Analyzing Priors on Deep Networks

David Duvenaud, Oren Rippel, Ryan Adams, Zoubin Ghahramani

Sheffield Workshop on Deep Probabilistic Models

October 2, 2014

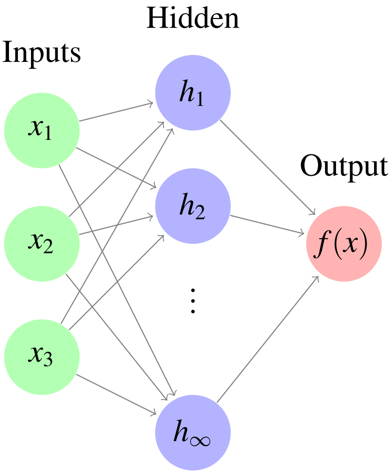
Designing neural nets

- Neural nets require lots of design decisions whose implications hard to understand.
- We want to understand them without reference to a specific dataset, loss function, or training method.
- We can analyze different network architectures by looking at nets whose parameters are drawn randomly.

Why look at priors if I'm going to learn everything anyways?

- When using Bayesian neural nets:
  - Can't learn types of networks having vanishing probability under the prior.
- Even when non-probabilistic:
  - Good prior $\rightarrow$ a good initialization strategy.
  - Good prior $\rightarrow$ a good regularization strategy.
  - Good prior $\rightarrow$ higher fraction of parameters specify reasonable models $\rightarrow$ easier optimization problem.

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 $K \to \infty$

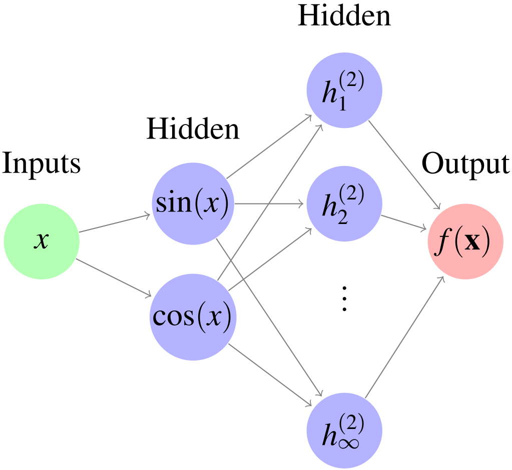
$$\text{cov}\begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{x}') \end{bmatrix} \to \frac{\sigma^2}{K} \sum_{i=1}^{K} h_i(\mathbf{x}) h_i(\mathbf{x}')$$

Kernel learning as feature learning

- GPs have fixed features, integrate out feature weights.
- Mapping between kernels and features:
  $k(\mathbf{x}, \mathbf{x}') = \mathbf{h}(\mathbf{x})^\mathsf{T}\mathbf{h}(\mathbf{x}')$.
- Any PSD kernel can be written as inner product of features. (Mercer's Theorem)
- Kernel learning = feature learning

- What if we make the GP nueral network deep?

Example deep kernel: Periodic



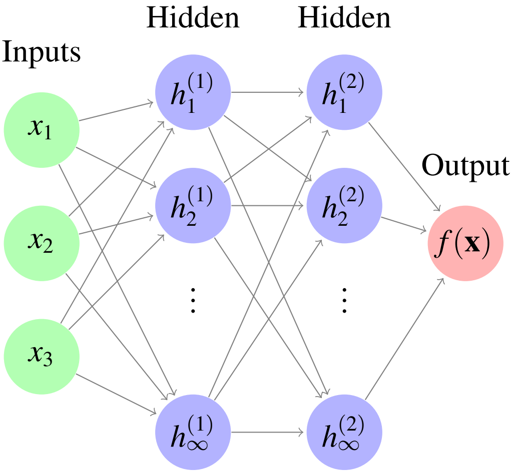
Now our model is:

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

we have "deep kernel":

$$k_2(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2}\left(\mathbf{h}^1(\mathbf{x})) - \mathbf{h}^1(\mathbf{x}')\right)$$

Deep nets, deep kernels



Now our model is:

$$f(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^{K} w_i h_i^{(2)}\left(\mathbf{h}^{(1)}(\mathbf{x})\right)$$
$$= \boldsymbol{w}^\mathsf{T} \mathbf{h}^{(2)}\left(\mathbf{h}^{(1)}(\mathbf{x})\right)$$

Instead of

$$k_1(\mathbf{x}, \mathbf{x}') = \mathbf{h}^{(1)}(\mathbf{x})^\mathsf{T} \mathbf{h}^{(1)}(\mathbf{x}'),$$

we have "deep kernel":

$$k_2(\mathbf{x}, \mathbf{x}')$$
$$= \left[\mathbf{h}^{(2)}\left(\mathbf{h}^{(1)}(\mathbf{x})\right)\right]^\mathsf{T} \mathbf{h}^{(2)}\left(\mathbf{h}^{(1)}(\mathbf{x}')\right)$$

Deep Kernels

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

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

$$= \left(\mathbf{h}^{SE}\left(\mathbf{h}^1(\mathbf{x})\right)\right)^{\mathsf{T}} \mathbf{h}^{SE}\left(\mathbf{h}^1(\mathbf{x}')\right)$$

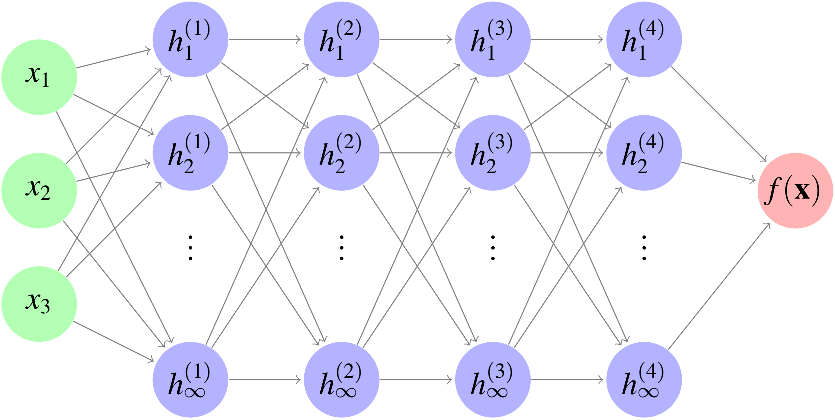
$$= \exp\left(-\frac{1}{2}||\mathbf{h}^1(\mathbf{x}) - \mathbf{h}^1(\mathbf{x}')||_2^2\right)$$

$$= \exp\left(-\frac{1}{2}\left[\mathbf{h}^1(\mathbf{x})^{\mathsf{T}}\mathbf{h}^1(\mathbf{x}) - 2\mathbf{h}^1(\mathbf{x})^{\mathsf{T}}\mathbf{h}^1(\mathbf{x}') + \mathbf{h}^1(\mathbf{x}')^{\mathsf{T}}\mathbf{h}^1(\mathbf{x}')\right]\right)$$

$$= \exp\left(-\frac{1}{2}\left[k_1(\mathbf{x}, \mathbf{x}) - 2k_1(\mathbf{x}, \mathbf{x}') + k_1(\mathbf{x}', \mathbf{x}')\right]\right)$$

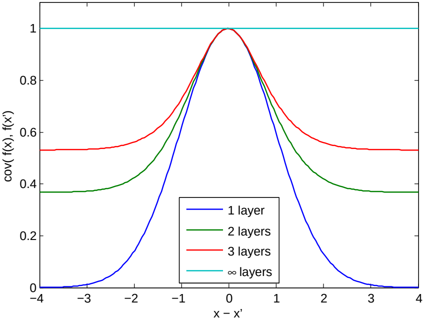
- A closed form…let's do it again!

Repeated Fixed Feature Mappings

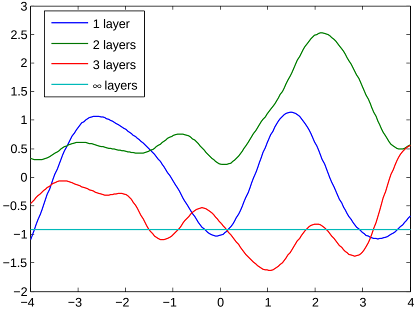


Infinitely Deep Kernels

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



Kernel



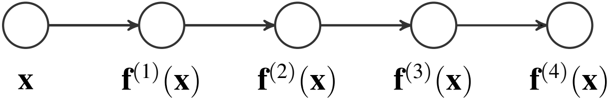
Draws from GP prior

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

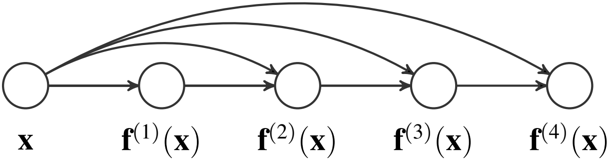
A simple fix

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

Standard architecture:



Input-connected architecture:



A simple fix

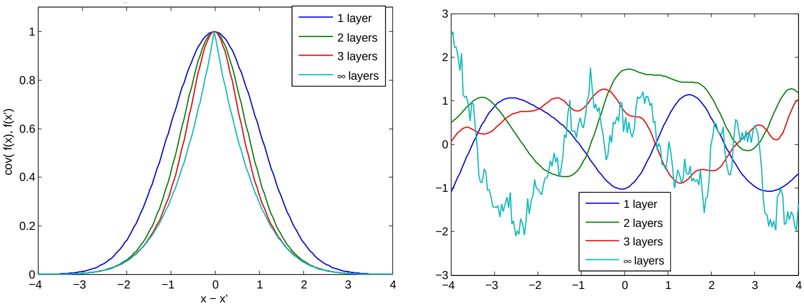
$$k_{L+1}(\mathbf{x}, \mathbf{x}') =$$

$$= \exp\left(-\frac{1}{2}\left|\left|\begin{bmatrix} \mathbf{h}^L(\mathbf{x}) \\ \mathbf{x} \end{bmatrix} - \begin{bmatrix} \mathbf{h}^L(\mathbf{x}') \\ \mathbf{x}' \end{bmatrix}\right|\right|_2^2\right)$$

$$= \exp\left(-\frac{1}{2}\left[k_L(\mathbf{x}, \mathbf{x}) - 2k_L(\mathbf{x}, \mathbf{x}') + k_L(\mathbf{x}', \mathbf{x}')\right] - \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2\right)$$

Infinitely deep kernels, take two

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



Kernels | Draws from GP priors

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

Not very exciting...

- Fixed feature mapping, unlikely to be useful for anything
- Power of neural nets comes from learning a custom representation.

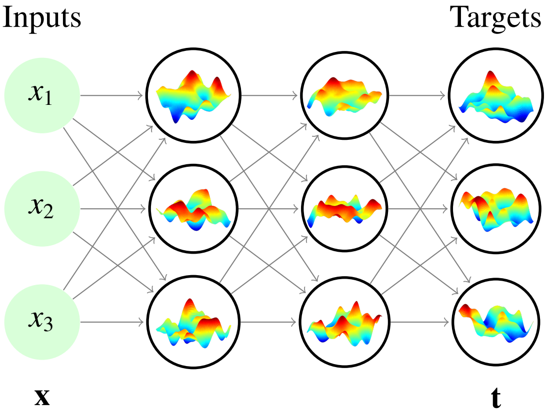
Deep Gaussian Processes

- A prior over compositions of functions:

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

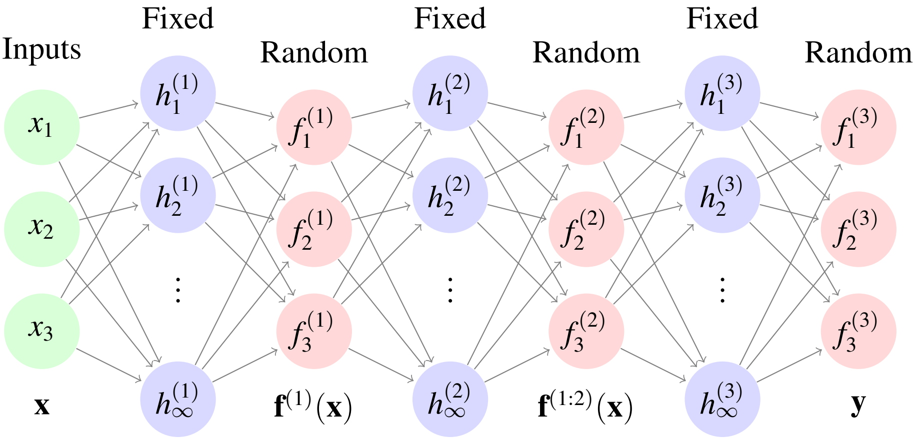
  with each $\mathbf{f}_d^{(\ell)} \overset{\text{ind}}{\sim} \mathcal{GP}\big(0, k_d^{\ell}(\mathbf{x}, \mathbf{x}')\big)$.
- Can be seen as a "simpler" version of Bayesian neural nets
- Two equivalent architectures.

Deep GPs as nonparametric nets



- 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 $\mathbf{y} = \mathbf{f}(\mathbf{x})$.
- In this talk we only consider noiseless functions.

Deep GPs as infinitely wide parametric nets

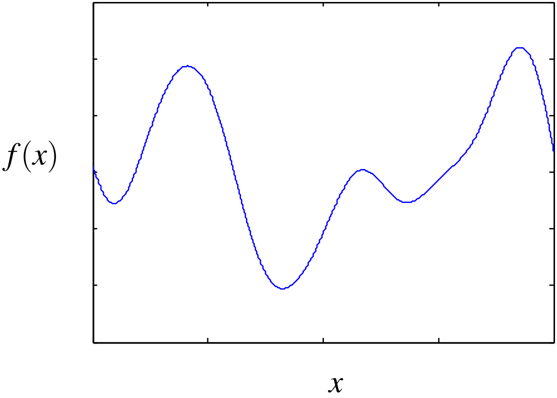


- Infinitely-wide fixed feature maps alternating with finite linear information bottlenecks:

$$\mathbf{h}^{(\ell)}(\mathbf{x}) = \sigma\left(\mathbf{b}^{(\ell)} + \left[\mathbf{V}^{(\ell)}\mathbf{W}^{(\ell-1)}\right]\mathbf{h}^{(\ell-1)}(\mathbf{x})\right)$$

Priors on deep networks

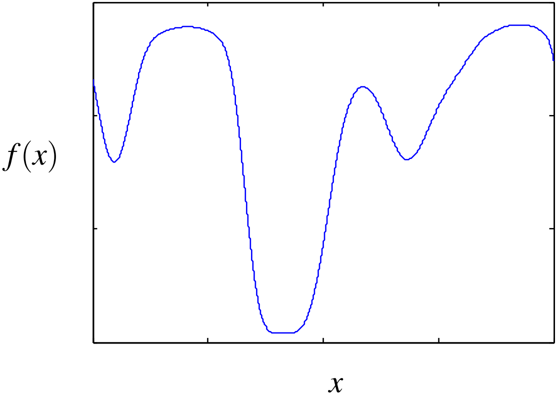
- A draw from a one-neuron-per-layer deep GP:



1 Layer

Priors on deep networks

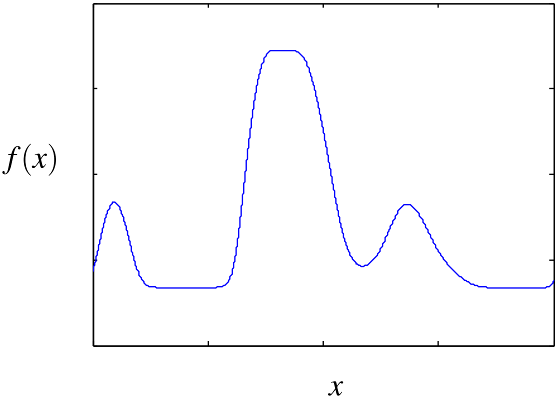
- A draw from a one-neuron-per-layer deep GP:



2 Layers

Priors on deep networks

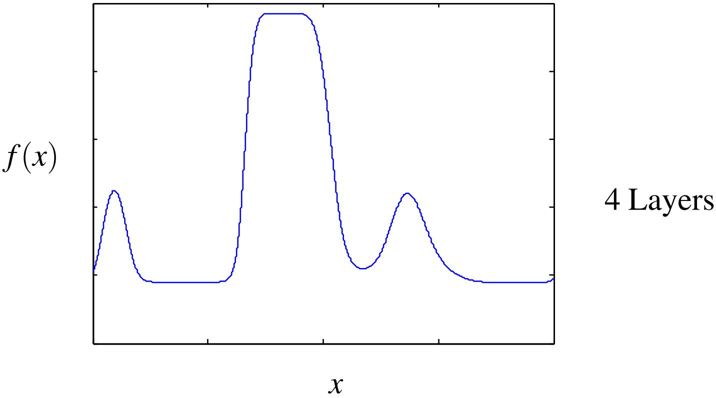
- A draw from a one-neuron-per-layer deep GP:



3 Layers

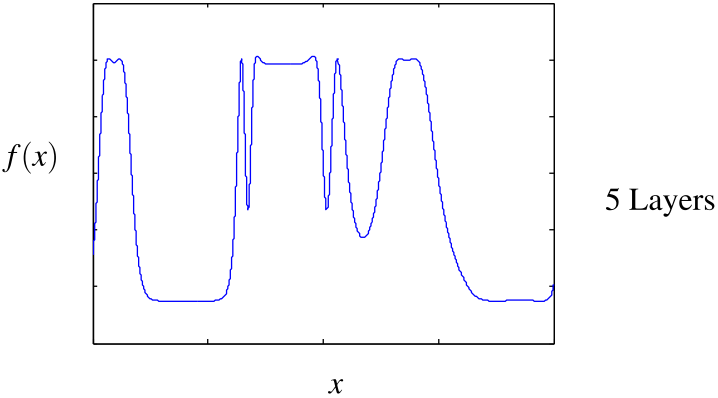
Priors on deep networks

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



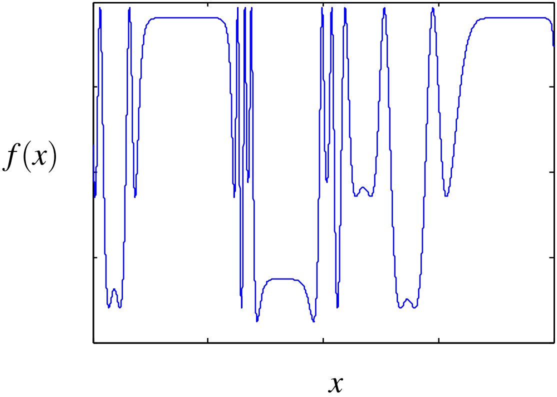
Priors on deep networks

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



Priors on deep networks

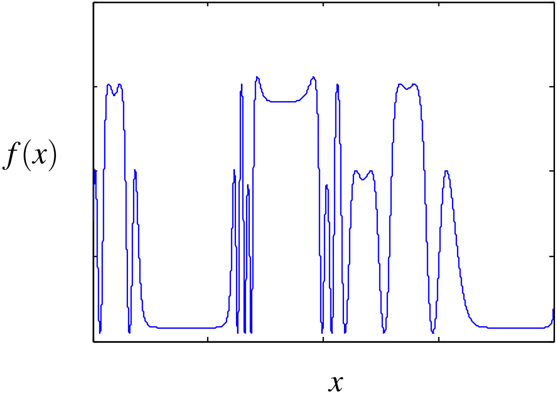
- A draw from a one-neuron-per-layer deep GP:



6 Layers

Priors on deep networks

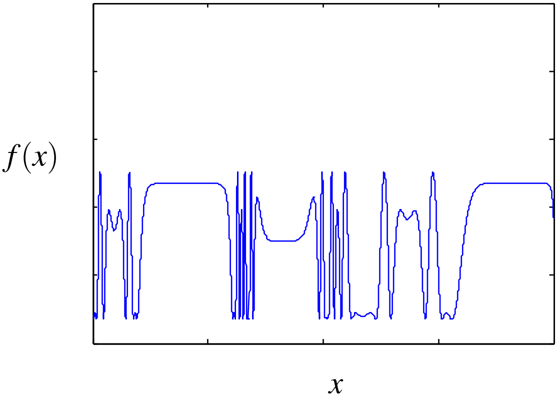
- A draw from a one-neuron-per-layer deep GP:



7 Layers

Priors on deep networks

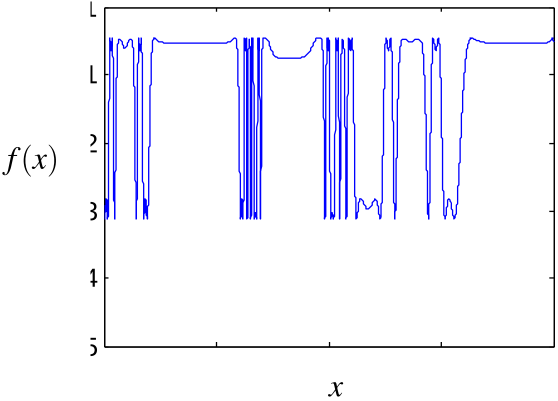
- A draw from a one-neuron-per-layer deep GP:



8 Layers

Priors on deep networks

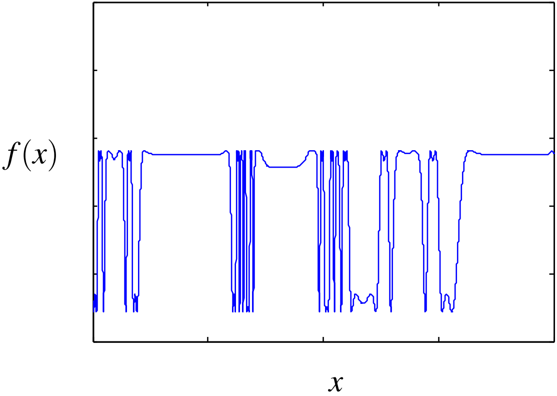
- A draw from a one-neuron-per-layer deep GP:



9 Layers

Priors on deep networks

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

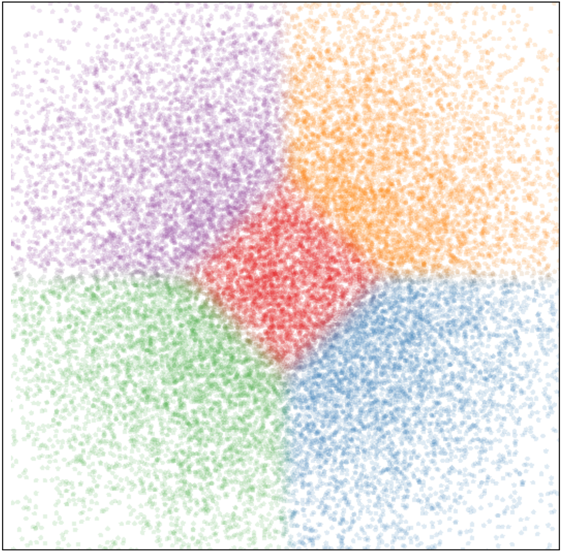


10 Layers

Size of derivative becomes log-normal distributed.

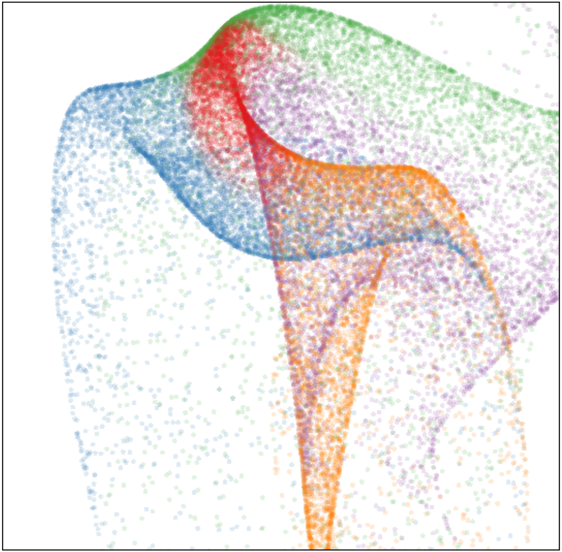
Priors on deep networks

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



Priors on deep networks

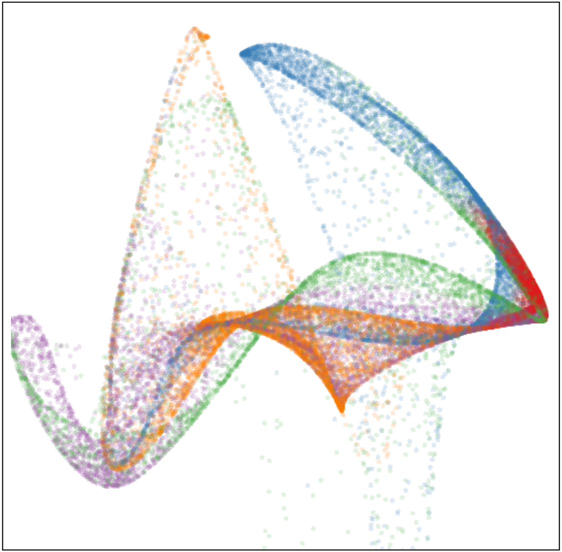
- 2D to 2D warpings of a set of coloured points:



1 Layer

Priors on deep networks

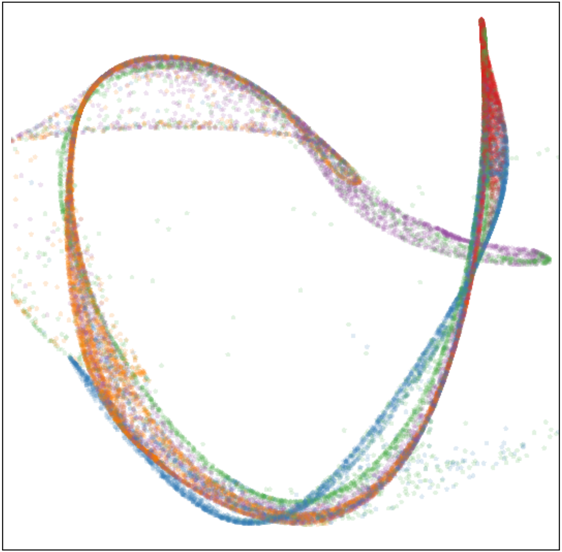
- 2D to 2D warpings of a set of coloured points:



2 Layers

Priors on deep networks

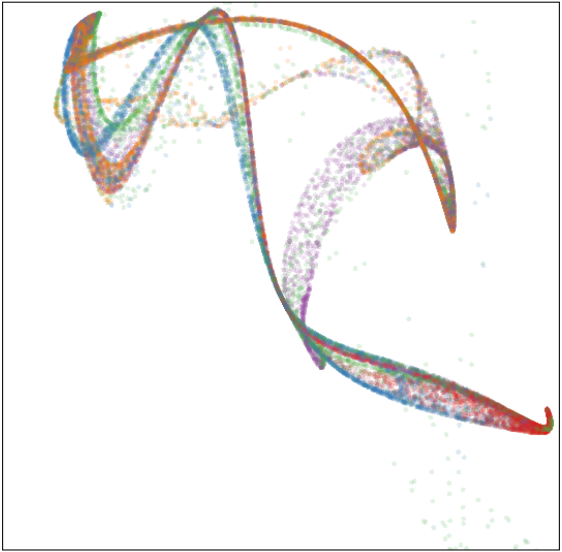
- 2D to 2D warpings of a set of coloured points:



3 Layers

Priors on deep networks

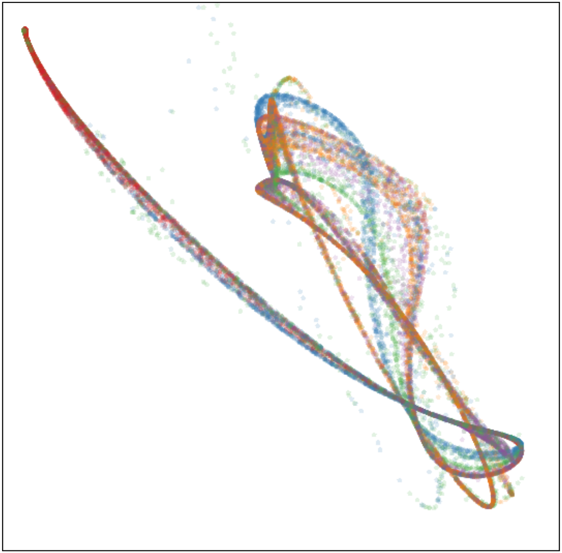
- 2D to 2D warpings of a set of coloured points:



4 Layers

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

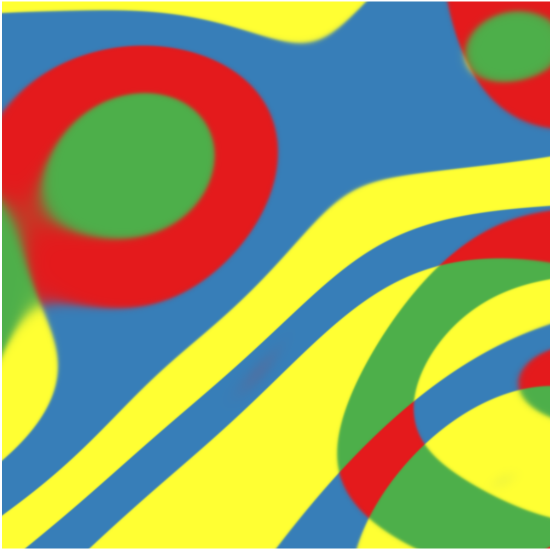
Priors on deep networks

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

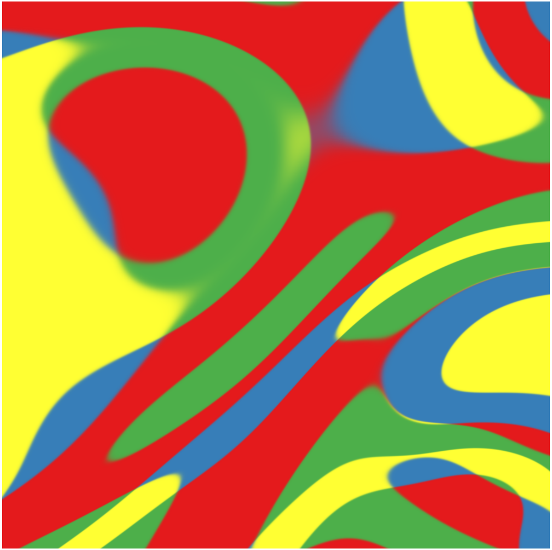
Priors on deep networks

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

3 Layers

Priors on deep networks

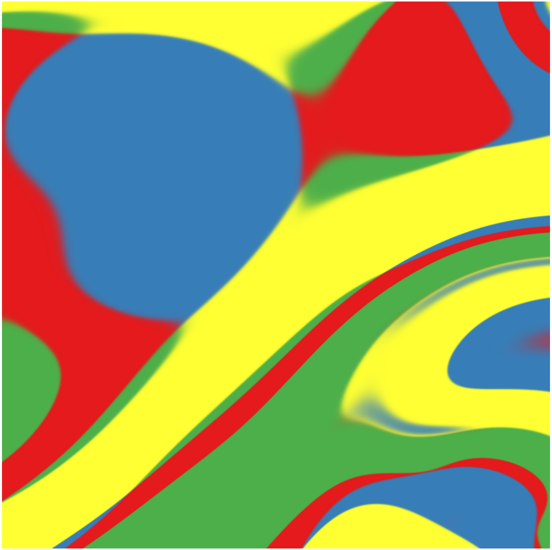
Color shows **y** that each **x** is mapped to (decision boundary)



4 Layers

Priors on deep networks

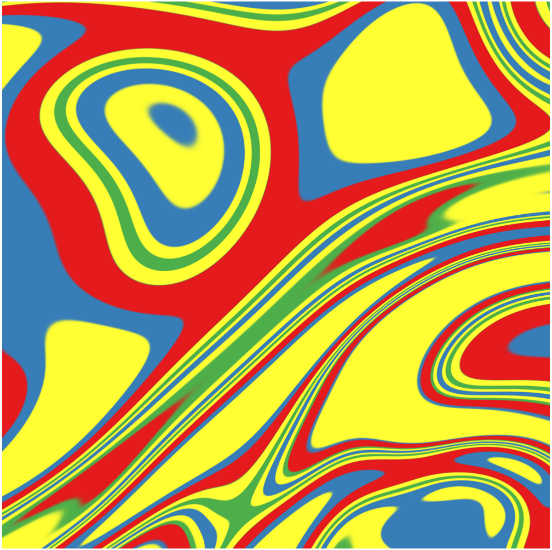
Color shows **y** that each **x** is mapped to (decision boundary)



5 Layers

Priors on deep networks

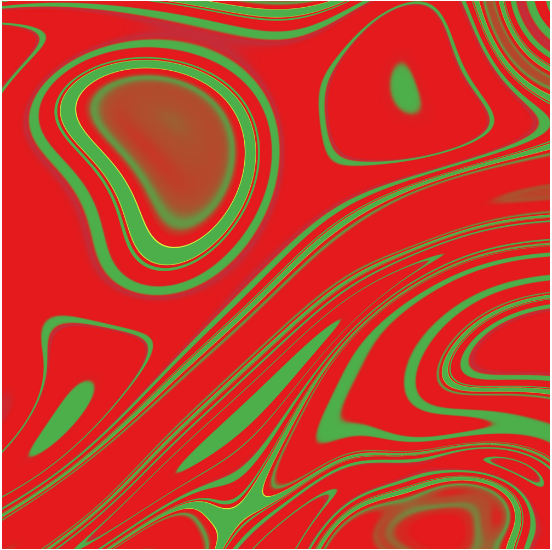
Color shows **y** that each **x** is mapped to (decision boundary)



10 Layers

Priors on deep networks

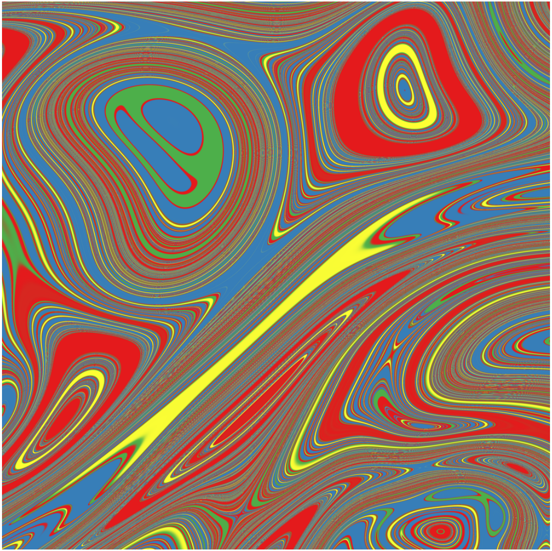
Color shows **y** that each **x** is mapped to (decision boundary)



20 Layers

Priors on deep networks

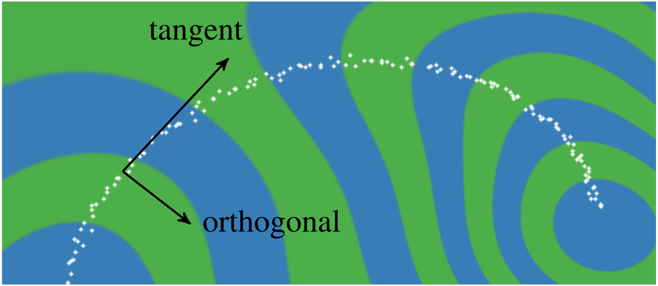
Color shows **y** that each **x** is mapped to (decision boundary)



40 Layers

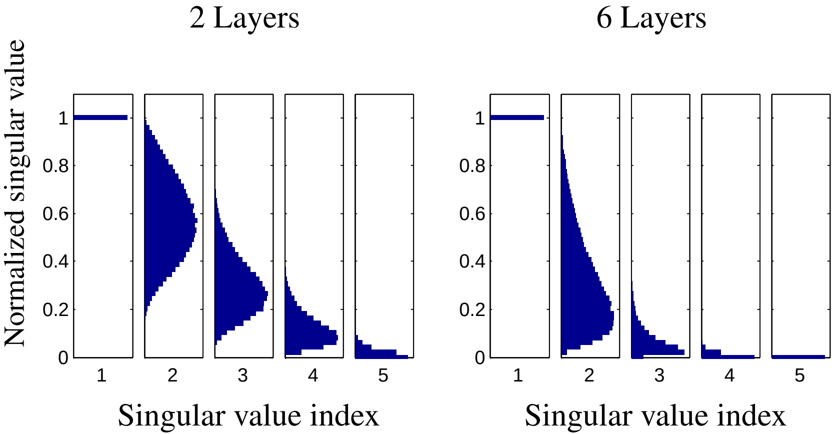
Representation only changes in one direction locally.

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.

Analysis of Jacobian



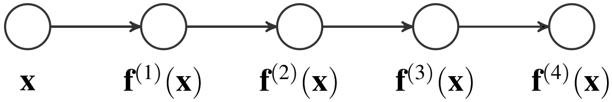
The distribution of normalized singular values of the Jacobian of functions drawn from a 5-dimensional deep GP prior.

- Lemma from paper: The Jacobian of a deep GP is a product of i.i.d. random Gaussian matrices.
- Output only changes in w.r.t. one direction as net deepens.

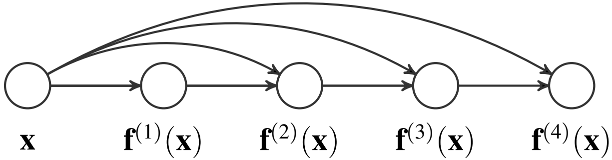
A simple fix

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

Standard architecture:

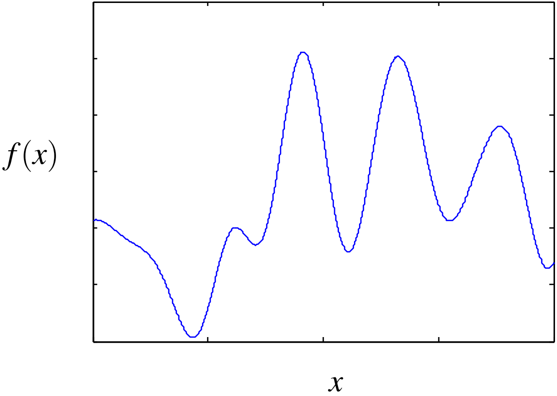


Input-connected architecture:



A different architecture

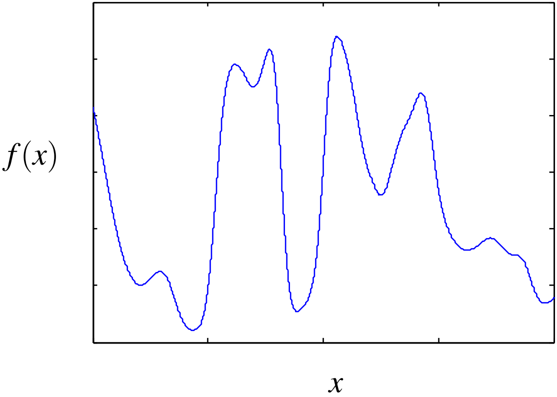
- A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



1 layer

A different architecture

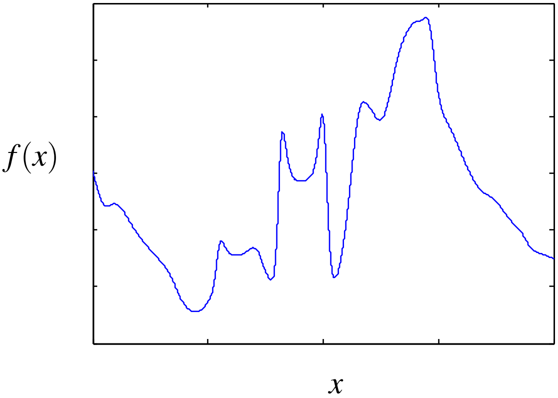
- A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



2 layers

A different architecture

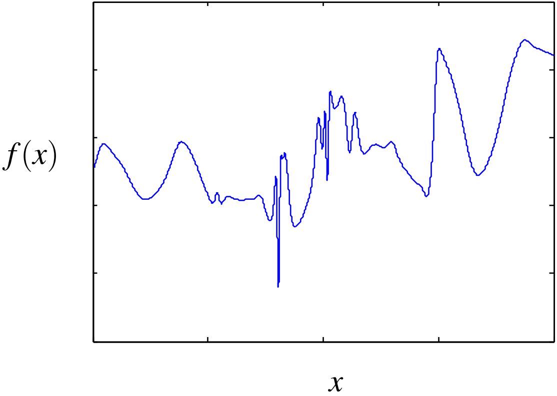
- A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



3 layers

A different architecture

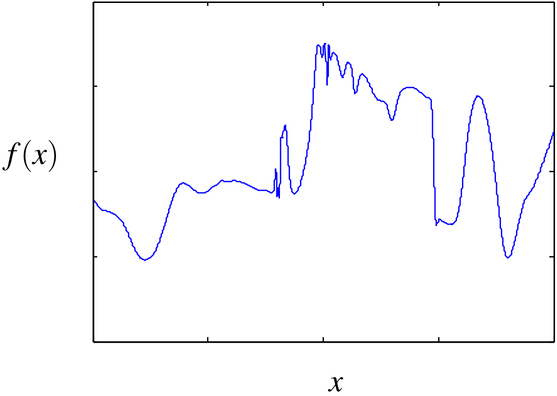
- A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



4 layers

A different architecture

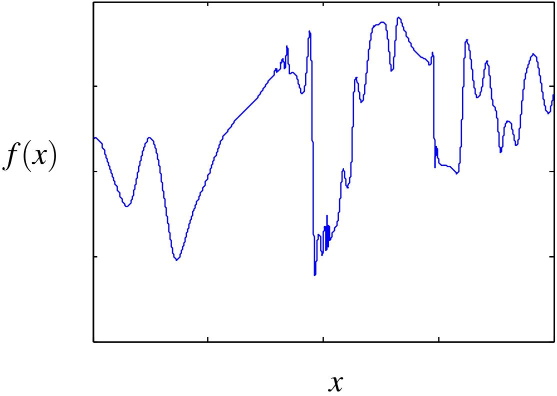
- A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



5 layers

A different architecture

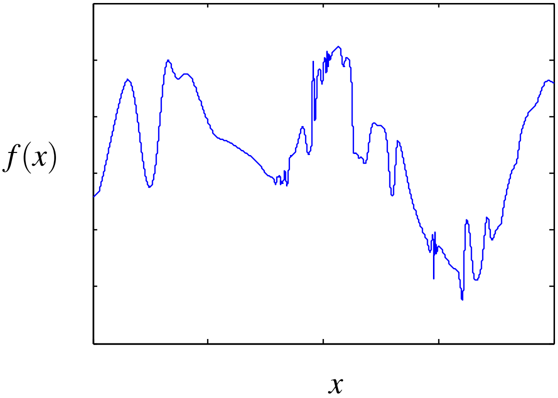
- A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



6 layers

A different architecture

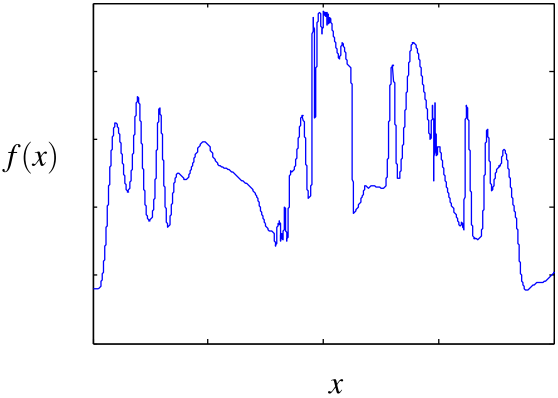
- A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



7 layers

A different architecture

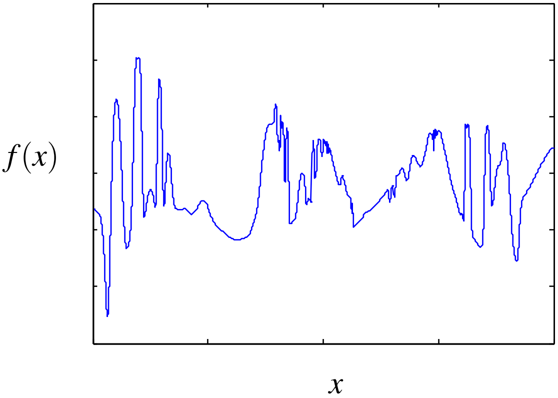
- A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



8 layers

A different architecture

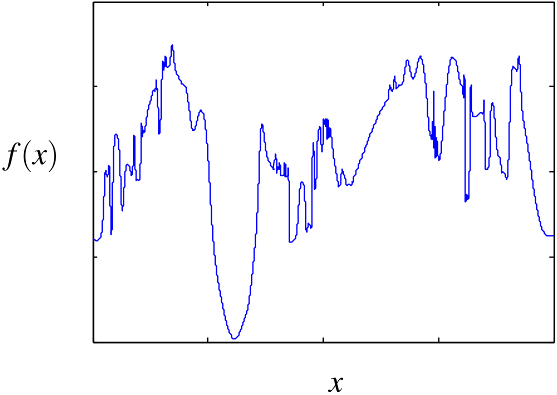
- A draw from a one-neuron-per-layer deep GP, with the input also connected to each layer:



9 layers

A different architecture

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

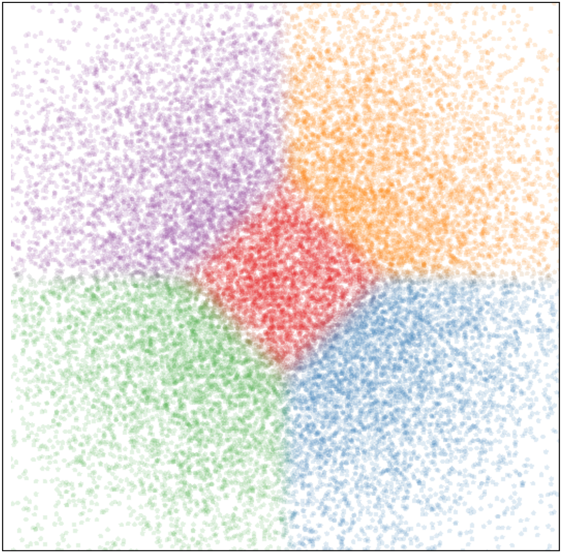


10 layers

Greater variety of derivatives.

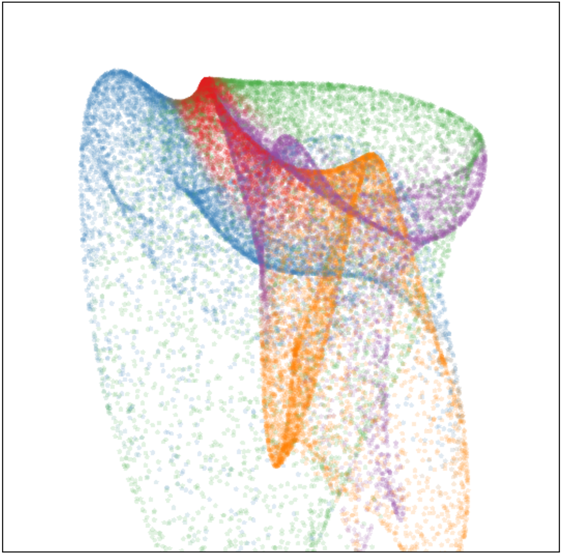
A different architecture

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



A different architecture

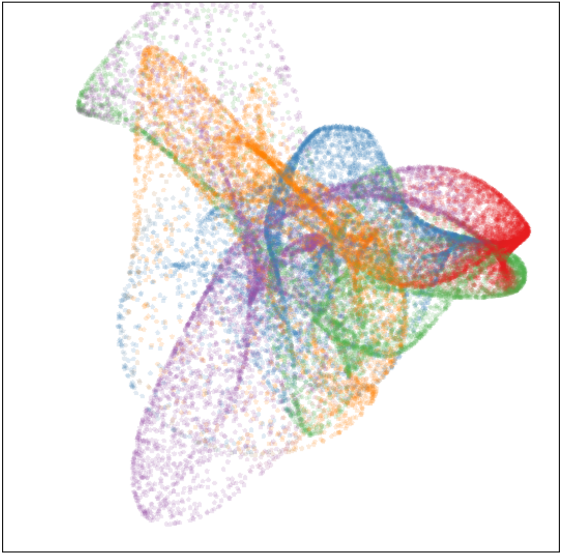
- Input-connected 2D to 2D warpings of coloured points:



1 Layer

A different architecture

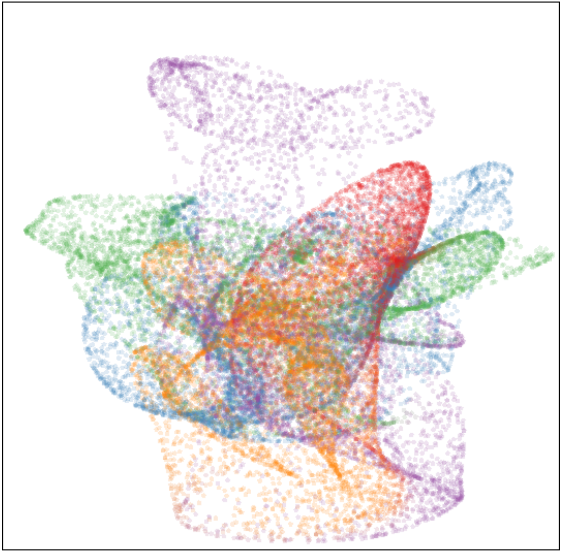
- Input-connected 2D to 2D warpings of coloured points:



2 Layers

A different architecture

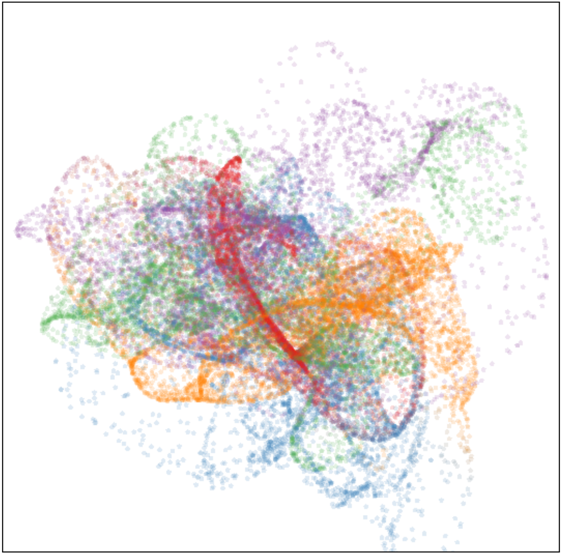
- Input-connected 2D to 2D warpings of coloured points:



3 Layers

A different architecture

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



4 Layers

A different architecture

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

5 Layers

Density becomes more complex but remains 2D.

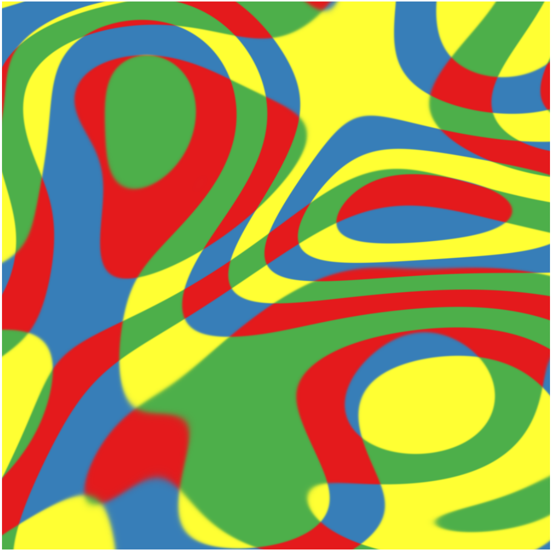
A different architecture (show video)

- Color shows **y** that each **x** is mapped to

No warping

A different architecture (show video)

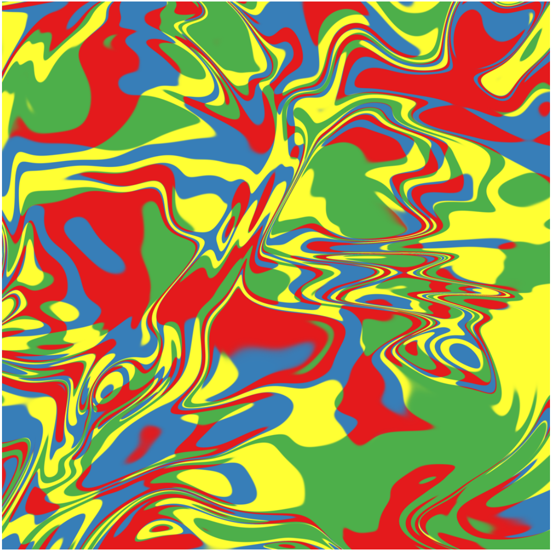
- Color shows **y** that each **x** is mapped to



2 Layers

A different architecture (show video)

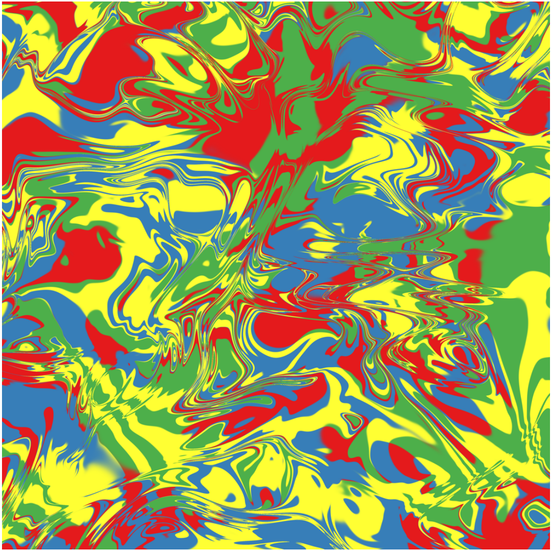
- Color shows **y** that each **x** is mapped to



10 Layers

A different architecture (show video)

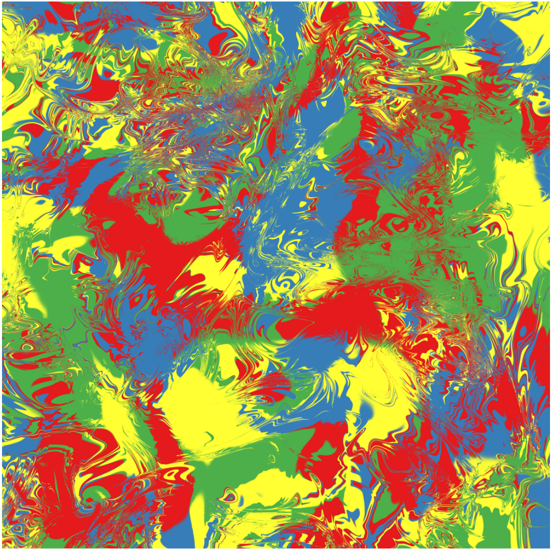
- Color shows **y** that each **x** is mapped to



20 Layers

A different architecture (show video)

- Color shows **y** that each **x** is mapped to



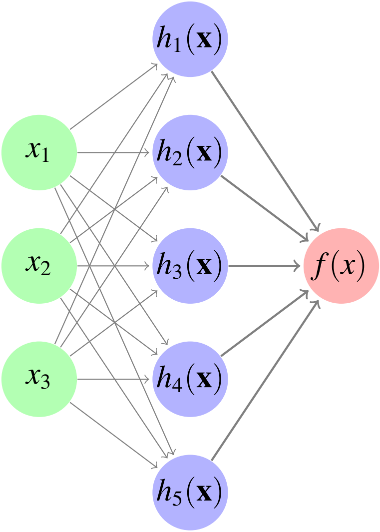
40 Layers

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 on Feature Activations



Original formulation:

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

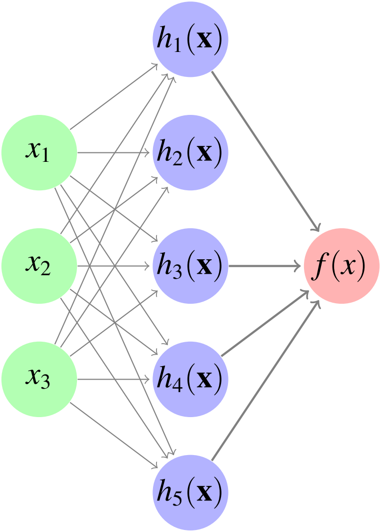
with any weight distribution,

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

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

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

Dropout on Feature Activations



Remove units with probability $\frac{1}{2}$:

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

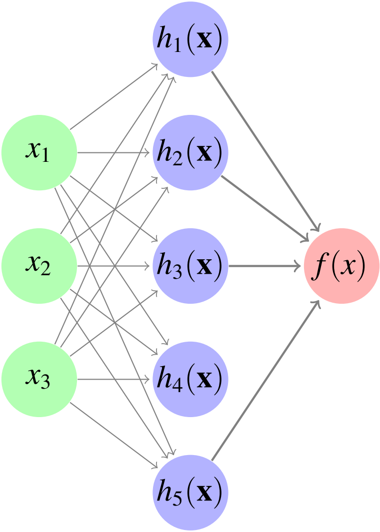
with any weight distribution,

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

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

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

Dropout on Feature Activations



Remove units with probability $\frac{1}{2}$:

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

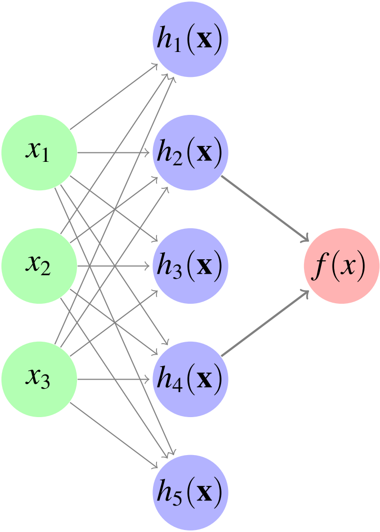
with any weight distribution,

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

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

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

Dropout on Feature Activations



Remove units with probability $\frac{1}{2}$:

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

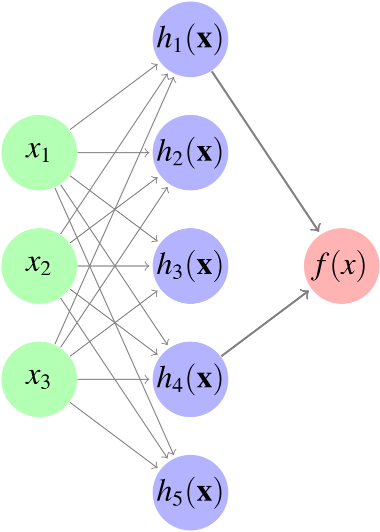
with any weight distribution,

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

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

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

Dropout on Feature Activations



Remove units with probability $\frac{1}{2}$:

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

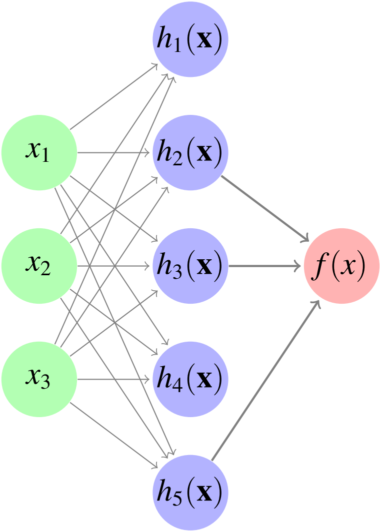
with any weight distribution,

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

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

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

Dropout on Feature Activations



Remove units with probability $\frac{1}{2}$:

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

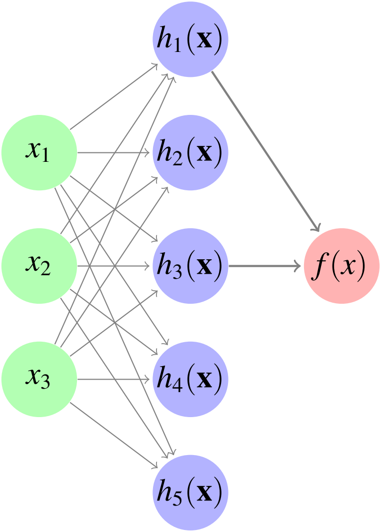
with any weight distribution,

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

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

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

Dropout on Feature Activations



Remove units with probability $\frac{1}{2}$:

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

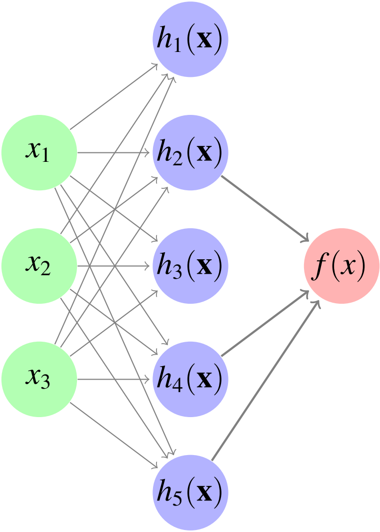
with any weight distribution,

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

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

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

Dropout on Feature Activations



Remove units with probability $\frac{1}{2}$:

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

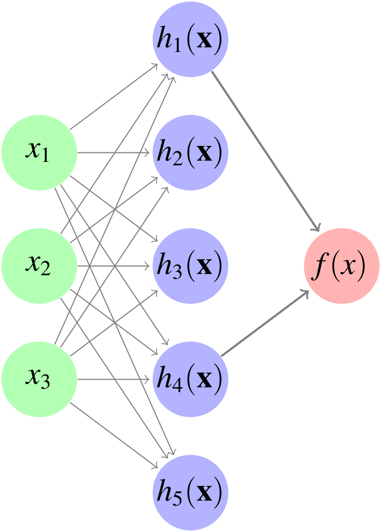
with any weight distribution,

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

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

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

Dropout on Feature Activations



Remove units with probability $\frac{1}{2}$:

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

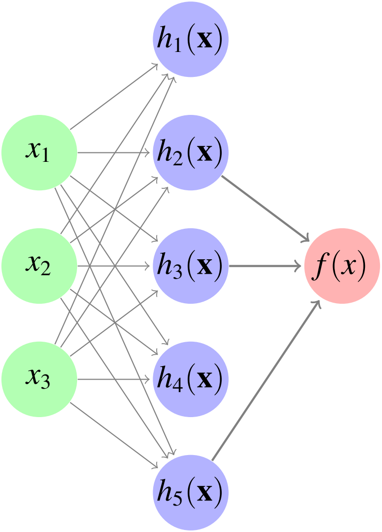
with any weight distribution,

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

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

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

Dropout on Feature Activations



Remove units with probability $\frac{1}{2}$:

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

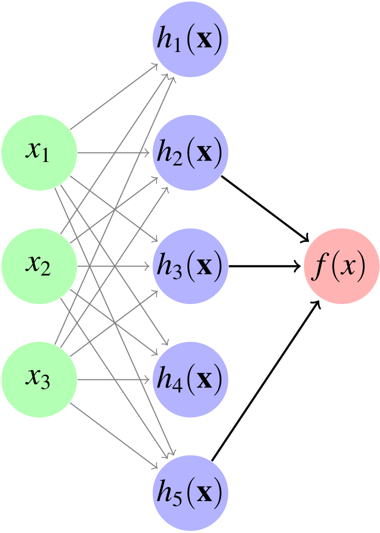
with any weight distribution,

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

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

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

Dropout on Feature Activations



Double output variance:

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

with any weight distribution,

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

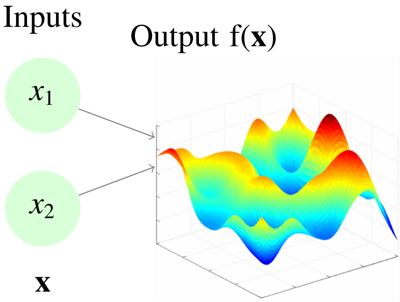
by CLT, gives a GP as $K \to \infty$

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

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 are strange because any one feature doesn't matter.
- Is there a better way to drop out features that would lead to robustness?

Dropout on GP inputs

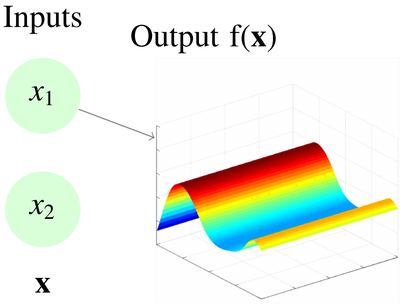


- Each function only depends on some input dimensions.
- Given prior covariance $\text{cov}\left[f(\mathbf{x}), f(\mathbf{x}')\right] = k(\mathbf{x}, \mathbf{x}')$, exact dropout gives a mixture of GPs:

$$p\big(f(\mathbf{x})\big) = \frac{1}{2^D} \sum_{\mathbf{r} \in \{0,1\}^D} \text{GP}\left(0, k(\mathbf{r}^\mathsf{T}\mathbf{x}, \mathbf{r}^\mathsf{T}\mathbf{x}')\right)$$

- Can be viewed as spike-and-slab ARD prior.

Dropout on GP inputs

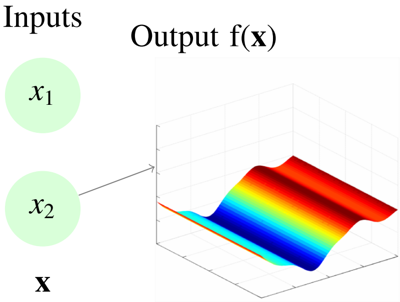


- Each function only depends on some input dimensions.
- Given prior covariance $\text{cov}\left[f(\mathbf{x}), f(\mathbf{x}')\right] = k(\mathbf{x}, \mathbf{x}')$, exact dropout gives a mixture of GPs:

$$p\big(f(\mathbf{x})\big) = \frac{1}{2^D} \sum_{\mathbf{r} \in \{0,1\}^D} \text{GP}\left(0, k(\mathbf{r}^\mathsf{T}\mathbf{x}, \mathbf{r}^\mathsf{T}\mathbf{x}')\right)$$

- Can be viewed as spike-and-slab ARD prior.

Dropout on GP inputs



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

$$p\big(f(\mathbf{x})\big) = \frac{1}{2^D} \sum_{\mathbf{r} \in \{0,1\}^D} \mathrm{GP}\left(0, k(\mathbf{r}^\mathsf{T}\mathbf{x}, \mathbf{r}^\mathsf{T}\mathbf{x}')\right)$$

- Can be viewed as spike-and-slab ARD prior.

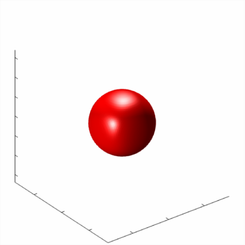
Covariance before and after dropout

Original squared-exp:

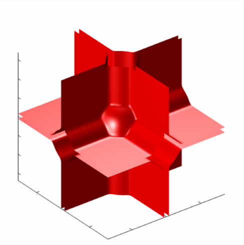
$$\text{cov}\left[f(\mathbf{x}), f(\mathbf{x}')\right] = k(\mathbf{x}, \mathbf{x}')$$

After dropout:

$$\text{cov}\left[f(\mathbf{x}), f(\mathbf{x}')\right] = \sum_{\mathbf{r} \in \{0,1\}^D} k(\mathbf{r}^\mathsf{T}\mathbf{x}, \mathbf{r}^\mathsf{T}\mathbf{x}')$$



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



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

- Sum of many functions, each depends only on a subset of inputs.
- 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 input 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?

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 input 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?

Thanks!