Automatic construction and description of nonparametric models

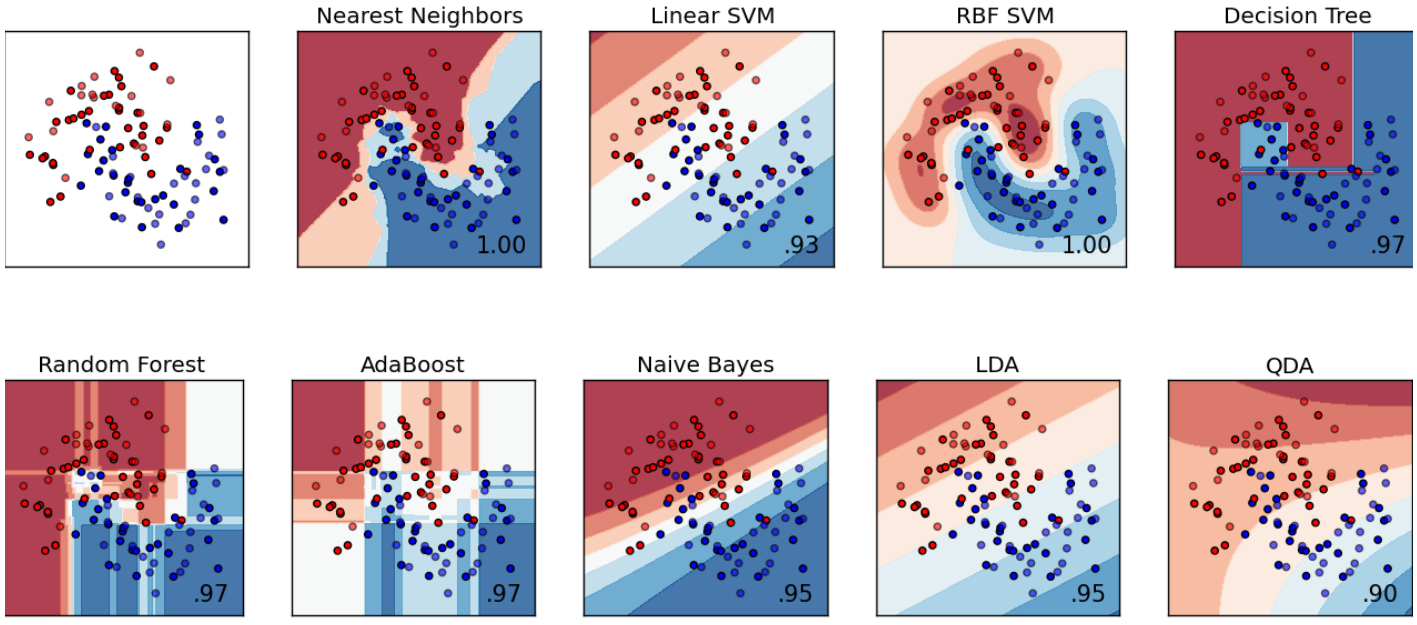


David Duvenaud, James Robert Lloyd, Roger Grosse,
Joshua Tenenbaum, Zoubin Ghahramani

January 8, 2014

TYPICAL STATISTICAL MODELLING

- models typically built by hand, or chosen from a fixed set

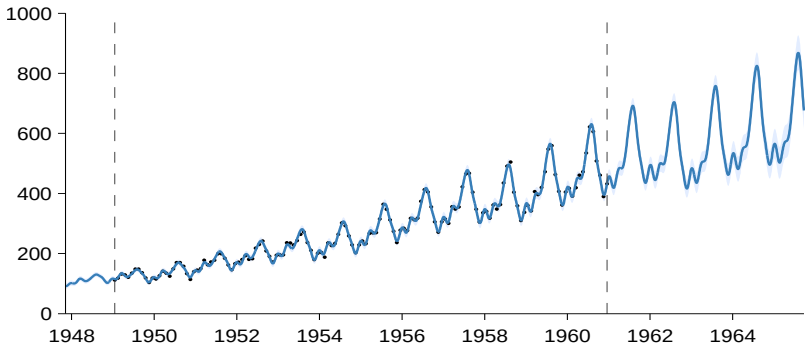


- Building by hand requires considerable expertise
- Just being nonparametric isn't good enough
  - Nonparametric does not mean assumption-free!
- Can silently fail
  - If none of the models tried fit the data well, how can you tell?

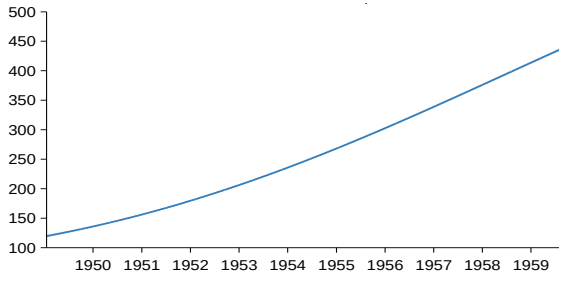
CAN WE DO BETTER?

- Andrew Gelman asks: How can an AI do statistics?
- An artificial statistician would need:
  - a language that could describe arbitrarily complicated models
  - a method of searching over those models
  - a procedure to check model fit
- This talk:We construct such a language over regression models, a procedure to search over it, and a method to describe in natural language the properties of the resulting models
  - Working on automatic model-checking...

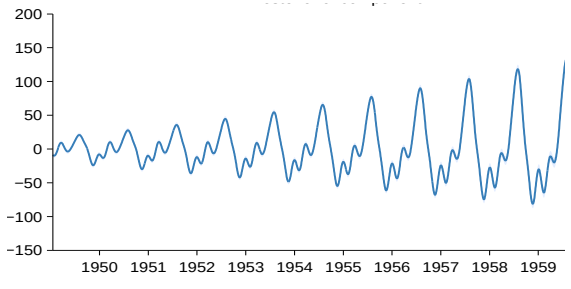
EXAMPLE: AN AUTOMATIC ANALYSIS



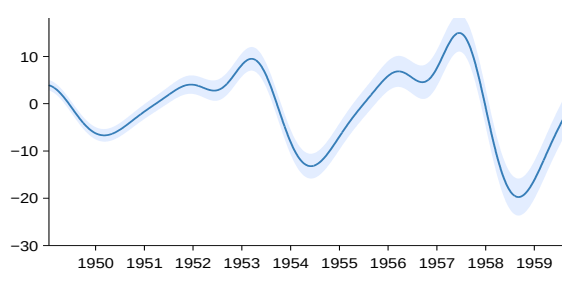
=



+



+



A very smooth, monotonically increasing function

An approximately periodic function with a period of 1.0 years and with approximately linearly increasing amplitude

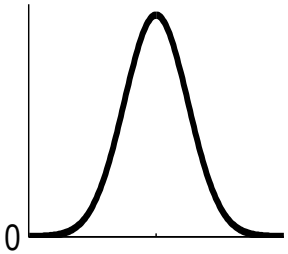
An exactly periodic function with a period of 4.3 years but with linearly increasing amplitude

A LANGUAGE OF REGRESSION MODELS

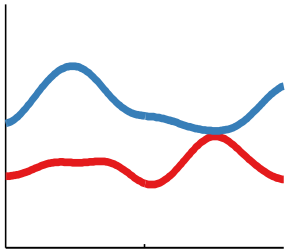
- We define a language of Gaussian process (GP) regression models by defining a language over kernel functions
- We start with a small set of base kernels and create a language with a generative grammar
  - $K \rightarrow K + K$
  - $K \rightarrow K \times K$
  - $K \rightarrow CP(K, K)$
  - $K \rightarrow \{\text{SE}, \text{LIN}, \text{PER}\}$
- The language is open-ended, but its structure makes natural-language description simple

KERNELS DETERMINE STRUCTURE OF GPS

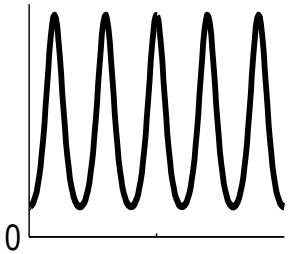
- Kernel determines almost all the properties of a GP prior
- Many different kinds, with very different properties:



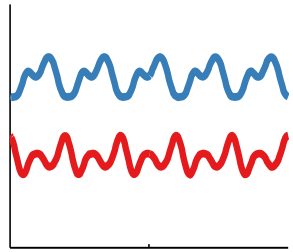
Squared-exp (SE)



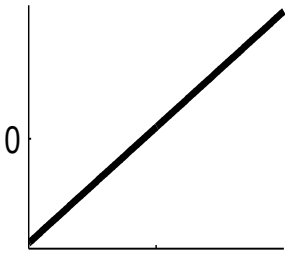
local variation



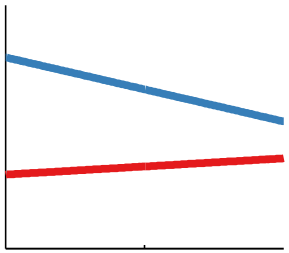
Periodic (PER)



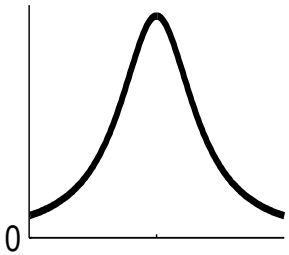
repeating structure



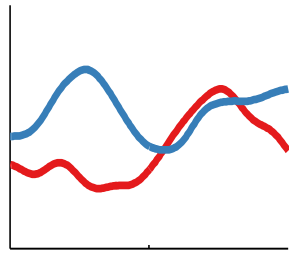
Linear (LIN)



linear functions



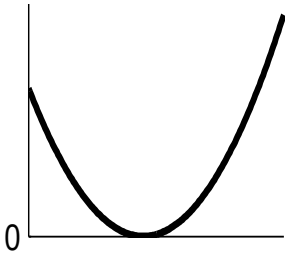
Rational-quadratic(RQ)



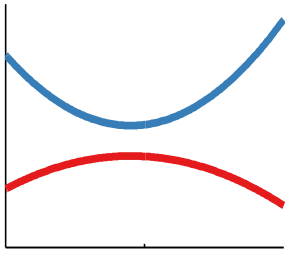
multi-scale variation

KERNELS CAN BE COMPOSED

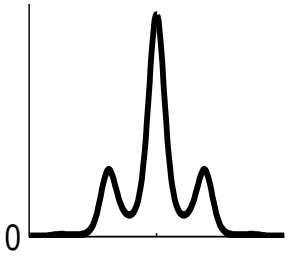
- Two main operations: addition, multiplication



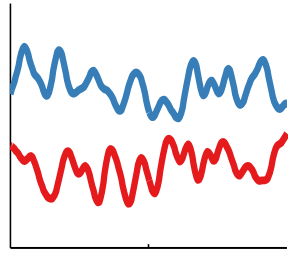
LIN × LIN



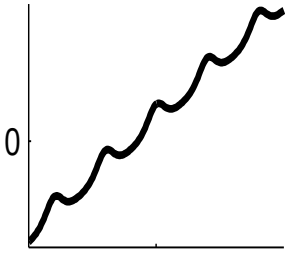
quadratic functions



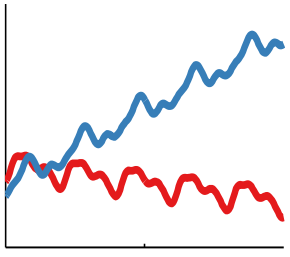
SE × PER



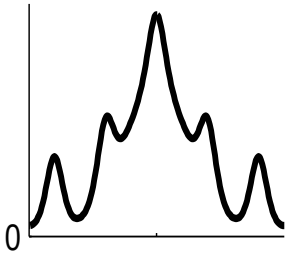
locally periodic



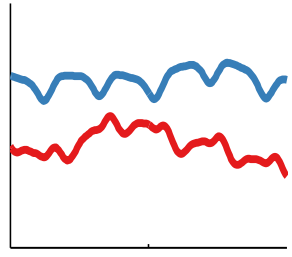
LIN + PER



periodic with trend



SE + PER

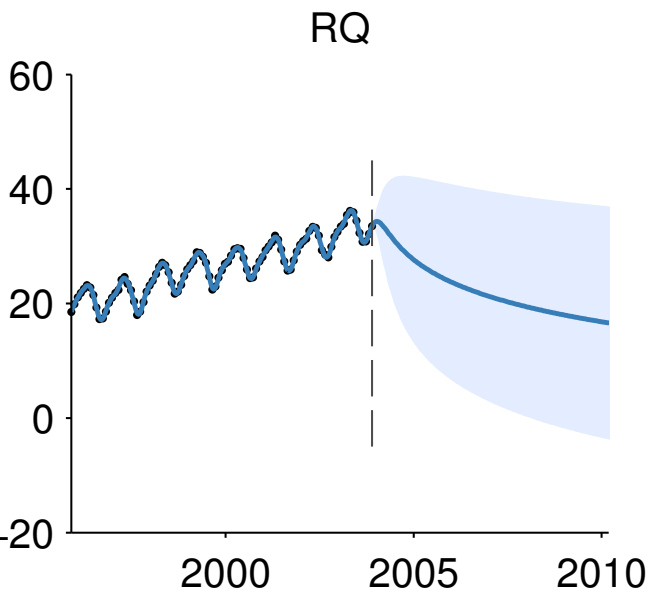


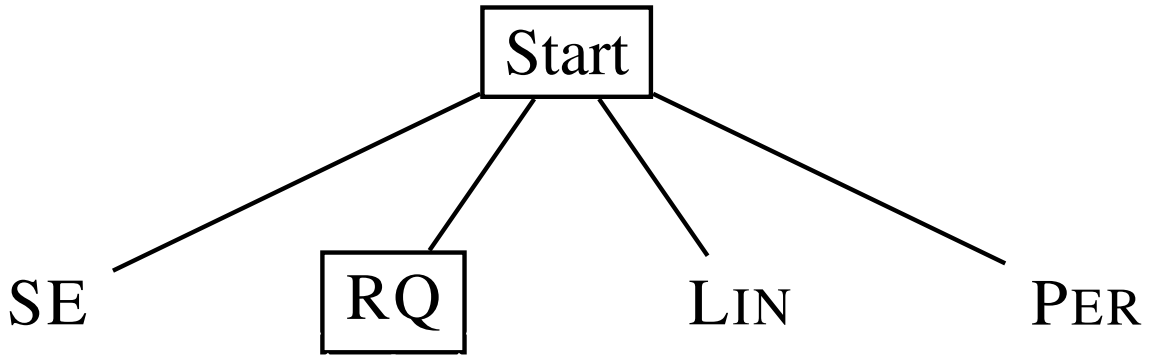
periodic with noise

SPECIAL CASES IN OUR LANGUAGE

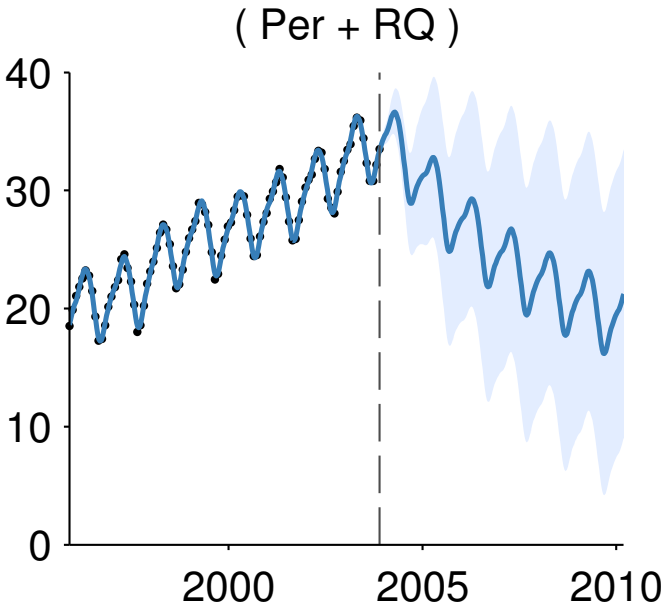
Regression motif	Example kernel
Linear regression	$\textsc{Lin}$
Quadratric regression	$\textsc{Lin} \times \textsc{Lin}$
Fourier analysis	$\sum \cos$
Spectral kernels	$\sum \mathrm{SE} \times \cos$
Changepoints	e.g. $\mathrm{CP}(\textsc{Per}, \mathrm{SE})$
Heteroscedasticity	e.g. $\mathrm{SE} + \textsc{Lin} \times \mathrm{SE}$

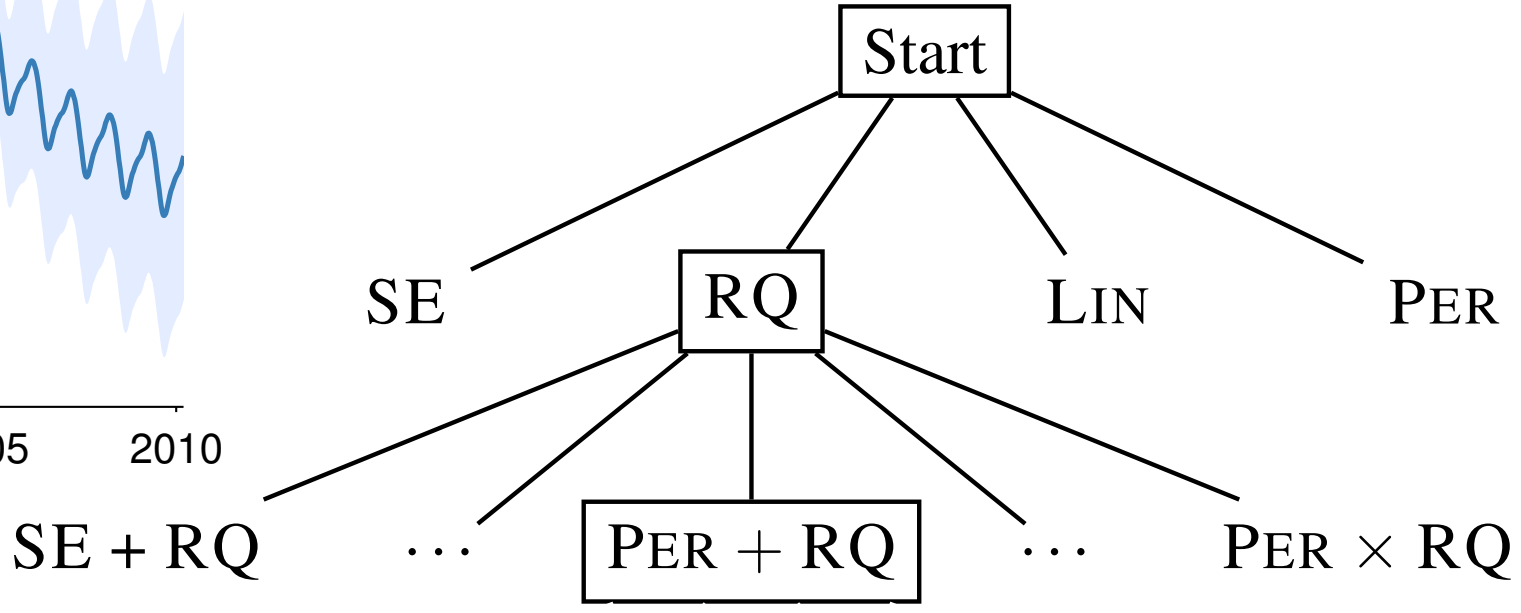
COMPOSITIONAL STRUCTURE SEARCH



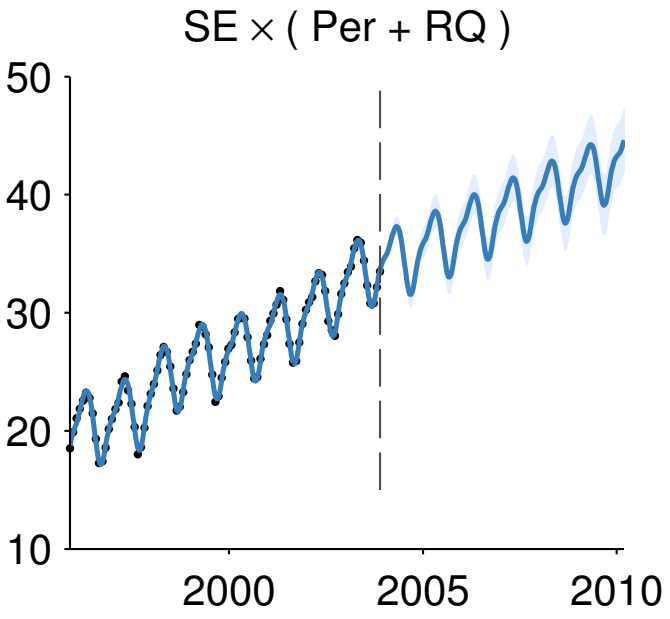


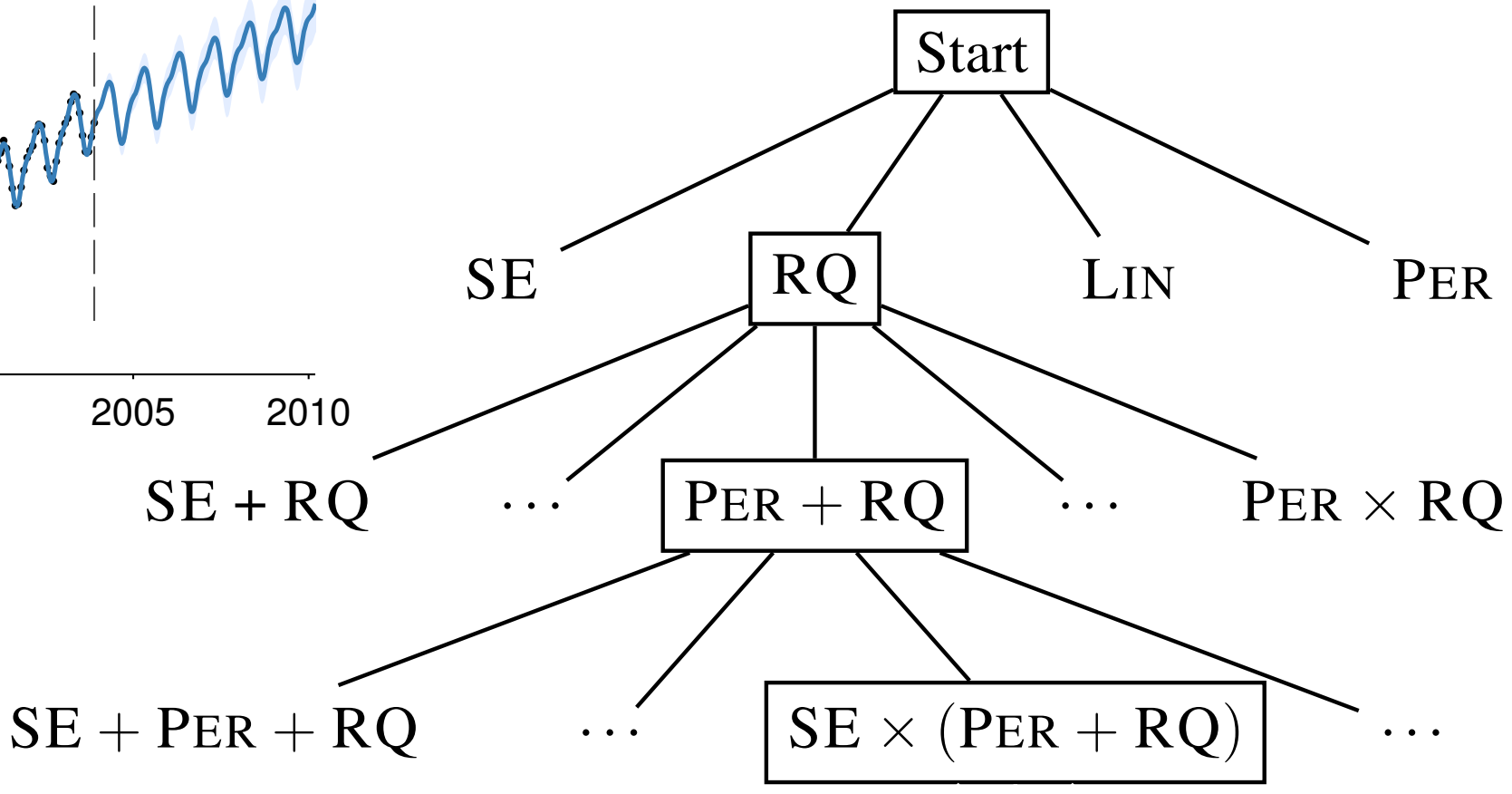
COMPOSITIONAL STRUCTURE SEARCH



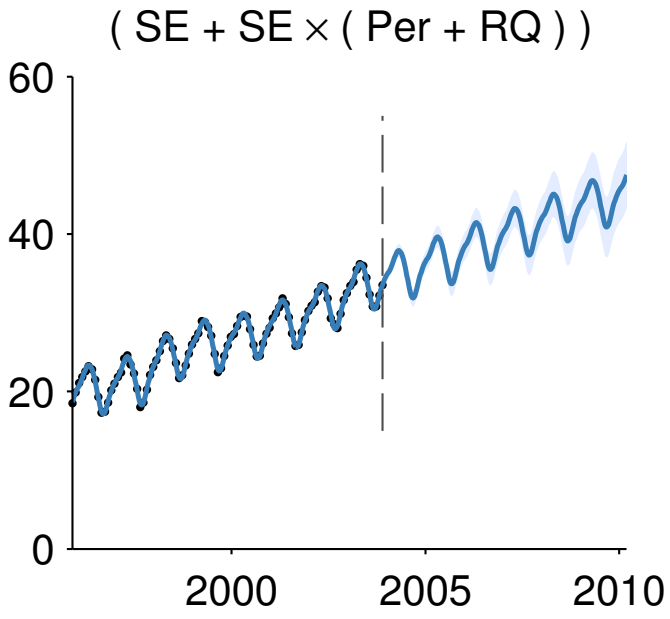


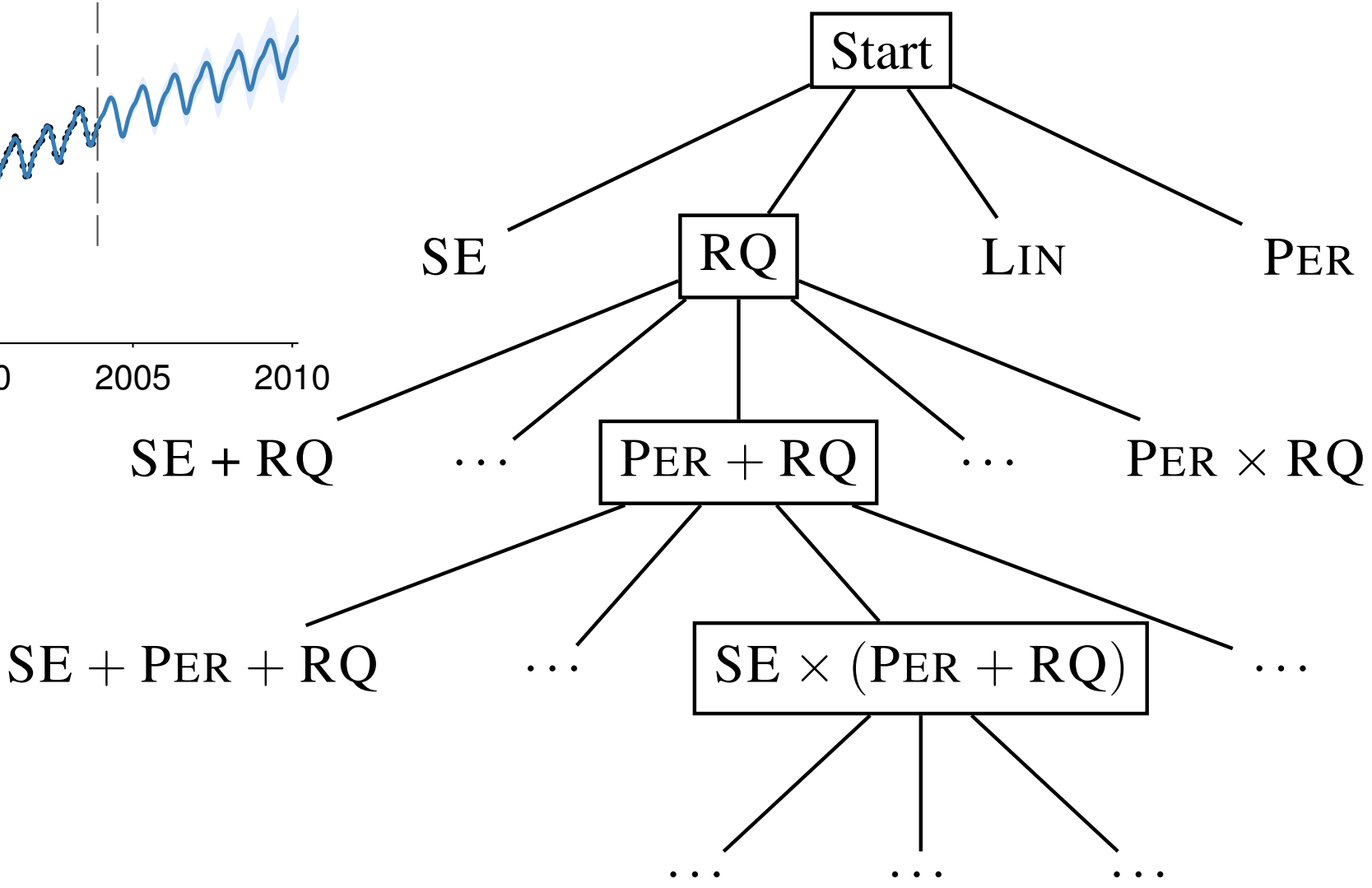
COMPOSITIONAL STRUCTURE SEARCH





COMPOSITIONAL STRUCTURE SEARCH





DISTRIBUTIVITY HELPS INTERPRETABILITY

We can write all kernels as sums of products of base kernels:

$$\text{SE} \times (\text{RQ} + \text{LIN}) = (\text{SE} \times \text{RQ}) + (\text{SE} \times \text{LIN}).$$

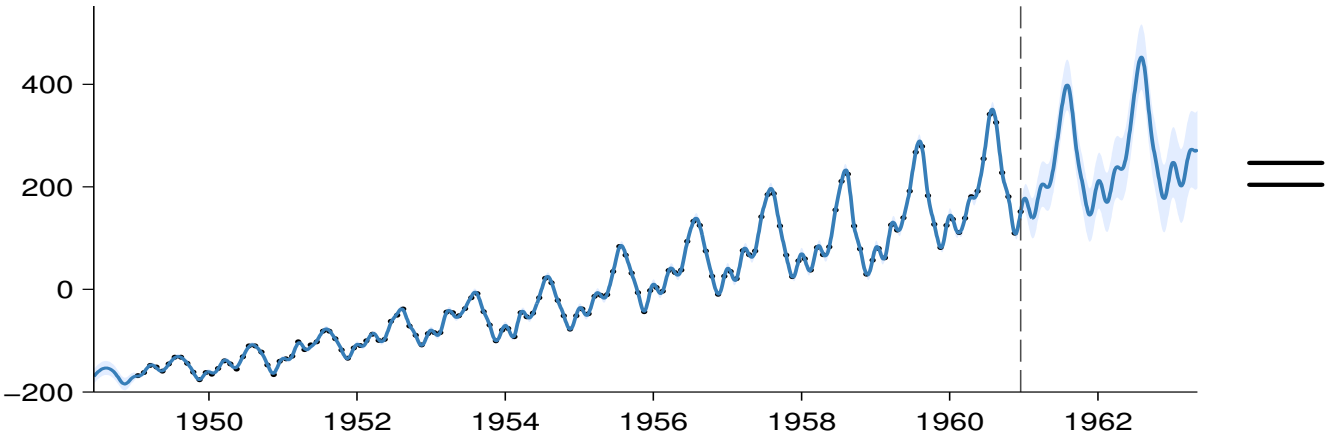
Sums of kernels are equivalent to sums of functions.

If f_1, f_2 are independent, and $f_1 \sim \mathcal{GP}(\mu_1, k_1), f_2 \sim \mathcal{GP}(\mu_2, k_2)$ then

$$(f_1 + f_2) \sim \mathcal{GP}(\mu_1 + \mu_2, k_1 + k_2)$$

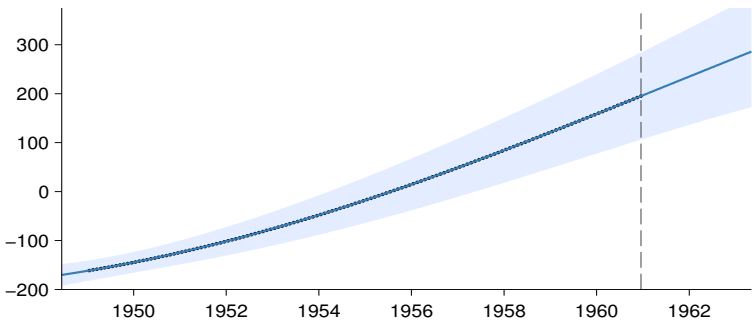
Example Decomposition: Airline

SE × (Lin + Lin × (Per + RQ))

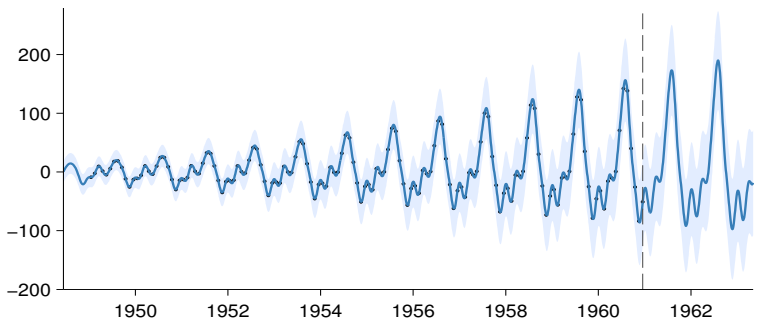


=

SE × Lin

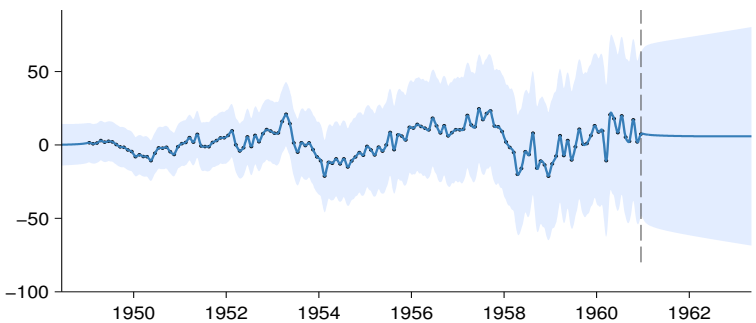


SE × Lin × Per

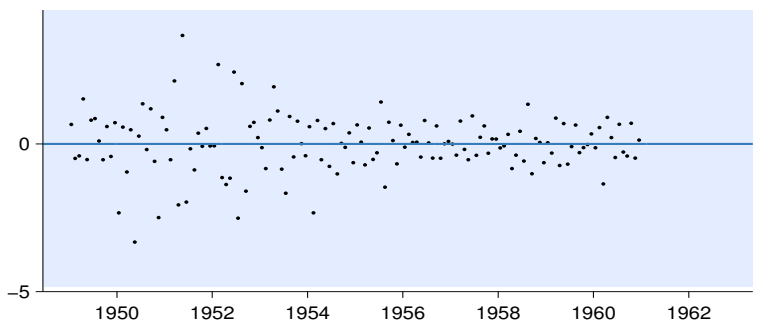


+

SE × Lin × RQ



Residuals



DESCRIBING KERNELS

Products of same type of kernel collapse.

Product of Kernels	Becomes
SE $\times$ SE $\times$ SE ...	SE
LIN $\times$ LIN $\times$ LIN ...	A polynomial
PER $\times$ PER $\times$ PER	Same covariance as product of periodic functions

DESCRIBING KERNELS

Each kernel acts as a modifier in a standard way: an “adjective”.

Kernel	Becomes
$K \times \mathrm{SE}$	’locally’ or ’approximately’
$K \times \mathrm{LIN}$	’with linearly growing amplitude’
$K \times \mathrm{PER}$	’periodic’
$\mathrm{CP}(K1, K2)$	’. . . changing at x to . . . ’

- Special cases for when they’re on their own
- Extra adjectives depending on hyperparameters

EXAMPLE KERNEL DESCRIPTIONS

Product of Kernels	Description
PER	An exactly periodic function
PER $\times$ SE	An approximately periodic function
PER $\times$ SE $\times$ LIN	An approximately periodic function with linearly varying amplitude

THIS ANALYSIS WAS AUTOMATICALLY GENERATED

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

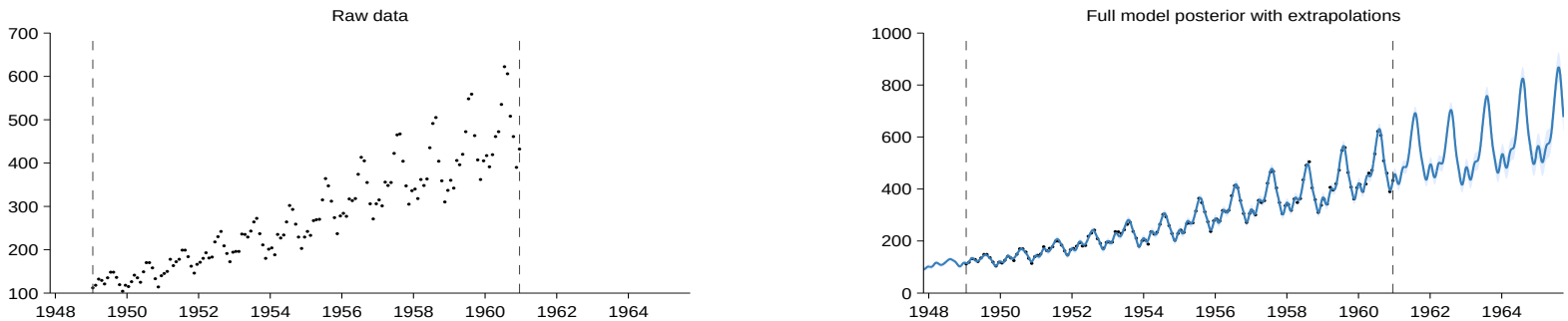


Figure 1: Raw data (left) and model posterior with extrapolation (right)

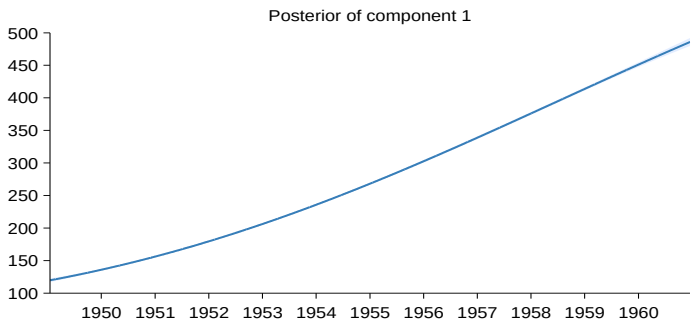
The structure search algorithm has identified four additive components in the data:

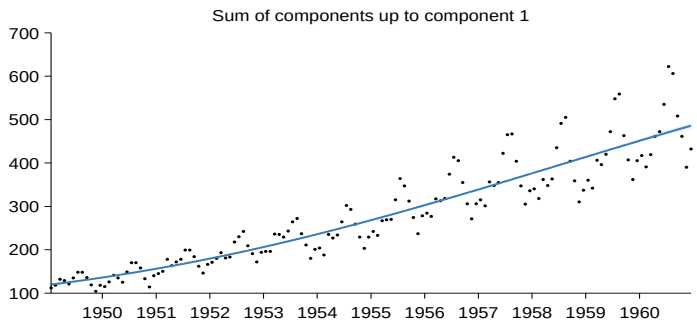
- A very smooth monotonically increasing function.
- An approximately periodic function with a period of 1.0 years and with approximately linearly increasing amplitude.
- An exactly periodic function with a period of 4.3 years but with linearly increasing amplitude.
- Uncorrelated noise with linearly increasing standard deviation.

THIS ANALYSIS WAS AUTOMATICALLY GENERATED

2.1 Component 1

This component is a very smooth and monotonically increasing function.

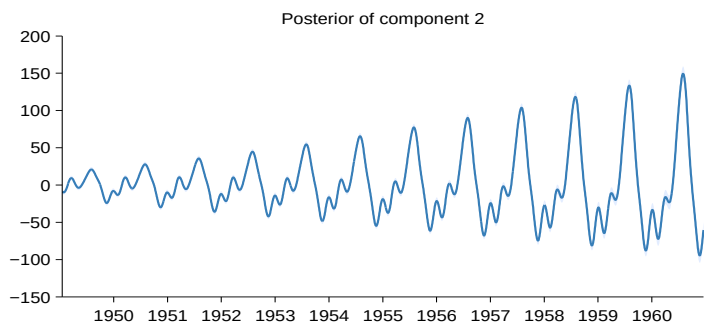


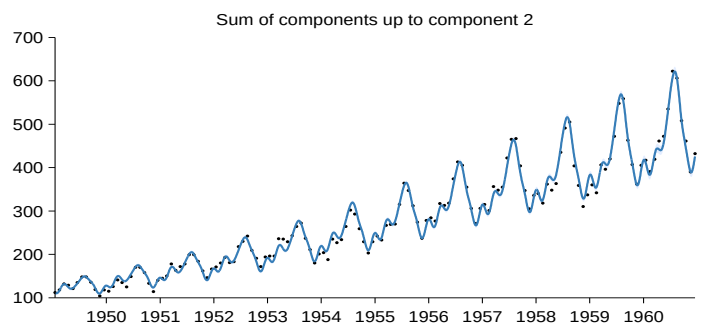


THIS ANALYSIS WAS AUTOMATICALLY GENERATED

2.2 Component 2

This component is approximately periodic with a period of 1.0 years and varying amplitude. Across periods the shape of this function varies very smoothly. The amplitude of the function increases approximately linearly. The shape of this function within each period has a typical lengthscale of 6.0 weeks.

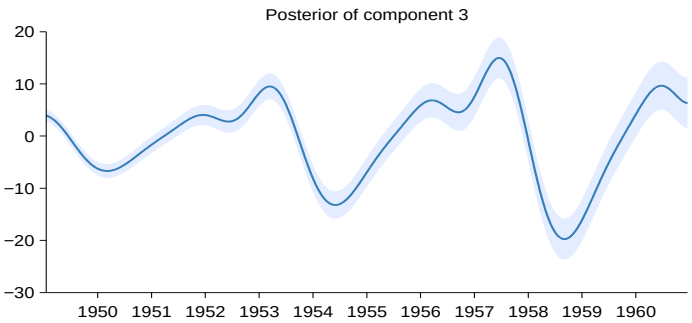


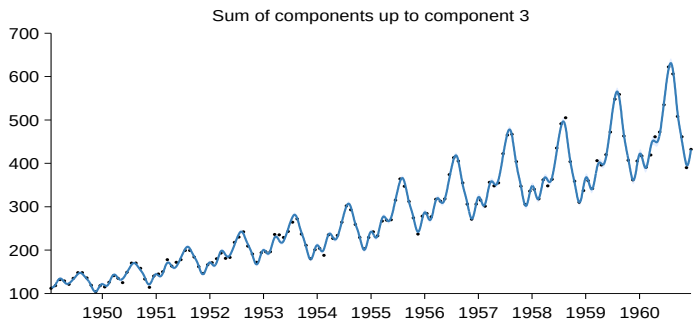


THIS ANALYSIS WAS AUTOMATICALLY GENERATED

2.3 Component 3

This component is exactly periodic with a period of 4.3 years but with varying amplitude. The amplitude of the function increases linearly. The shape of this function within each period has a typical lengthscale of 7.4 months.





SUMMARY

- Constructed a language of regression models through kernel composition
- Searched over this language greedily
- Kernels sums and products modify prior in predictable ways, allowing automatic natural-language description of models
- Open questions:
  - Interpretability versus flexibility
  - Automatic Model-checking

KERNEL LEARNING AS FEATURE LEARNING

- Kernels implicitly compute dot product between features of two items: $k(x, x') = \phi(x)^\mathsf{T}\phi(x')$.
- one-to-one mapping between kernels and feature maps
- feature learning with tractable marginal likelihood!

- Can compose feature maps:
- exaxmple: periodic kernel $k_{per}(x, x') = \exp(-\sin^2(x - x'))$ is equiavelent to $k_{se}(\sin(x), \cos(x), \sin(x'), \cos(x')$.
- What can we do with feature compositions?

Deep Kernels

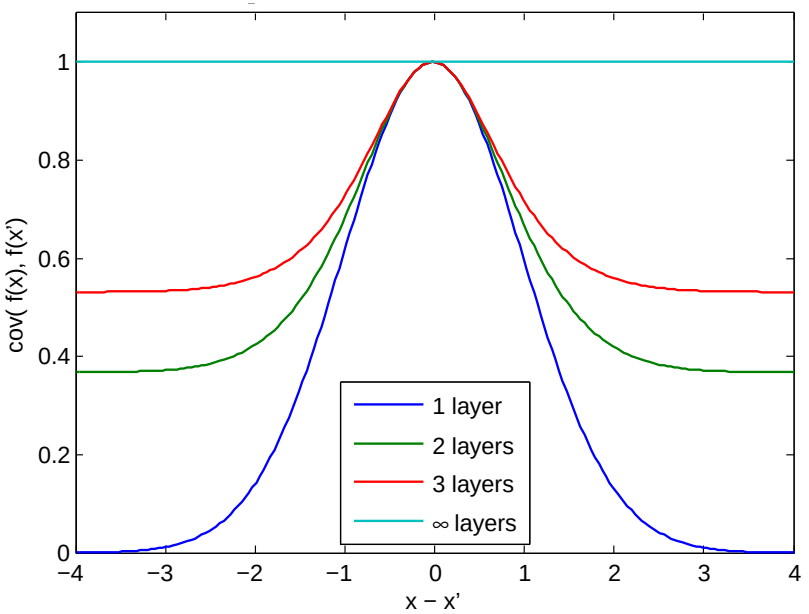
- (Cho, 2012) built kernels from multiple layers of feature mappings.
- given $k_1(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\mathsf{T}\Phi(\mathbf{x}')$, we can consider $k_2(\mathbf{x}, \mathbf{x}') = \Phi(\Phi(\mathbf{x}))^\mathsf{T}\Phi(\Phi(\mathbf{x}'))$. For SE kernel:

$$
\begin{aligned}
& k_2(\mathbf{x}, \mathbf{x}') = \\
& = \exp\left(-\frac{1}{2}||\Phi(\mathbf{x}) - \Phi(\mathbf{x}')||_2^2\right) \\
& = \exp\left(-\frac{1}{2}\left[k_1(\mathbf{x}, \mathbf{x}) - 2k_1(\mathbf{x}, \mathbf{x}') + k_1(\mathbf{x}', \mathbf{x}')\right]\right) \\
& = \exp\left(k_1(\mathbf{x}, \mathbf{x}') - 1\right) \qquad (\text{if } k_1(\mathbf{x}, \mathbf{x}) = 1)
\end{aligned}
$$

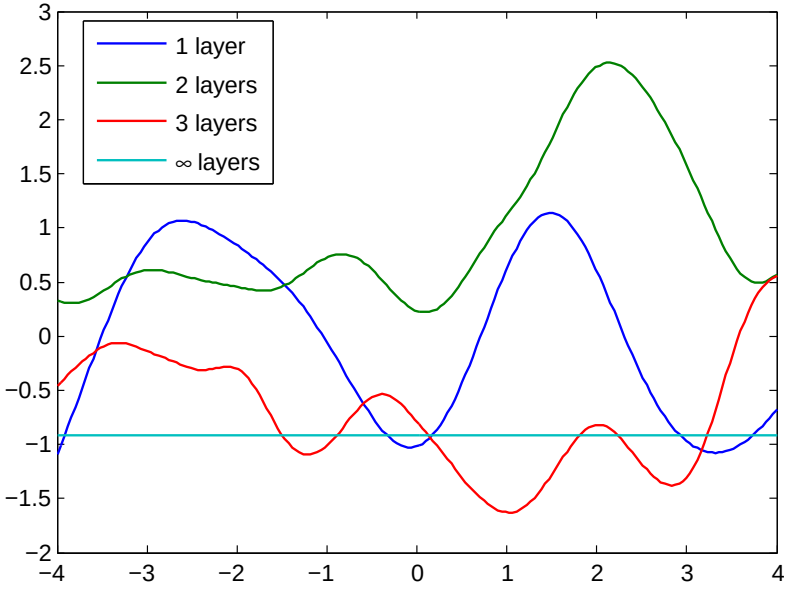
- Why not go deeper?

Infinitely Deep Kernels

- For SE kernel, $k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(k_1(\mathbf{x}, \mathbf{x}') - 1\right)$.
- What is the eventual limit?



Kernel



Draws from GP prior

- $k_\infty(\mathbf{x}, \mathbf{x}') = 1$ everywhere.

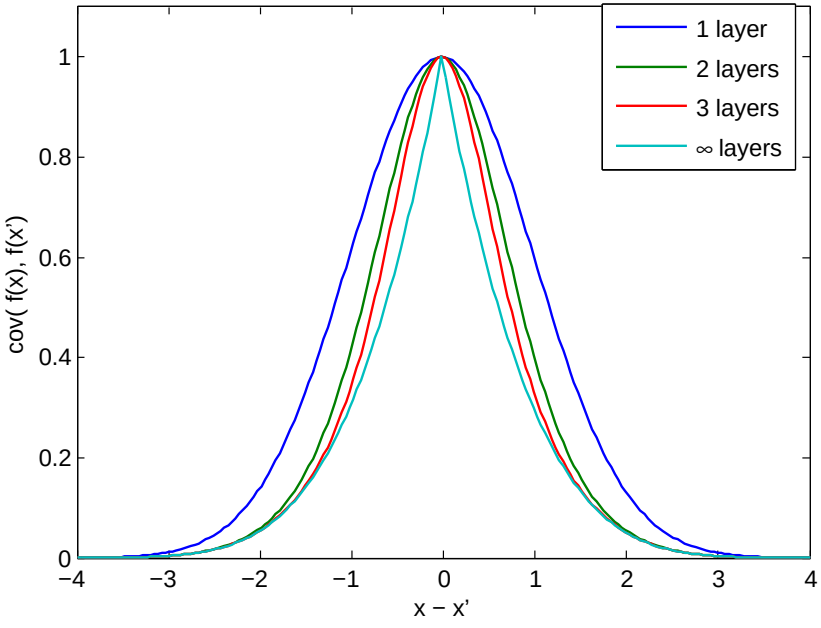
A SIMPLE FIX...

- Following a suggestion from (Neal, 1995), we connect the inputs $\mathbf{x}$ to each layer. (append x to features at each layer:

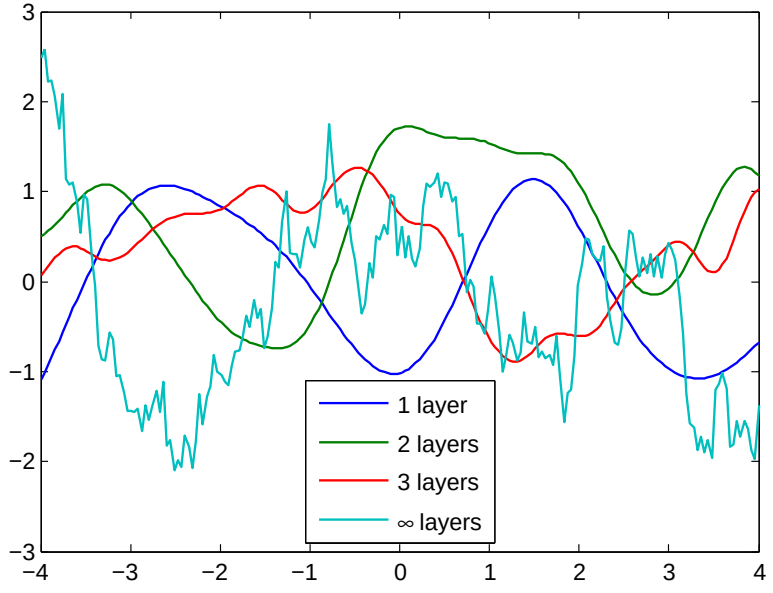
$$
\begin{aligned}
& k_{n+1}(\mathbf{x}, \mathbf{x}') = \\
& = \exp\left(-\frac{1}{2}\left|\left|\begin{bmatrix}\Phi_n(\mathbf{x}) \\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\Phi_n(\mathbf{x}') \\ \mathbf{x}'\end{bmatrix}\right|\right|_2^2\right) \\
& = \exp\left(k_n(\mathbf{x}, \mathbf{x}') - 1 - \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2\right) \qquad (1)
\end{aligned}
$$

Infinitely Deep Kernels

- $k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(k_n(\mathbf{x}, \mathbf{x}') - 1 - \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2\right)$.
- What is the eventual limit?



Kernel



Draws from GP prior

- Looks like an OU process with skinny tails, samples are non-differentiable (fractal).

What went wrong?

- Fixed feature mappings
- not capturing invariances
- not throwing away unnecessary information
- an illustration of why learning is usually necessary

Deep Gaussian Processes

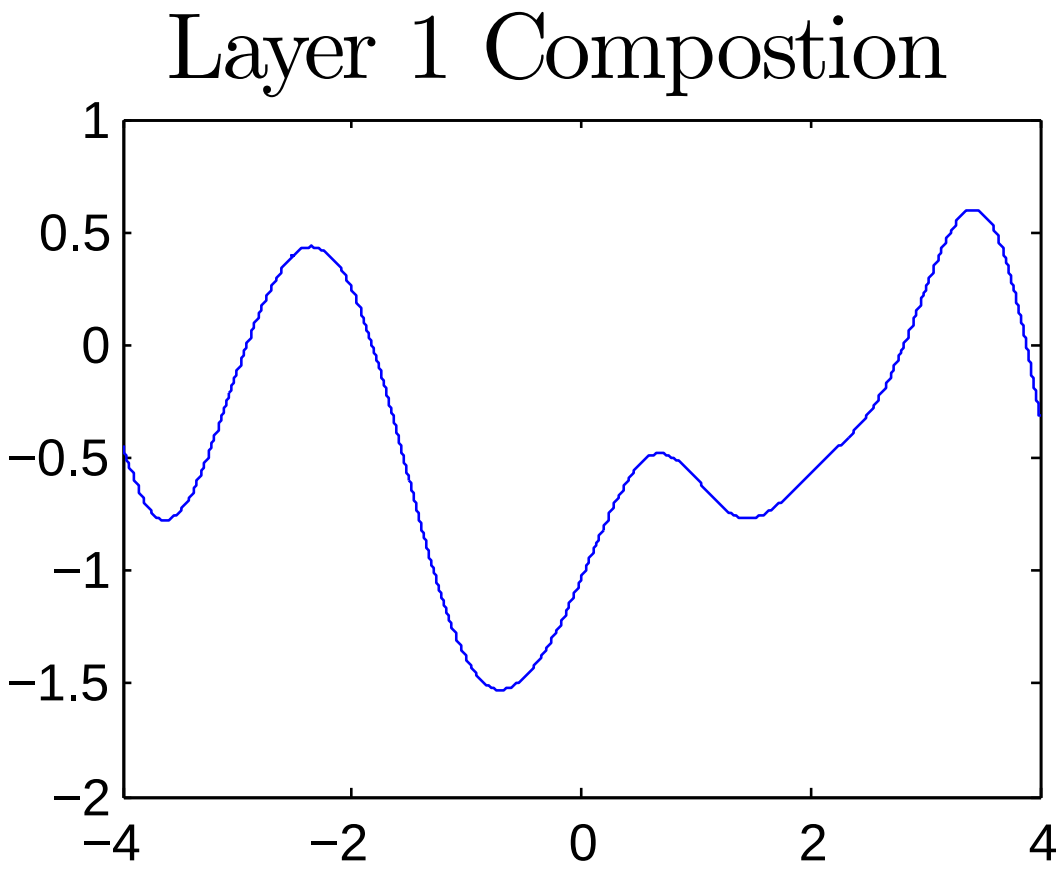
- a prior over compositions of functions:

$$\mathbf{f}^{(1:L)}(\mathbf{x}) = \mathbf{f}^{(L)}(\mathbf{f}^{(L-1)}(\dots \mathbf{f}^{(2)}(\mathbf{f}^{(1)}(\mathbf{x}))\dots)) \tag{2}$$

  where each $\mathbf{f}_d^{(\ell)} \overset{\text{ind}}{\sim} \mathcal{GP}\left(0, k_d^{\ell}(\mathbf{x}, \mathbf{x}')\right)$.
- Can be seen as a certain limit of infinitely-wide deep nets.
- inference is really hard.
- maybe we can learn something about deep models just from looking at prior draws?

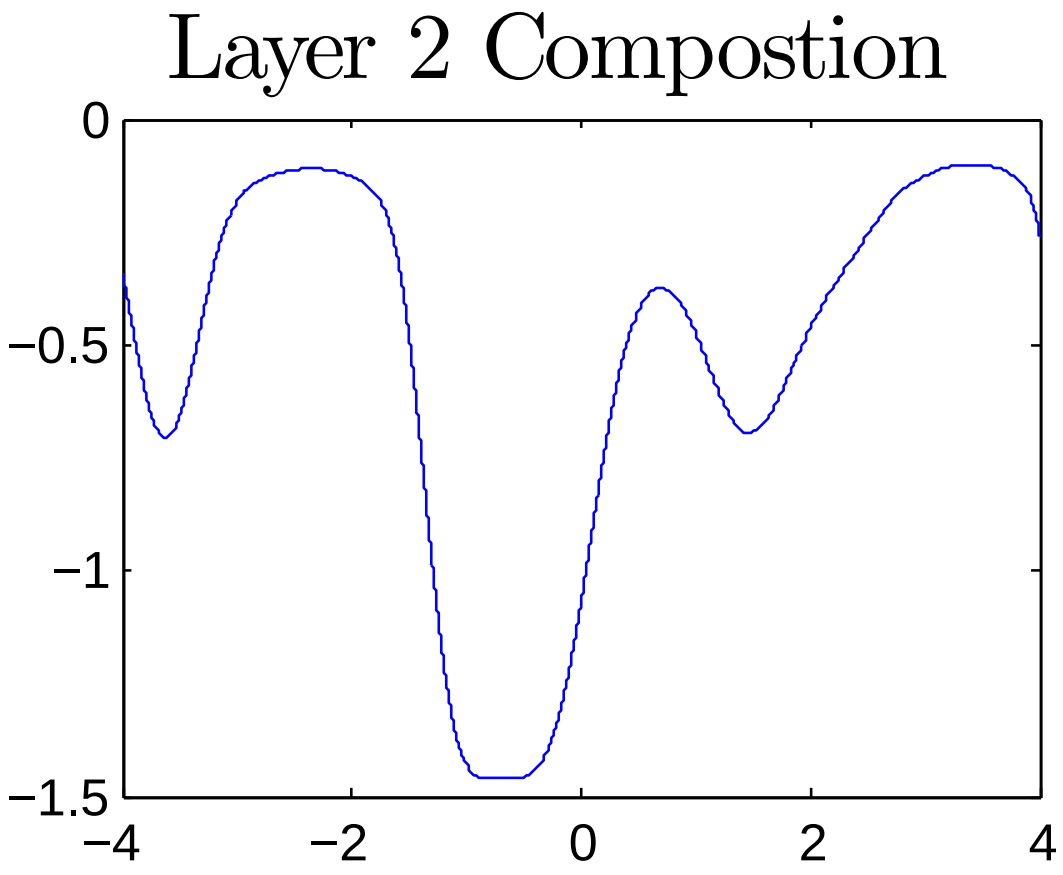
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



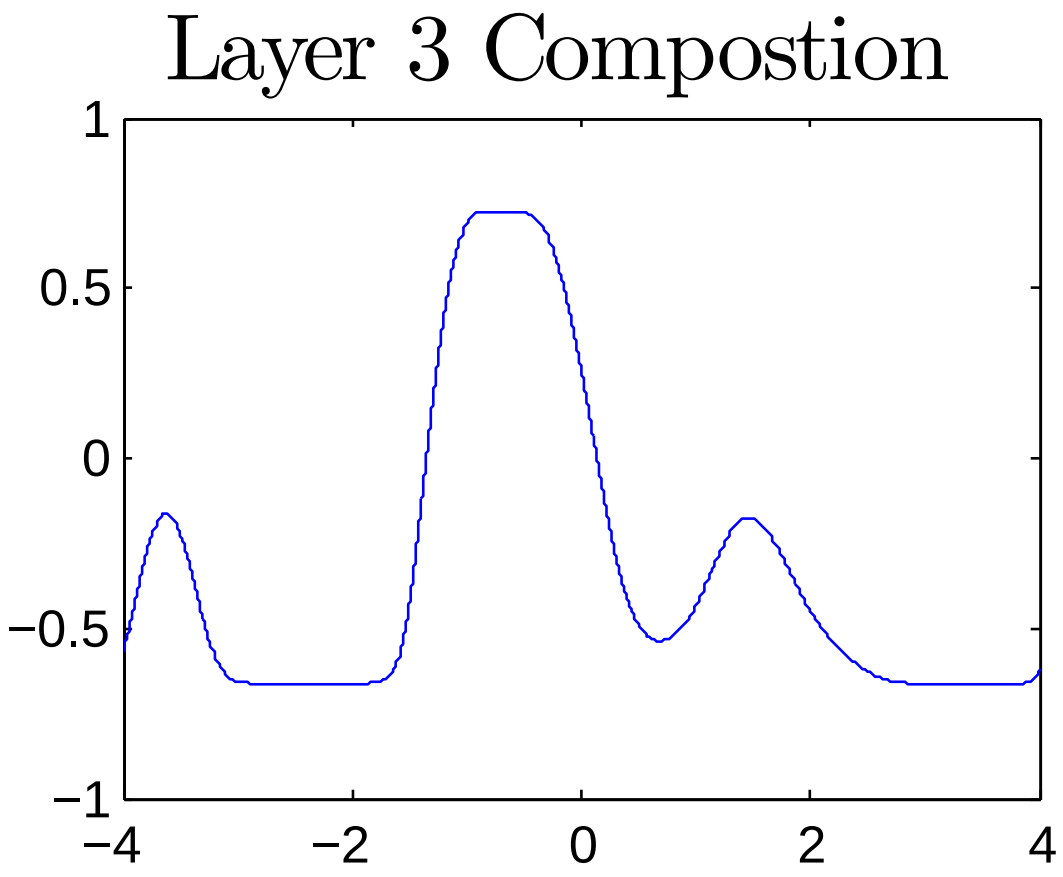
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



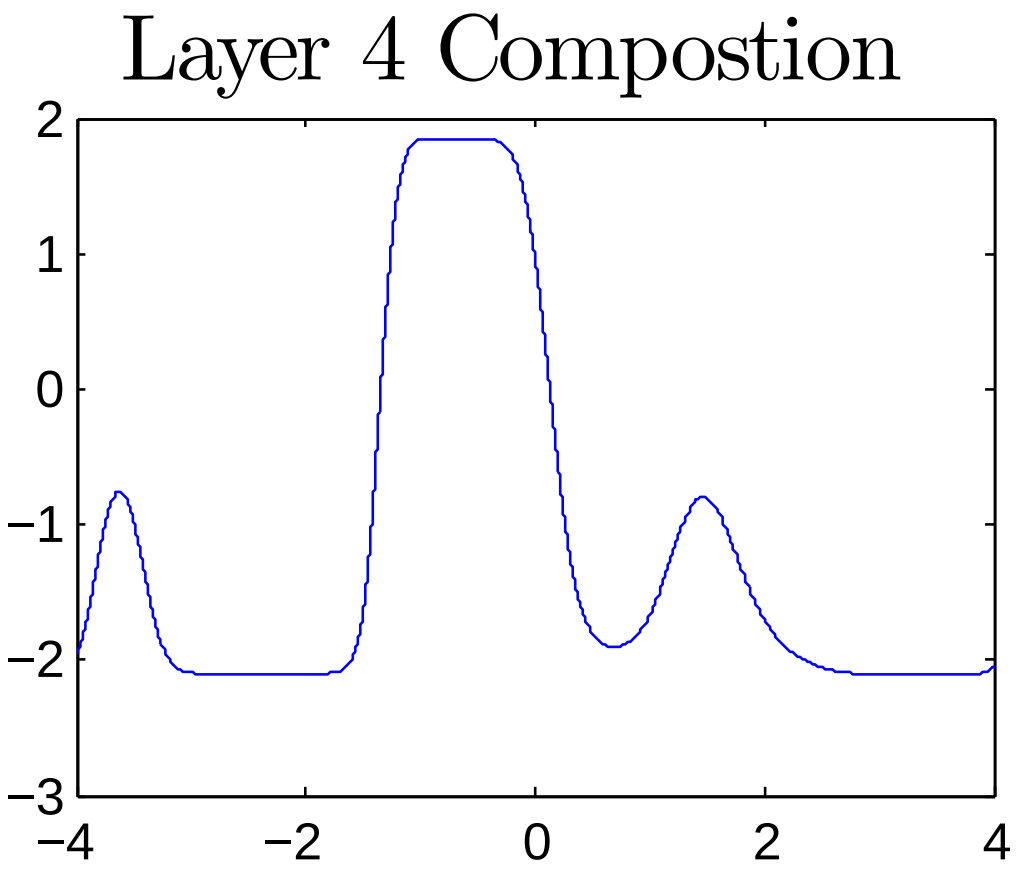
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



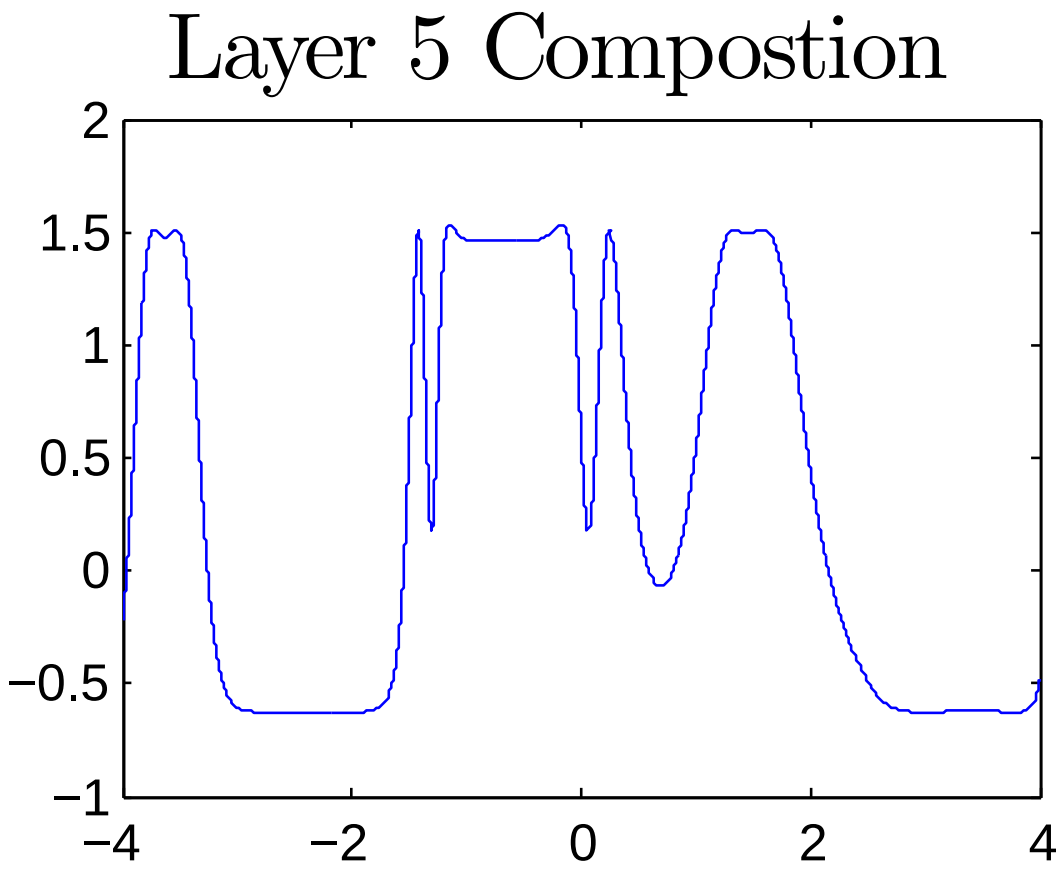
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



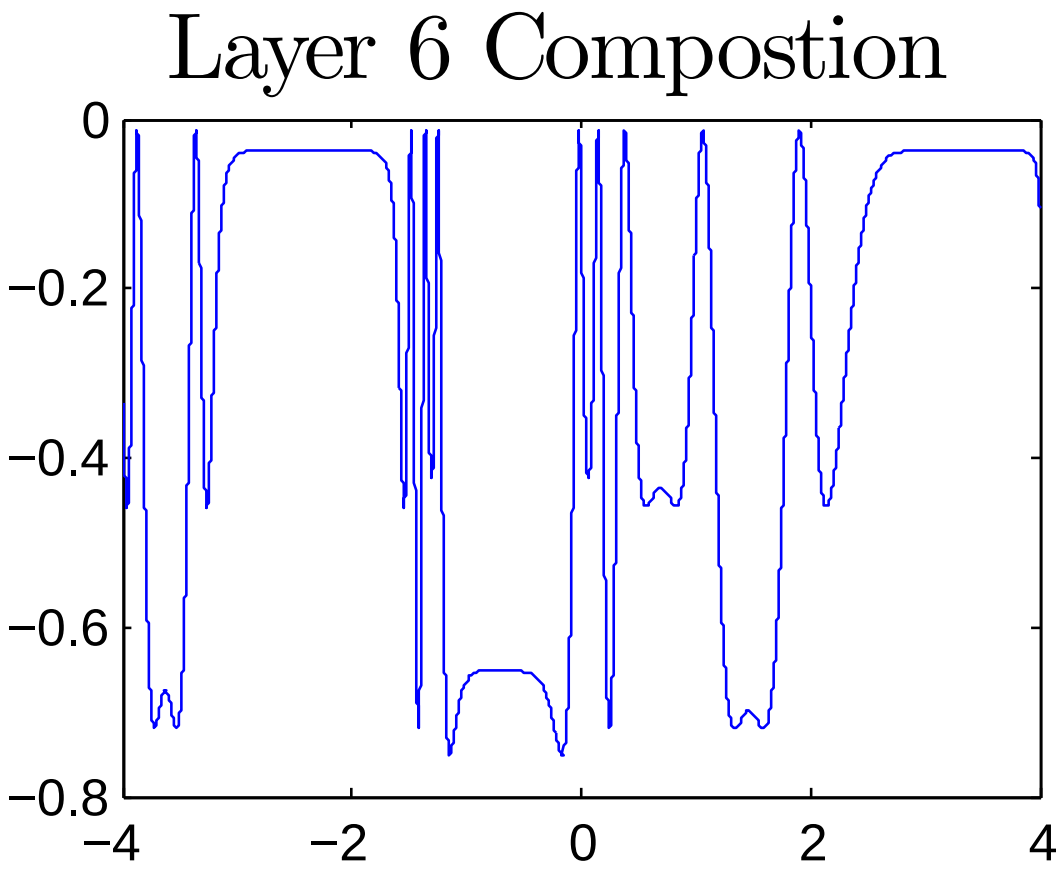
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



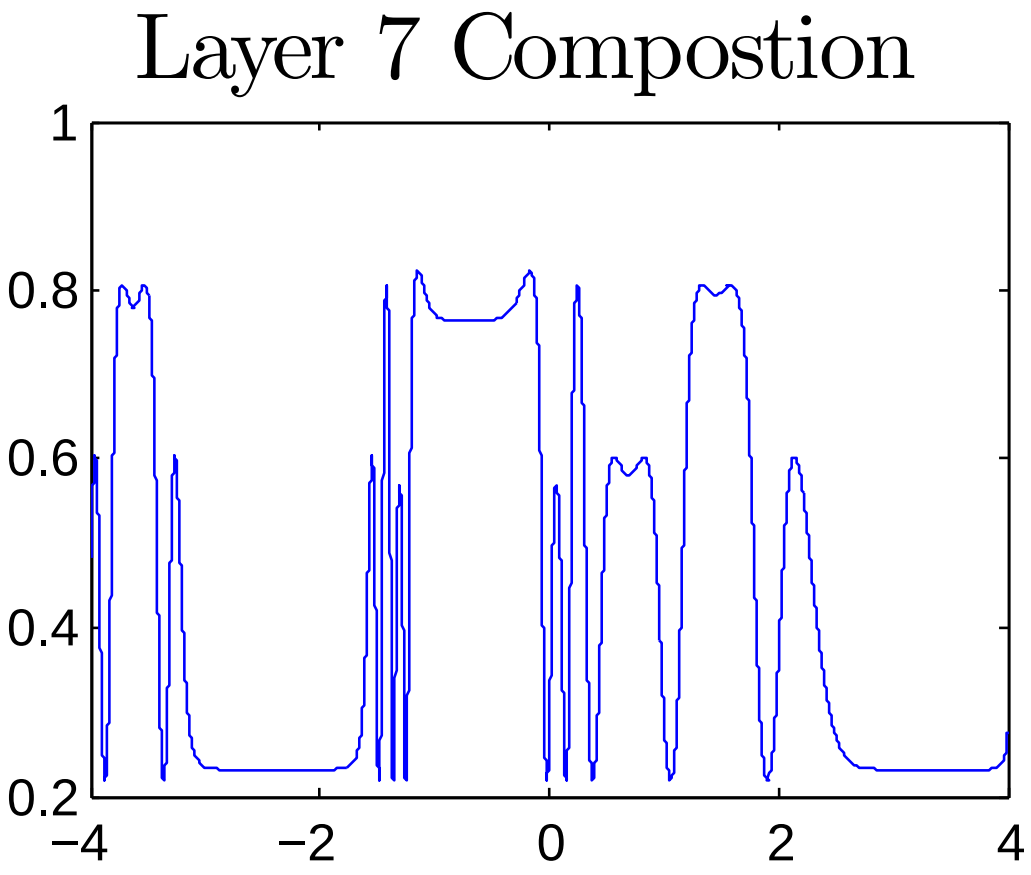
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



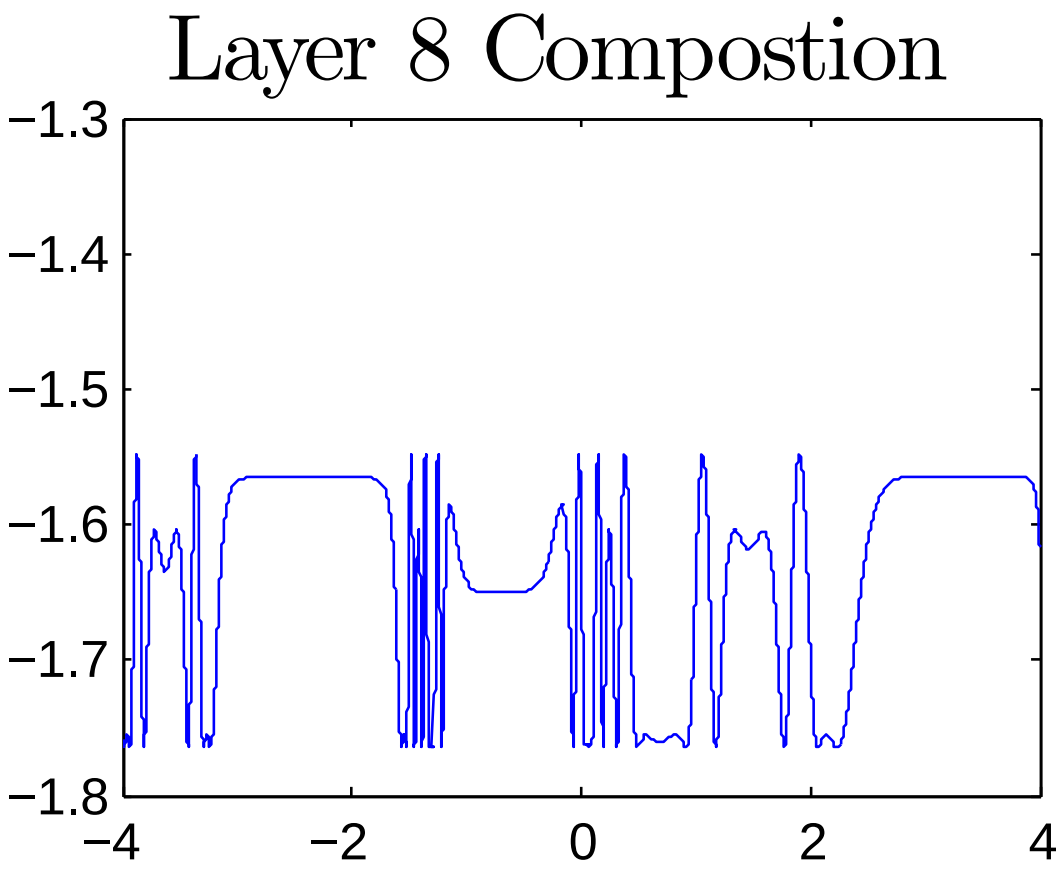
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



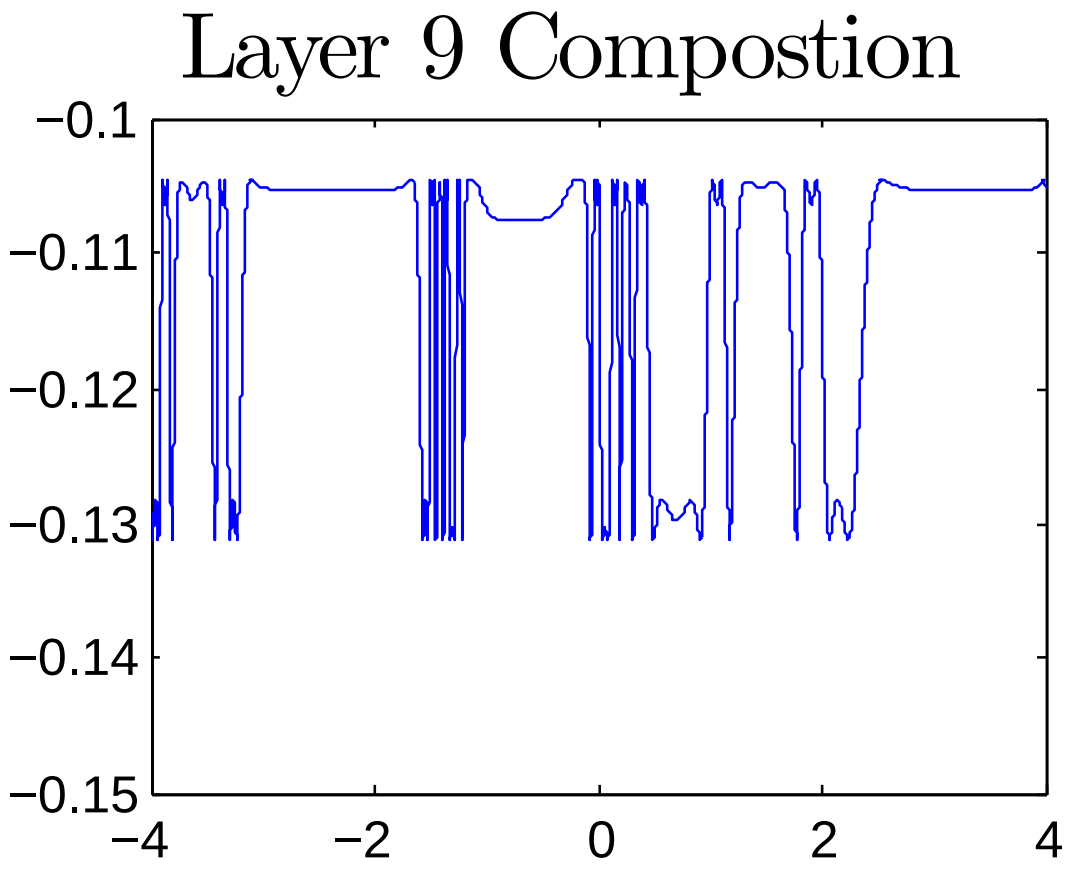
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



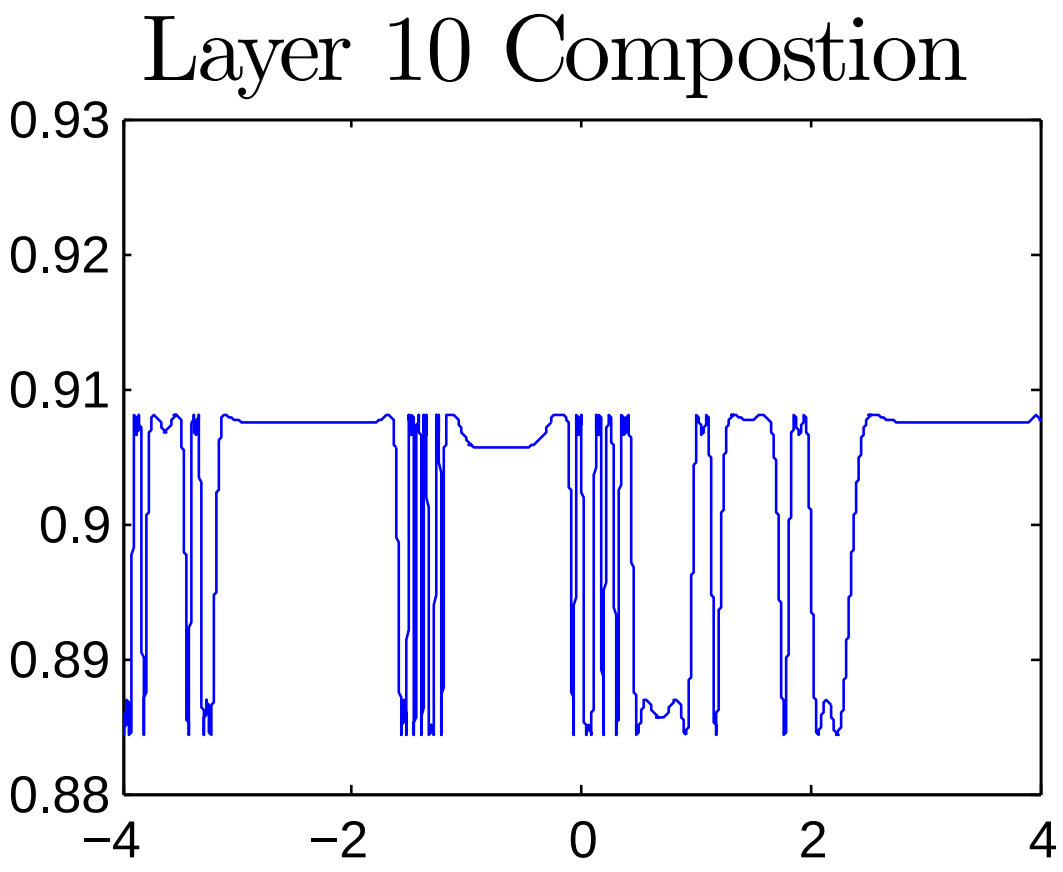
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



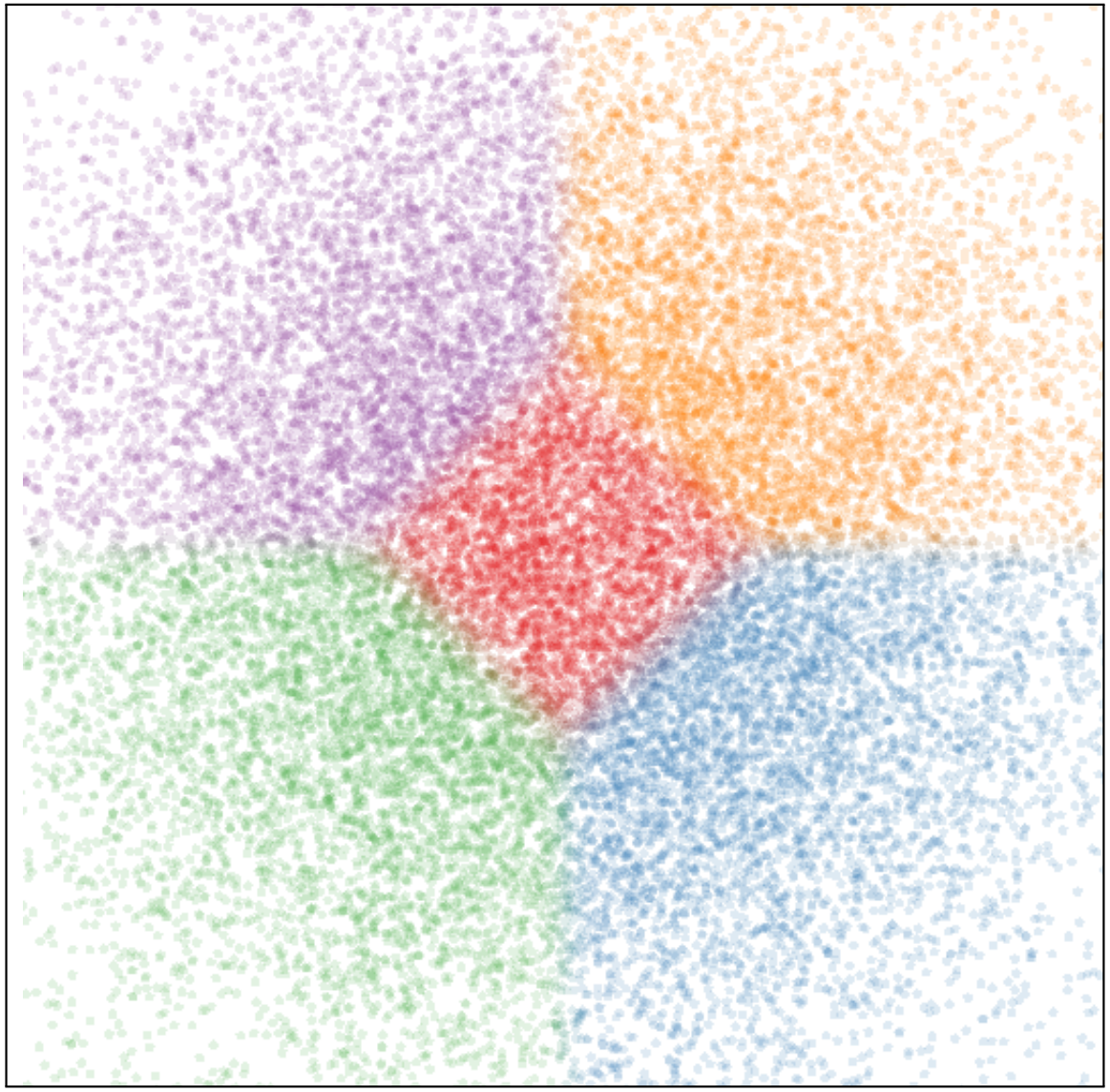
Deep Gaussian Processes

- Draws from one-dimensional deep GPs:



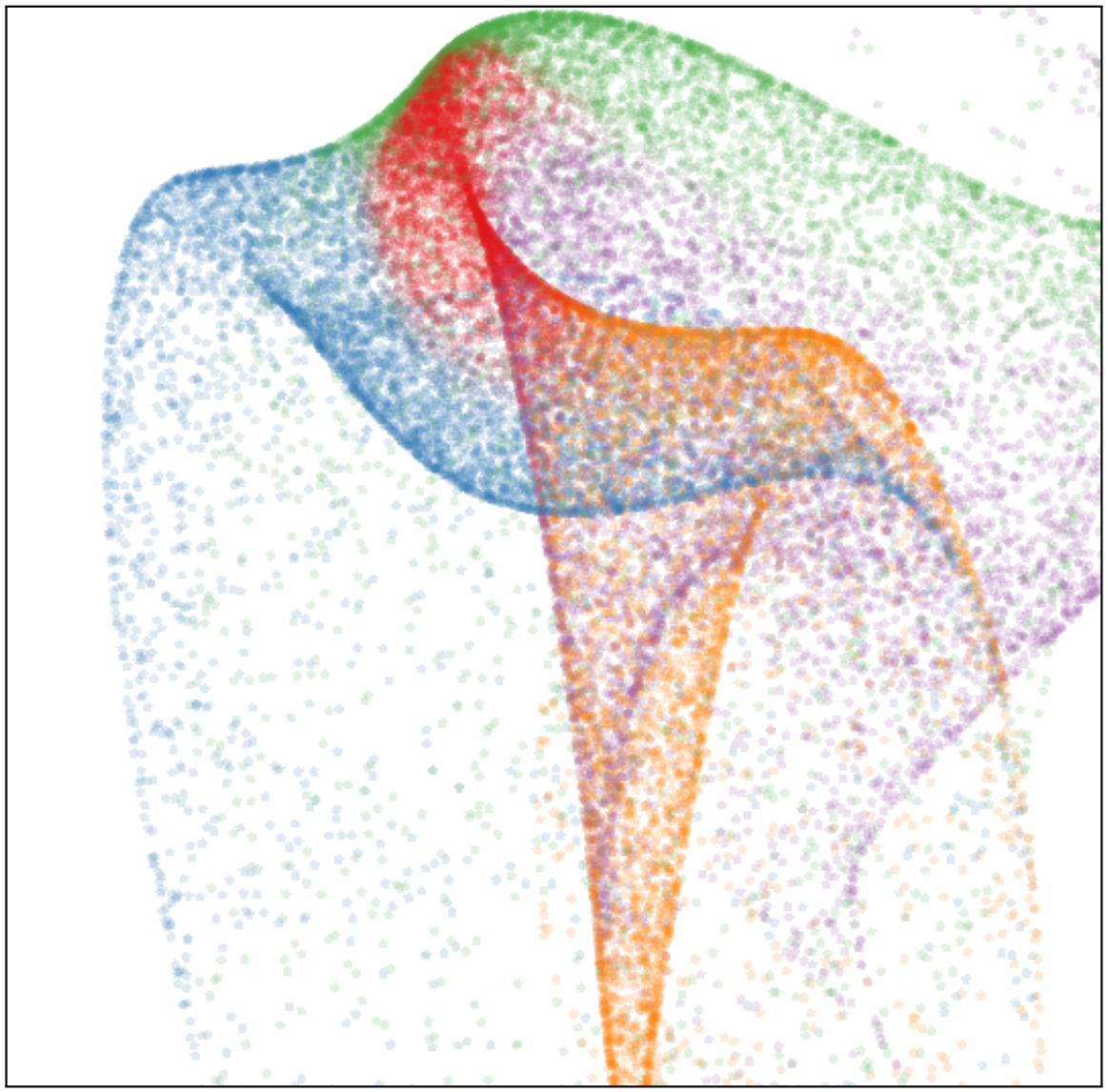
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



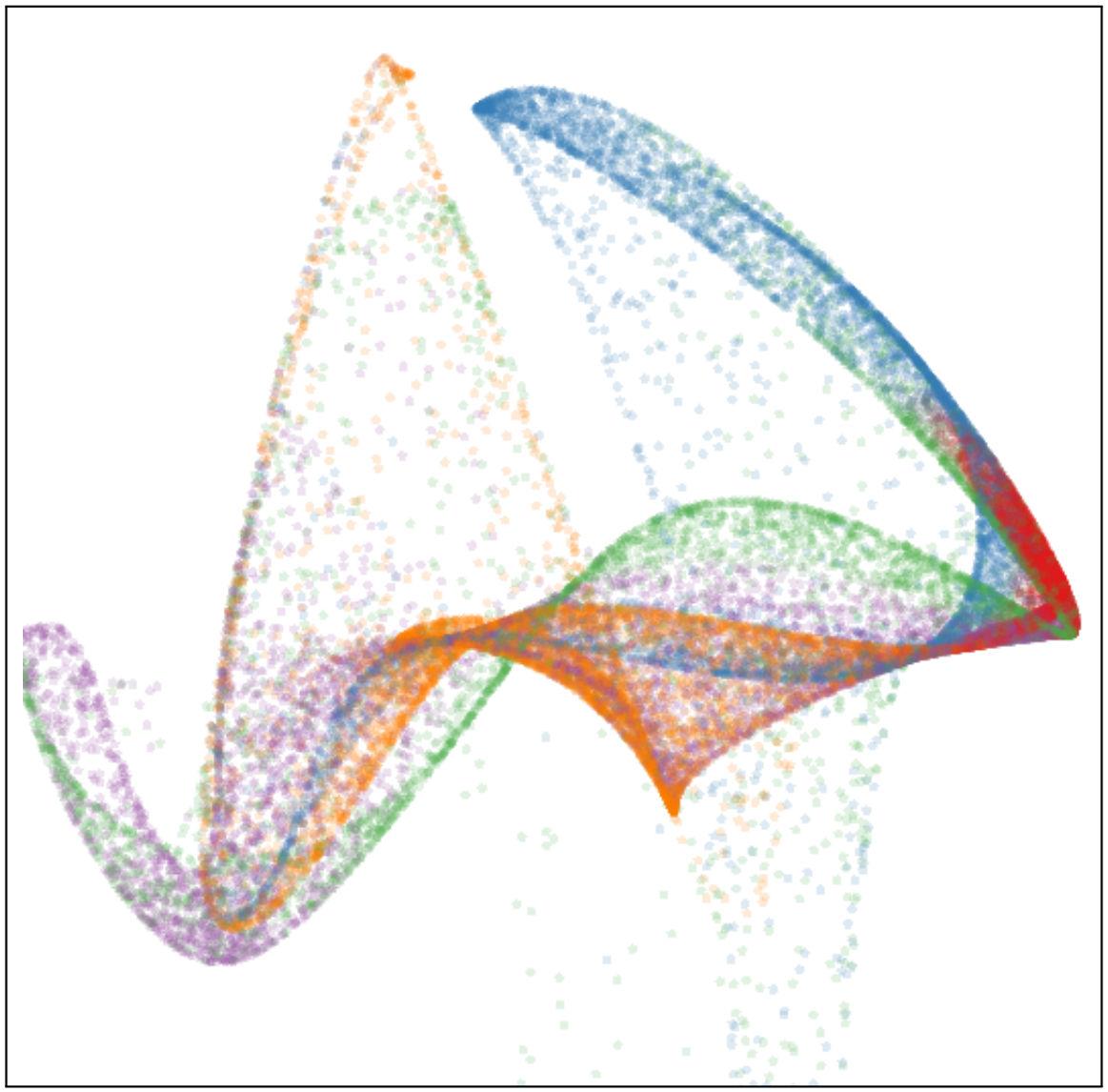
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



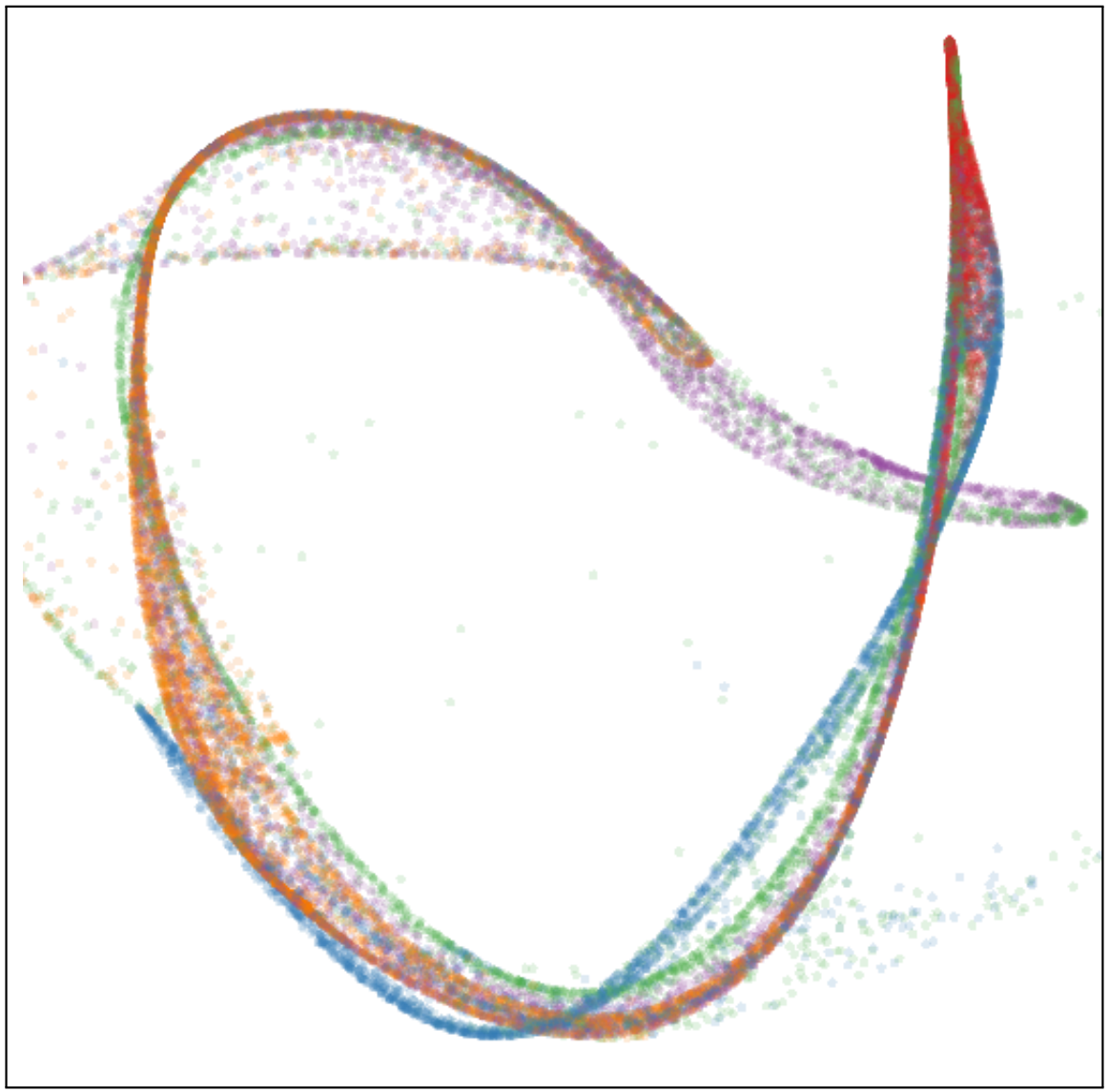
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



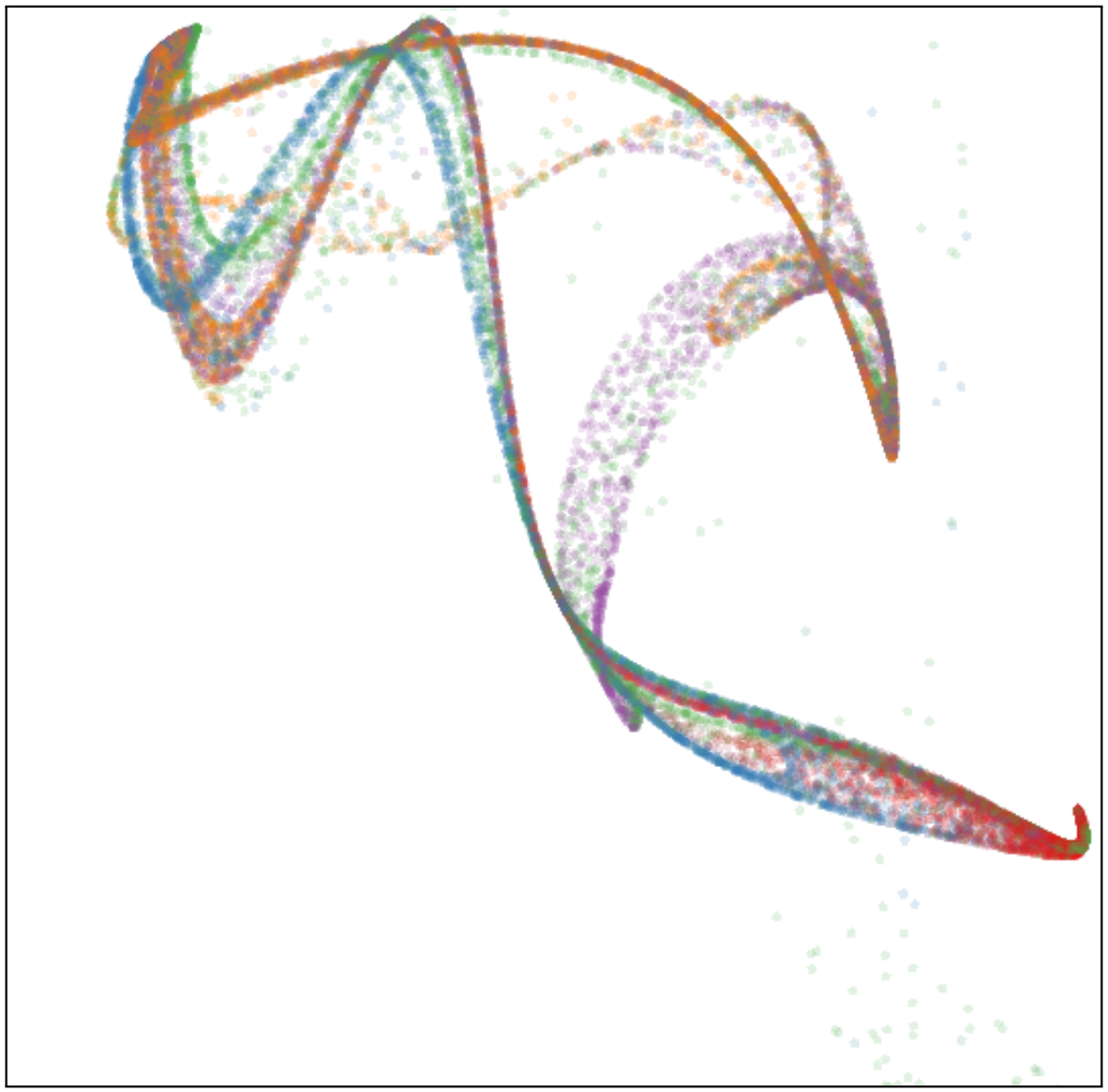
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



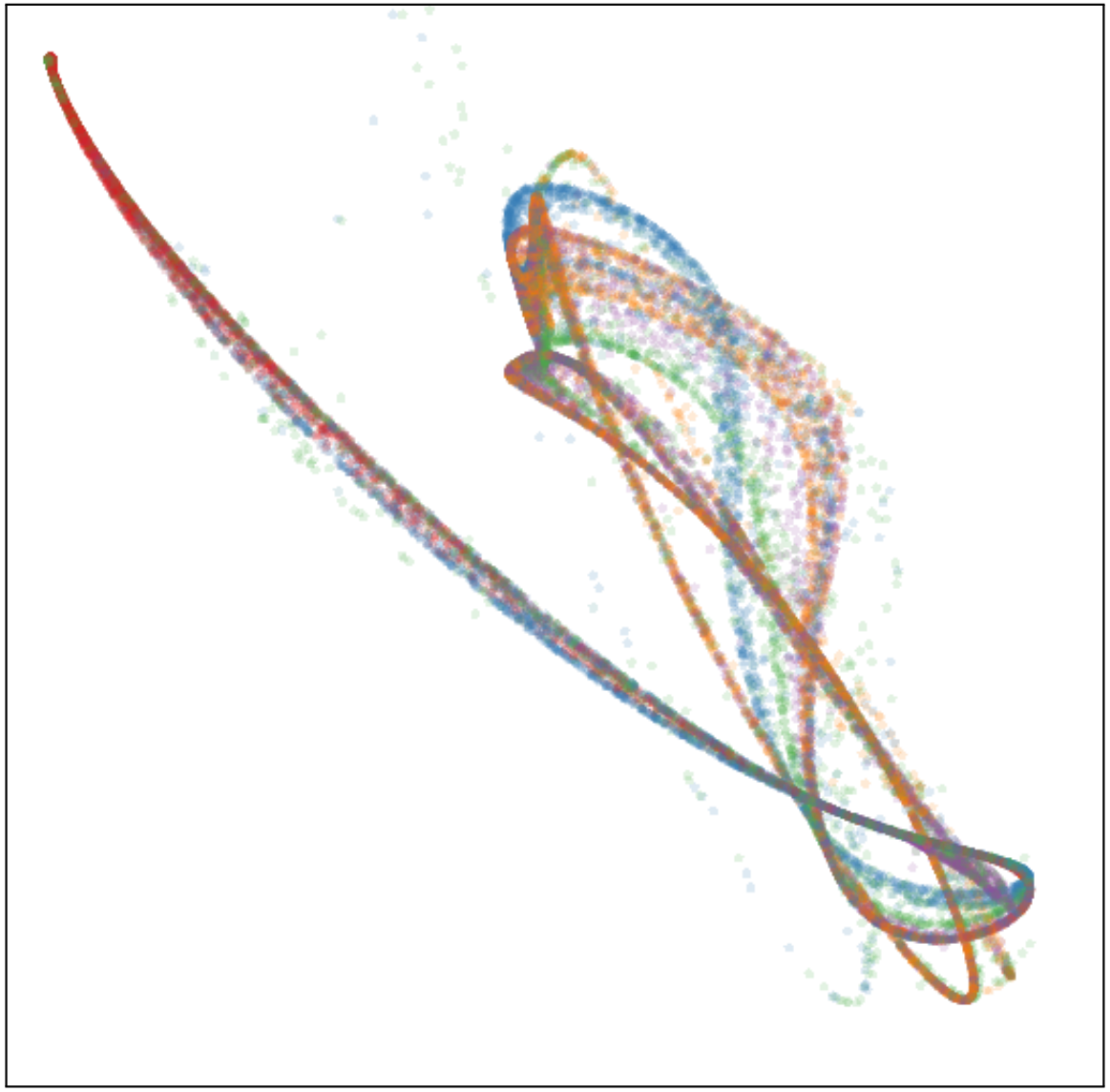
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



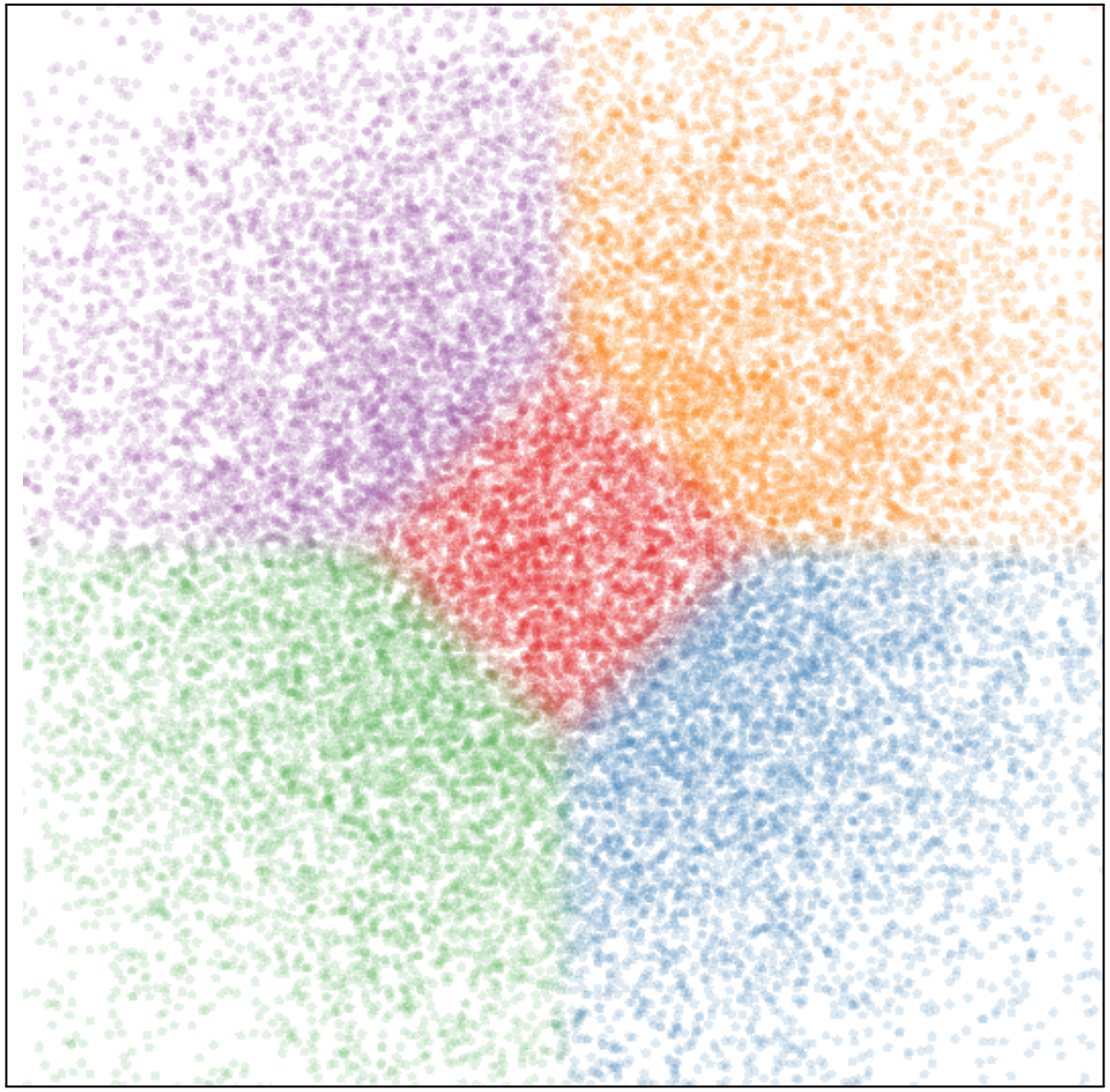
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



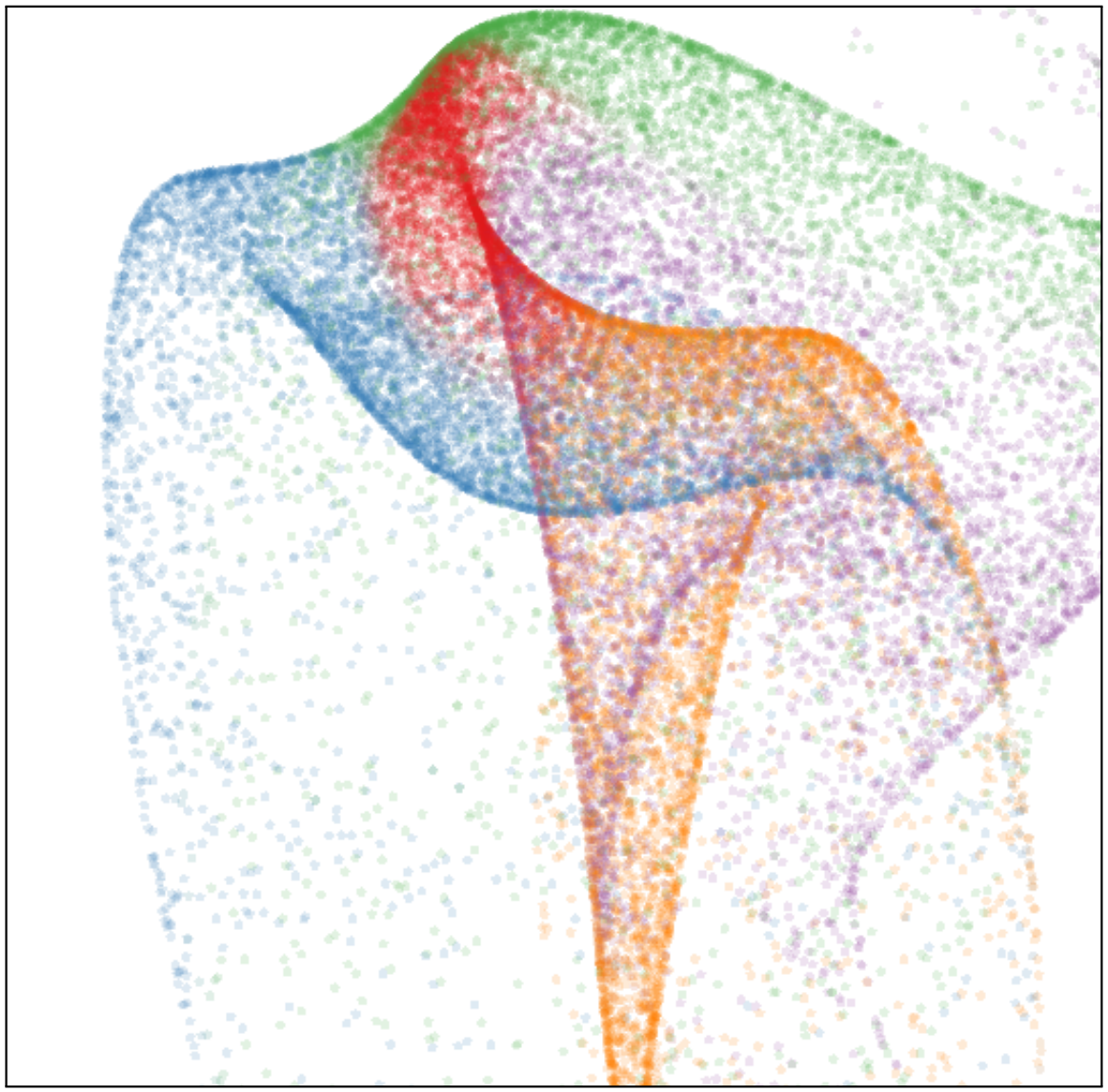
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



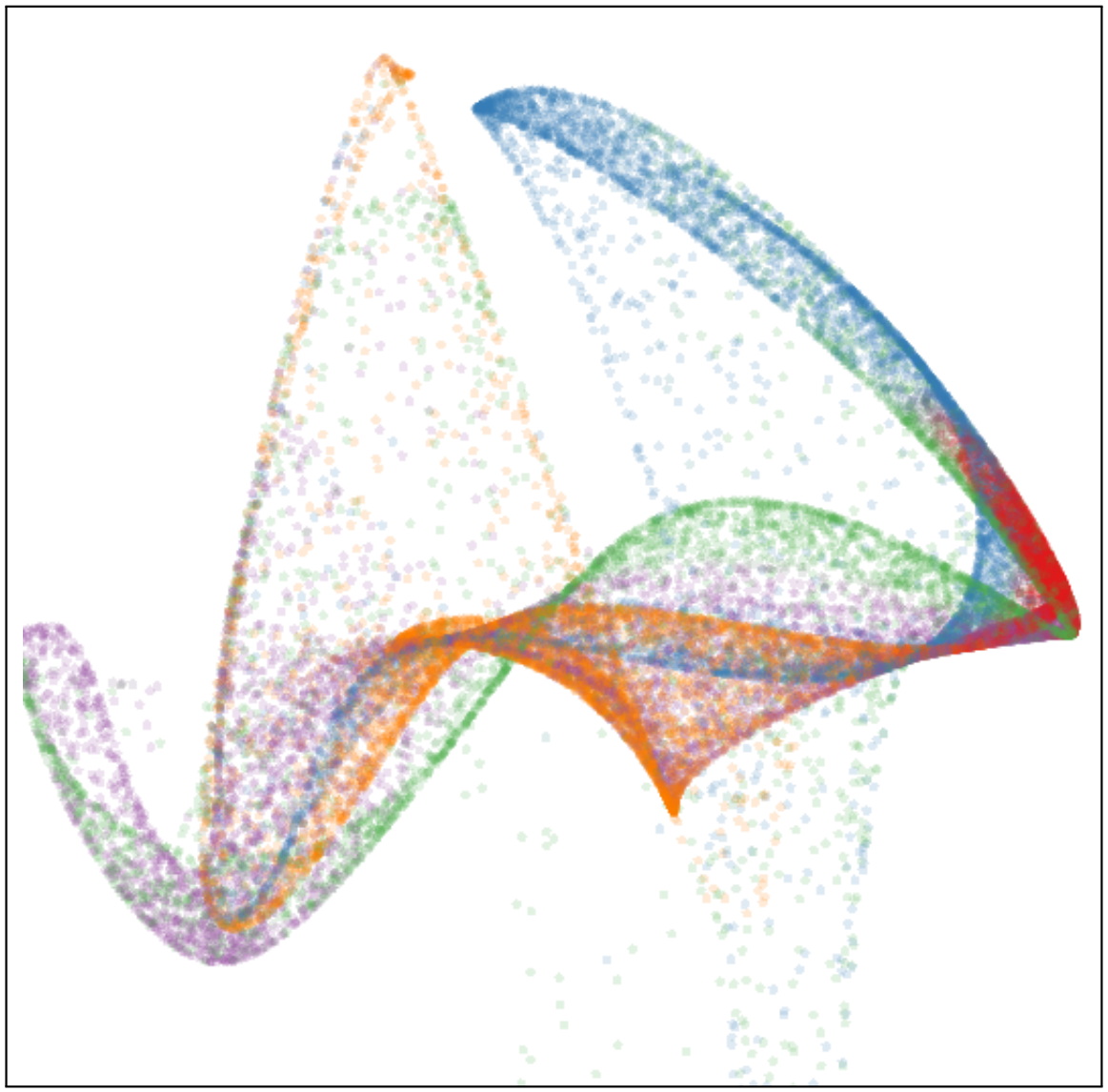
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



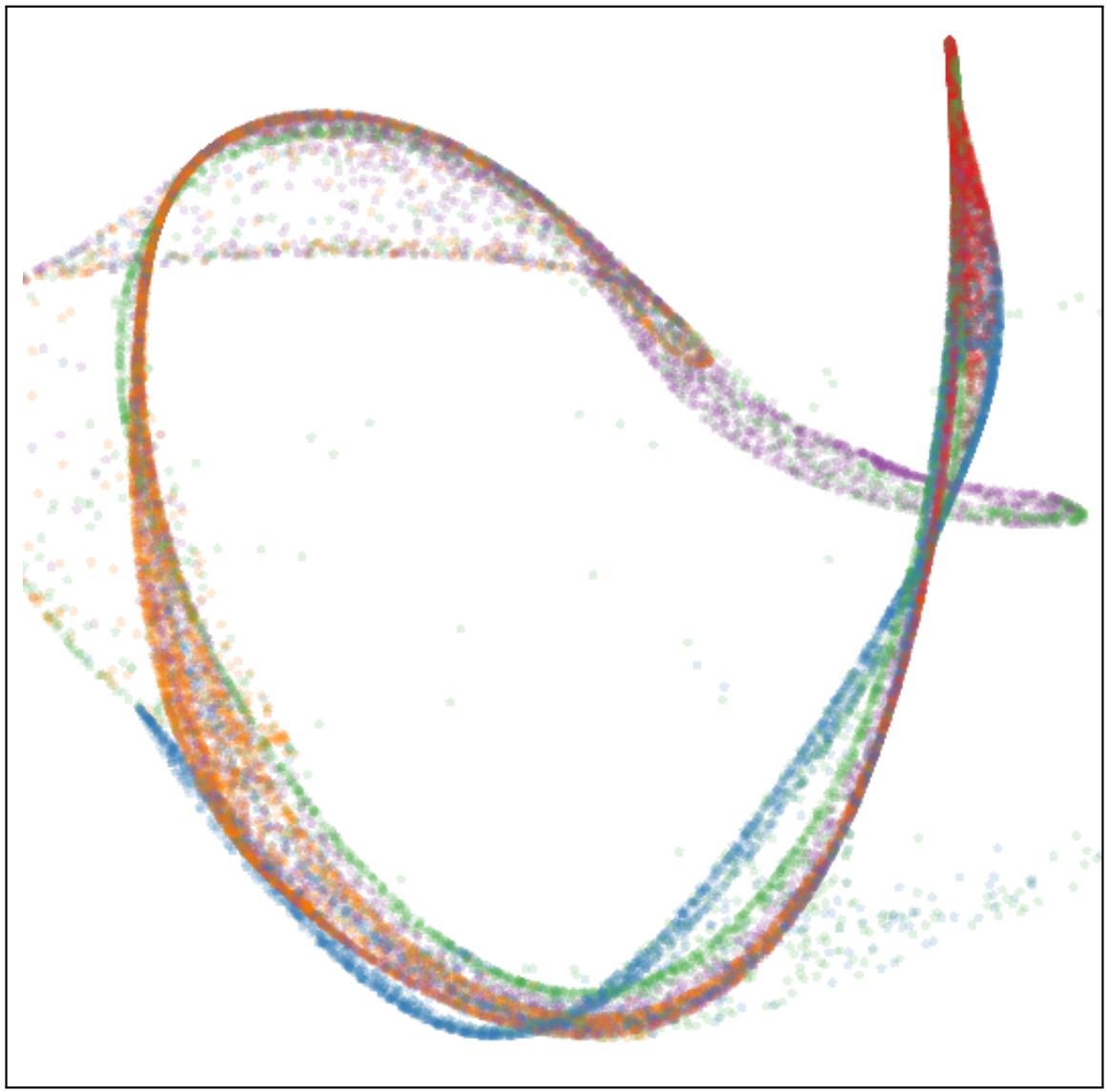
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



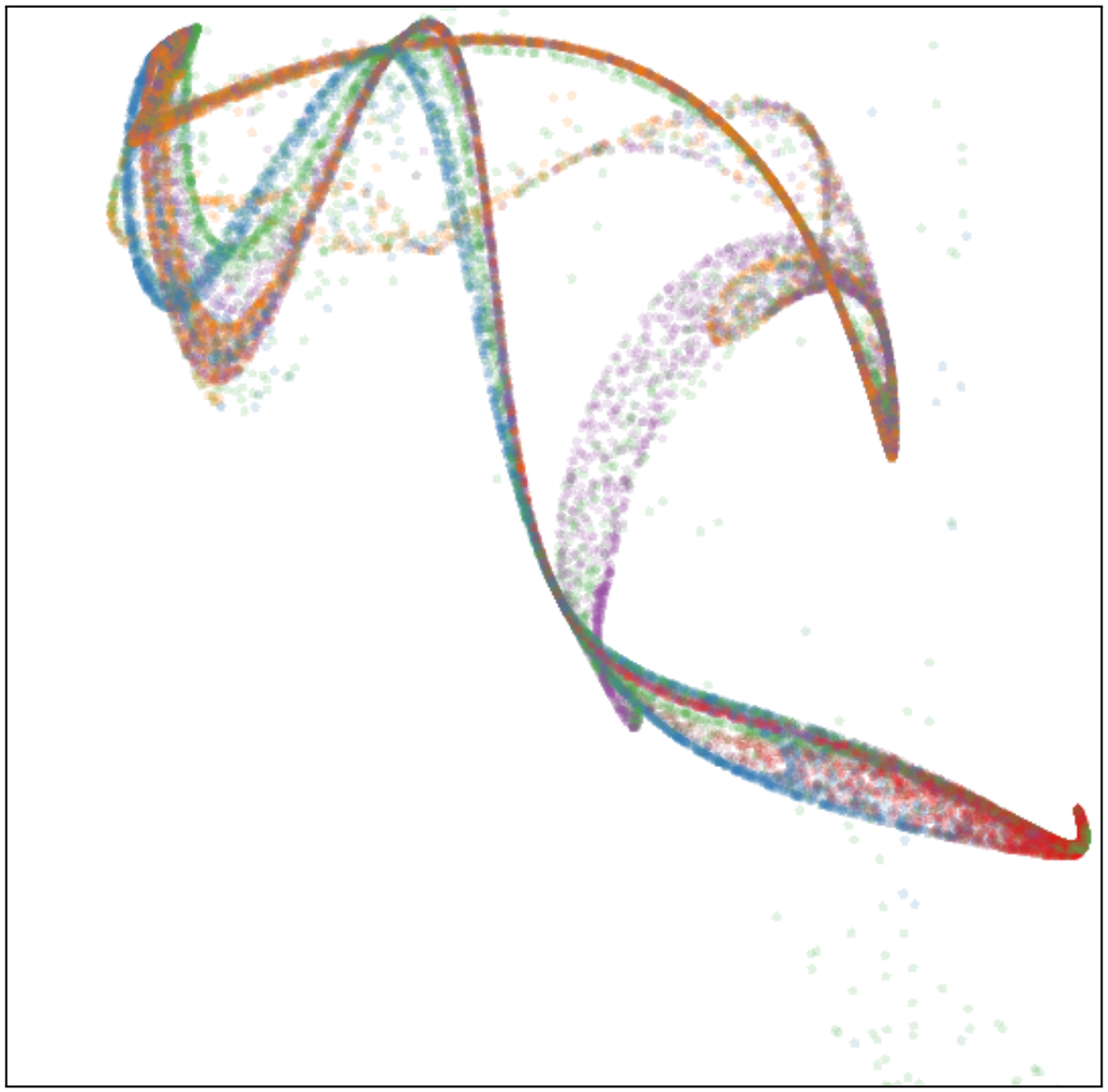
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



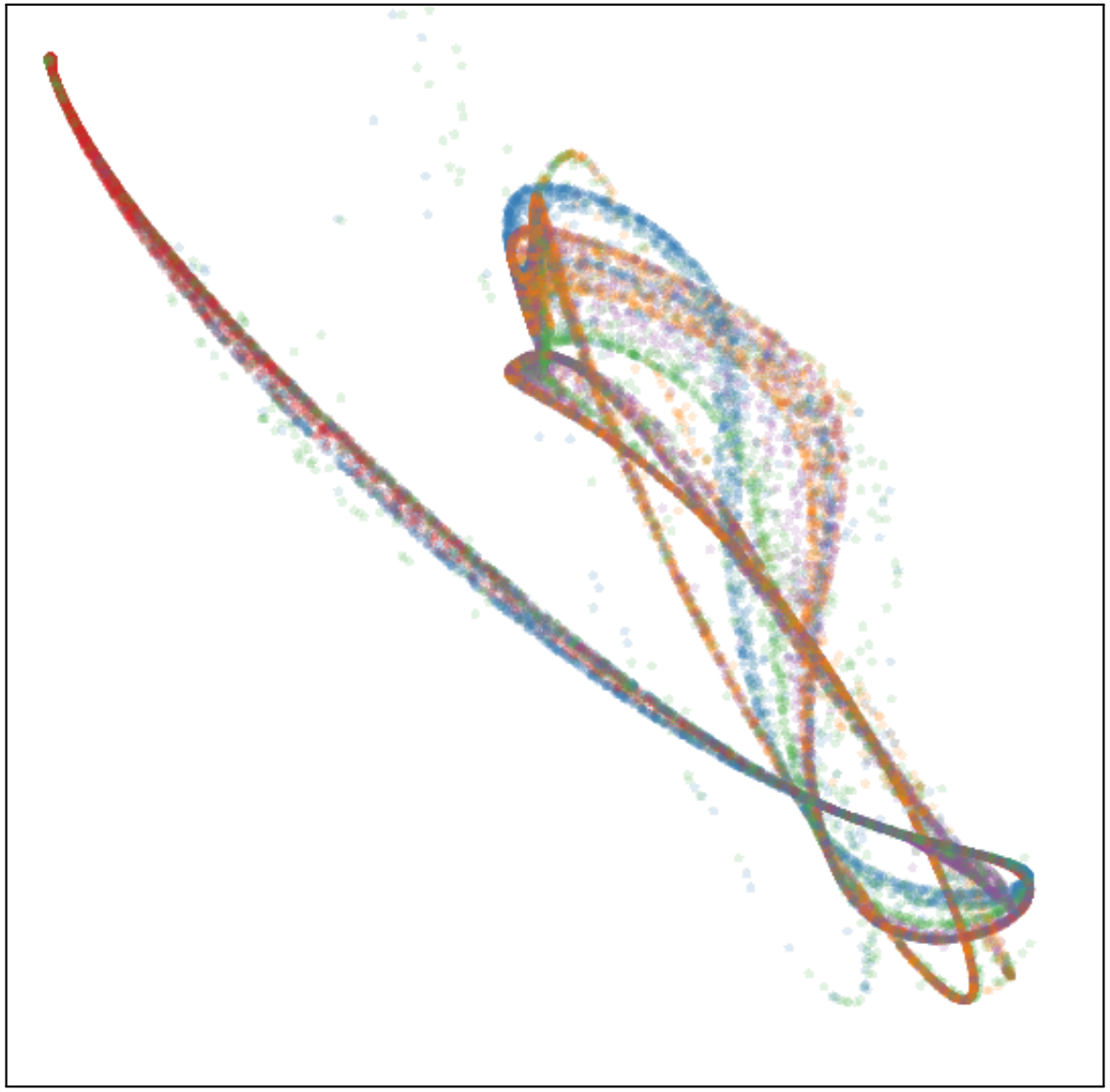
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



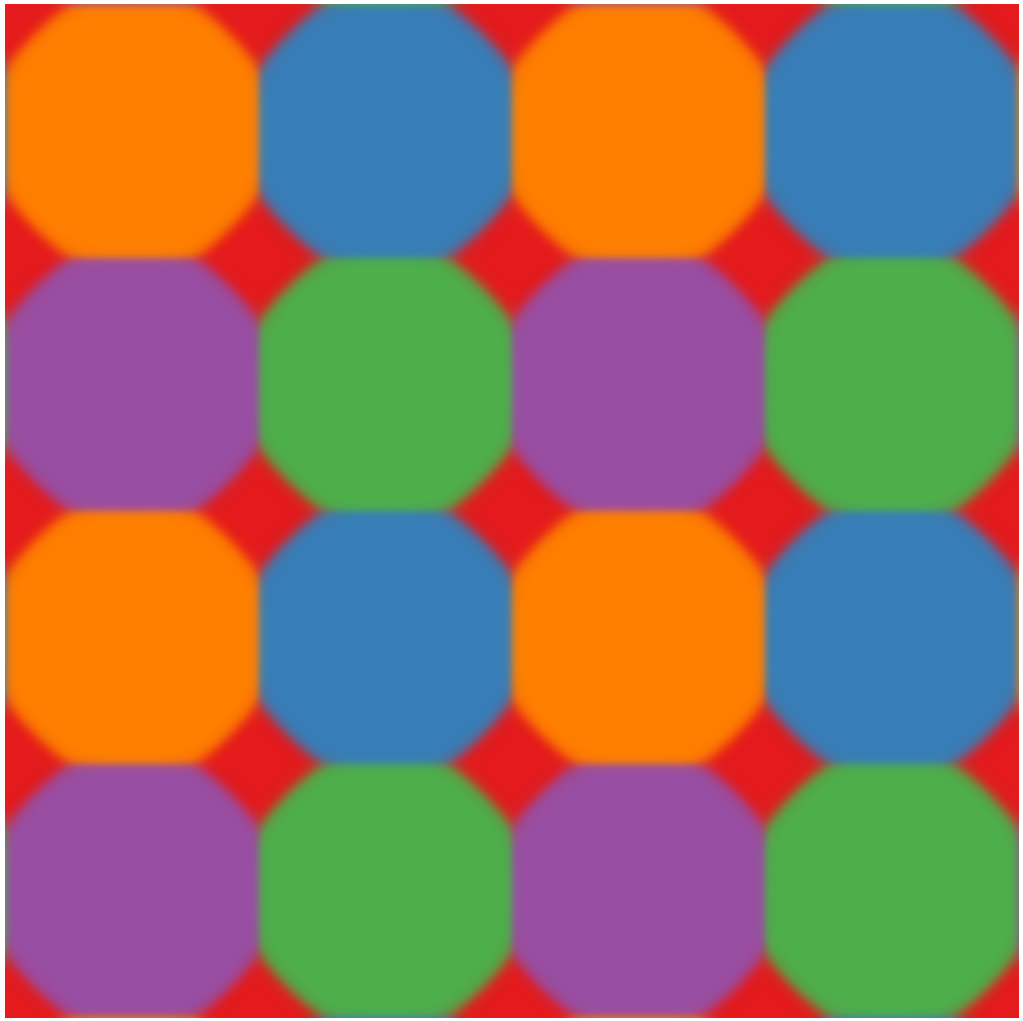
Deep Gaussian Processes

- 2D to 2D warpings of a Gaussian density:



Deep Gaussian Processes

- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



No warping

Deep Gaussian Processes

- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)

One Layers

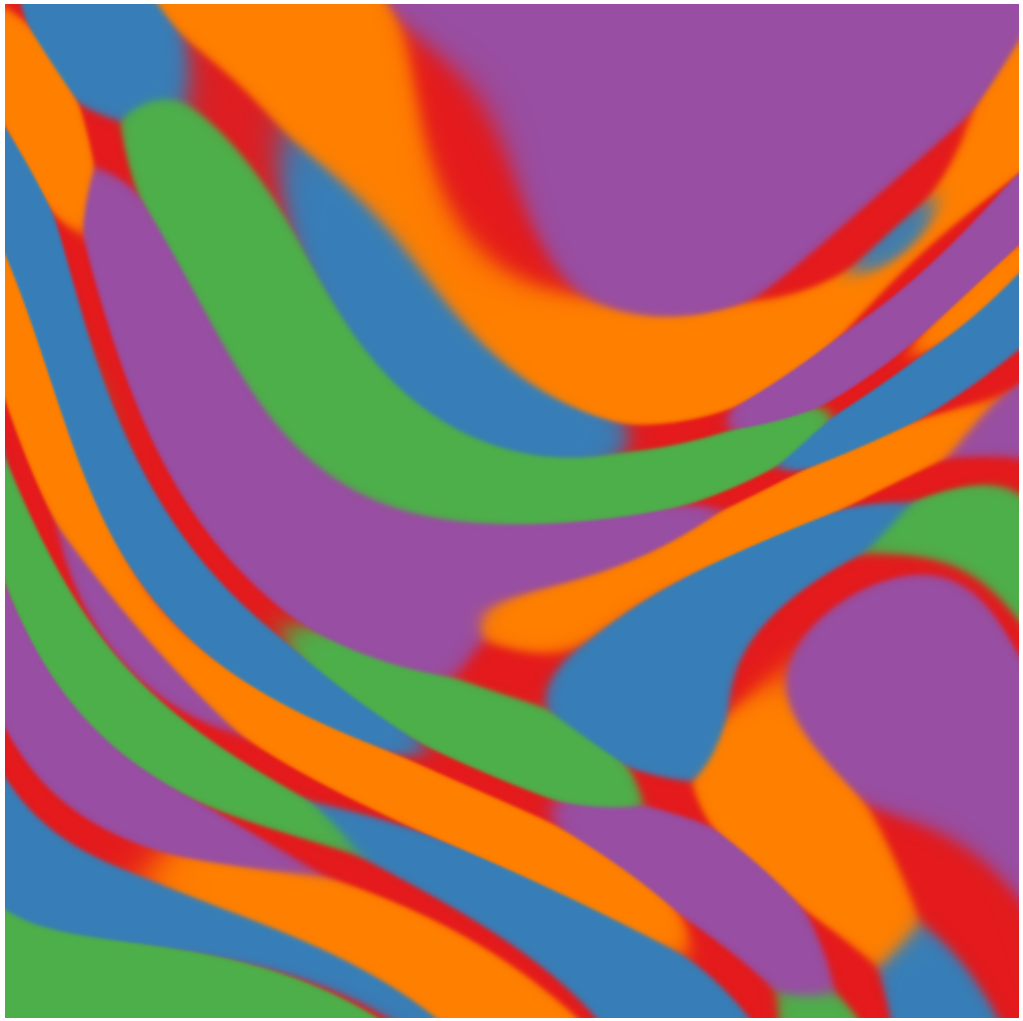
Deep Gaussian Processes

- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)

Two Layers

Deep Gaussian Processes

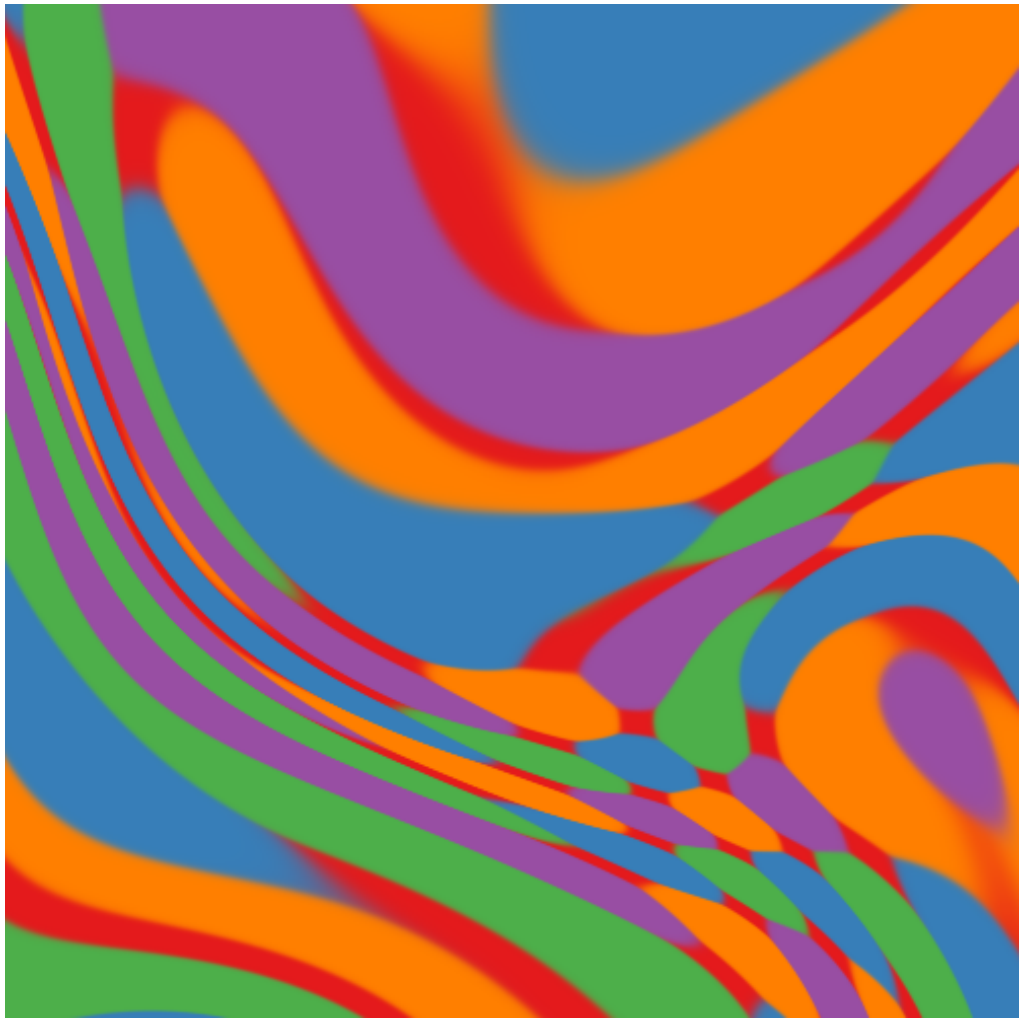
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



Three Layers

Deep Gaussian Processes

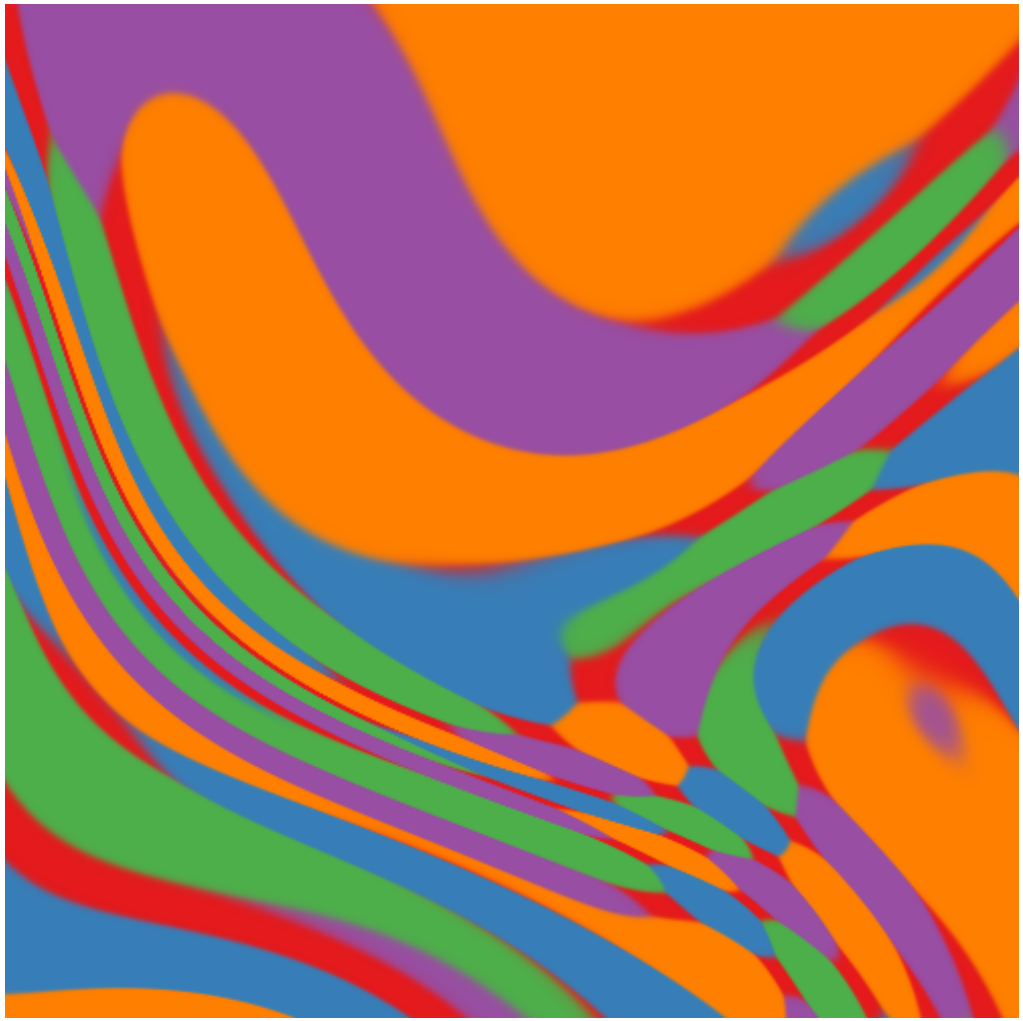
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



Four Layers

Deep Gaussian Processes

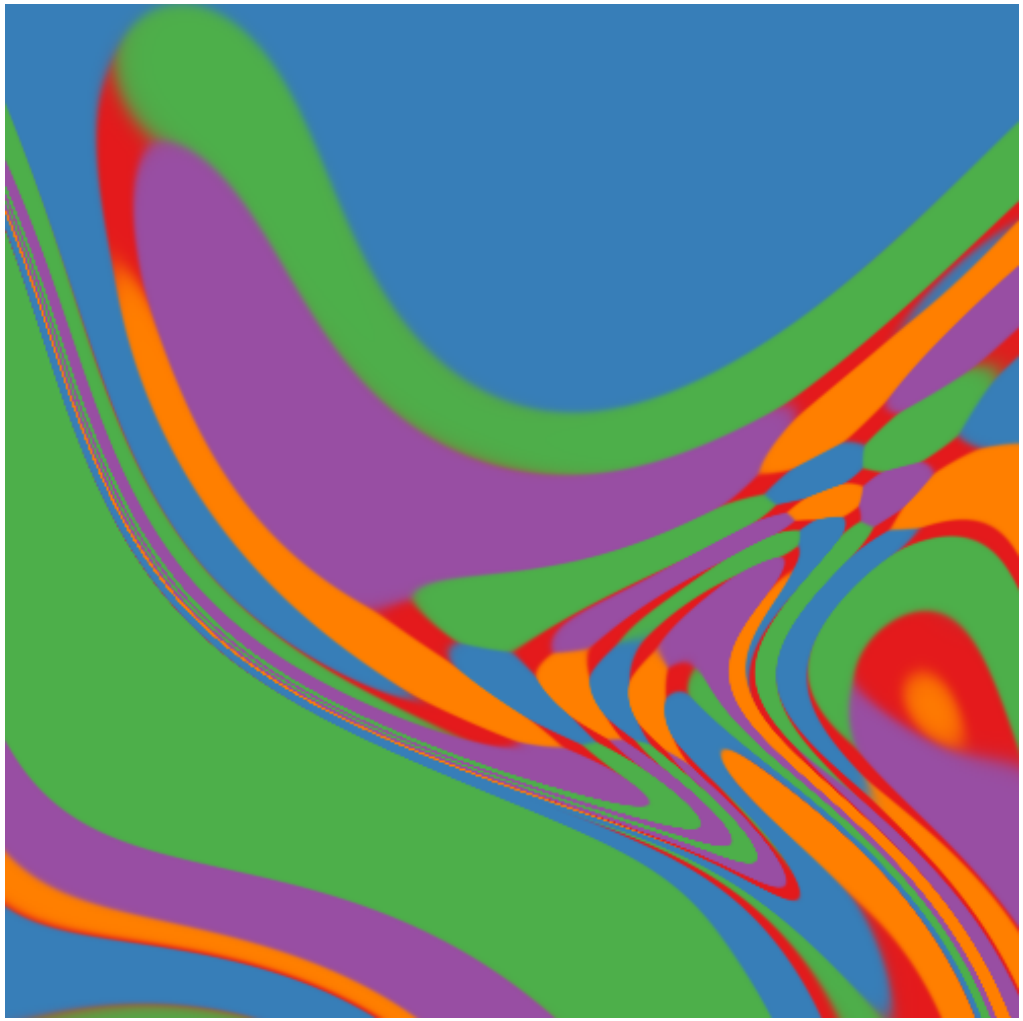
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



Five Layers

Deep Gaussian Processes

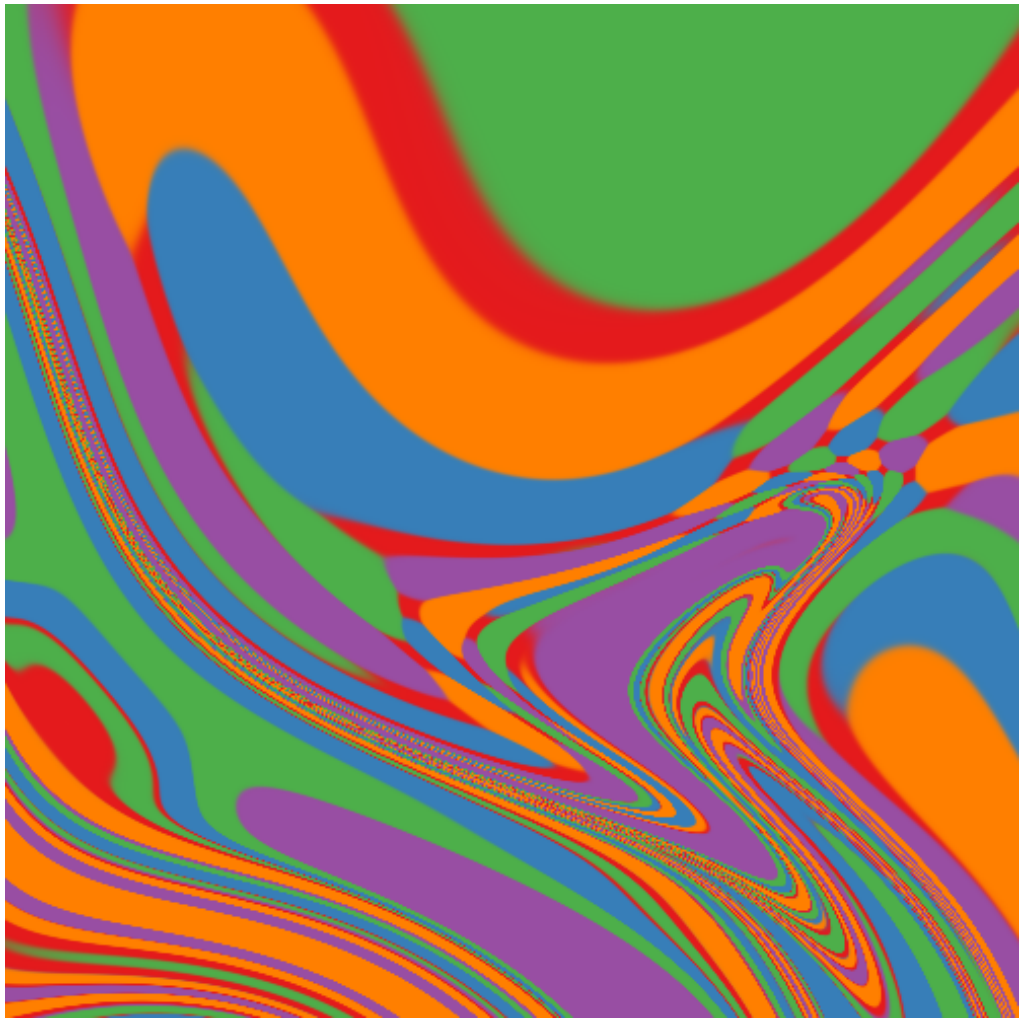
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



Ten Layers

Deep Gaussian Processes

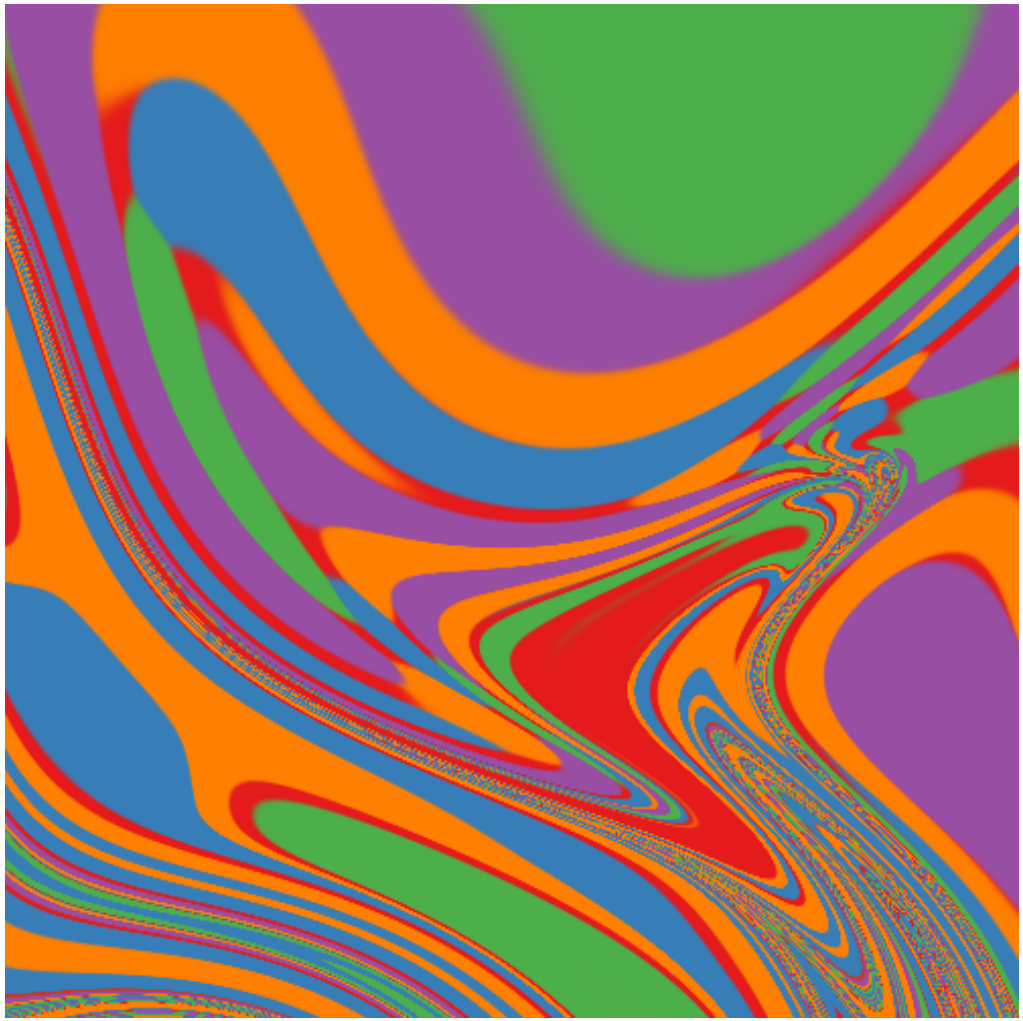
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



Twenty Layers

Deep Gaussian Processes

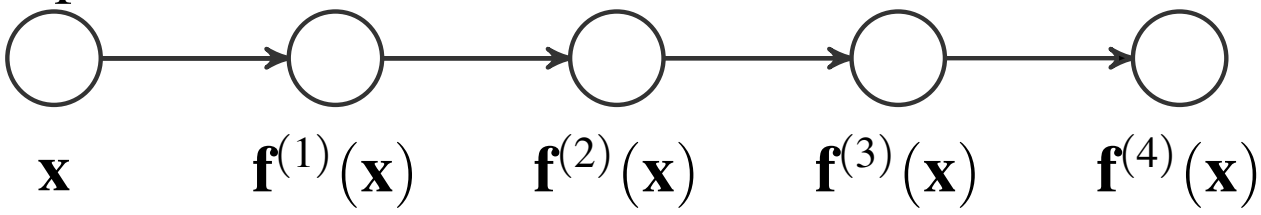
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



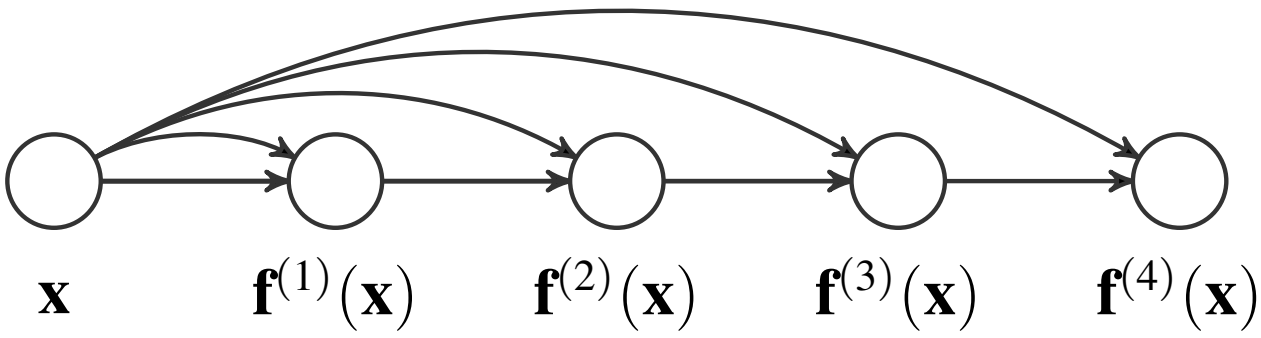
Forty Layers

Deep Gaussian Processes

- Again following Radford's thesis, connect every layer to input:



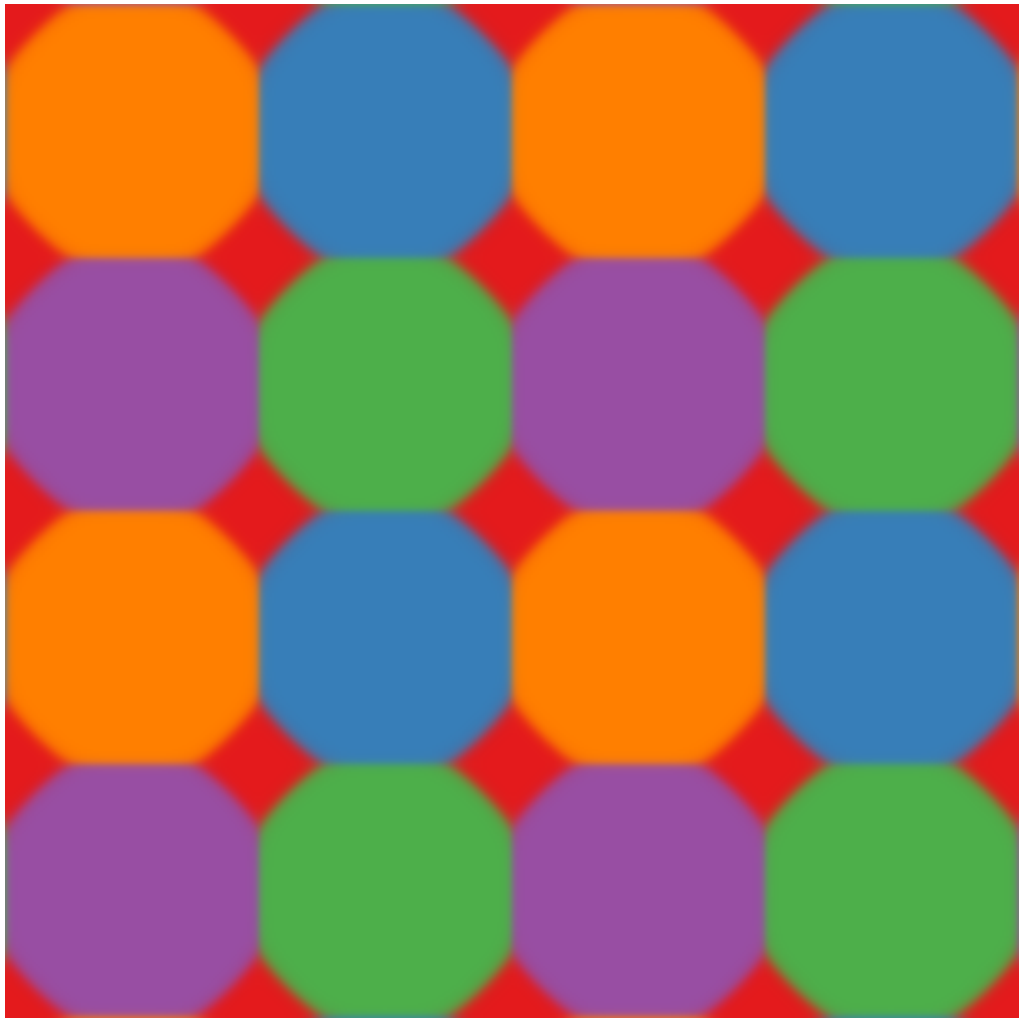
a) standard MLP architecture.



b) Input-connected architecture.

Deep Gaussian Processes

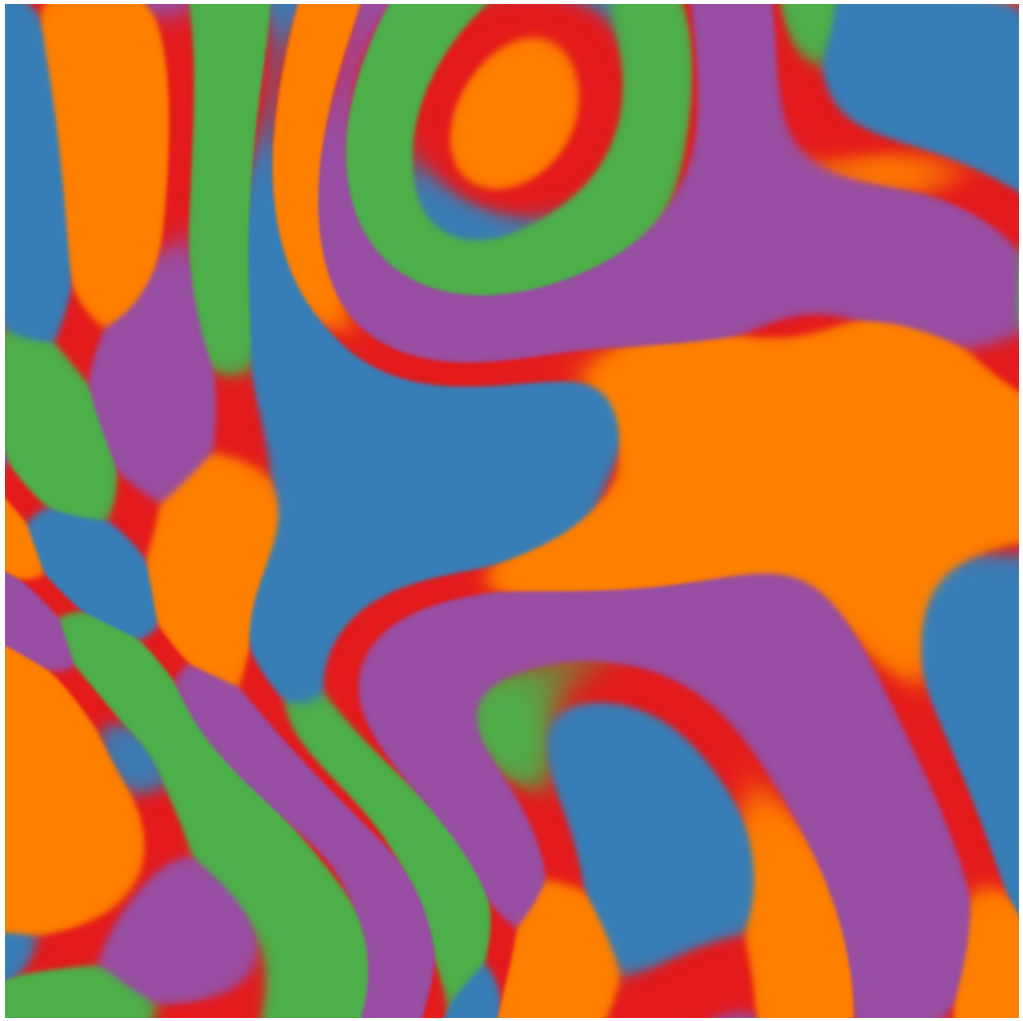
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



No warping

Deep Gaussian Processes

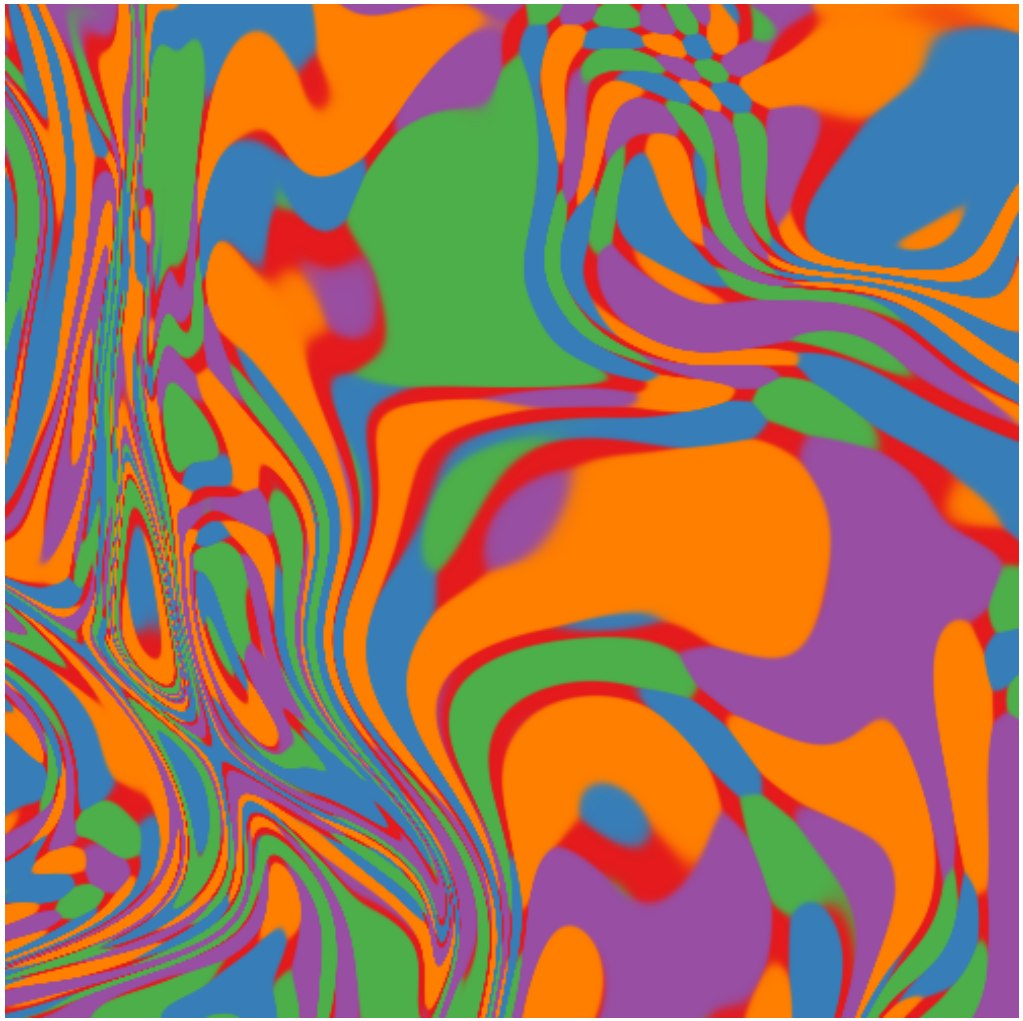
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



Two Layers

Deep Gaussian Processes

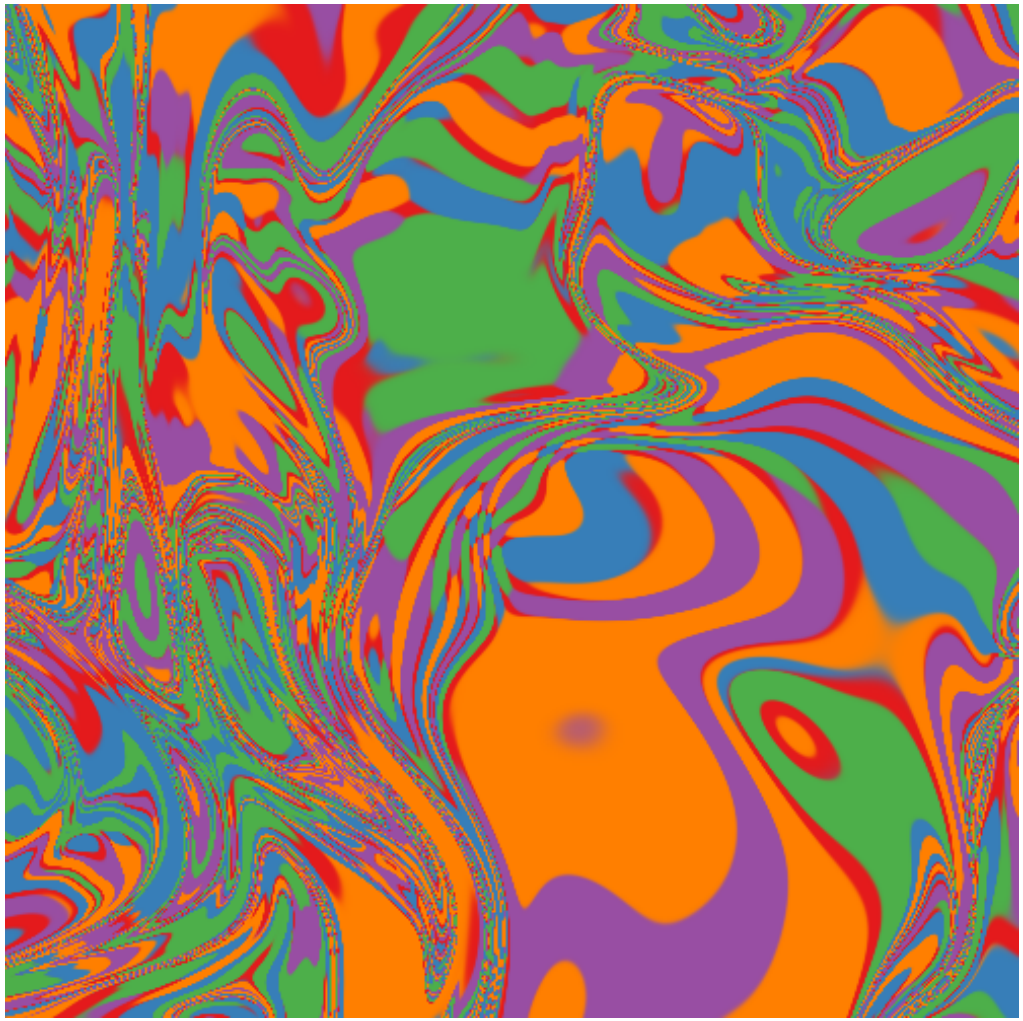
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



Ten Layers

Deep Gaussian Processes

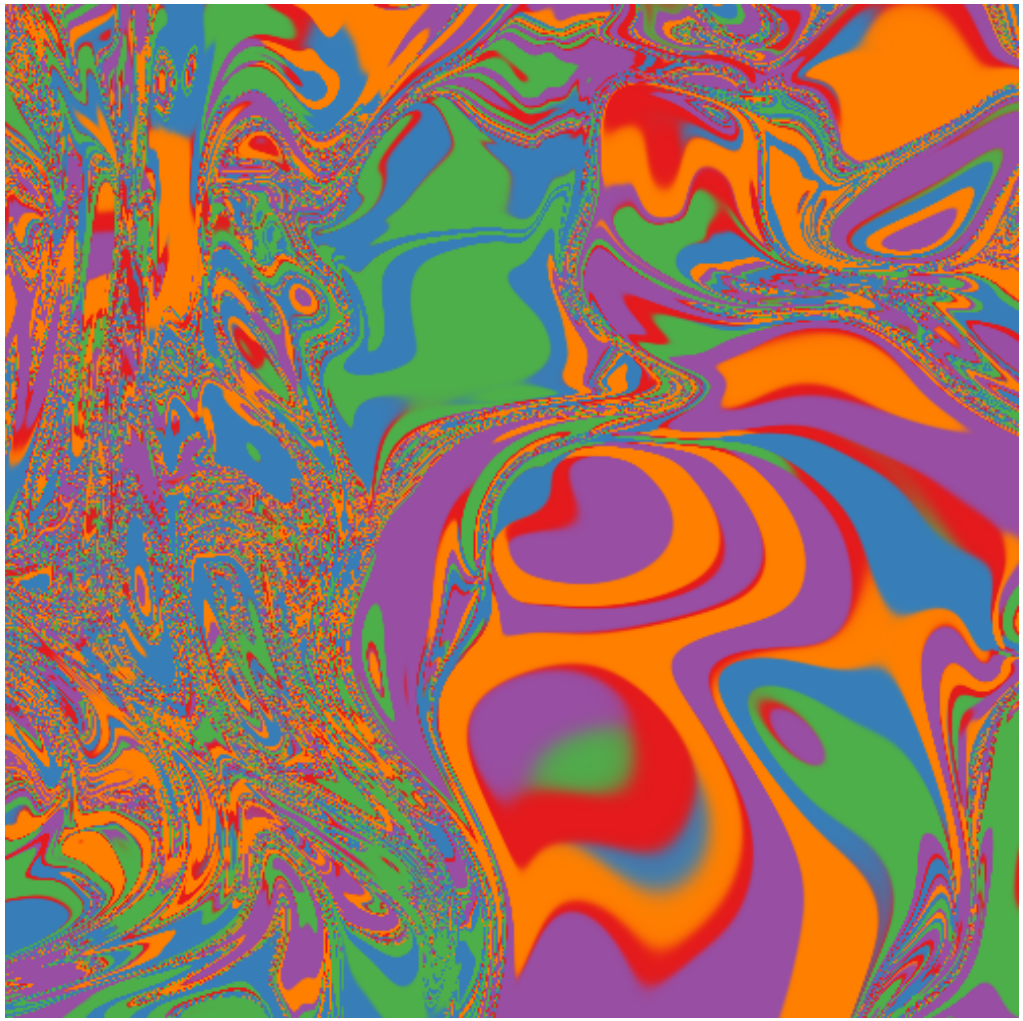
- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



Twenty Layers

Deep Gaussian Processes

- Showing the x that gave rise to a particular y
- (i.e. decision boundaries)



Forty Layers

SUMMARY

- GPs let us compute marginal likelihood, which enables model search
- Kernel learning is an example of representation learning
- Build priors over functions that put mass on the sorts of structures we'd like to be able to learn sheds might shed light on architectures or initialization strategies
- Open questions:
  - When is discrete kernel search a good way to find representations?
  - What could we do if we could compute the marginal likelihood of a neural net?

SUMMARY

- GPs let us compute marginal likelihood, which enables model search
- Kernel learning is an example of representation learning
- Build priors over functions that put mass on the sorts of structures we'd like to be able to learn sheds might shed light on architectures or initialization strategies
- Open questions:
  - When is discrete kernel search a good way to find representations?
  - What could we do if we could compute the marginal likelihood of a neural net?

Thanks!