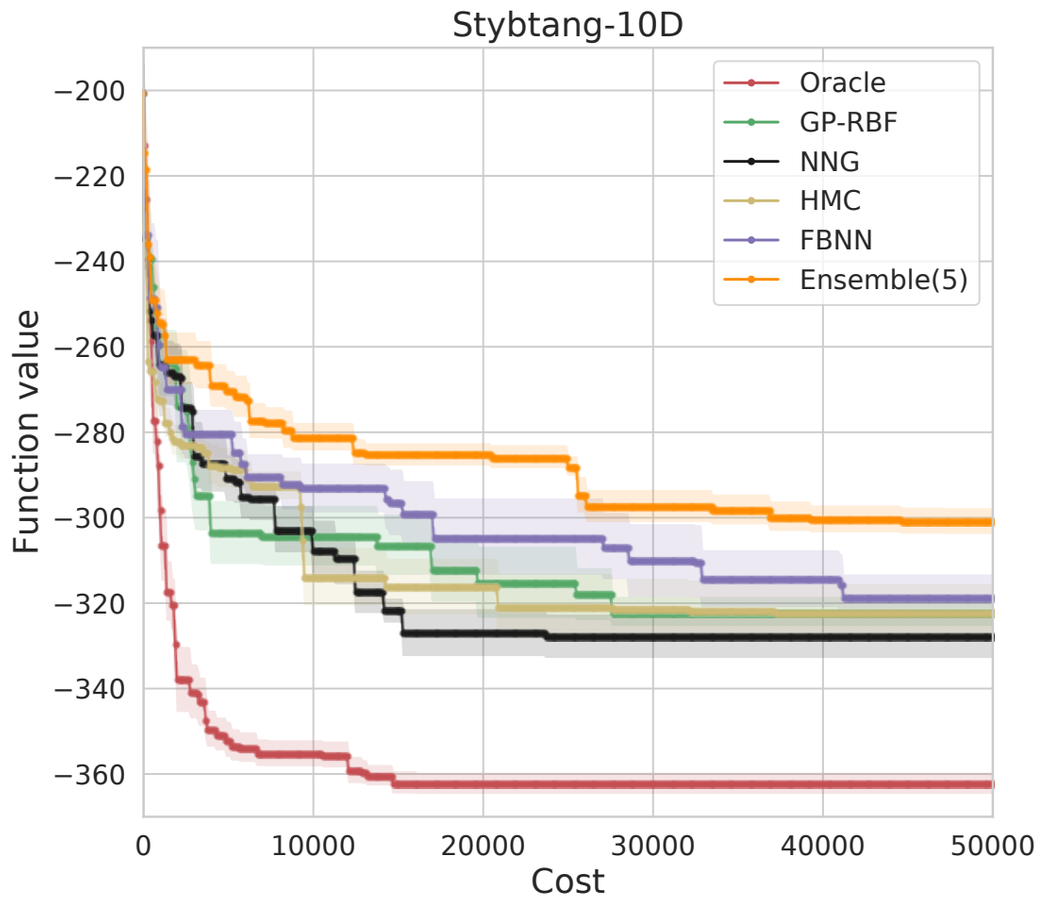
(b) Michalewicz



(c) Stybtang-transform

Structured Kernels for Bayes Opt

- Note: Bayesian neural nets don't achieve this by default.
  - Even though they're good at representing additive functions, they don't seem to enjoy the inductive bias.



Functional Variational BNNs

Shengyang Sun

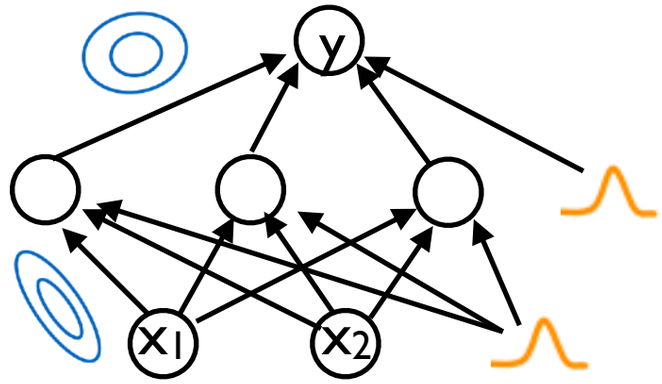
Guodong Zhang

Jiaxin Shi

ICLR 2019

Functional variational BNNs

- Define a stochastic process prior (e.g. a GP)
- Goal: train a generator network to produce functions as close as possible to the stochastic process posterior



- The stochastic weights and units are shared between all input locations. Hence, even the stochastic units represent epistemic, not aleatoric, uncertainty.

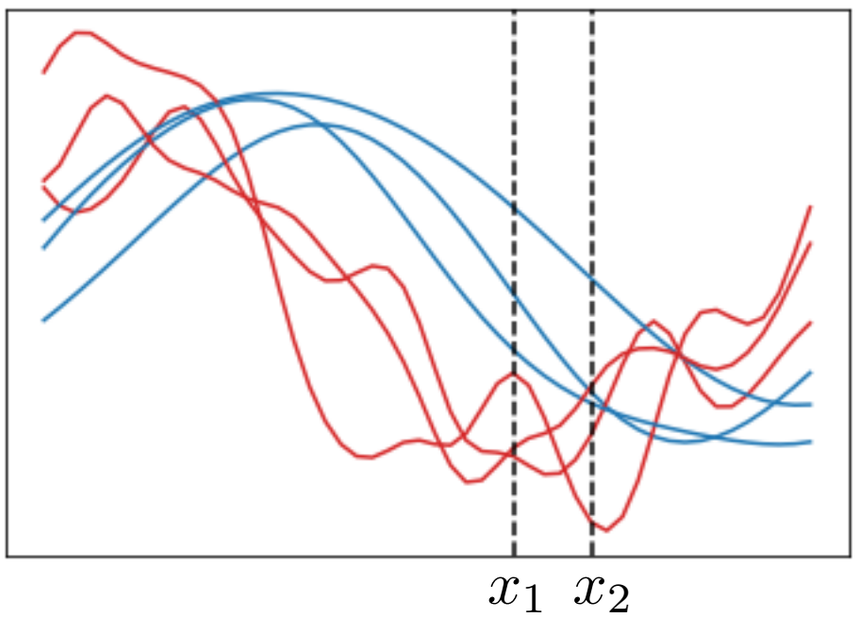
Functional ELBO

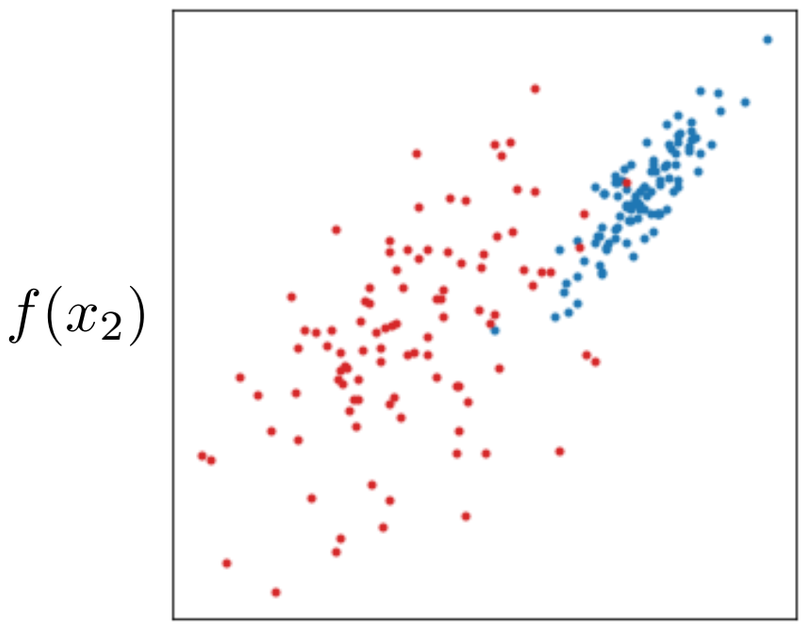
- Perform variational inference over stochastic processes using the functional ELBO (fELBO)

$$\mathcal{L}(q) := \mathbb{E}_q[\log p(\mathbf{y}^D|f)] - \mathrm{KL}[q(f)||p(f)].$$

- p(f) could be a GP, or it could be an implicit prior
- Useful result (based on Kolmogorov Extension Theorem)

$$\mathrm{KL}[P\|Q] = \sup_{n,\,\mathbf{x}_{1:n}} \mathrm{KL}[P_{\mathbf{x}_{1:n}}\|Q_{\mathbf{x}_{1:n}}].$$





Functional ELBO

- Rewrite the fELBO in terms of measurement points

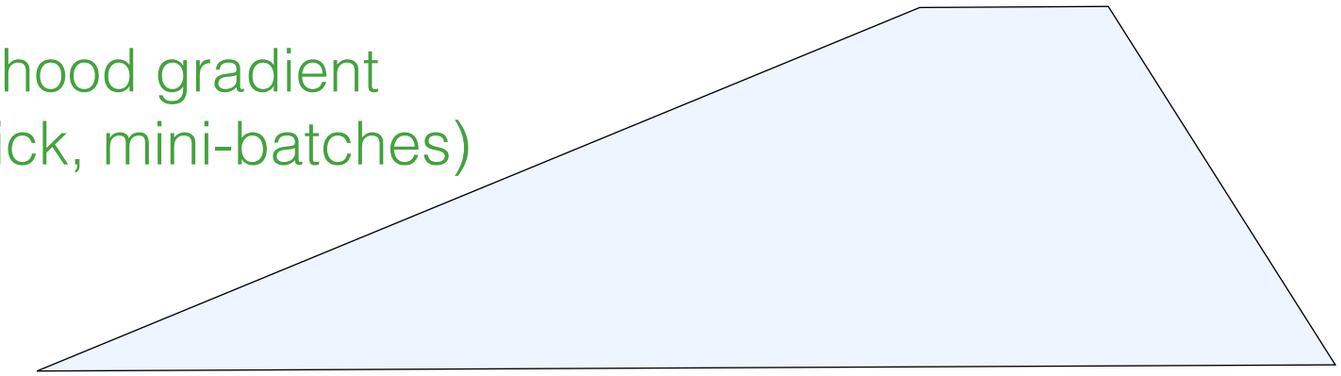
$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_q[\log p(\mathbf{y}^D|f)] - \sup_{\mathbf{X}} \mathrm{KL}[q(\mathbf{f}^{\mathbf{X}})||p(\mathbf{f}^{\mathbf{X}})] \\ &= \inf_{\mathbf{X}} \sum_{(\mathbf{x}^D, y^D)\in\mathcal{D}} \mathbb{E}_q[\log p(y^D|f(\mathbf{x}^D))] - \mathrm{KL}[q(\mathbf{f}^{\mathbf{X}})||p(\mathbf{f}^{\mathbf{X}})] \\ &:= \inf_{\mathbf{X}} \mathcal{L}_{\mathbf{X}}(q).\end{aligned}$$

- In principle, since this is a minmax objective, the measurement points could be chosen by an adversary.
- In practice, measurement points consist of random training examples and random points from the domain.

fELBO gradients

$$\nabla_\phi \mathcal{L}_{\mathbf{X}}(q) = \sum_{i=1}^{N} \nabla_\phi \mathbb{E}_q[\log p(y^{(i)} \mid \mathbf{x}^{(i)}, f)] \quad - \quad \nabla_\phi \mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{f}^{\mathbf{X}}) \,\|\, p(\mathbf{f}^{\mathbf{X}}))$$

log-likelihood gradient
(reparam trick, mini-batches)



$$\mathbb{E}_q\left[\nabla_\phi \log q_\phi(\mathbf{f}^{\mathbf{X}})\right] + \mathbb{E}_\xi\left[\nabla_\phi \mathbf{f}^{\mathbf{X}}(\nabla_{\mathbf{f}} \log q(\mathbf{f}^{\mathbf{X}}) - \nabla_{\mathbf{f}} \log p(\mathbf{f}^{\mathbf{X}}))\right]$$

$= 0$

estimate using spectral Stein gradient estimator (SSGE)

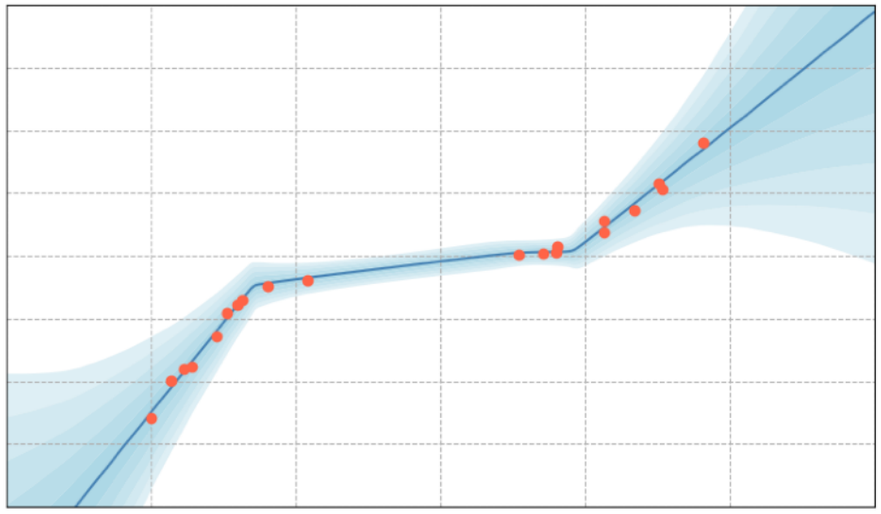
SSGE only works in low-dimensional spaces, but we can choose a small measurement set

handles implicit p (sort of)

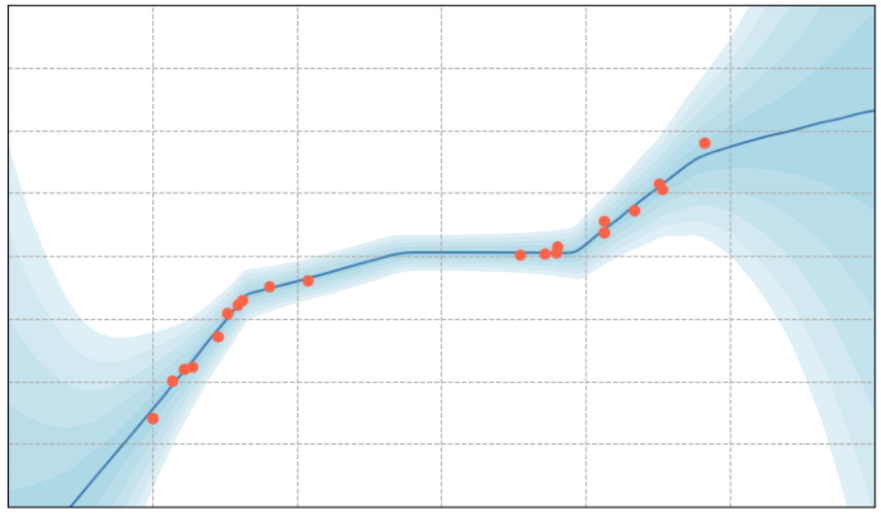
Experiments: $y = x^3$

BNN, variational inference (Bayes By Backprop)

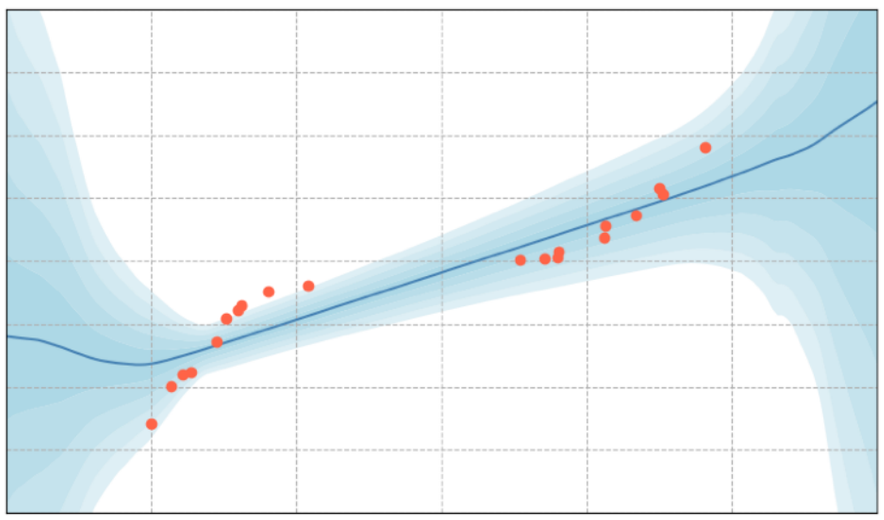
1x100



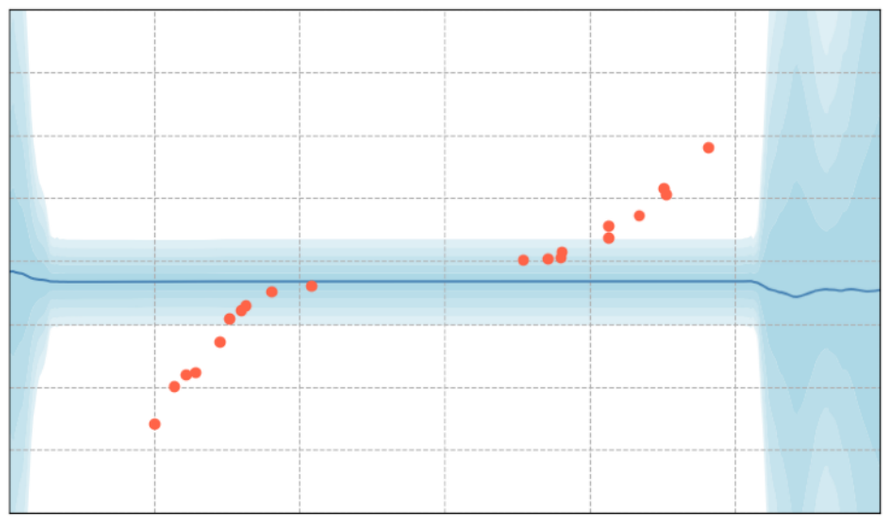
2x100



3x100



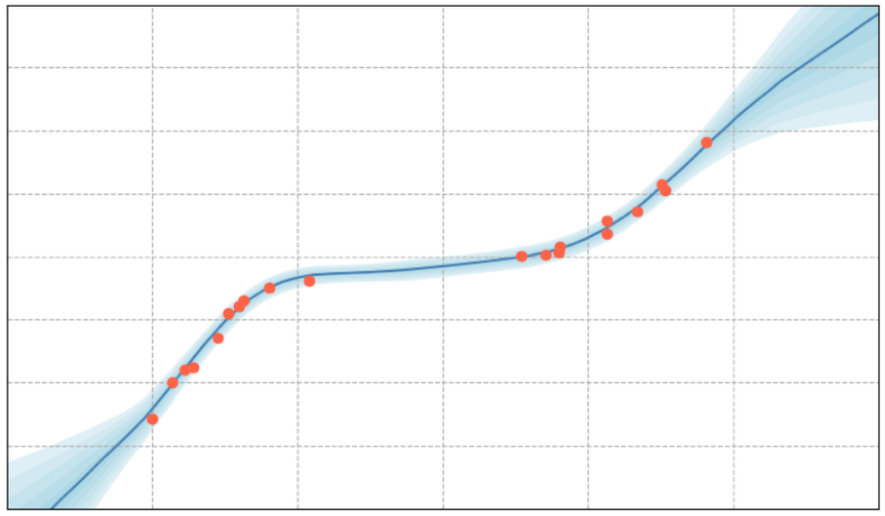
5x100



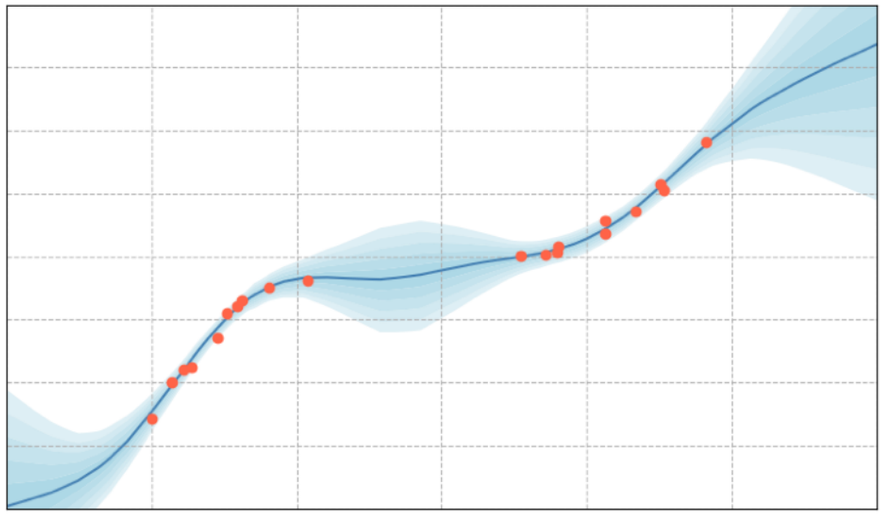
Experiments: $y = x^3$

Functional variational BNN

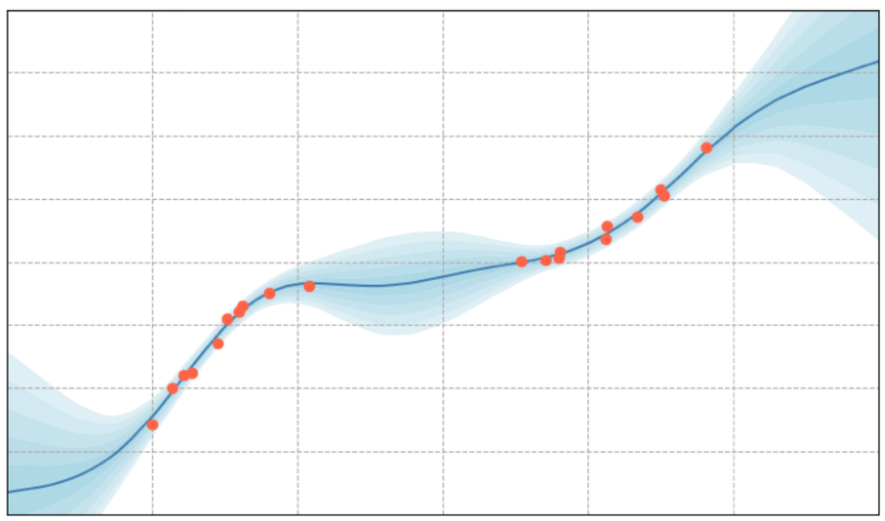
1x100



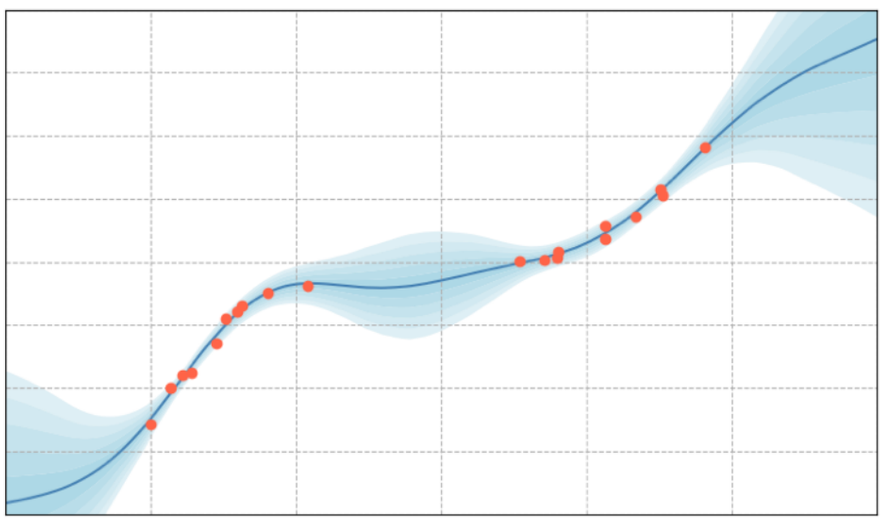
2x100



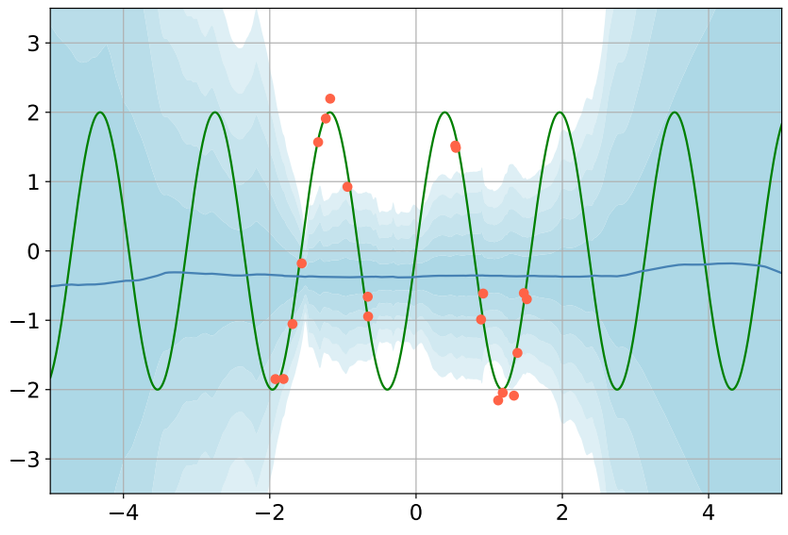
3x100



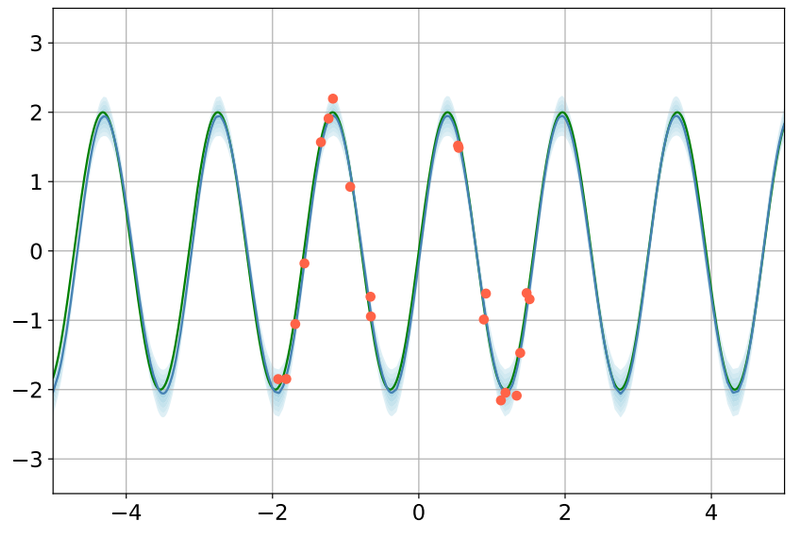
5x100



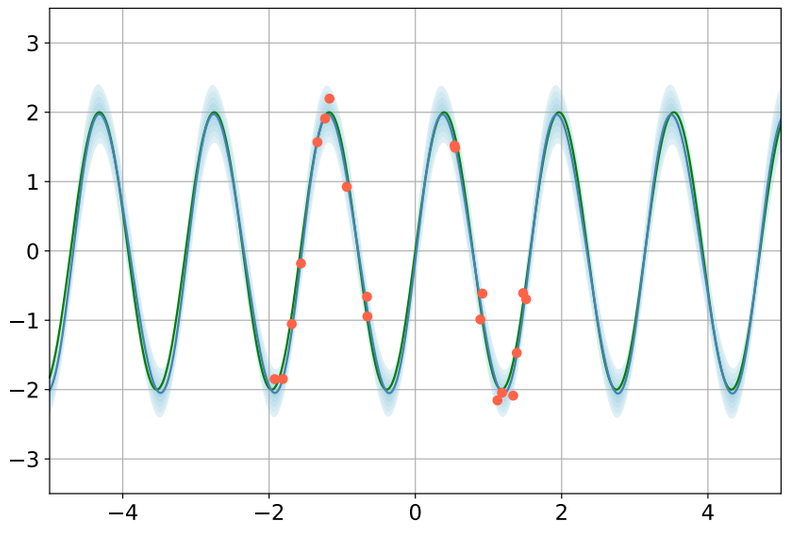
Experiments: periodic function



Variational BNN
(Bayes By Backprop)



fBNN, prior =
GP w/ periodic kernel

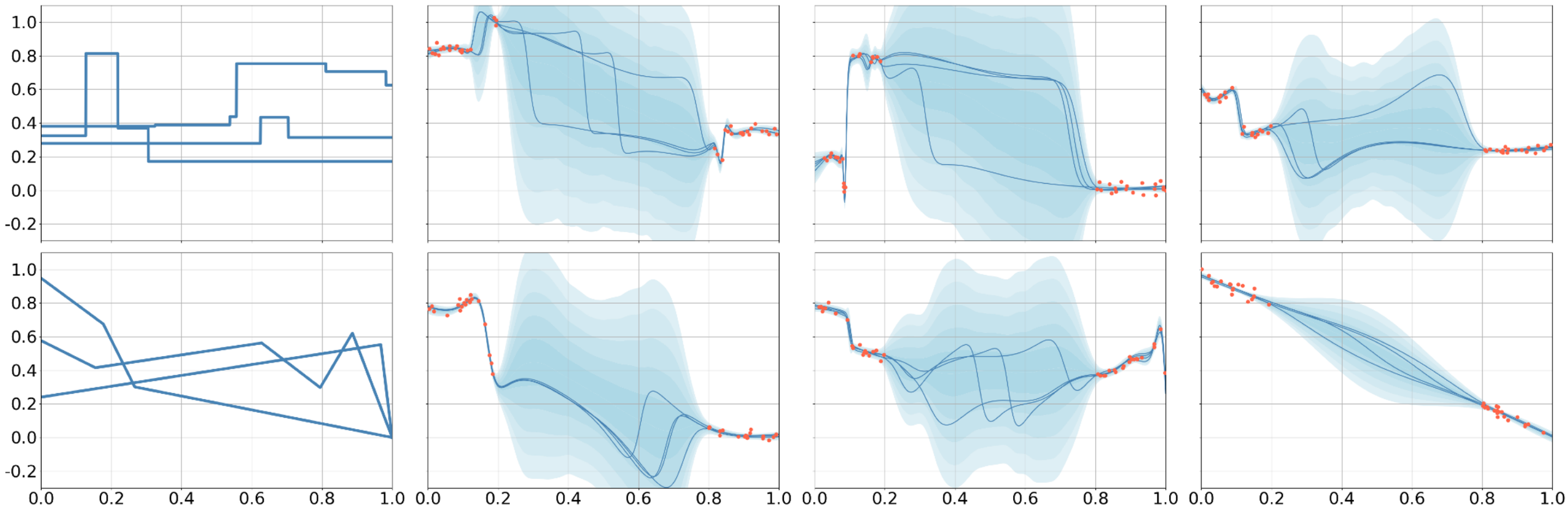


GP w/ periodic kernel

Experiments: implicit priors

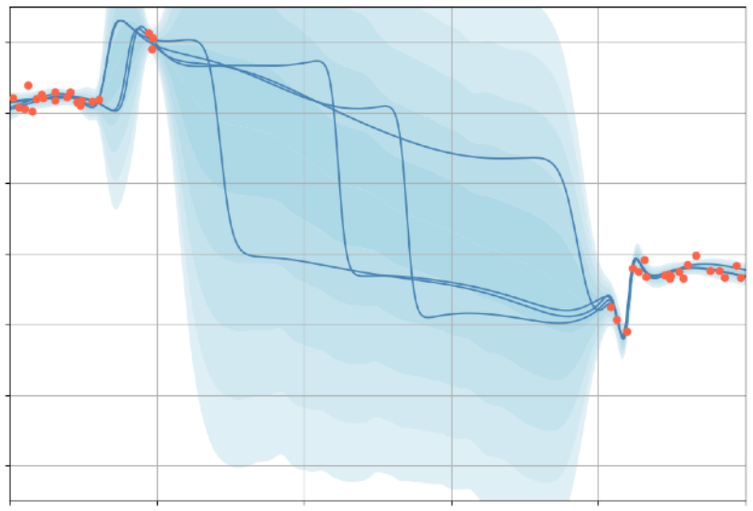
Prior Samples

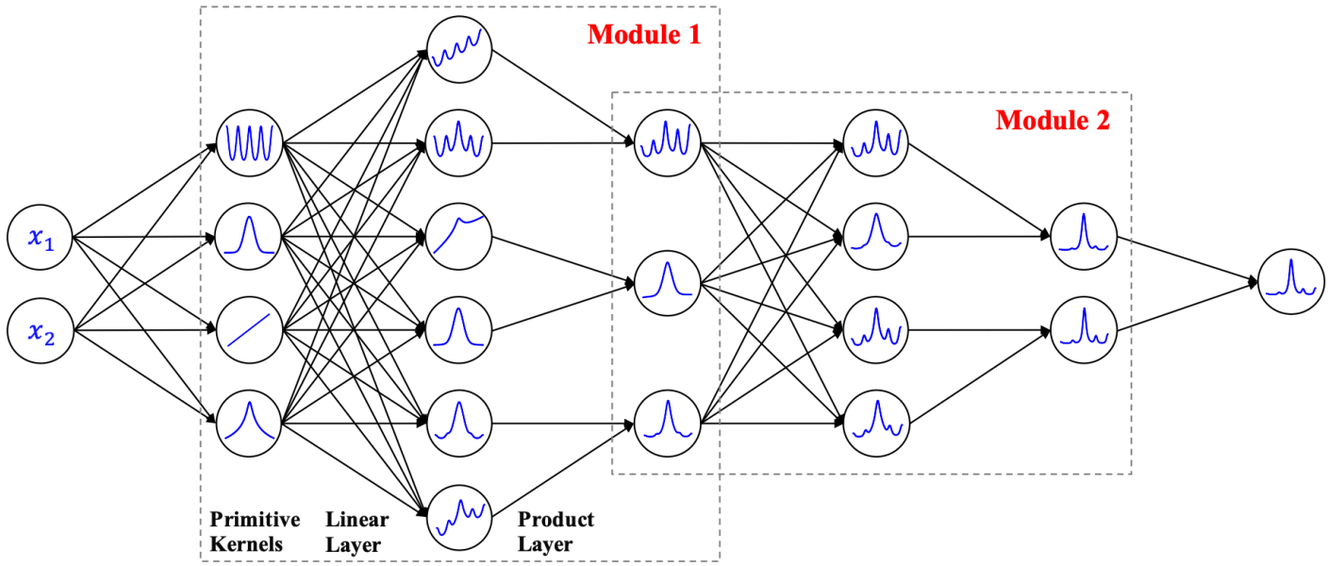
Variational Posterior

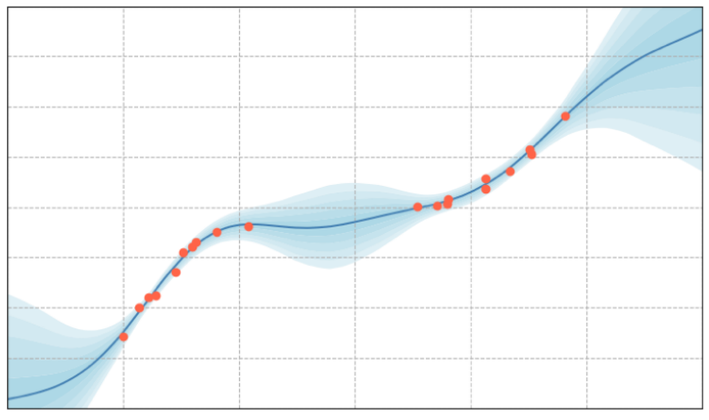


Conclusions

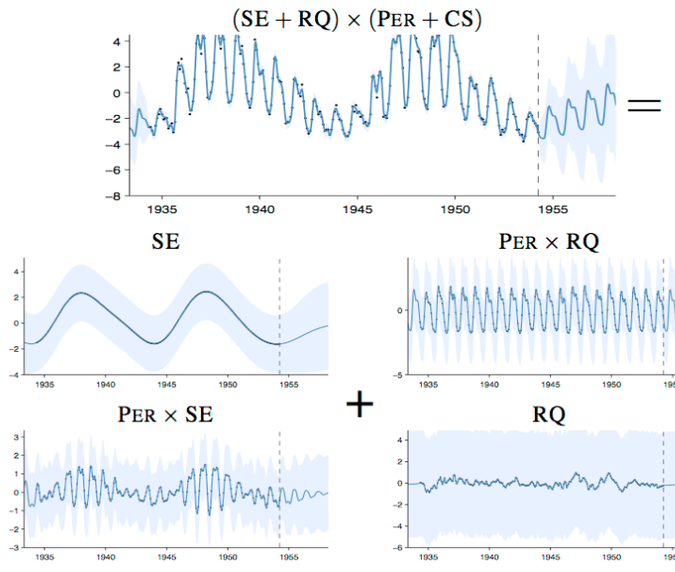
- Specify a stochastic process prior, not a prior over weights
- Perform variational inference over functions, not weight space
- functional ELBO
  - approximate KL term with marginal KL over a measurement set
  - approximate the marginal KL using SSGE
- why not just use a GP?
  - explicit function samples
  - scalability in # examples
  - implicit priors
  - derivative observations?
  - easily compose functional BNNs into larger networks?

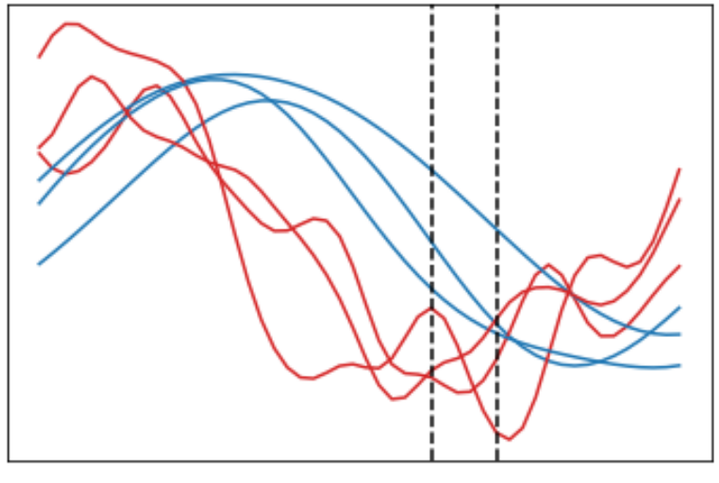


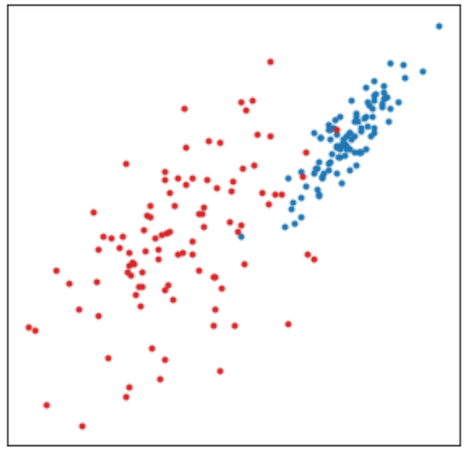

Module 1
Module 2
x_1
x_2
Primitive Kernels
Linear Layer
Product Layer

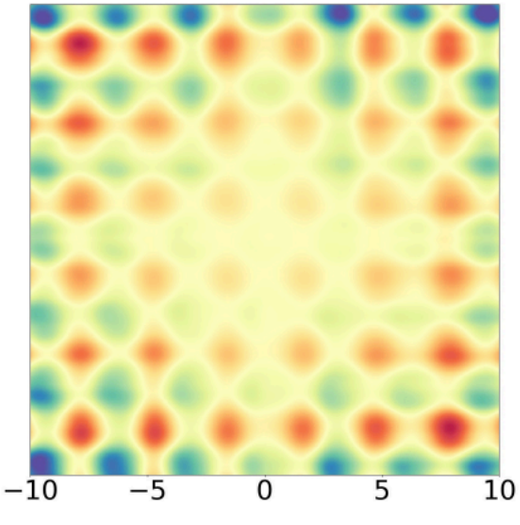


Thank you!


(SE + RQ) × (PER + CS)
=
SE
PER × RQ
PER × SE
+
RQ






−10
−5
0
5
10