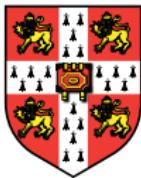


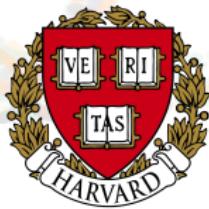
Visualizing Priors on Deep Networks



David Duvenaud, Oren Rippel, Ryan Adams, Zoubin Ghahramani



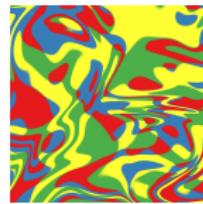
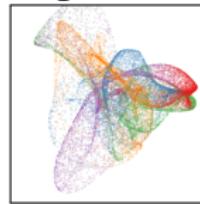
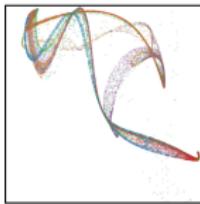
UNIVERSITY OF
CAMBRIDGE



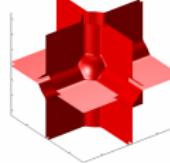
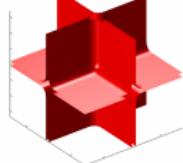
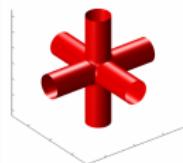
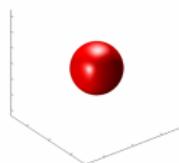
June 16, 2014

What this talk is about

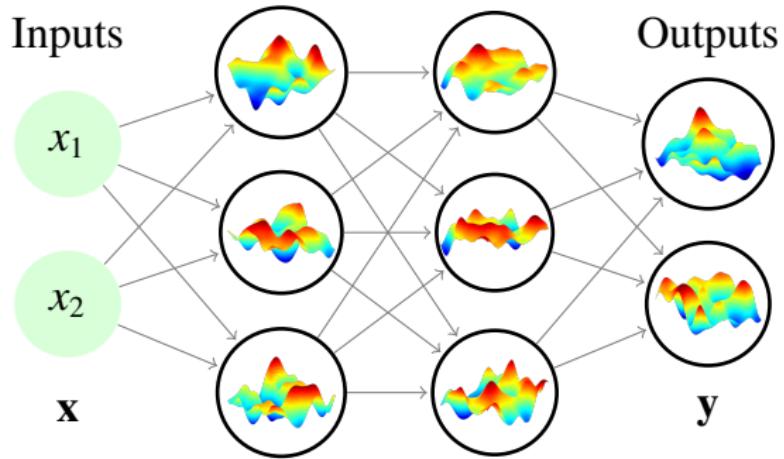
- ▶ Neural nets require lots of design choices, implications hard to understand.
- ▶ Why not just sample from priors on deep nets?
- ▶ Might help understand them without reference to a specific dataset, loss function, or training method.
- ▶ We'll visualize and analyze sampled neural nets:



- ▶ We'll also see how model choices (such as dropout) affect the resulting models.



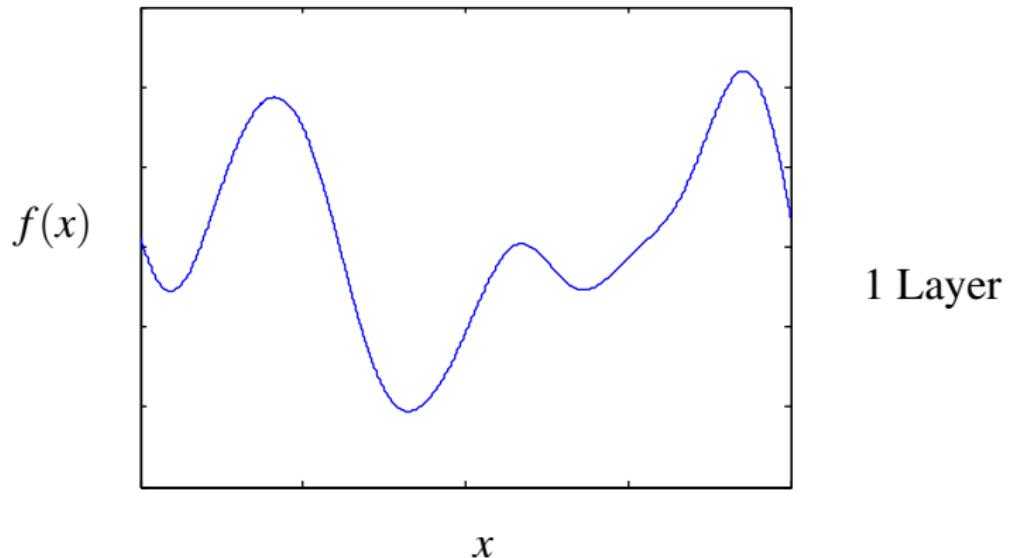
A prior on deep nets: deep Gaussian processes



- ▶ A neural net where each neuron's activation function is drawn from a Gaussian process prior.
- ▶ Avoids problem of unit saturation (with sigmoidal units).
- ▶ Each draw from neural net prior gives a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$.

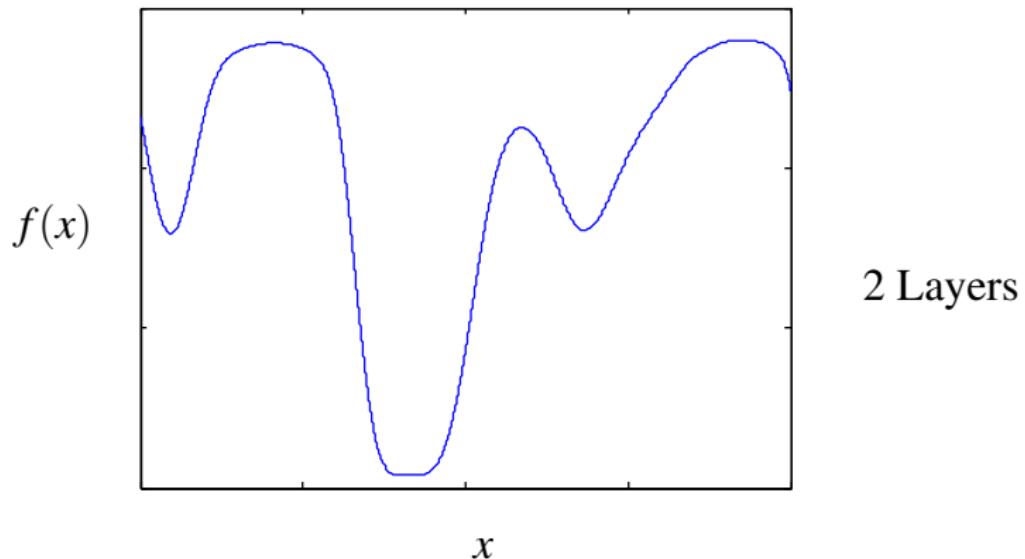
Priors on deep networks

- ▶ A draw from a one-neuron-per-layer deep GP:



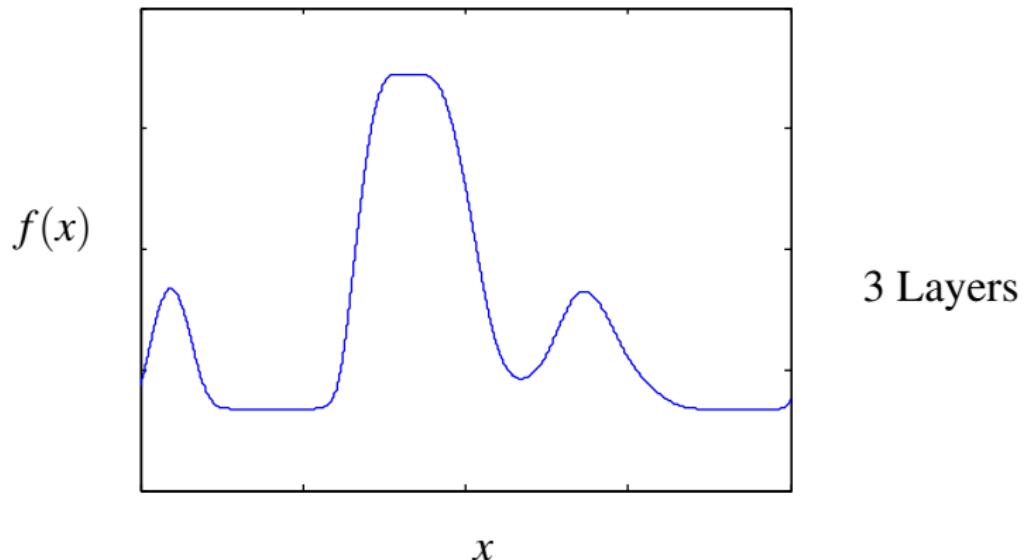
Priors on deep networks

- ▶ A draw from a one-neuron-per-layer deep GP:



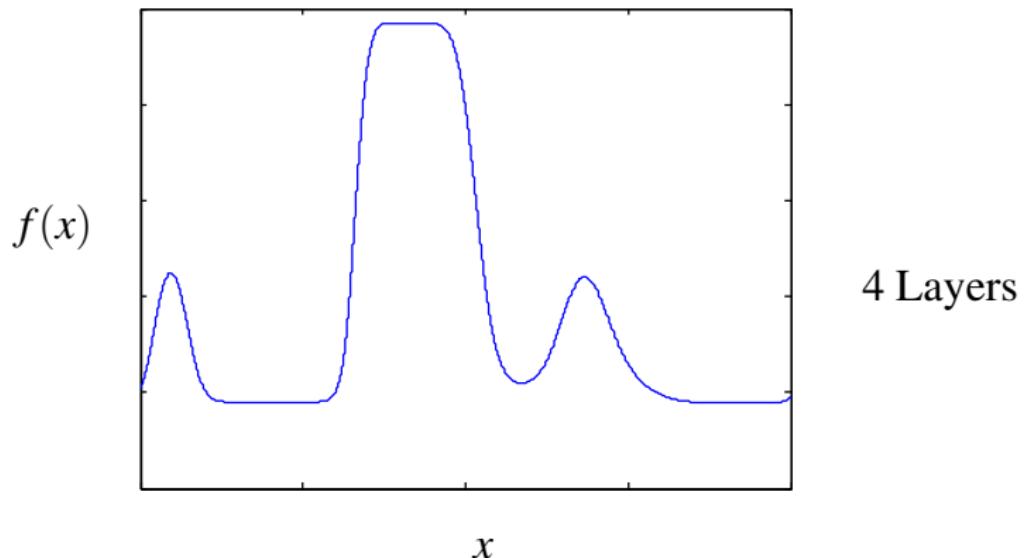
Priors on deep networks

- ▶ A draw from a one-neuron-per-layer deep GP:



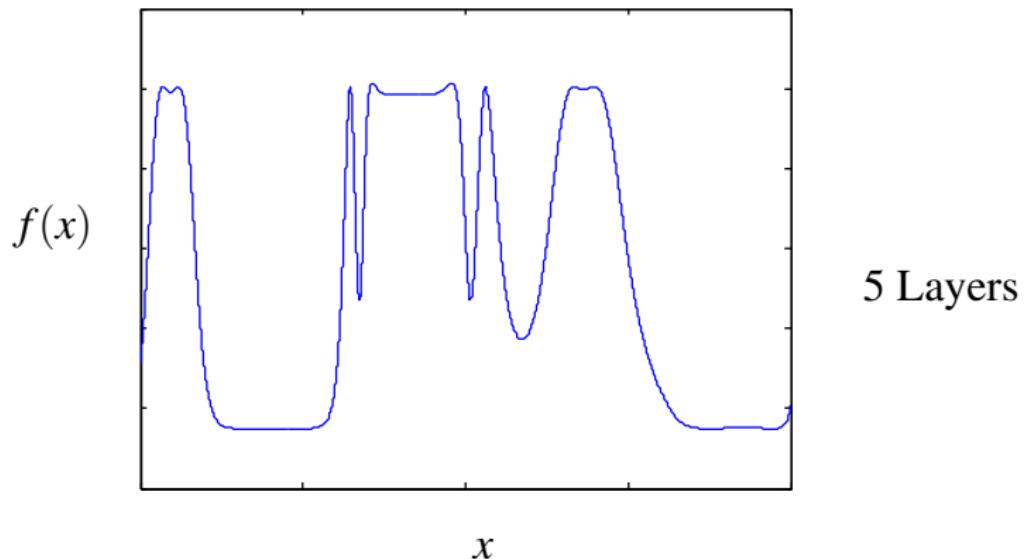
Priors on deep networks

- ▶ A draw from a one-neuron-per-layer deep GP:



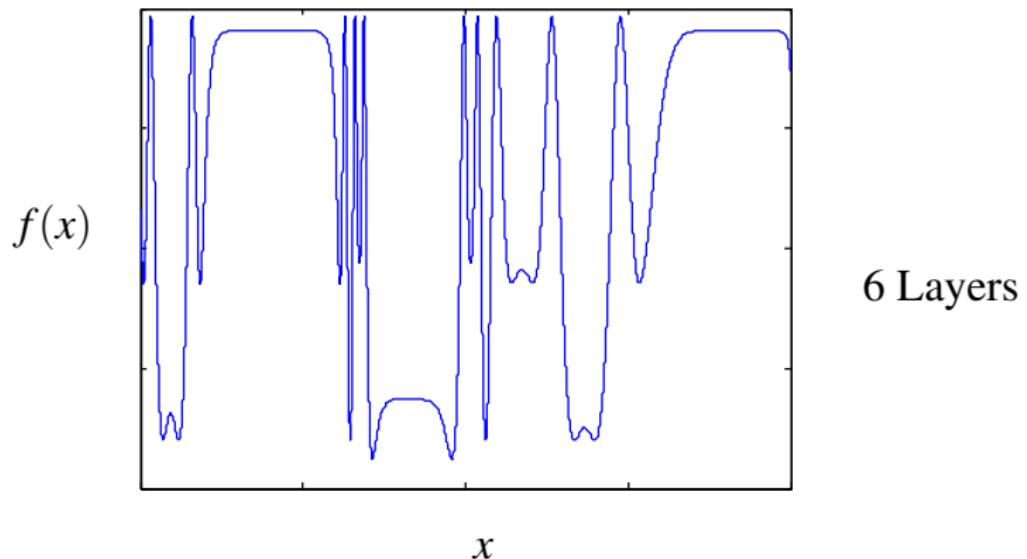
Priors on deep networks

- ▶ A draw from a one-neuron-per-layer deep GP:



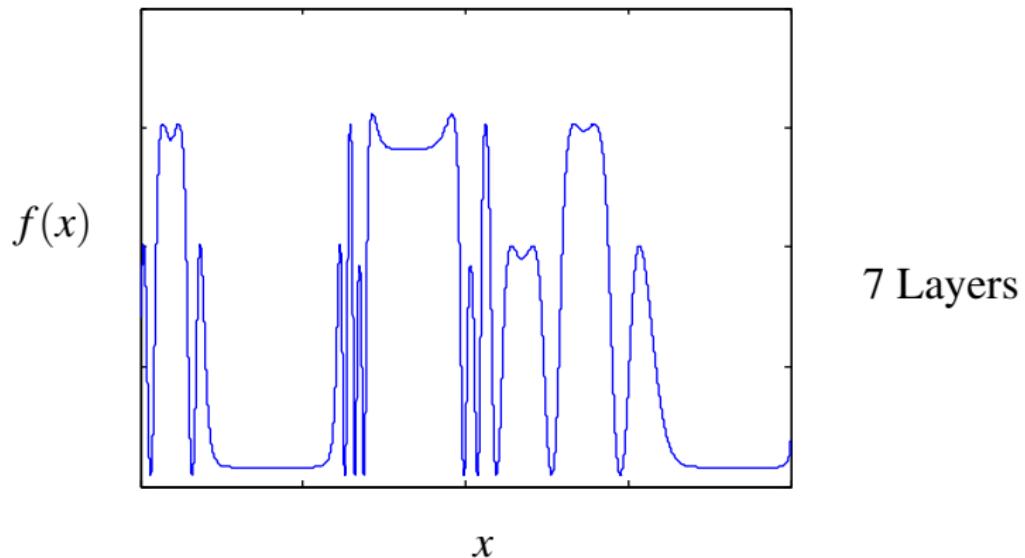
Priors on deep networks

- ▶ A draw from a one-neuron-per-layer deep GP:



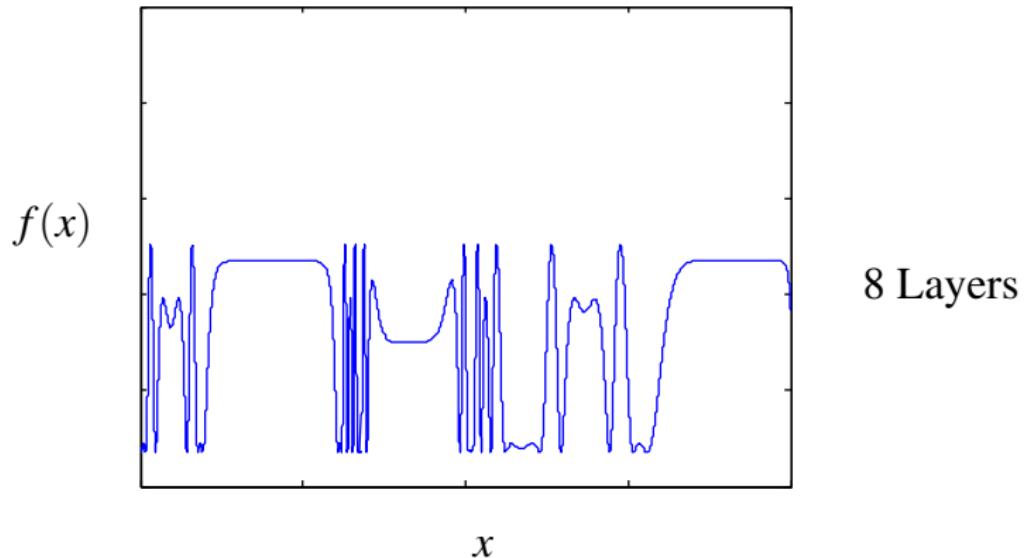
Priors on deep networks

- ▶ A draw from a one-neuron-per-layer deep GP:



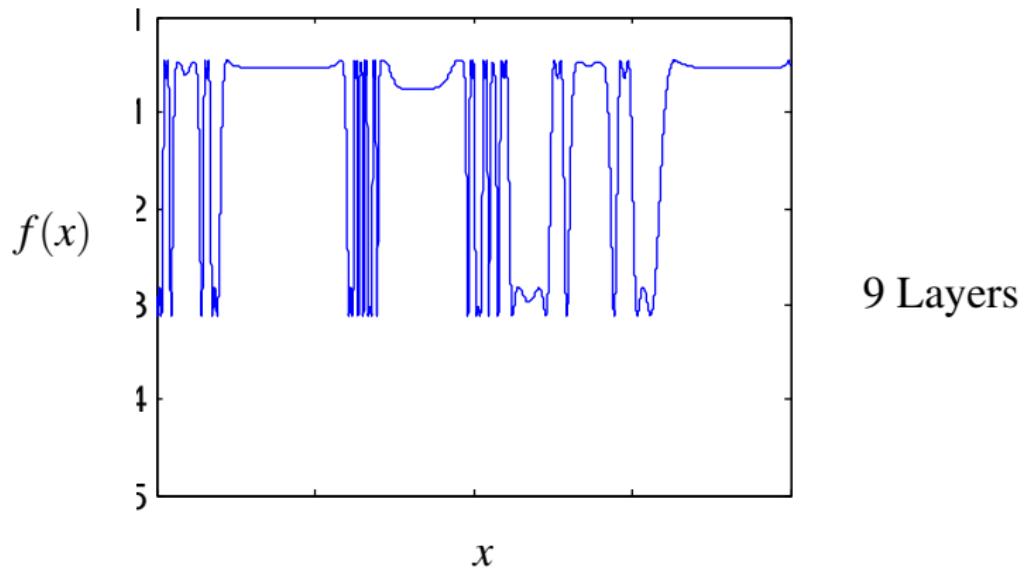
Priors on deep networks

- ▶ A draw from a one-neuron-per-layer deep GP:



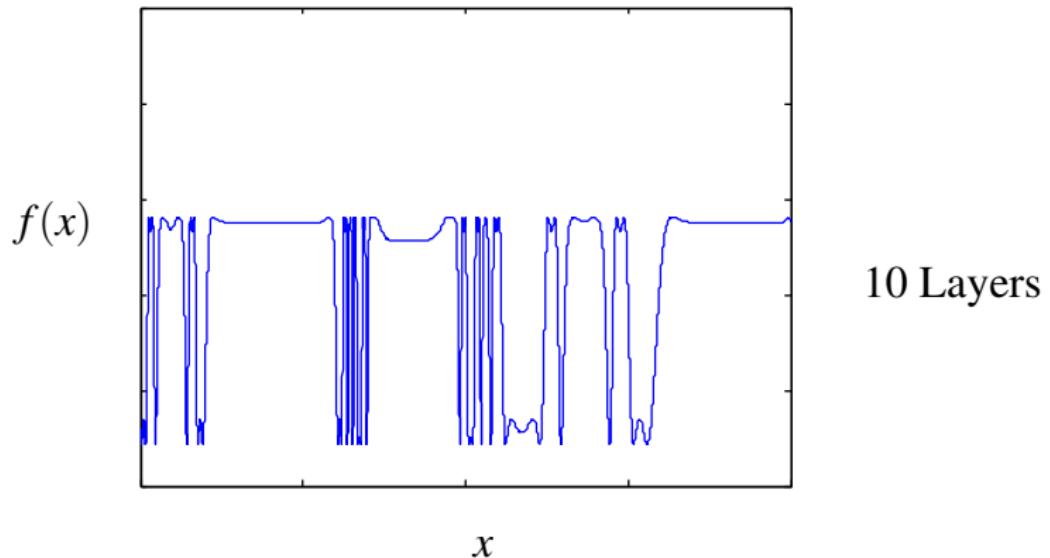
Priors on deep networks

- ▶ A draw from a one-neuron-per-layer deep GP:



Priors on deep networks

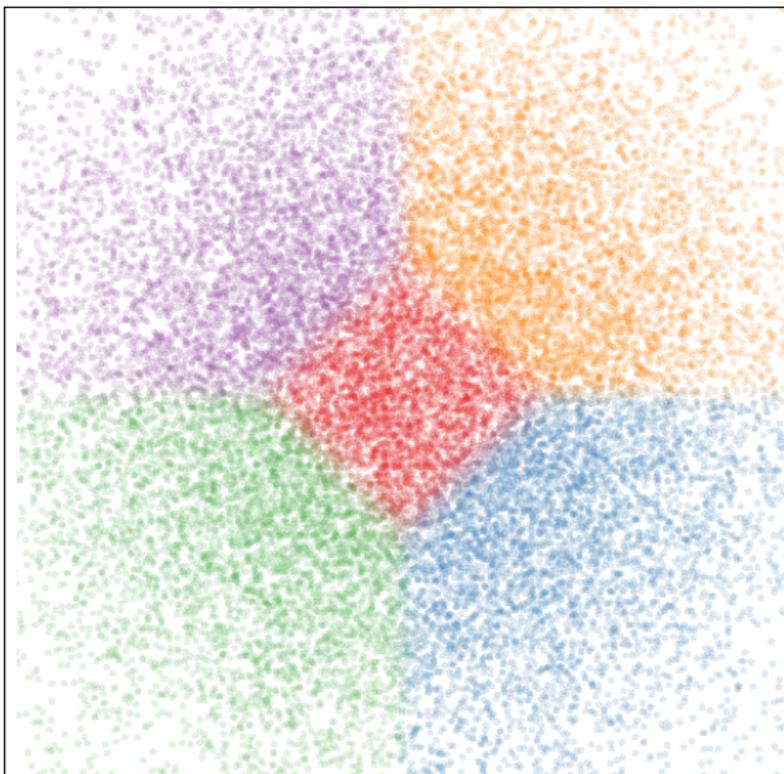
- ▶ A draw from a one-neuron-per-layer deep GP:



Size of derivative becomes log-normal distributed.

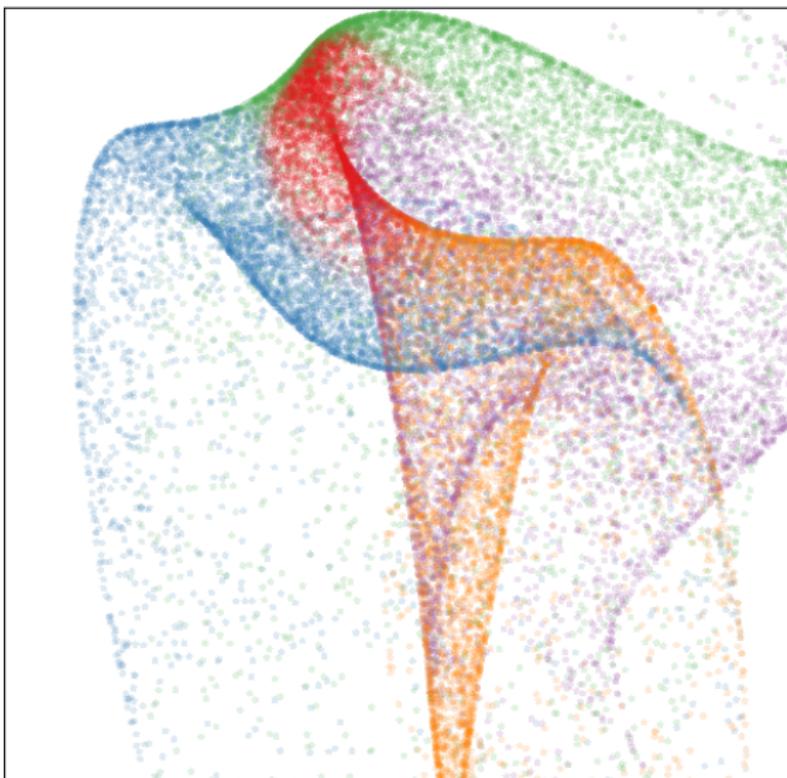
Priors on deep networks

- ▶ 2D to 2D warpings of a set of coloured points:



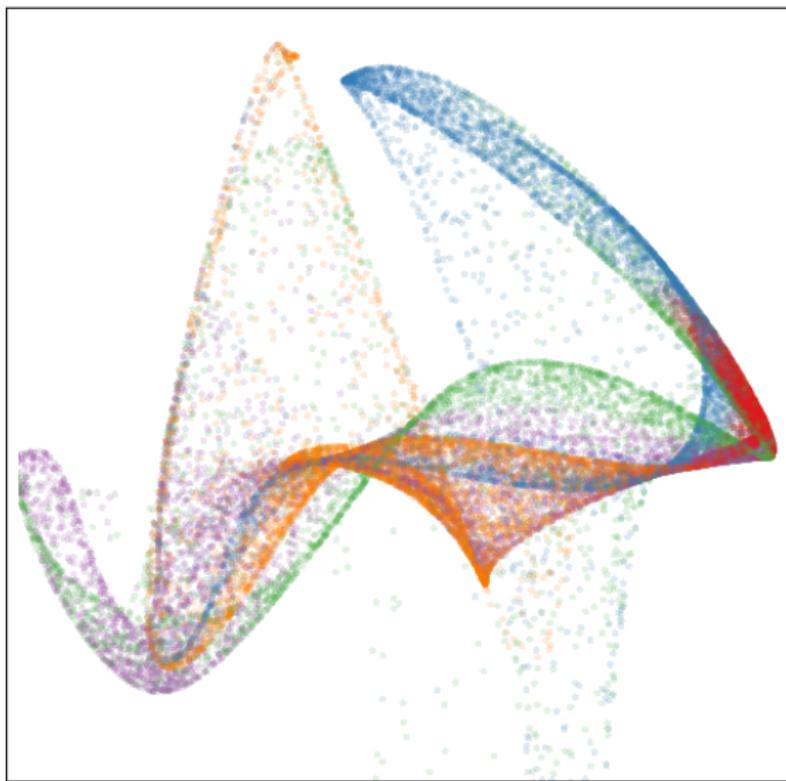
Priors on deep networks

- ▶ 2D to 2D warpings of a set of coloured points:



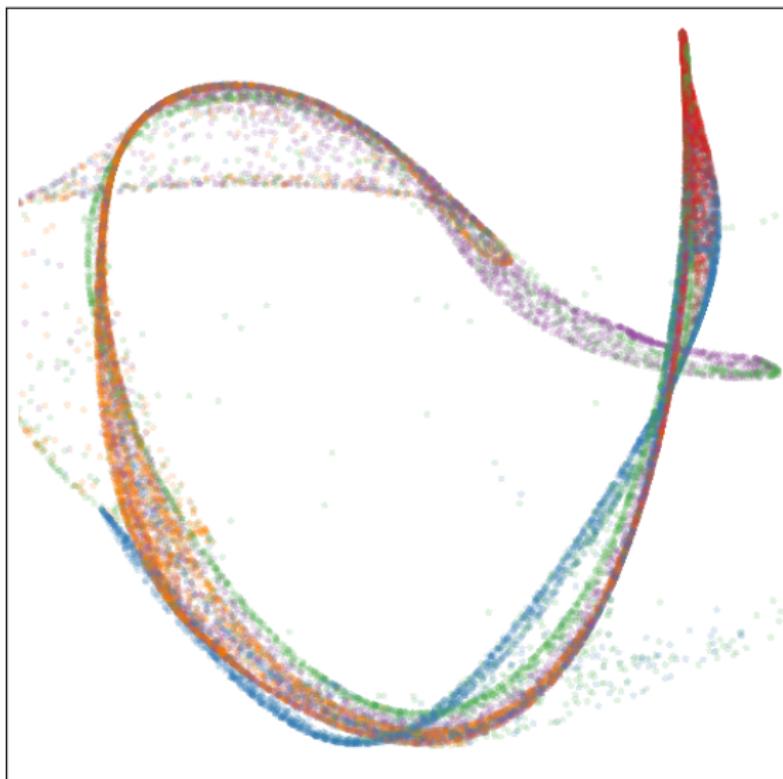
Priors on deep networks

- ▶ 2D to 2D warpings of a set of coloured points:



Priors on deep networks

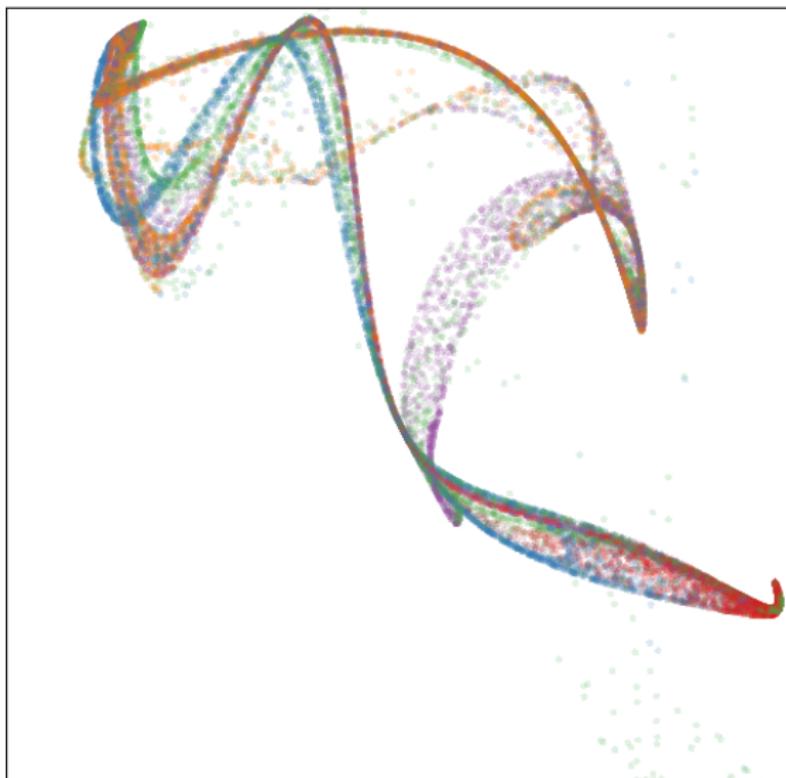
- ▶ 2D to 2D warpings of a set of coloured points:



3 Layers

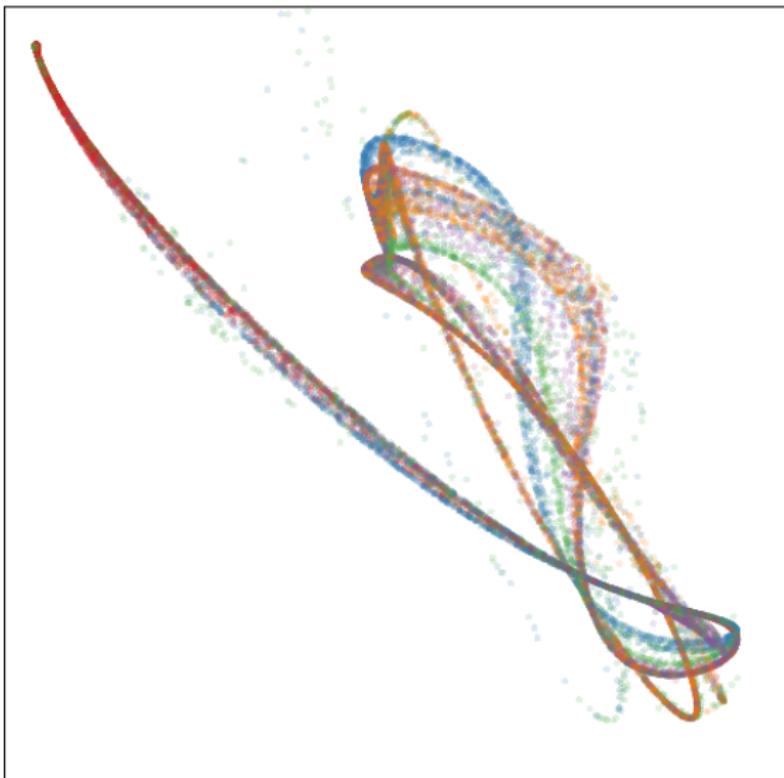
Priors on deep networks

- ▶ 2D to 2D warpings of a set of coloured points:



Priors on deep networks

- ▶ 2D to 2D warpings of a set of coloured points:



5 Layers

Density concentrates along filaments.

Priors on deep networks

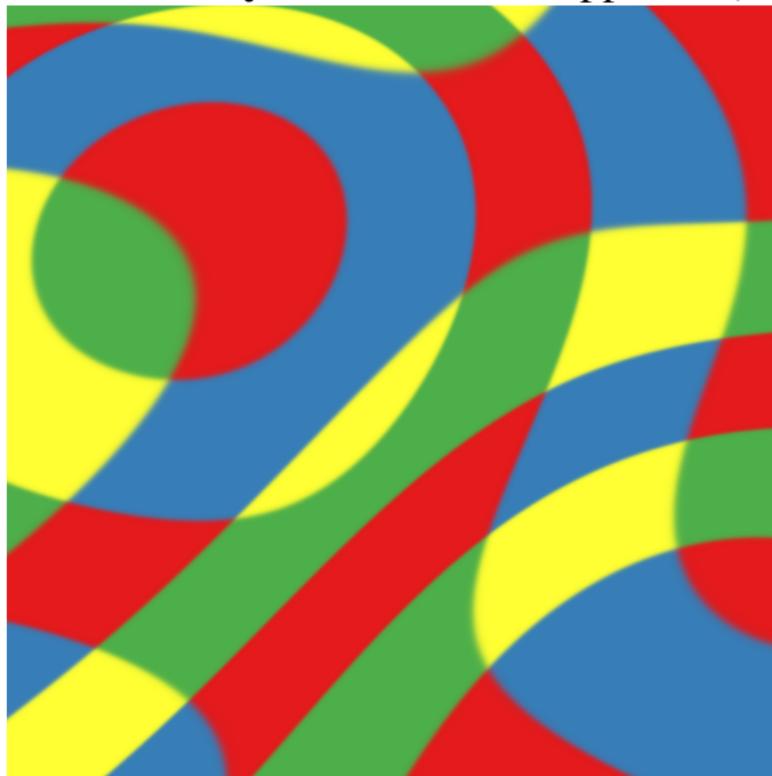
Color shows y that each x is mapped to (decision boundary)



No warping

Priors on deep networks

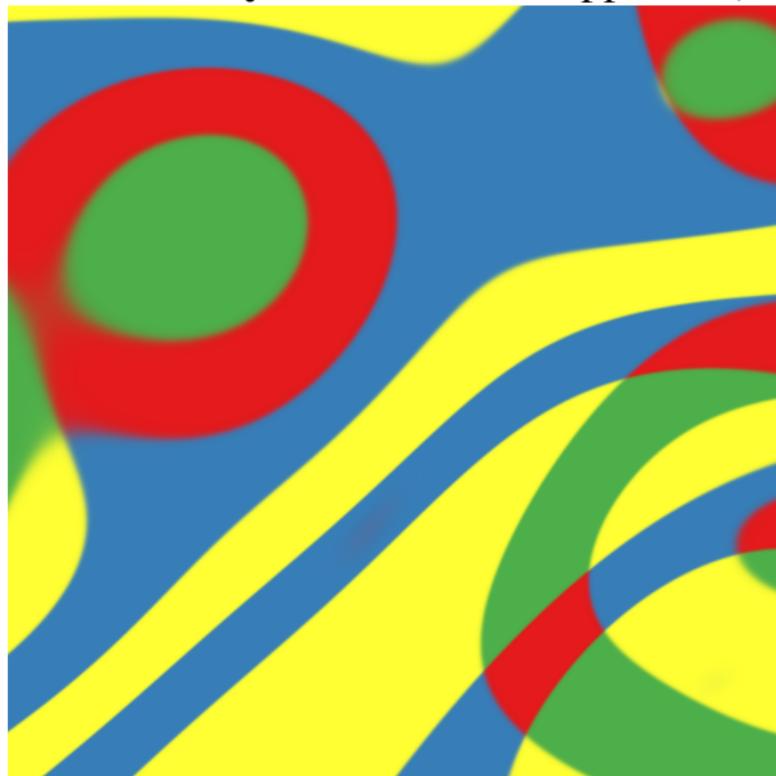
Color shows y that each x is mapped to (decision boundary)



1 Layer

Priors on deep networks

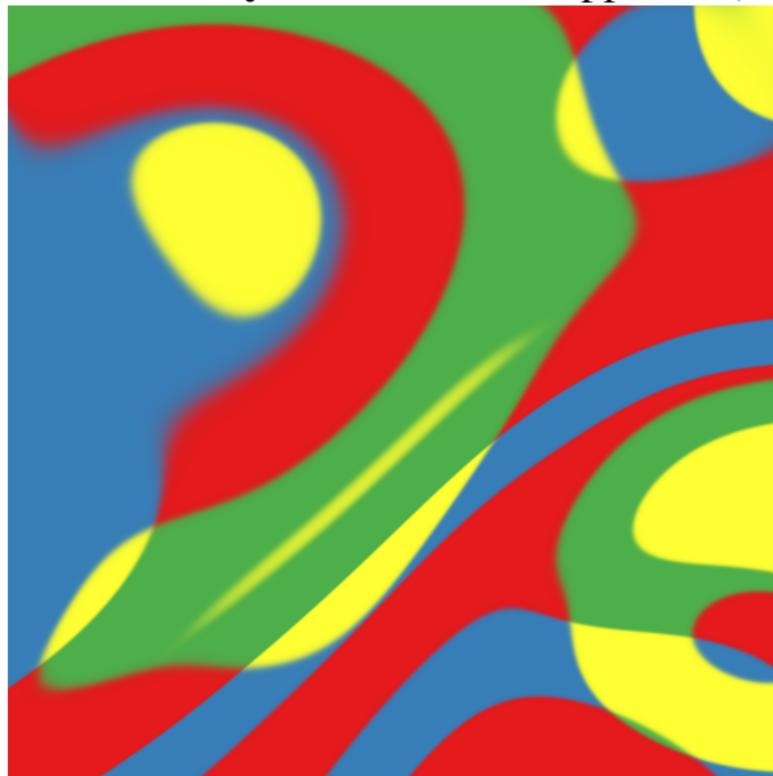
Color shows y that each x is mapped to (decision boundary)



2 Layers

Priors on deep networks

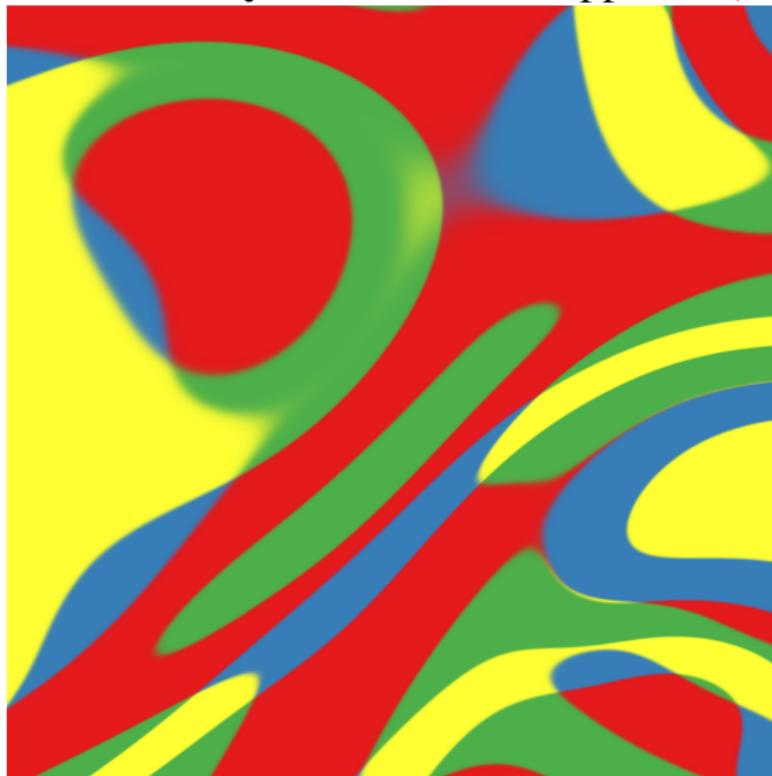
Color shows y that each x is mapped to (decision boundary)



3 Layers

Priors on deep networks

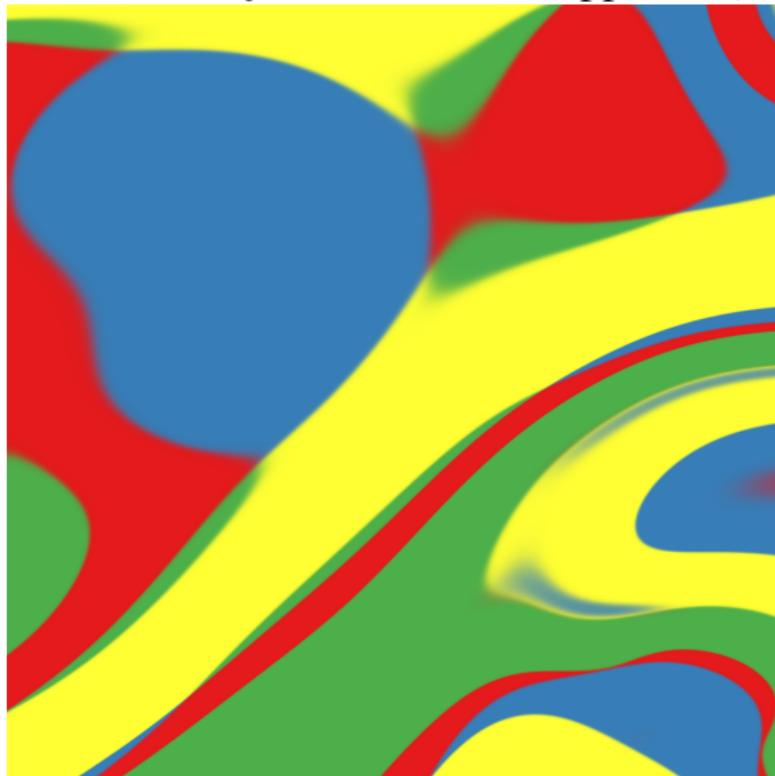
Color shows y that each x is mapped to (decision boundary)



4 Layers

Priors on deep networks

Color shows y that each x is mapped to (decision boundary)



5 Layers

Priors on deep networks

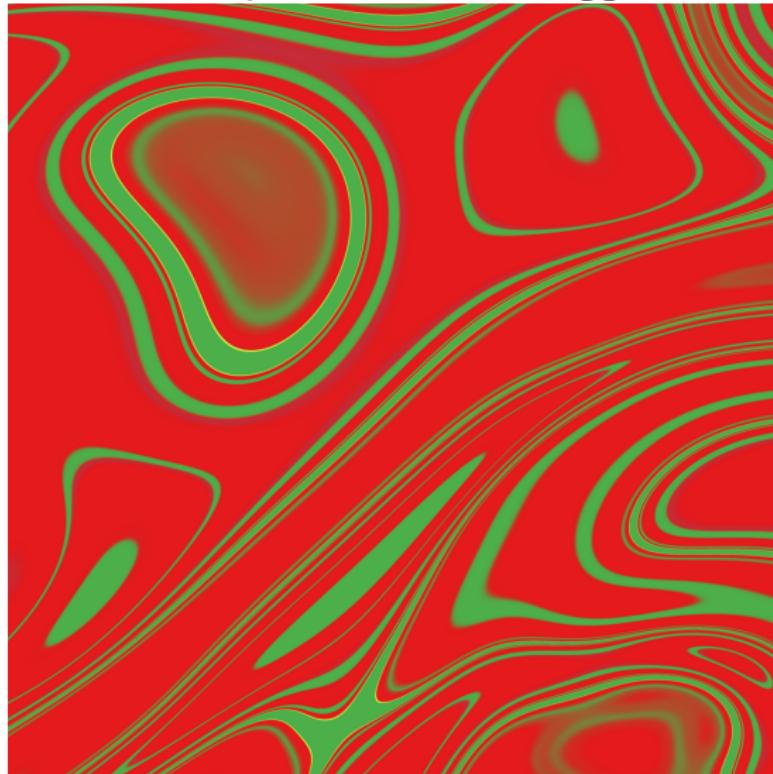
Color shows y that each x is mapped to (decision boundary)



10 Layers

Priors on deep networks

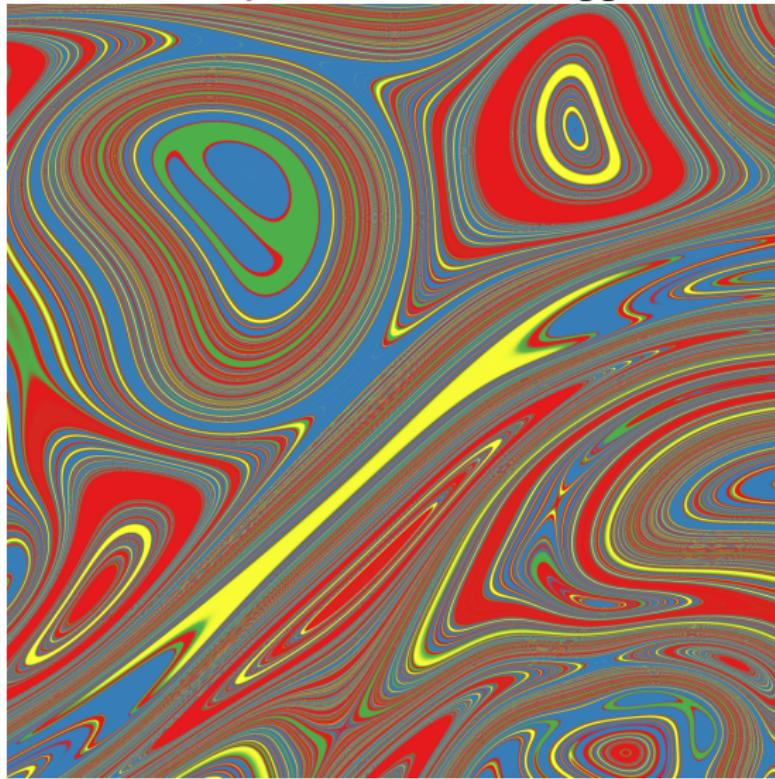
Color shows y that each x is mapped to (decision boundary)



20 Layers

Priors on deep networks

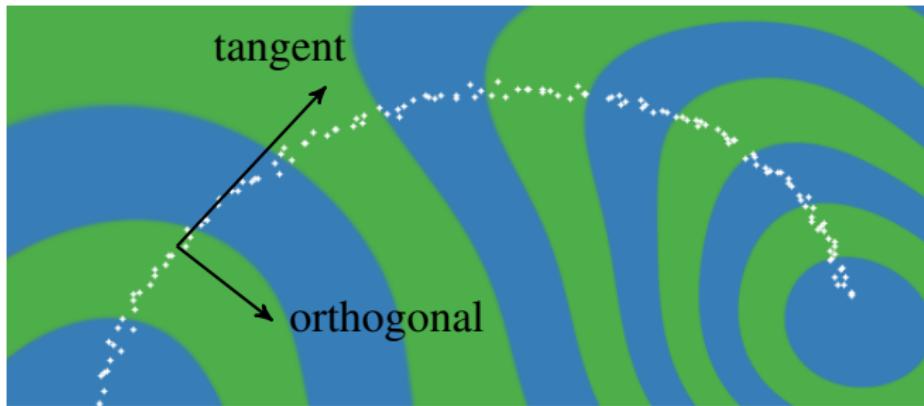
Color shows y that each x is mapped to (decision boundary)



40 Layers

Representation only changes in one direction locally.

What makes a good representation?

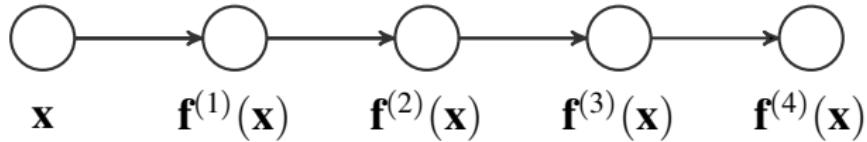


- ▶ Good representations of data manifolds don't change in directions orthogonal to the manifold. ([Rifai et. al. 2011](#))
- ▶ Good representations also change in directions tangent to the manifold, to preserve information.
- ▶ Representation of a D -dimensional manifold should change in D orthogonal directions, locally.
- ▶ Our prior on functions might be too restrictive.

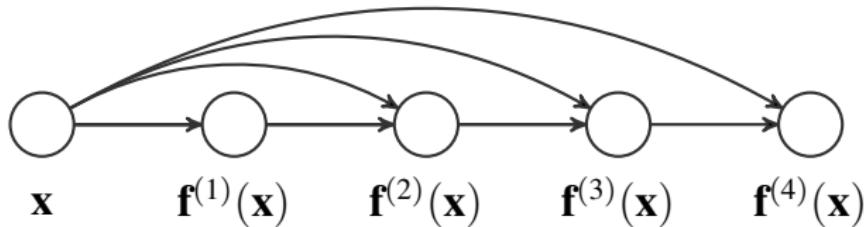
A simple fix

- ▶ Following a suggestion from [Neal \(1995\)](#), we connect the inputs \mathbf{x} to each layer:

Standard architecture:

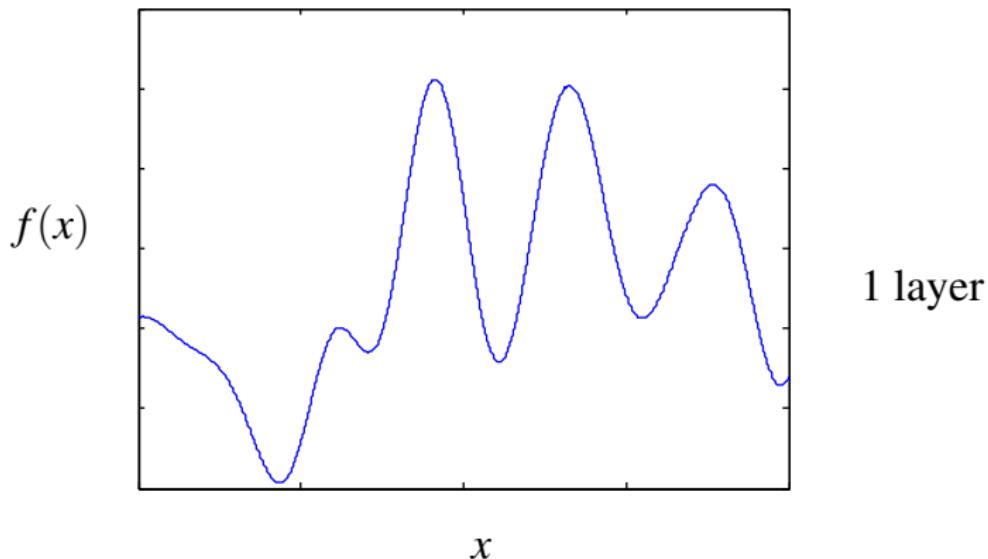


Input-connected architecture:



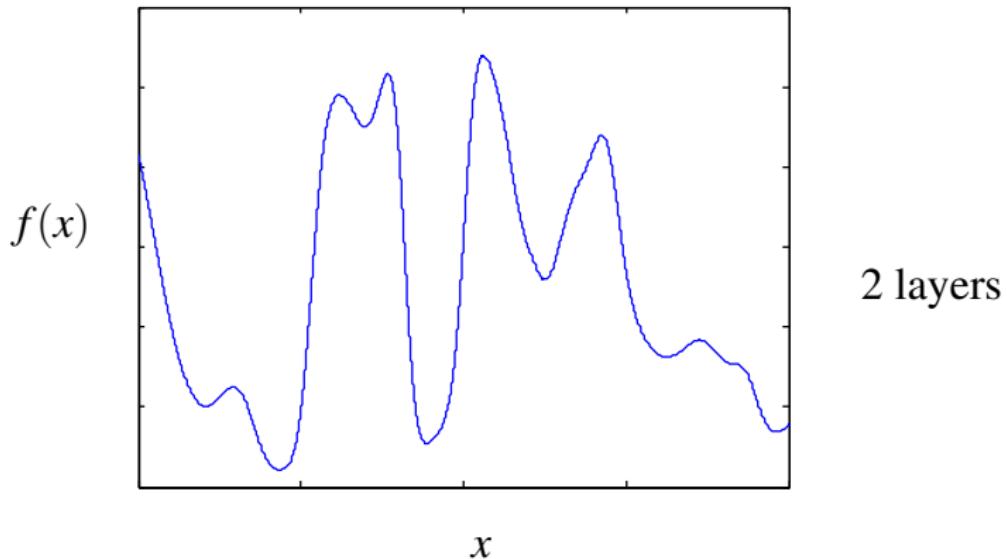
A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



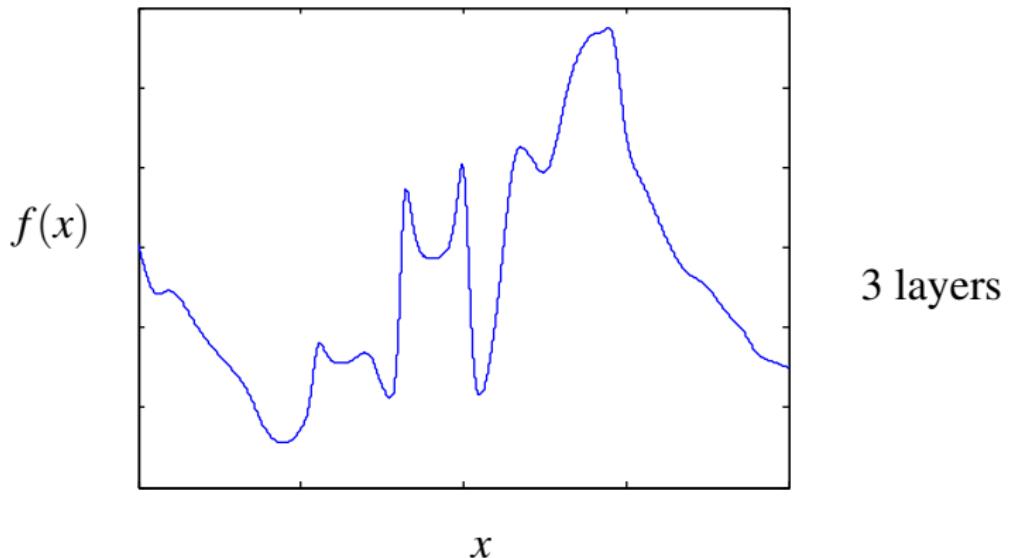
A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



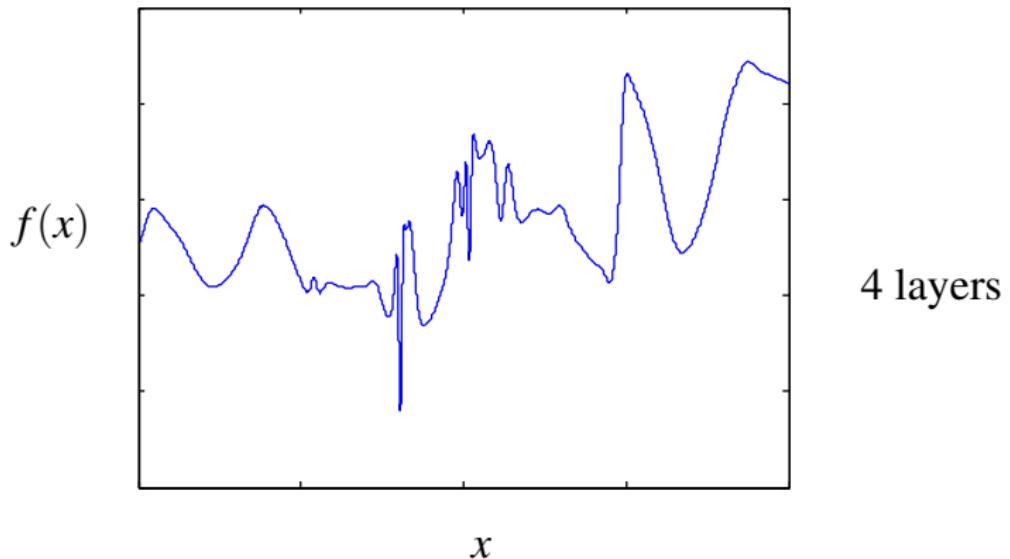
A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



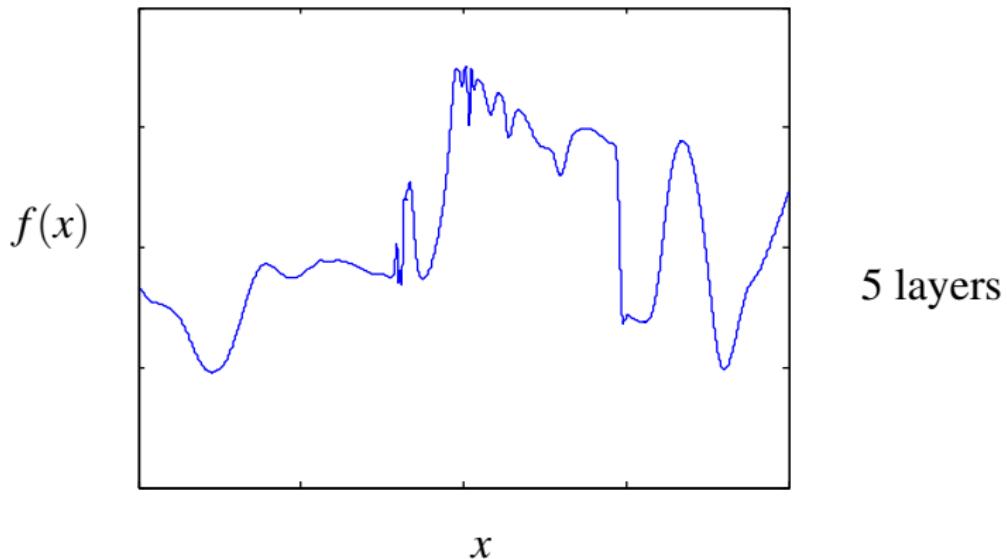
A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



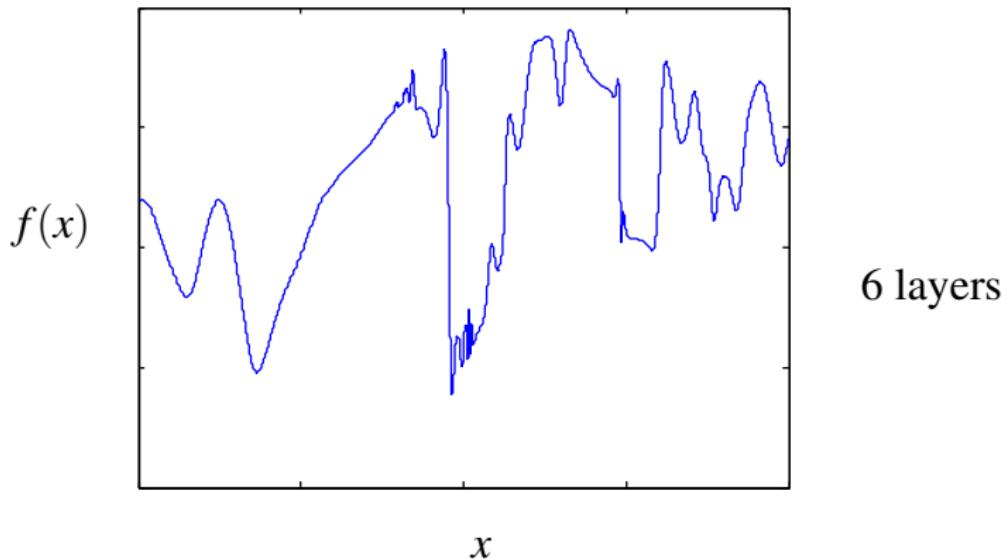
A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



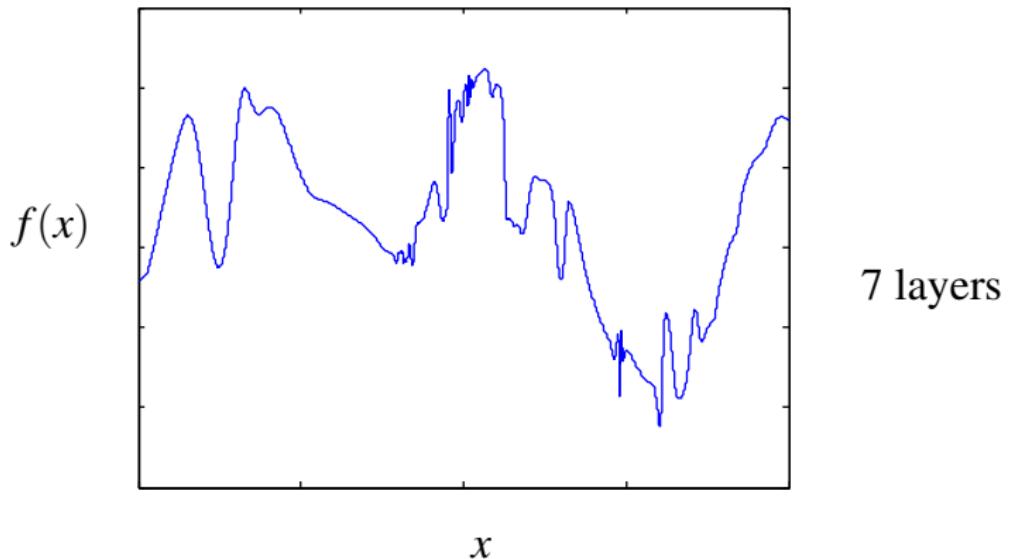
A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



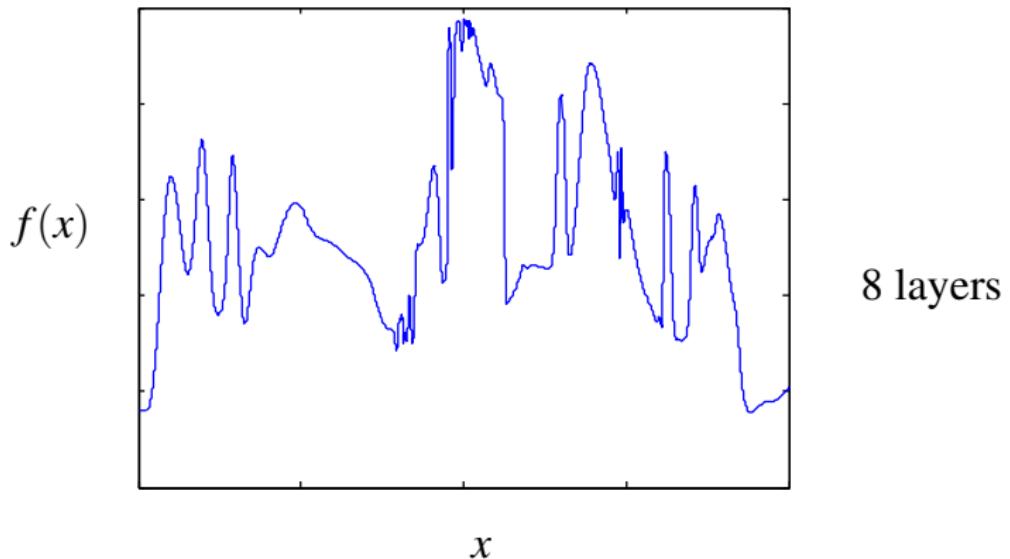
A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



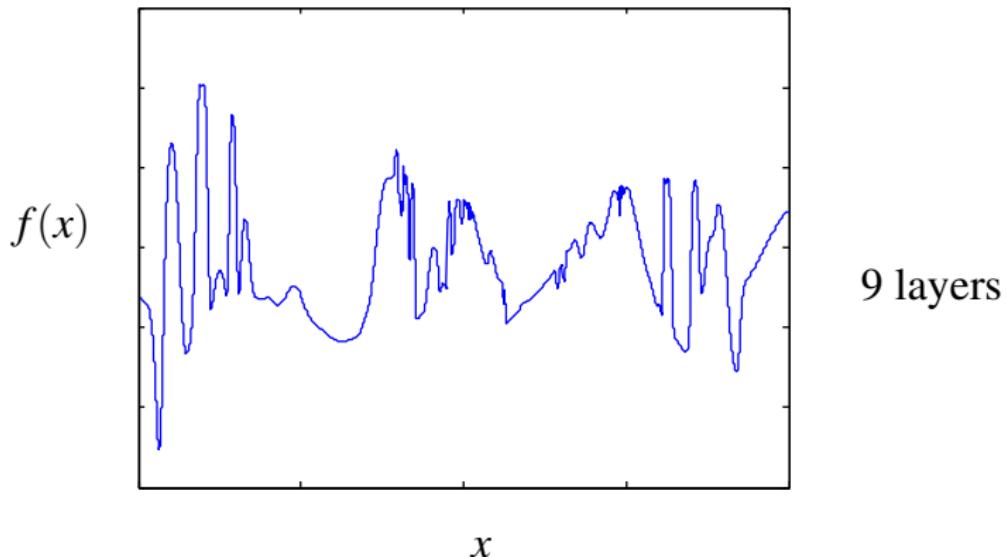
A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



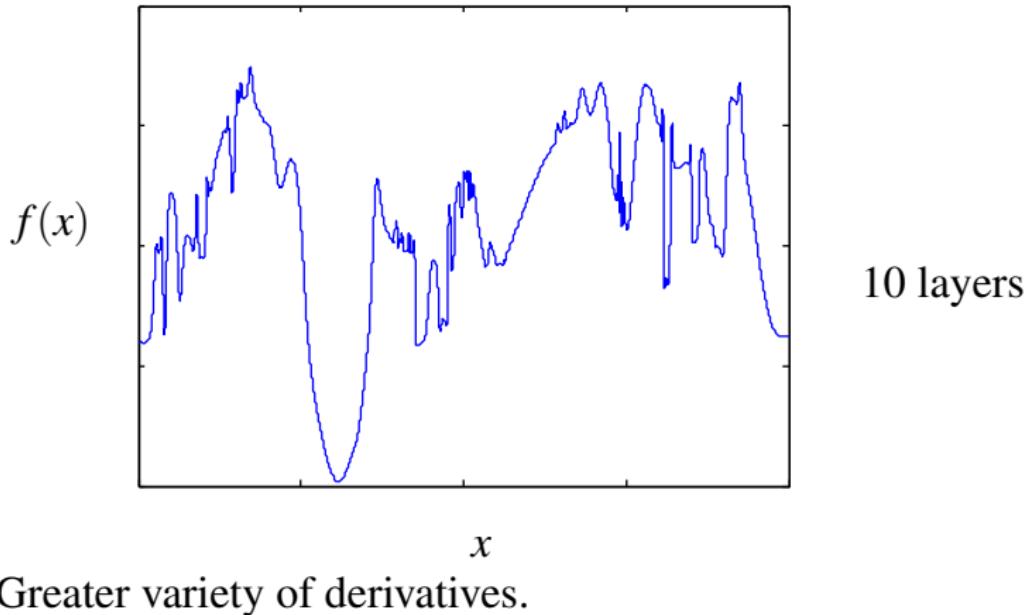
A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



A different architecture

- ▶ A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



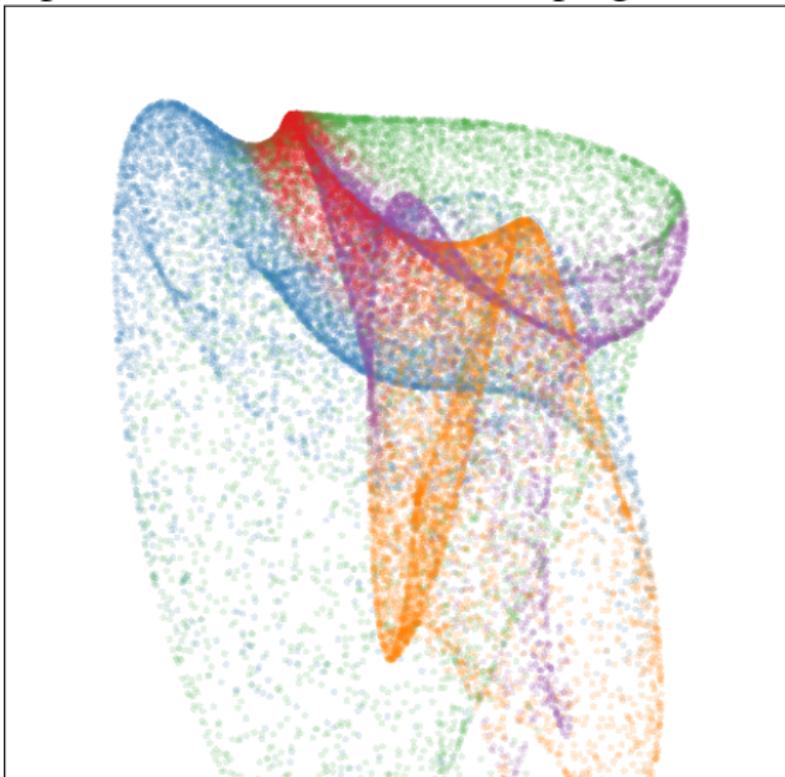
A different architecture

- ▶ Input-connected 2D to 2D warpings of coloured points:



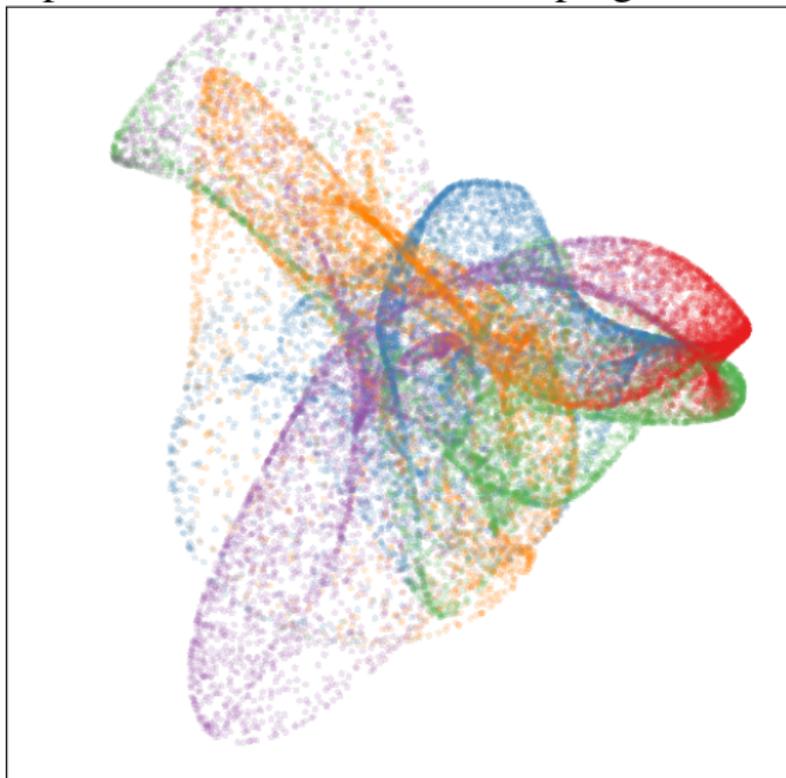
A different architecture

- ▶ Input-connected 2D to 2D warpings of coloured points:



A different architecture

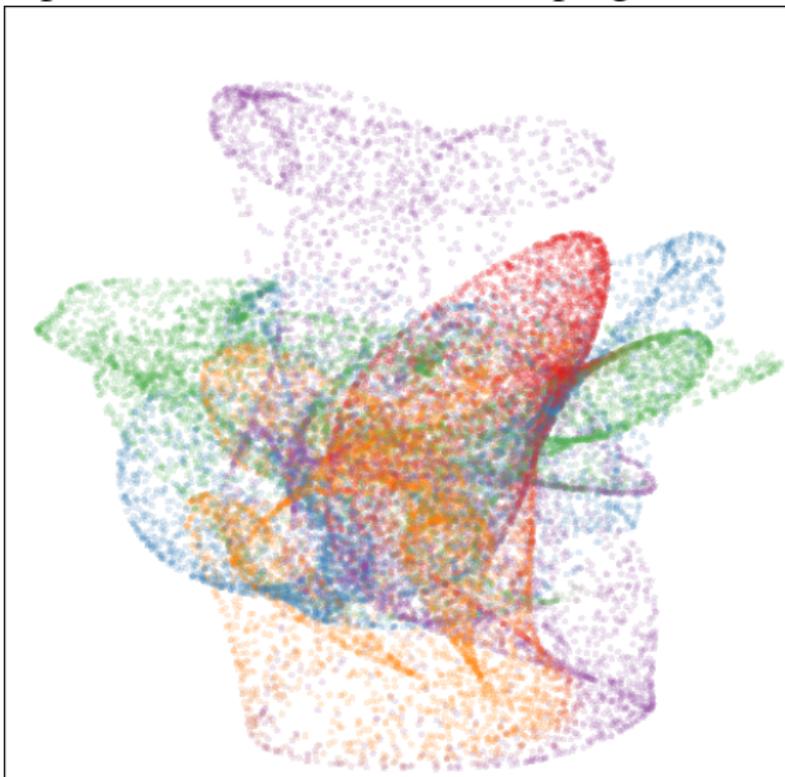
- ▶ Input-connected 2D to 2D warpings of coloured points:



2 Layers

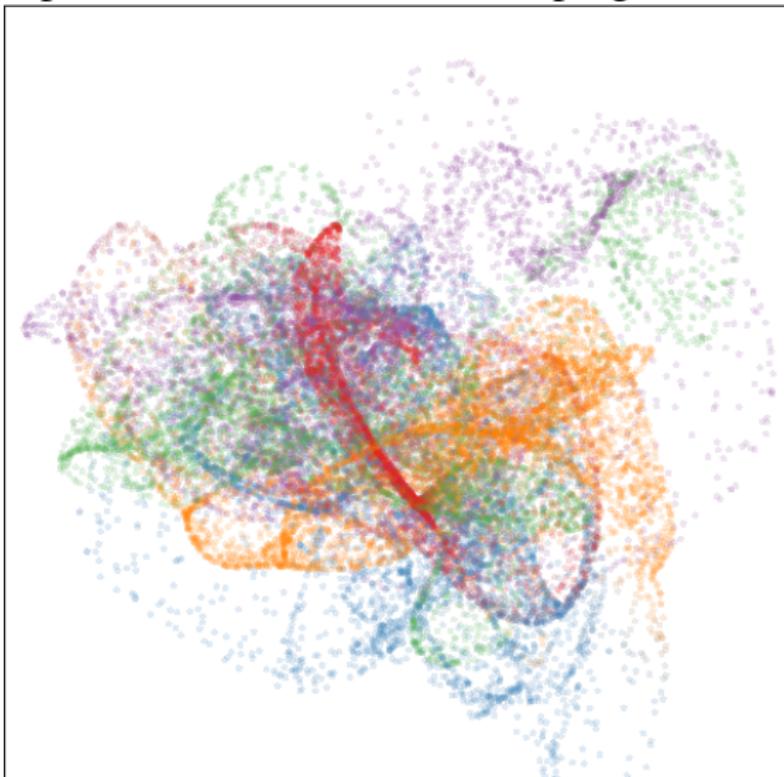
A different architecture

- ▶ Input-connected 2D to 2D warpings of coloured points:



A different architecture

- ▶ Input-connected 2D to 2D warpings of coloured points:



A different architecture

- ▶ Input-connected 2D to 2D warpings of coloured points:



5 Layers

Density becomes more complex but remains 2D.

A different architecture

- ▶ Color shows y that each x is mapped to



No warping

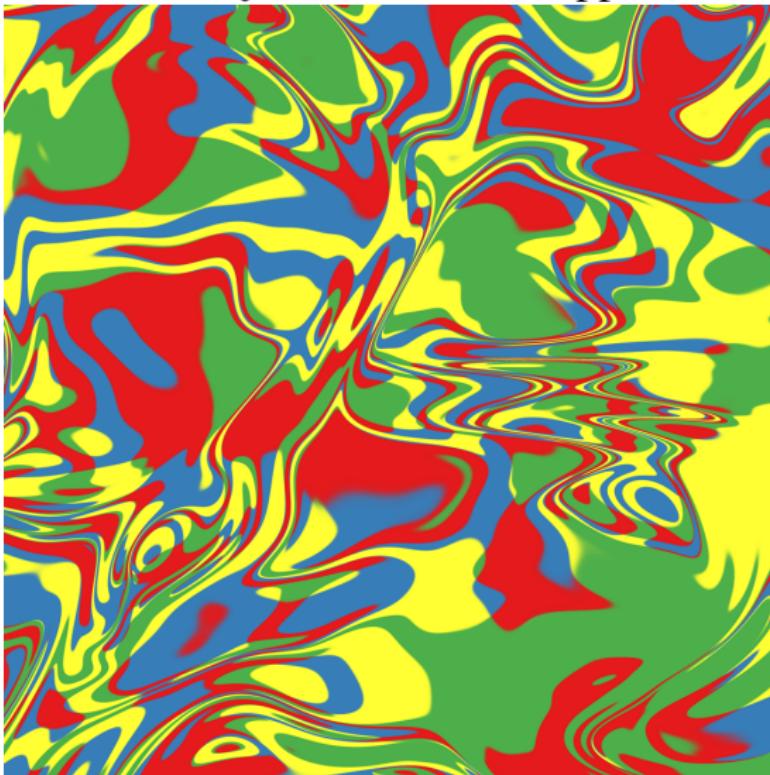
A different architecture

- ▶ Color shows y that each x is mapped to



A different architecture

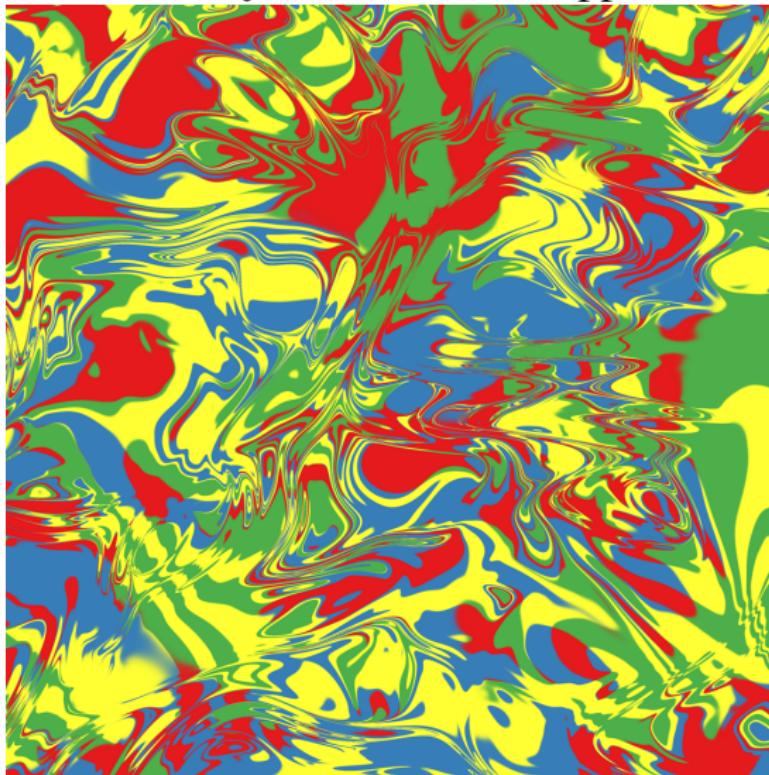
- ▶ Color shows y that each x is mapped to



10 Layers

A different architecture

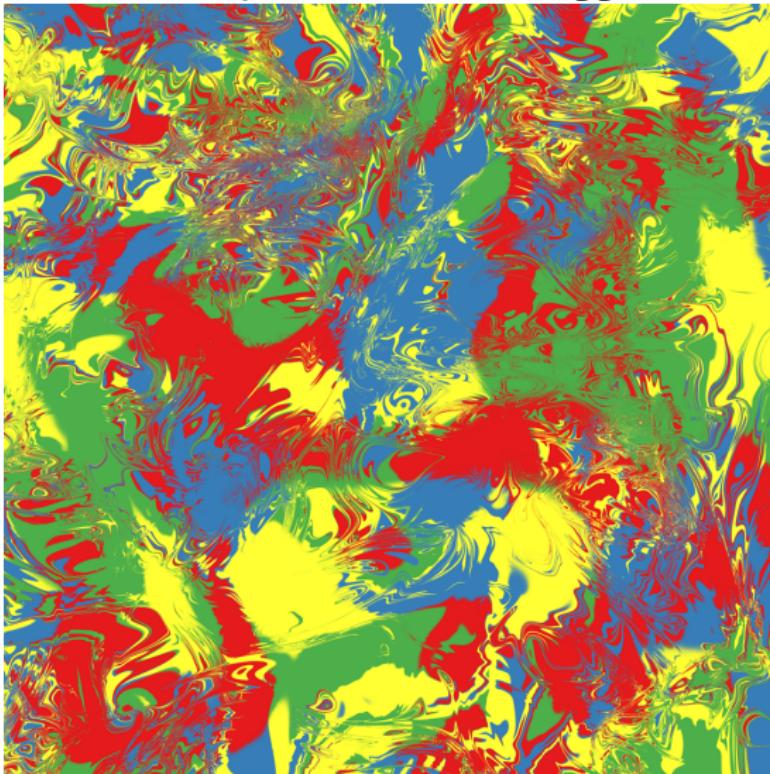
- ▶ Color shows y that each x is mapped to



20 Layers

A different architecture

- ▶ Color shows y that each x is mapped to



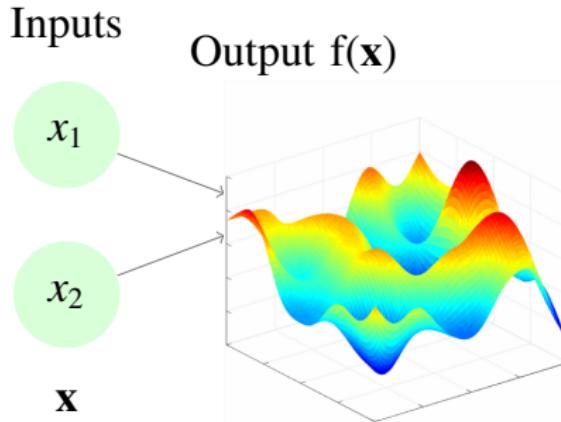
40 Layers

Representation sometimes depends on all directions.

Understanding dropout

- ▶ Dropout is a method for regularizing neural networks (Hinton et al., 2012; Srivastava, 2013).
- ▶ Recipe:
 1. Randomly set to zero (drop out) some neuron activations.
 2. Average over all possible ways of doing this.
- ▶ Gives robustness since neurons can't depend on each other.
- ▶ How does dropout affect priors on functions?
- ▶ Related work: (Baldi and Sadowski, 2013; Cho, 2013; Wager, Wang and Liang, 2013)

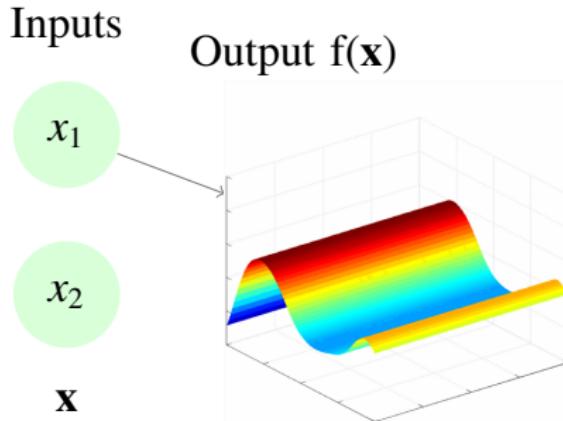
Dropout in shallow Gaussian processes



- ▶ Each function only depends on some input dimensions.
- ▶ Given prior covariance $\text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}')$, exact dropout gives a mixture of GPs:

$$p(f(\mathbf{x})) = \frac{1}{2^D} \sum_{\mathbf{r} \in \{0,1\}^D} \text{GP}\left(0, k(\mathbf{r}^\top \mathbf{x}, \mathbf{r}^\top \mathbf{x}')\right)$$

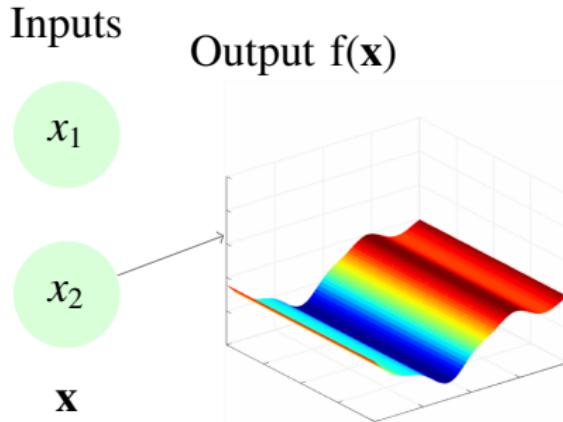
Dropout in shallow Gaussian processes



- ▶ Each function only depends on some input dimensions.
- ▶ Given prior covariance $\text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}')$, exact dropout gives a mixture of GPs:

$$p(f(\mathbf{x})) = \frac{1}{2^D} \sum_{\mathbf{r} \in \{0,1\}^D} \text{GP}\left(0, k(\mathbf{r}^\top \mathbf{x}, \mathbf{r}^\top \mathbf{x}')\right)$$

Dropout in shallow Gaussian processes



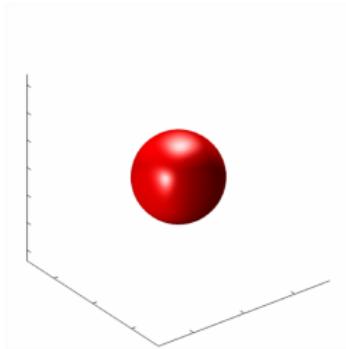
- ▶ Each function only depends on some input dimensions.
- ▶ Given prior covariance $\text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}')$, exact dropout gives a mixture of GPs:

$$p(f(\mathbf{x})) = \frac{1}{2^D} \sum_{\mathbf{r} \in \{0,1\}^D} \text{GP}\left(0, k(\mathbf{r}^\top \mathbf{x}, \mathbf{r}^\top \mathbf{x}')\right)$$

Covariance before and after dropout

Original squared-exp:

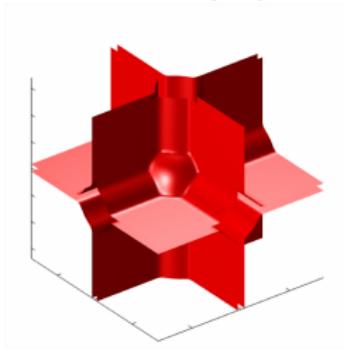
$$\text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}')$$



$$\mathbf{x} - \mathbf{x}'$$

After dropout:

$$\text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = \sum_{\mathbf{r} \in \{0,1\}^D} k(\mathbf{r}^\top \mathbf{x}, \mathbf{r}^\top \mathbf{x}')$$



$$\mathbf{x} - \mathbf{x}'$$

- ▶ Output similar even if some input dimensions change a lot.

Summary

- ▶ Priors on functions can shed light on design choices in a data-independent way.
- ▶ Example 1: Increasing depth makes net outputs change in fewer directions.
- ▶ Example 2: Dropout makes output similar even if some inputs change a lot.
- ▶ What sorts of structures do we want to be able to learn?
- ▶ Code at github.com/duvenaud/deep-limits

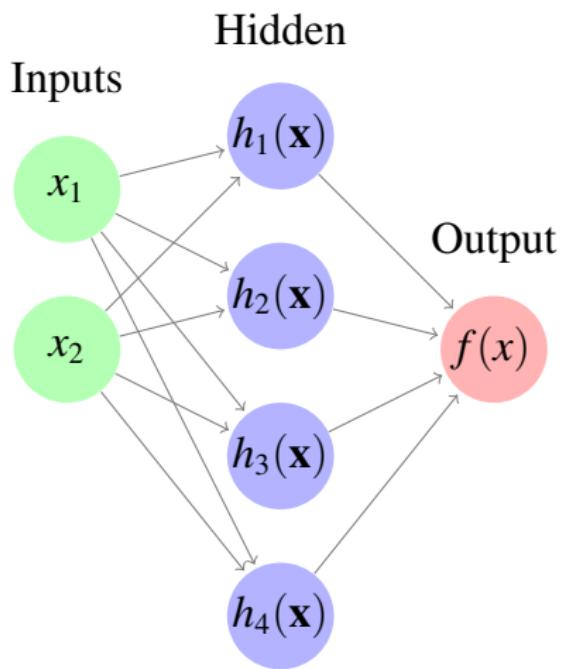
Summary

- ▶ Priors on functions can shed light on design choices in a data-independent way.
- ▶ Example 1: Increasing depth makes net outputs change in fewer directions.
- ▶ Example 2: Dropout makes output similar even if some inputs change a lot.
- ▶ What sorts of structures do we want to be able to learn?
- ▶ Code at github.com/duvenaud/deep-limits

Thanks!

Extra slides

GPs as Neural Nets



A weighted sum of features,

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K w_i h_i(\mathbf{x})$$

with any weight distribution,

$$\mathbb{E}[w_i] = 0, \quad \mathbb{V}[w_i] = \sigma^2, \quad i.i.d.$$

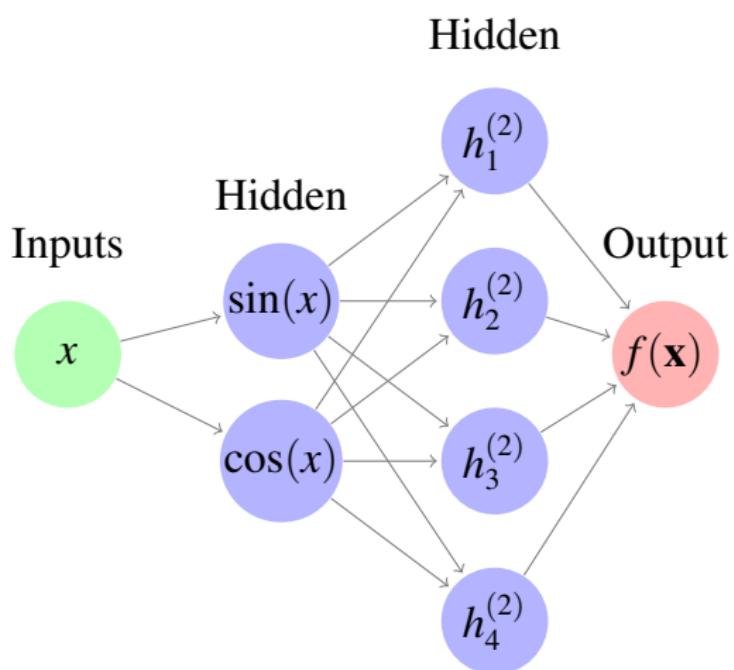
by CLT, gives a GP as $K \rightarrow \infty$!

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Kernel learning as feature learning

- ▶ GPs have fixed features, integrate out feature weights.
- ▶ Mapping between kernels and features:
 $k(\mathbf{x}, \mathbf{x}') = \mathbf{h}(\mathbf{x})^\top \mathbf{h}(\mathbf{x}').$
- ▶ Any PSD kernel can be written as inner product of features. (Mercer's Theorem)
- ▶ Kernel learning = feature learning
- ▶ What if we make the GP neural network deep?

Example deep kernel: Periodic



Now our model is:

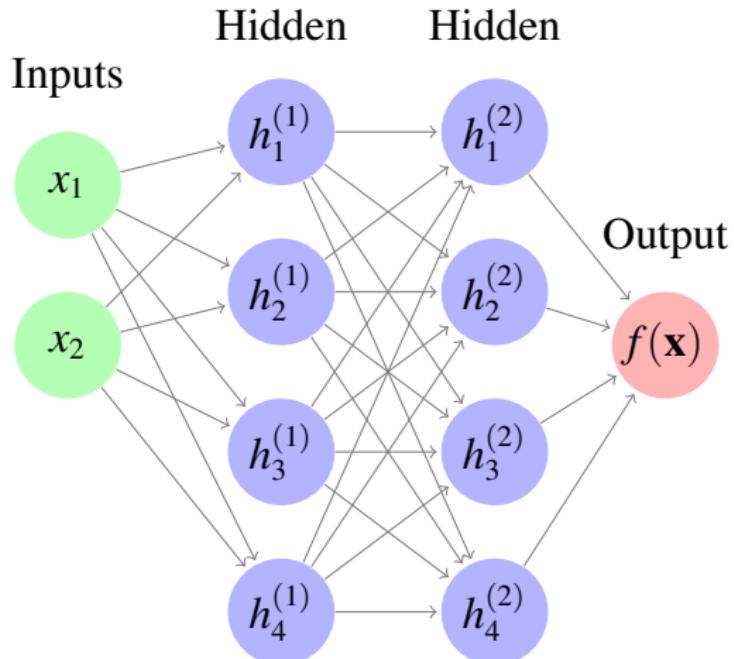
$$\mathbf{h}^1(\mathbf{x}) = [\sin(\mathbf{x}), \cos(\mathbf{x})]$$

we have “deep kernel”:

$$k_2(\mathbf{x}, \mathbf{x}')$$

$$= \exp\left(-\frac{1}{2} (\mathbf{h}^1(\mathbf{x}) - \mathbf{h}^1(\mathbf{x}'))\right)$$

Deep nets, deep kernels



Now our model is:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K w_i h_i^{(2)}(\mathbf{h}^{(1)}(\mathbf{x}))$$
$$= \mathbf{w}^\top \mathbf{h}^{(2)}(\mathbf{h}^{(1)}(\mathbf{x}))$$

Instead of

$$k_1(\mathbf{x}, \mathbf{x}') = \mathbf{h}^{(1)}(\mathbf{x})^\top \mathbf{h}^{(1)}(\mathbf{x}'),$$

we have “deep kernel”:

$$k_2(\mathbf{x}, \mathbf{x}')$$
$$= [\mathbf{h}^{(2)}(\mathbf{h}^{(1)}(\mathbf{x}))]^\top \mathbf{h}^{(2)}(\mathbf{h}^{(1)}(\mathbf{x}'))$$

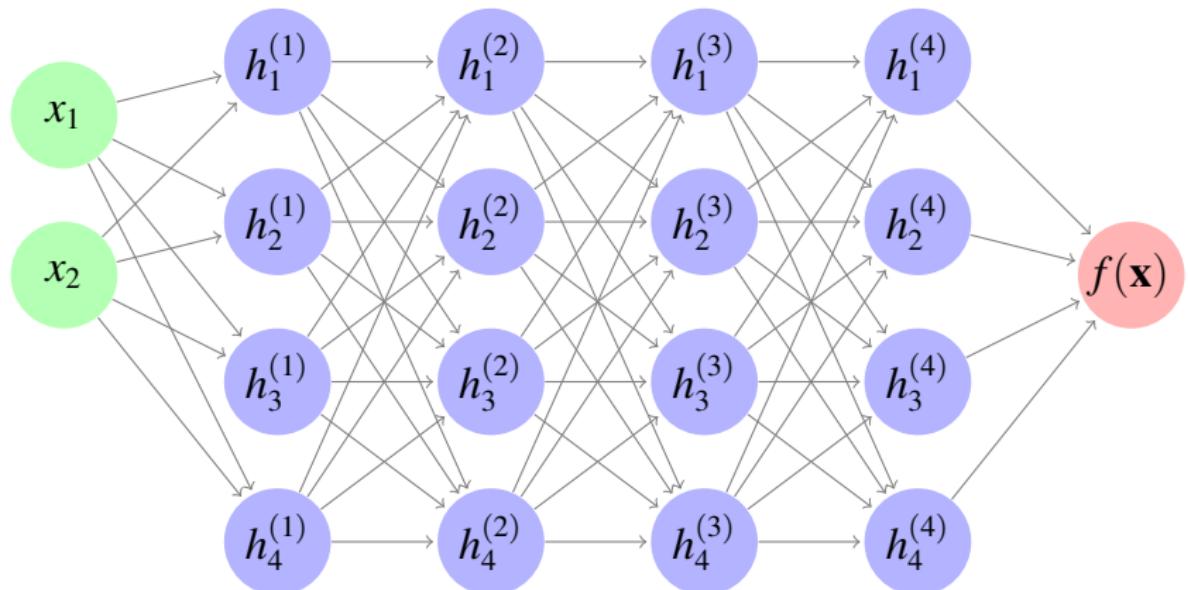
Deep Kernels

- ▶ (Cho, 2012) built kernels by composing feature mappings.
- ▶ Composing any kernel k_1 with a squared-exp kernel (SE):

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

- ▶ A closed form... let's do it again!

We need to go deeper



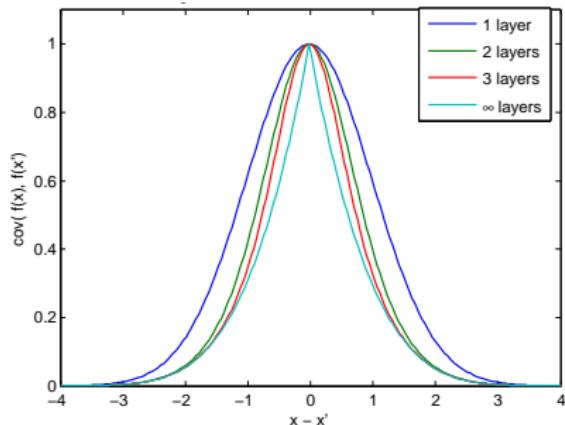
A simple fix...

- ▶ Following a suggestion from [Neal \(1995\)](#), we connect the inputs \mathbf{x} to each layer:

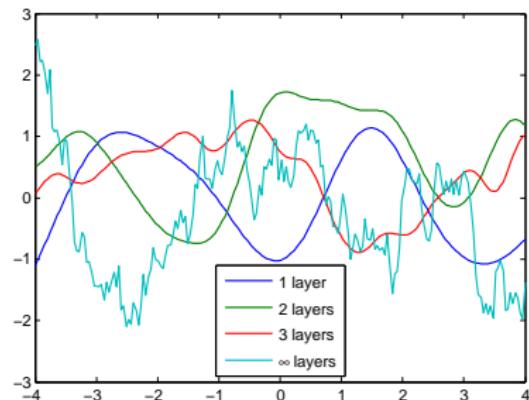
$$\begin{aligned} k_{L+1}(\mathbf{x}, \mathbf{x}') &= \\ &= \exp \left(-\frac{1}{2} \left\| \begin{bmatrix} \mathbf{h}^L(\mathbf{x}) \\ \mathbf{x} \end{bmatrix} - \begin{bmatrix} \mathbf{h}^L(\mathbf{x}') \\ \mathbf{x}' \end{bmatrix} \right\|_2^2 \right) \\ &= \exp \left(-\frac{1}{2} [k_L(\mathbf{x}, \mathbf{x}) - 2k_L(\mathbf{x}, \mathbf{x}') + k_L(\mathbf{x}', \mathbf{x}')] - \frac{1}{2} \|\mathbf{x} - \mathbf{x}'\|_2^2 \right) \end{aligned}$$

Infinitely Deep Kernels

- ▶ What is the limit of compositions of input-connected SE features?
- ▶ $k_{L+1}(\mathbf{x}, \mathbf{x}') = \exp(k_L(\mathbf{x}, \mathbf{x}') - 1 - \frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|_2^2)$.



Kernels



Draws from GP priors

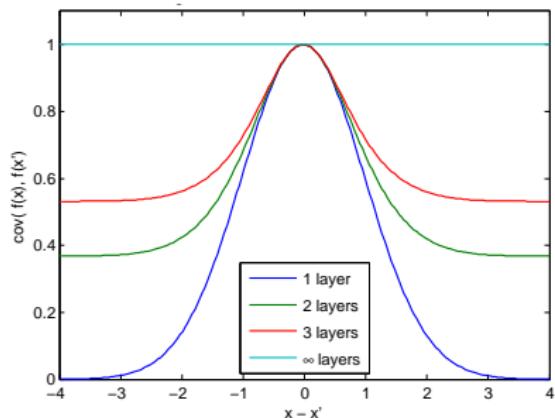
- ▶ Like an Ornstein-Uhlenbeck process with skinny tails
- ▶ Samples are non-differentiable (fractal).

What went wrong?

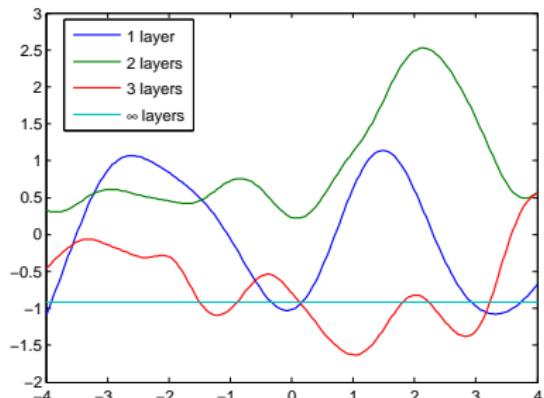
- ▶ Fixed feature mapping, unlikely to be useful for anything
- ▶ power of neural nets comes from learning a custom representation
- ▶ Need to search over feature mappings!
- ▶ Can try to learn kernels, or even better, integrate over feature mappings

Infinitely Deep Kernels

- ▶ For SE kernel, $k_{L+1}(\mathbf{x}, \mathbf{x}') = \exp(k_L(\mathbf{x}, \mathbf{x}') - 1)$.
- ▶ What is the limit of composing SE features?



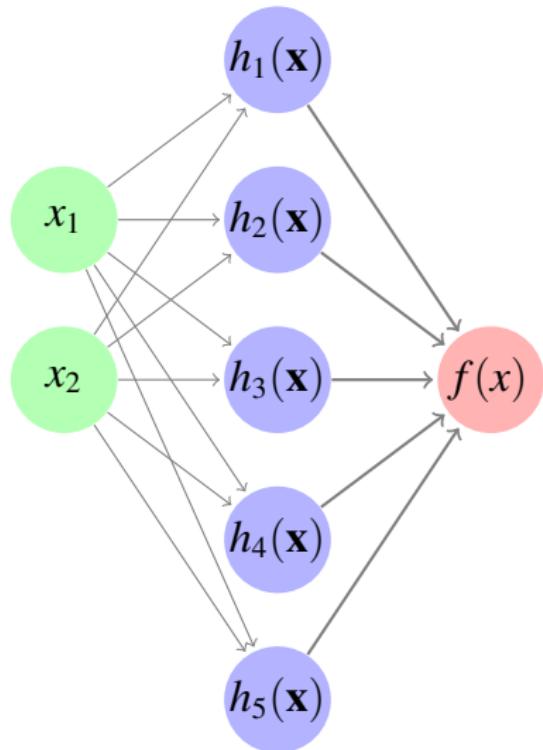
Kernel



Draws from GP prior

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

Dropout on Feature Activations



Original formulation:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K w_i h_i(\mathbf{x})$$

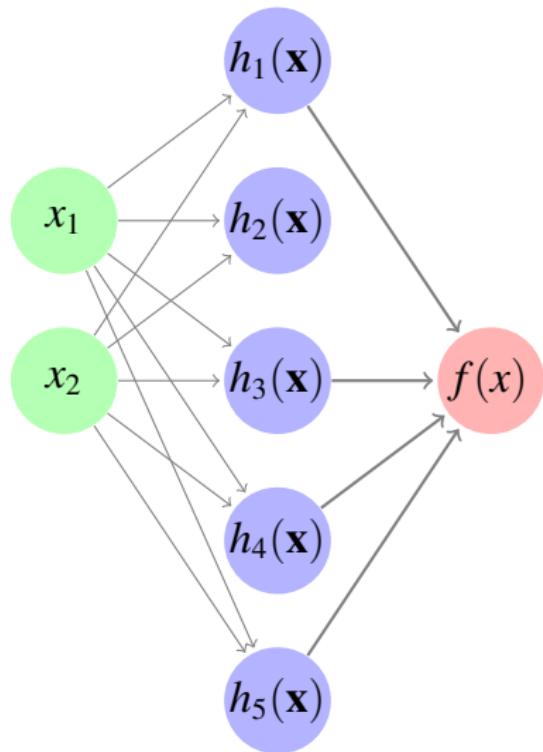
with any weight distribution,

$$\mathbb{E}[w_i] = 0, \quad \mathbb{V}[w_i] = \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Remove units with probability $1/2$:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K \mathbf{r}_i w_i h_i(\mathbf{x}) \quad \mathbf{r}_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

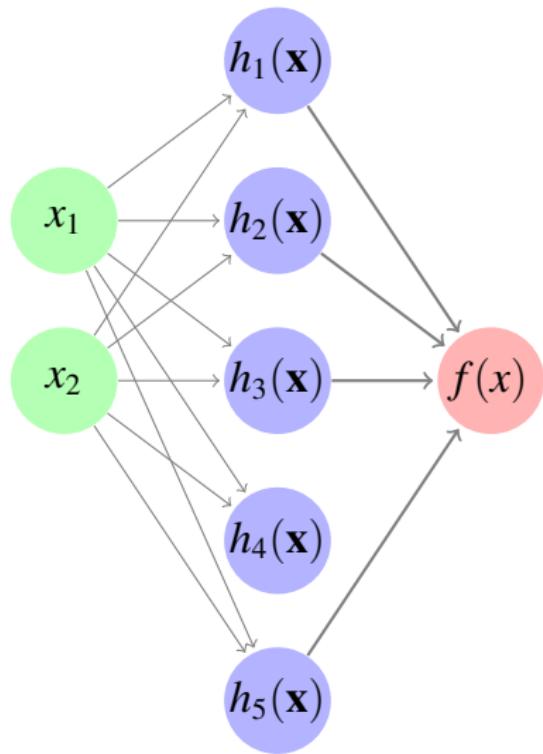
with any weight distribution,

$$\mathbb{E} [\mathbf{r}_i w_i] = 0, \quad \mathbb{V} [\mathbf{r}_i w_i] = \frac{1}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{1}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Remove units with probability $1/2$:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K \mathbf{r}_i w_i h_i(\mathbf{x}) \quad \mathbf{r}_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

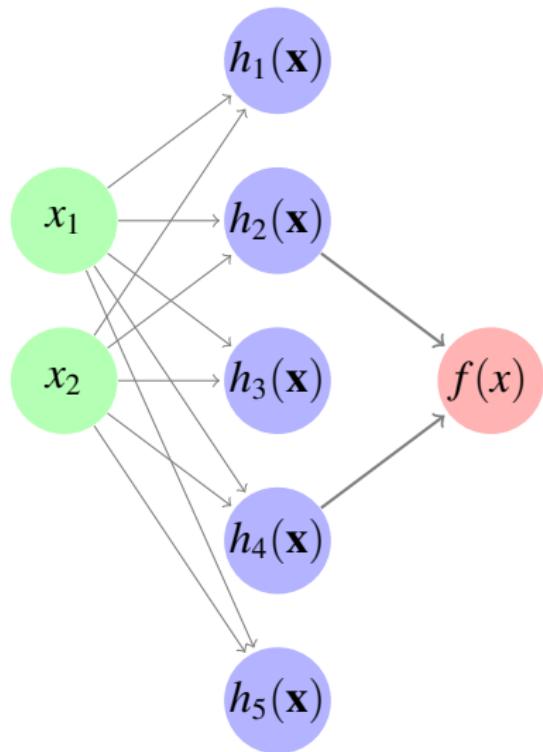
with any weight distribution,

$$\mathbb{E} [\mathbf{r}_i w_i] = 0, \quad \mathbb{V} [\mathbf{r}_i w_i] = \frac{1}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{1}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Remove units with probability $1/2$:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K r_i w_i h_i(\mathbf{x}) \quad r_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

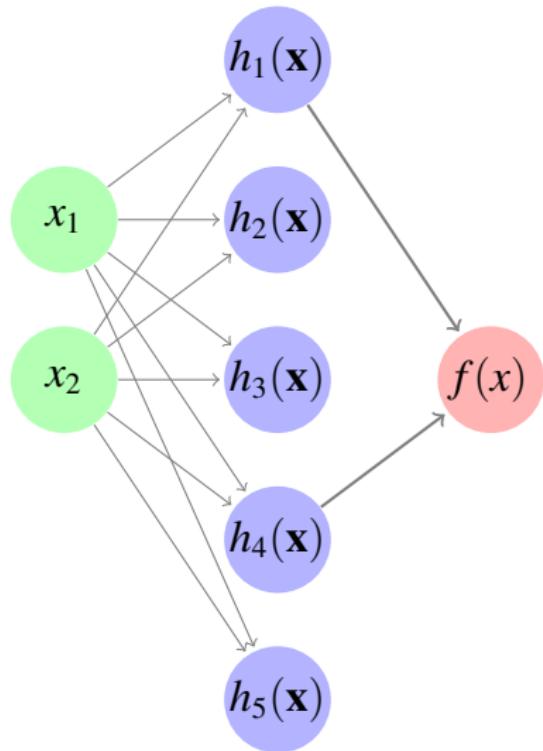
with any weight distribution,

$$\mathbb{E}[r_i w_i] = 0, \quad \mathbb{V}[r_i w_i] = \frac{1}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{1}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Remove units with probability $1/2$:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K \mathbf{r}_i w_i h_i(\mathbf{x}) \quad \mathbf{r}_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

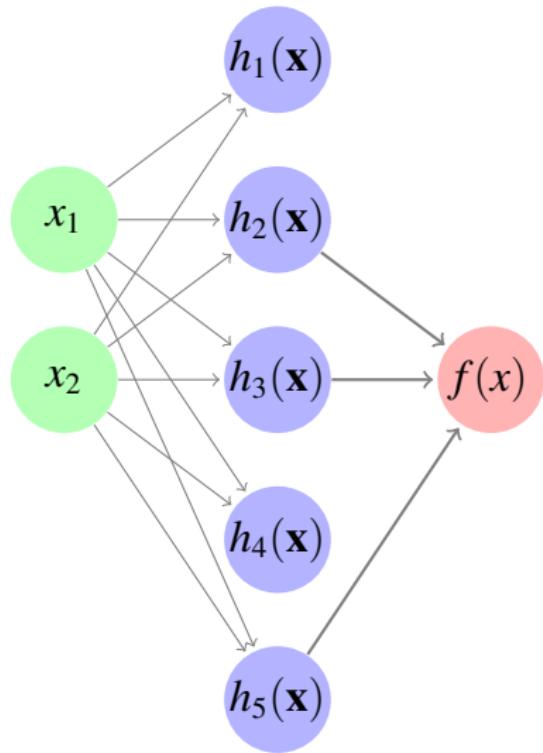
with any weight distribution,

$$\mathbb{E} [\mathbf{r}_i w_i] = 0, \quad \mathbb{V} [\mathbf{r}_i w_i] = \frac{1}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{1}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Remove units with probability $1/2$:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K r_i w_i h_i(\mathbf{x}) \quad r_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

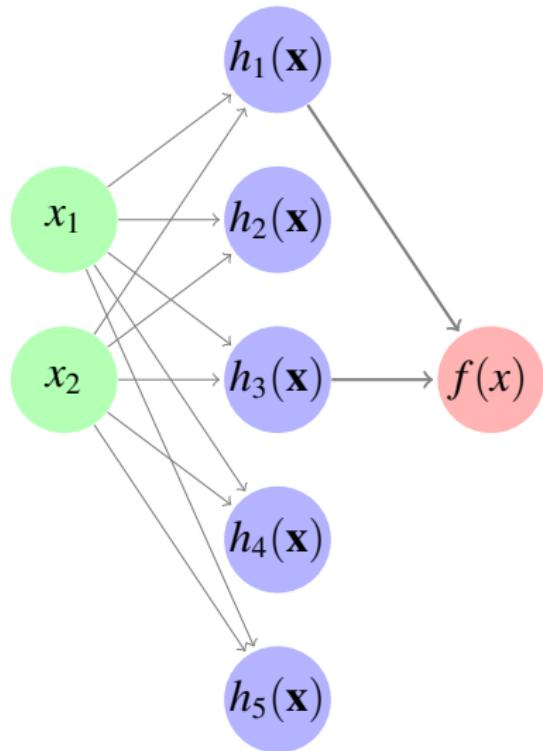
with any weight distribution,

$$\mathbb{E}[r_i w_i] = 0, \quad \mathbb{V}[r_i w_i] = \frac{1}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{1}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Remove units with probability $1/2$:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K \mathbf{r}_i w_i h_i(\mathbf{x}) \quad \mathbf{r}_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

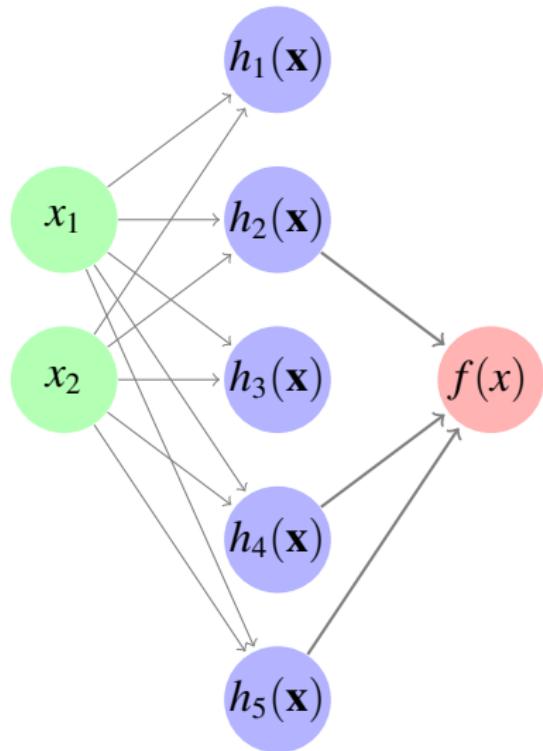
with any weight distribution,

$$\mathbb{E} [\mathbf{r}_i w_i] = 0, \quad \mathbb{V} [\mathbf{r}_i w_i] = \frac{1}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{1}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Remove units with probability $1/2$:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K r_i w_i h_i(\mathbf{x}) \quad r_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

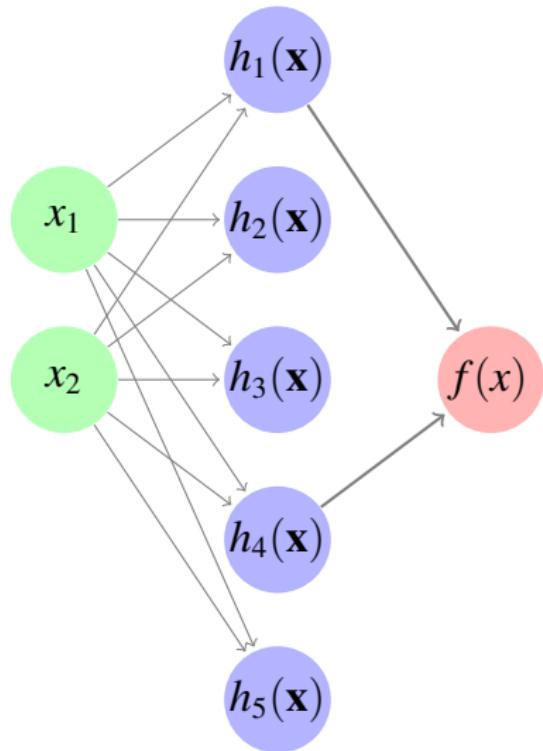
with any weight distribution,

$$\mathbb{E}[r_i w_i] = 0, \quad \mathbb{V}[r_i w_i] = \frac{1}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{1}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Remove units with probability $1/2$:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K \mathbf{r}_i w_i h_i(\mathbf{x}) \quad \mathbf{r}_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

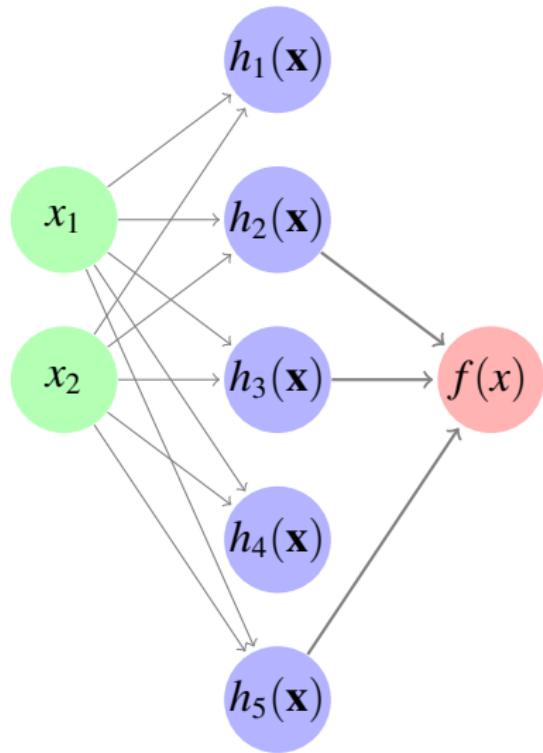
with any weight distribution,

$$\mathbb{E} [\mathbf{r}_i w_i] = 0, \quad \mathbb{V} [\mathbf{r}_i w_i] = \frac{1}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{1}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Remove units with probability $1/2$:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^K r_i w_i h_i(\mathbf{x}) \quad r_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

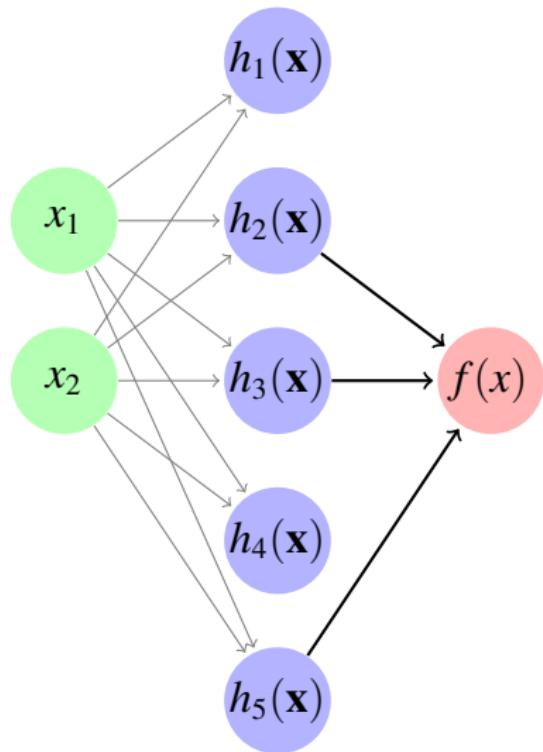
with any weight distribution,

$$\mathbb{E}[r_i w_i] = 0, \quad \mathbb{V}[r_i w_i] = \frac{1}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{1}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations



Double output variance:

$$f(\mathbf{x}) = \frac{2}{K} \sum_{i=1}^K r_i w_i h_i(\mathbf{x}) \quad r_i \stackrel{\text{iid}}{\sim} \text{Ber}(1/2)$$

with any weight distribution,

$$\mathbb{E}[2r_i w_i] = 0, \quad \mathbb{V}[2r_i w_i] = \frac{4}{4} \sigma^2$$

by CLT, gives a GP as $K \rightarrow \infty$

$$\text{cov} \begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \rightarrow \frac{4}{4} \frac{\sigma^2}{K} \sum_{i=1}^K h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Dropout on Feature Activations

- ▶ Dropout on feature activations gives same GP
 - ▶ Averaging the same model doesn't do anything
- ▶ GPs were doing dropout all along? ☺
- ▶ GPs strange because any one feature doesn't matter.
- ▶ Is there a better way to drop out features that would lead to robustness?

Deep Gaussian Processes

- ▶ A prior over compositions of functions:

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

with each $\mathbf{f}_d^{(\ell)} \stackrel{\text{ind}}{\sim} \mathcal{GP}\left(0, k_d^\ell(\mathbf{x}, \mathbf{x}')\right)$.

- ▶ A neural net where each neuron's activation function is drawn from a GP.
- ▶ Inference prevents overfitting, but is really hard.
- ▶ Maybe we can learn something just from looking at draws?