Deep Gaussian processes

Mauricio A. Álvarez

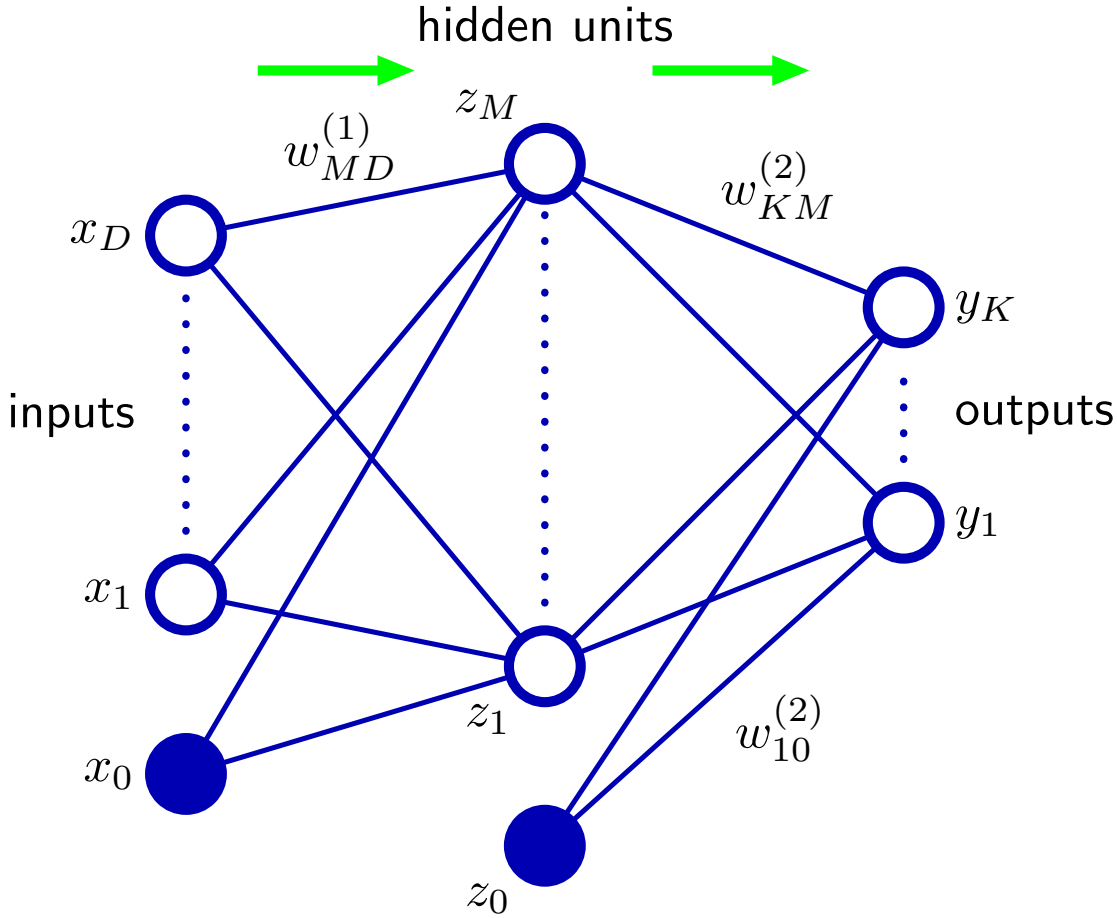
Topics on Deep Probabilistic Models

Acknowledgements

Contents

Contents

Typical architecture of a neural network



Model (I)

- The basic model of a neural network (NN) can be described by a series of functional transformations.

- We first construct M linear combinations of the input variables $x_1, \ldots, x_D$ in the form

$$a_j = \sum_{i=1}^{D} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)},$$

where $j = 1, \ldots, M$, and the superindex (1) indicates the parameters corresponding to the first "layer" of the network.

Model (II)

- The quantities a_j are known as *activations*.
- The a_j are transformed using a non-linear activation function to give

$$z_j = h(a_j).$$

- In this context, these functions are known as *hidden nodes*.

Model (III)

- The z_j are linearly combined again to give *output activations*

$$a_k = \sum_{j=1}^{M} w_{k,j}^{(2)} z_j + w_{k,0}^{(2)},$$

  where $k = 1, \ldots, K$, and K is the total number of outputs.

- The super-index (2) refers to the parameters of the second "layer" of the network.

- The output activations are transformed or not depending on the problem to address
  - Regression $\rightarrow y_k = a_k$.
  - Classification $\rightarrow y_k = \sigma(a_k)$.

Model (IV)

- Combining both stages, we get

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{k,j}^{(2)} h\left(\sum_{i=1}^{D} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}\right) + w_{k,0}^{(2)}\right),$$

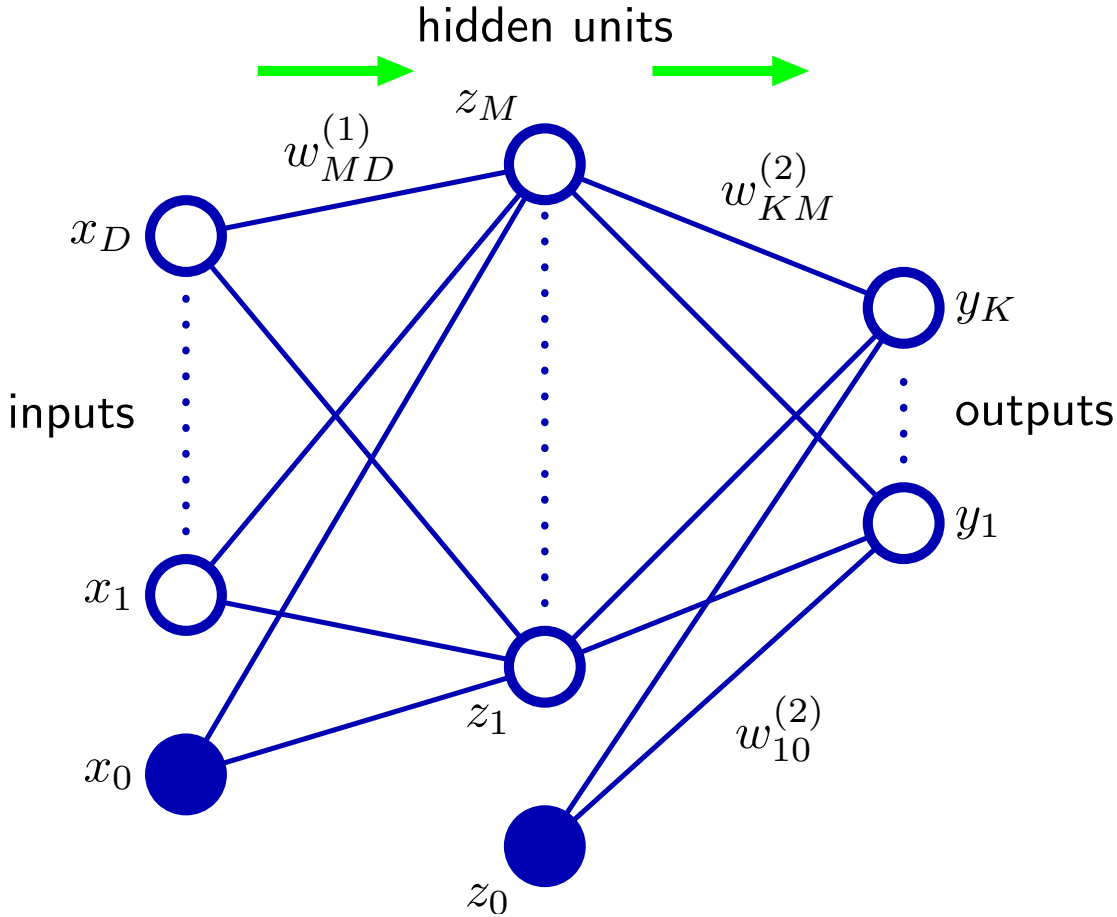
$$= \sigma\left(\sum_{j=0}^{M} w_{k,j}^{(2)} h\left(\sum_{i=0}^{D} w_{j,i}^{(1)} x_i\right)\right),$$

  with $x_0 = 1$.

- Notice that in the second equality, we have made use of $z_0 = h\left(\sum_{i=0}^{D} w_{0,i}^{(1)} x_i\right)$.
- Parameters $\{w_{j,i}\}_{j=1,i=0}^{M,D}$ y $\{w_{k,j}\}_{k=1,j=0}^{K,M}$ are joinly denoted as $\mathbf{w}$.
- Neural network: non-linear function of $\{x_i\}_{i=1}^{D}$ to $\{y_k\}_{k=1}^{K}$ controlled by $\mathbf{w}$.

Model (V)

The network in the figure is a two-layer NN due to two is the number of layers with adaptive weights.

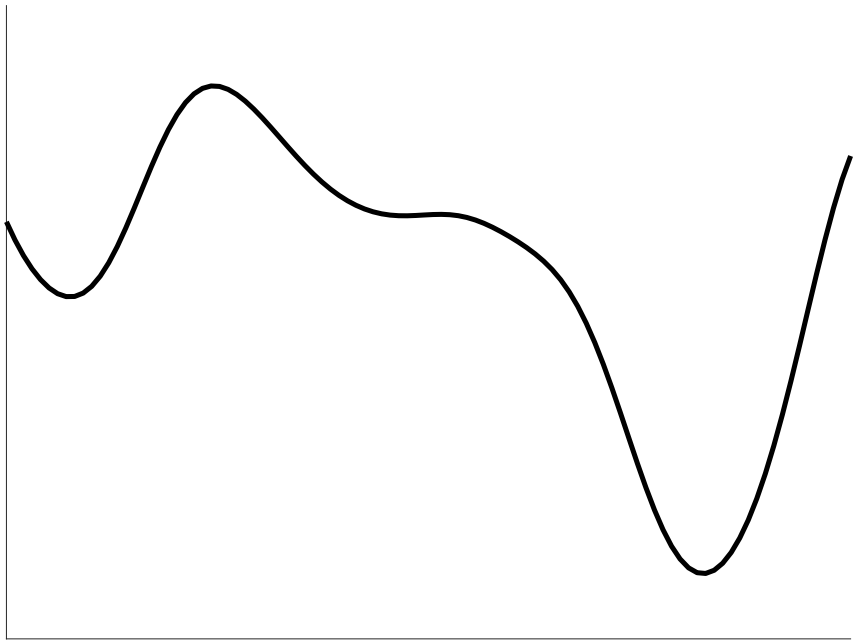


How to train a neural network?

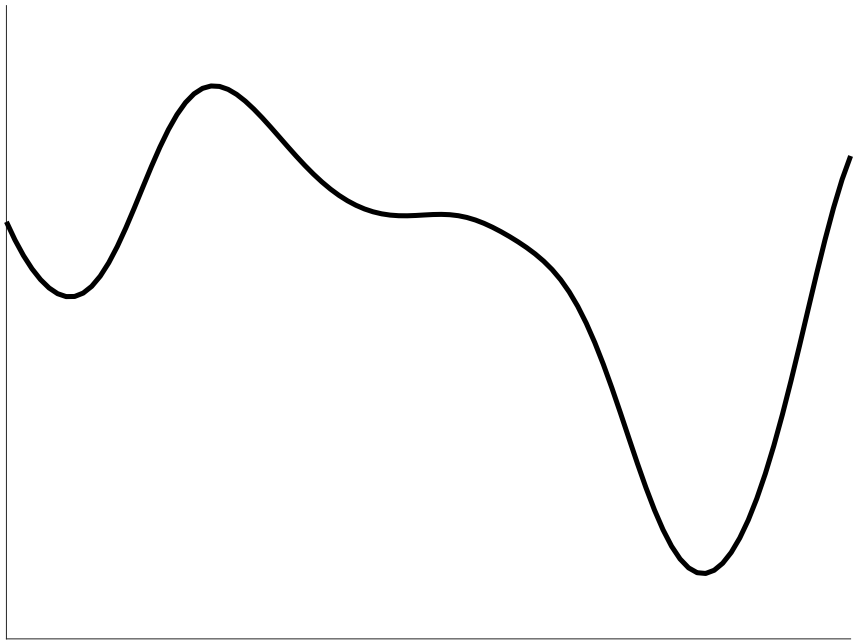
- The backpropagation algorithm.
- The core ideas behind modern feed-forward networks have not changed substantially since the 1980s.
- Improvements today are mainly due to: large datasets and, powerful computers and software infrastructure.
- Algorithmic changes:
  - Cross-entropy error functions instead of mean-squared error.
  - Replacement of sigmoid hidden units with piecewise linear units, such as rectified linear units.

Contents

Gaussian processes

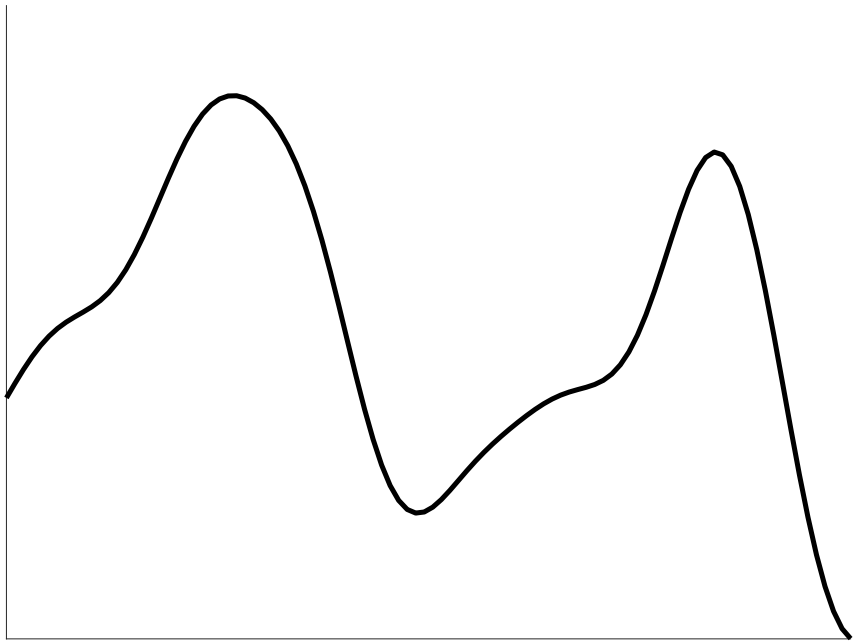


Gaussian processes



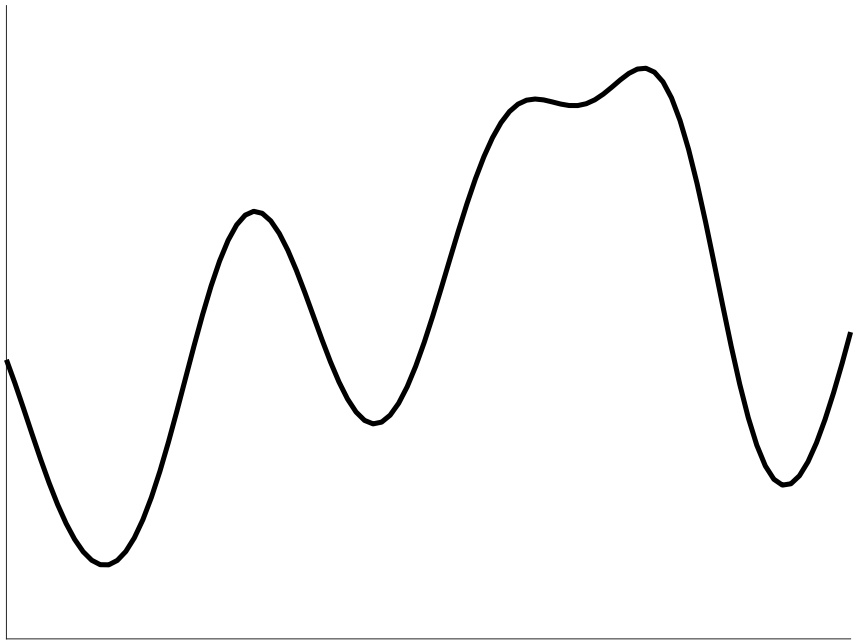
$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Gaussian processes



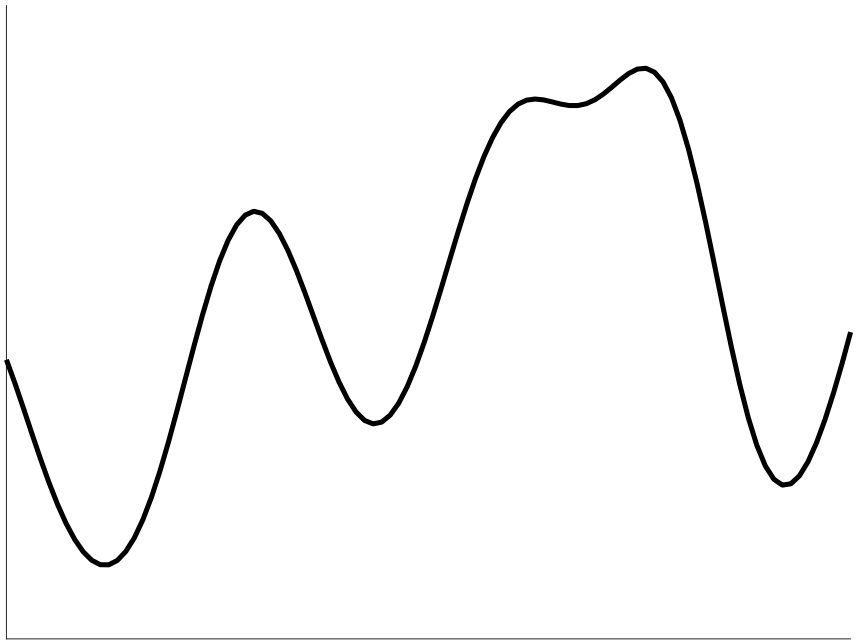
$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

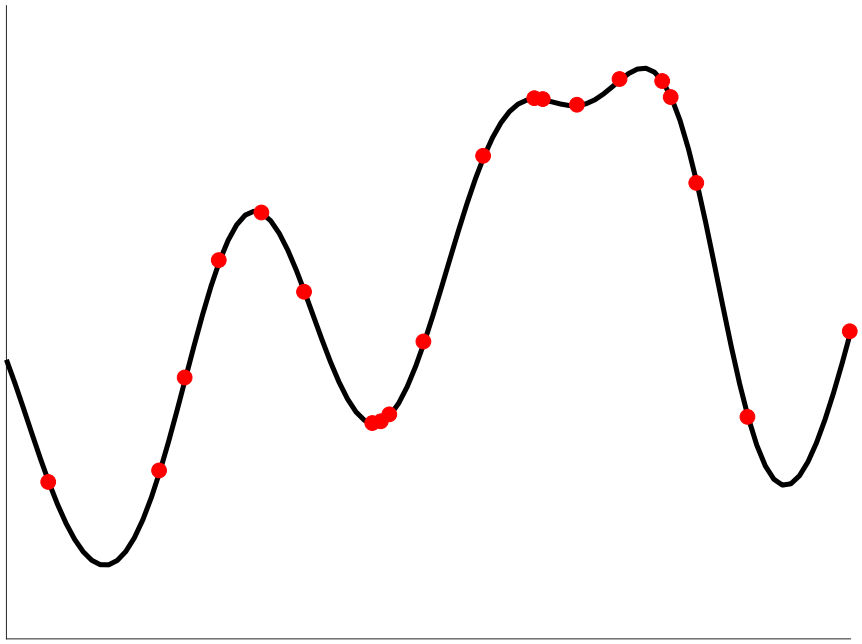
Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

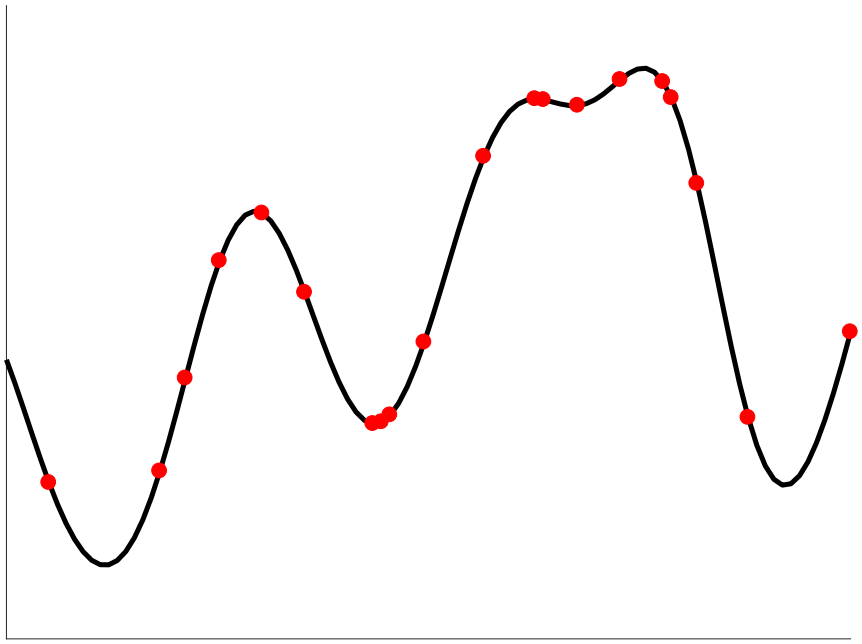
Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

Gaussian processes

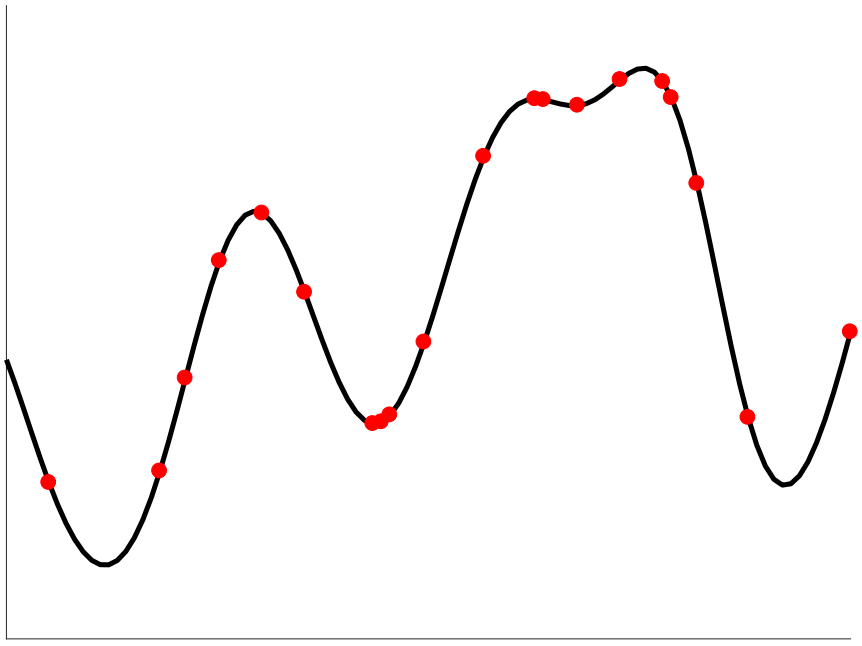


$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \right)$$

Gaussian processes



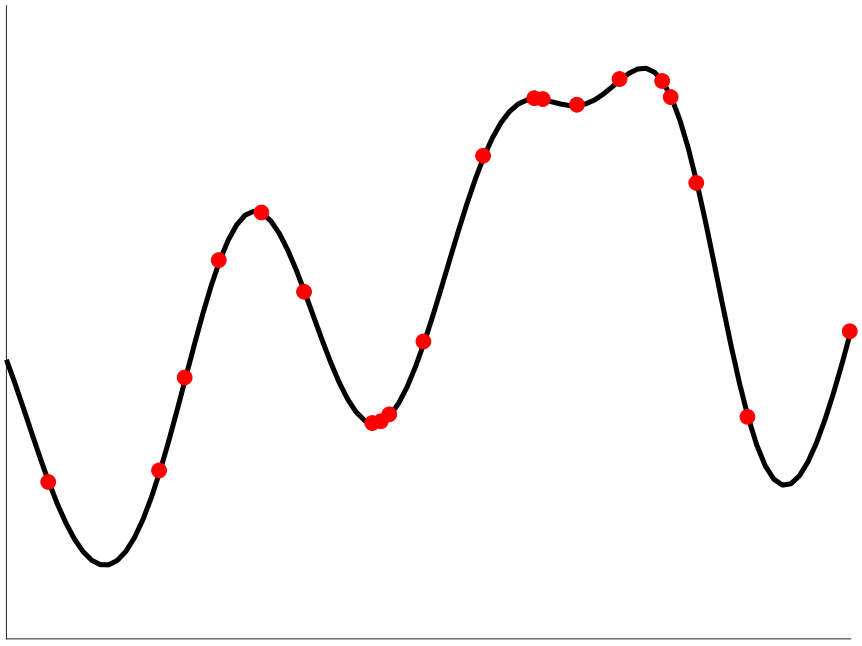
$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \right)$$

$$\mathbf{f}$$

Gaussian processes



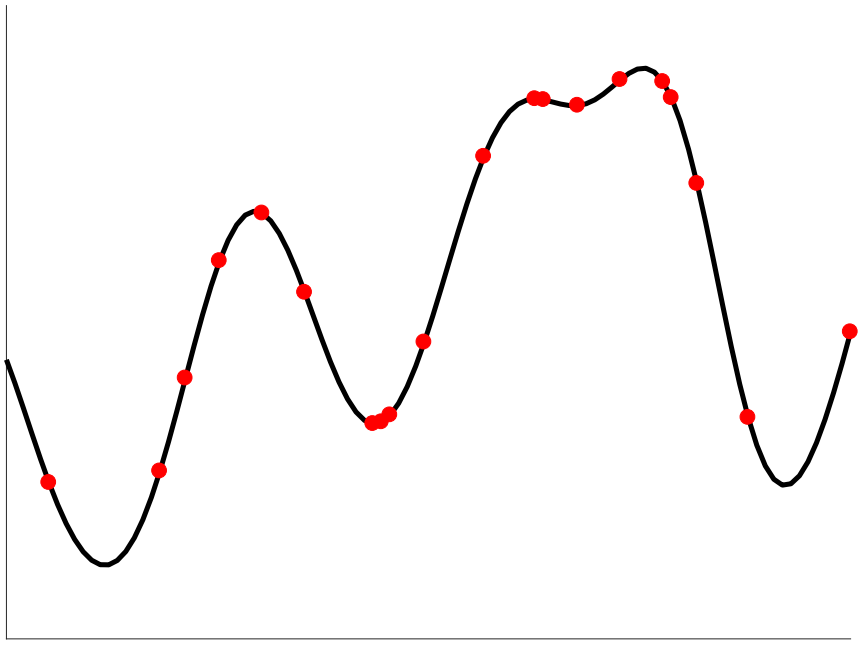
$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \right)$$

$\mathbf{f}$ $\qquad\qquad$ $\mathbf{K}$

Gaussian processes



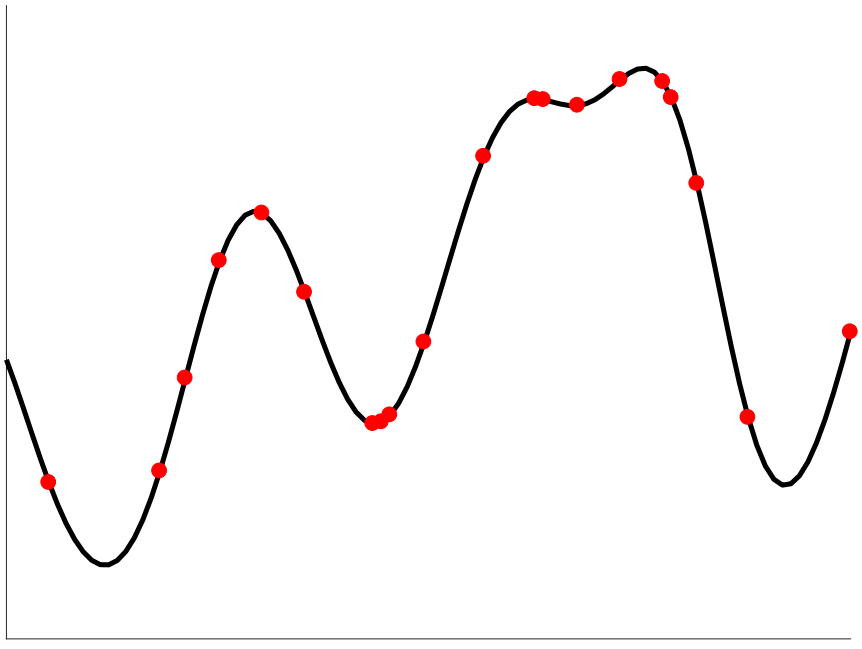
$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \right)$$

$$\mathbf{f} \qquad \mathbf{0} \qquad \mathbf{K}$$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

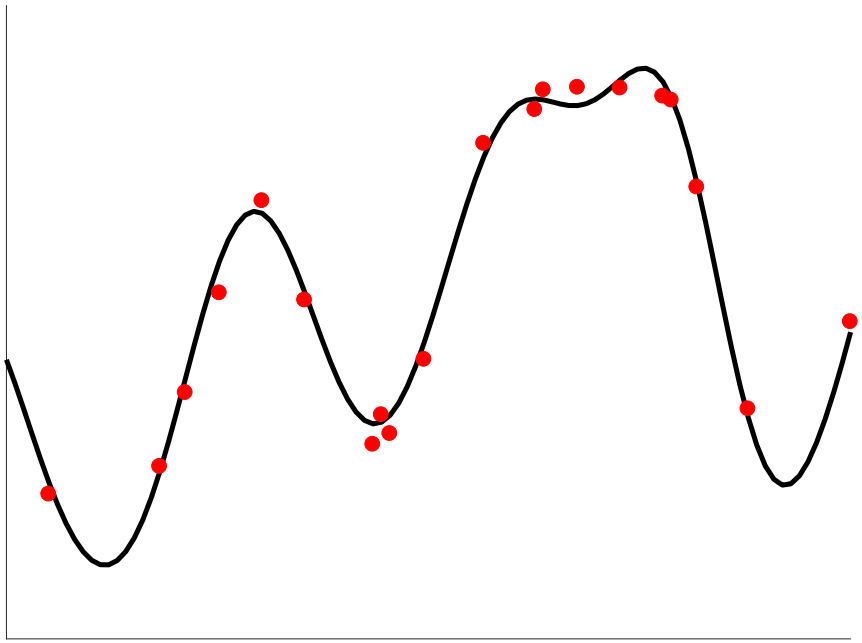
$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \right)$$

$\mathbf{f}$ $\quad$ $\mathbf{0}$ $\quad$ $\mathbf{K}$

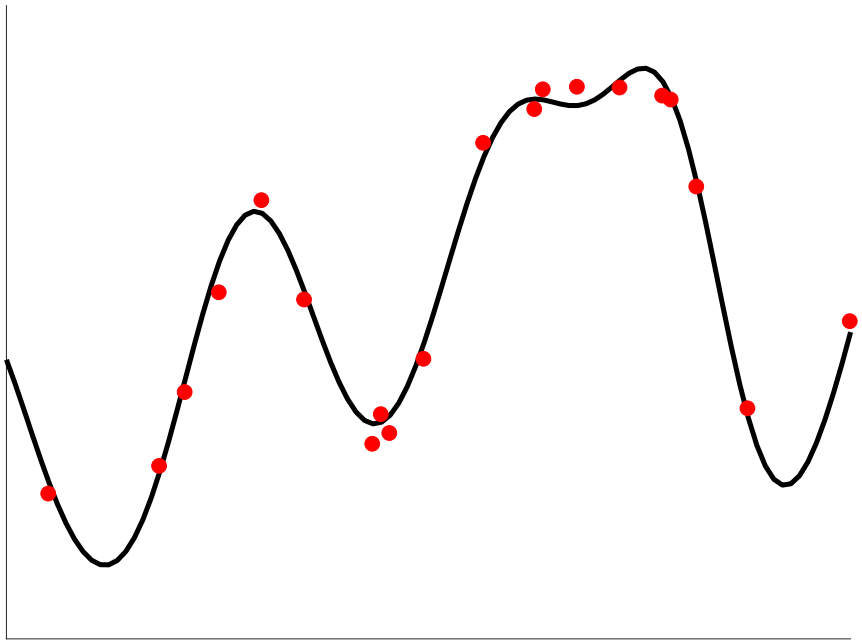
For prediction: $p(f(\mathbf{x}_*)|\mathbf{f})$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

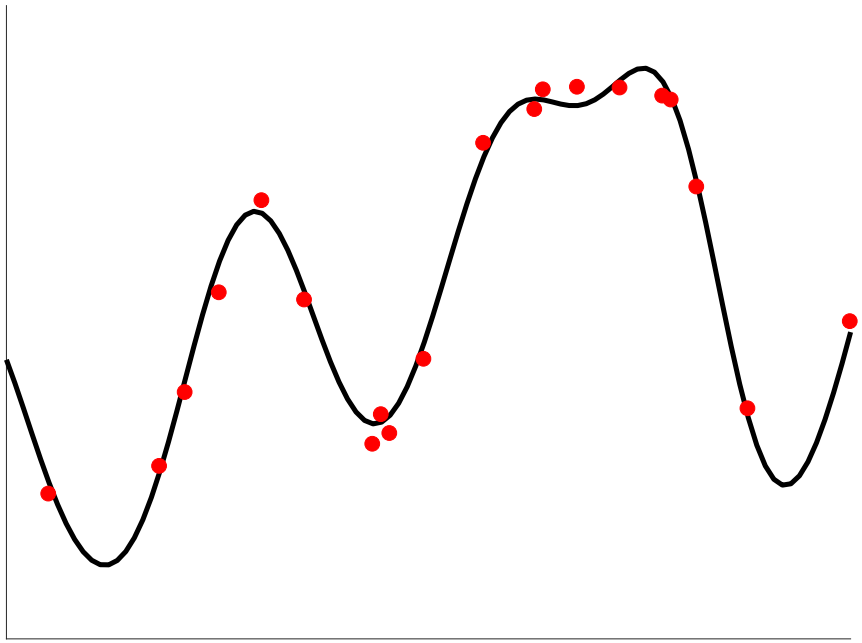
Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

Gaussian processes

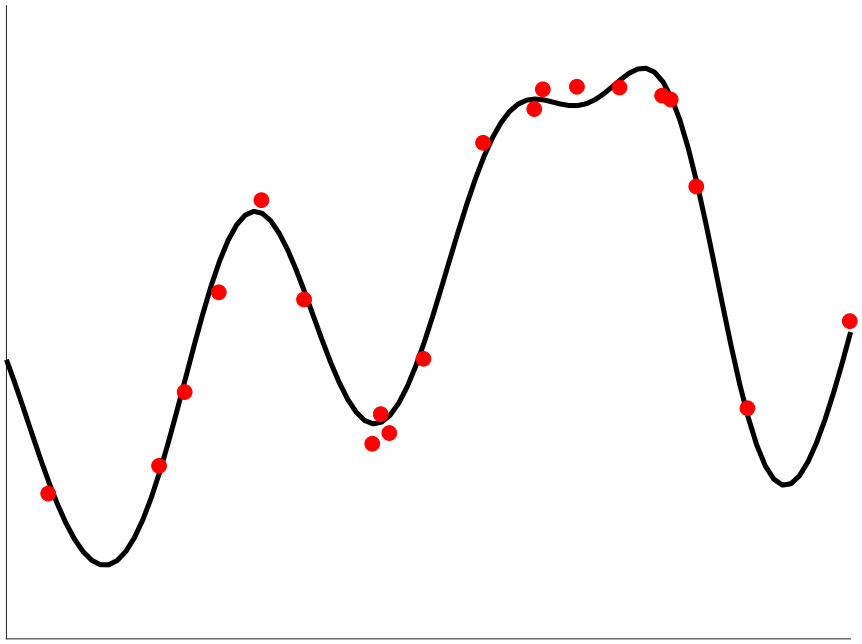


$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Gaussian processes



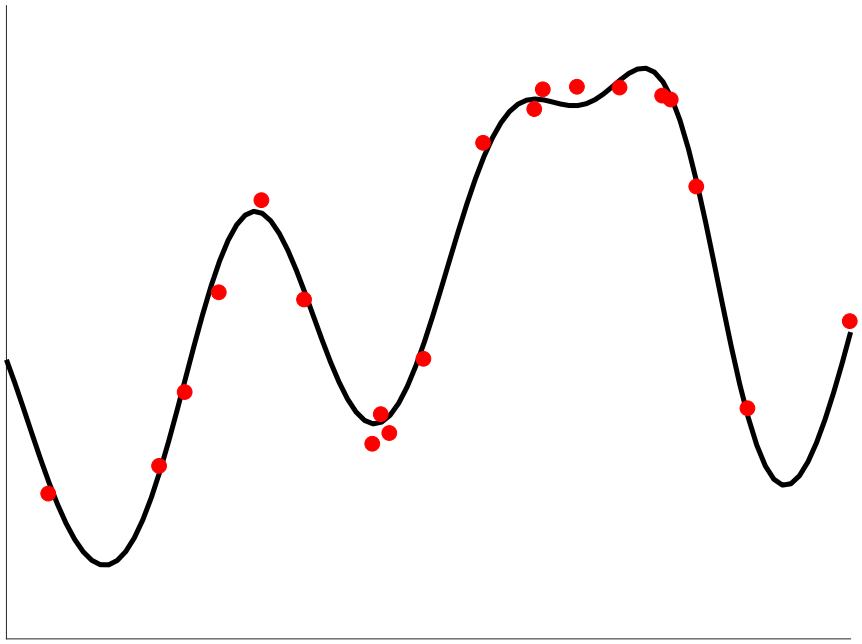
$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\mathcal{D} = \{(\mathbf{x}_i, y(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

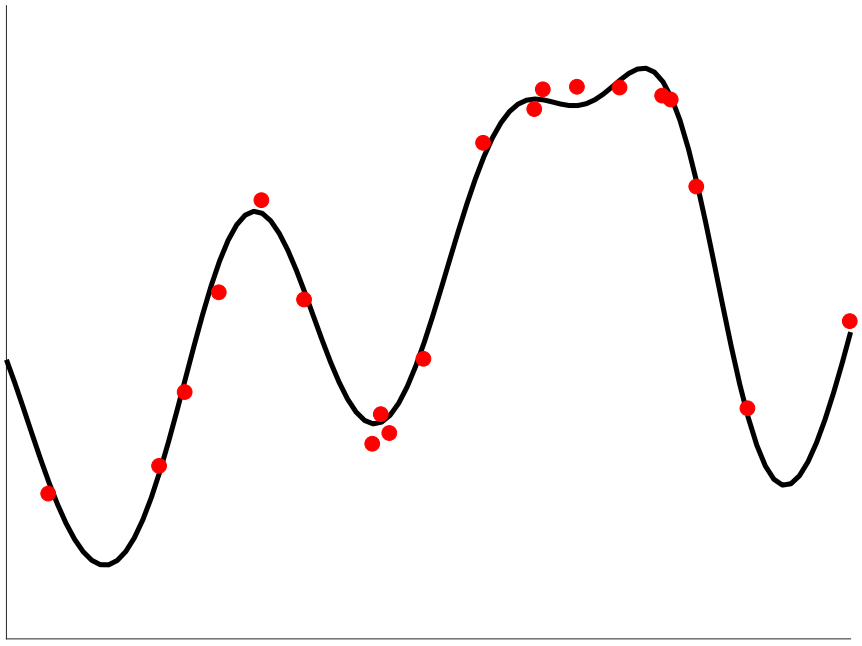
$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\mathcal{D} = \{(\mathbf{x}_i, y(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} y(\mathbf{x}_1) \\ \vdots \\ y(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \right)$$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

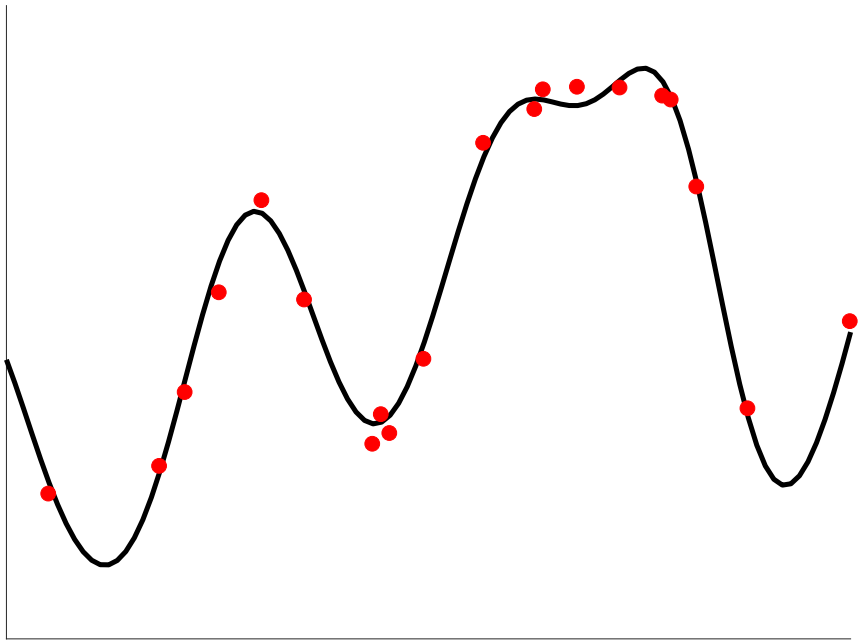
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\mathcal{D} = \{(\mathbf{x}_i, y(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} y(\mathbf{x}_1) \\ \vdots \\ y(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \right)$$

$$\mathbf{y}$$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

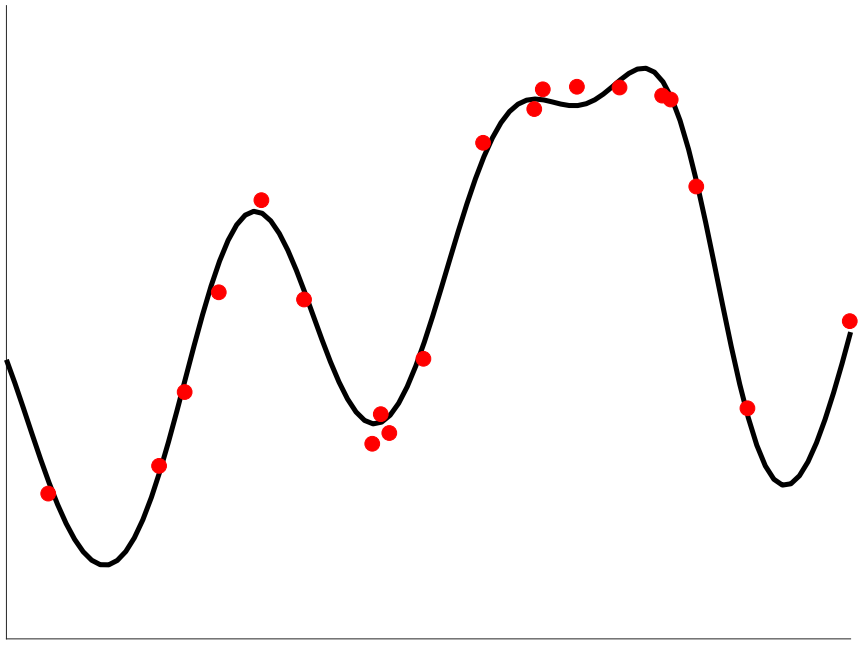
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\mathcal{D} = \{(\mathbf{x}_i, y(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} y(\mathbf{x}_1) \\ \vdots \\ y(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \right)$$

$$\mathbf{y} \qquad\qquad \mathbf{K}$$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

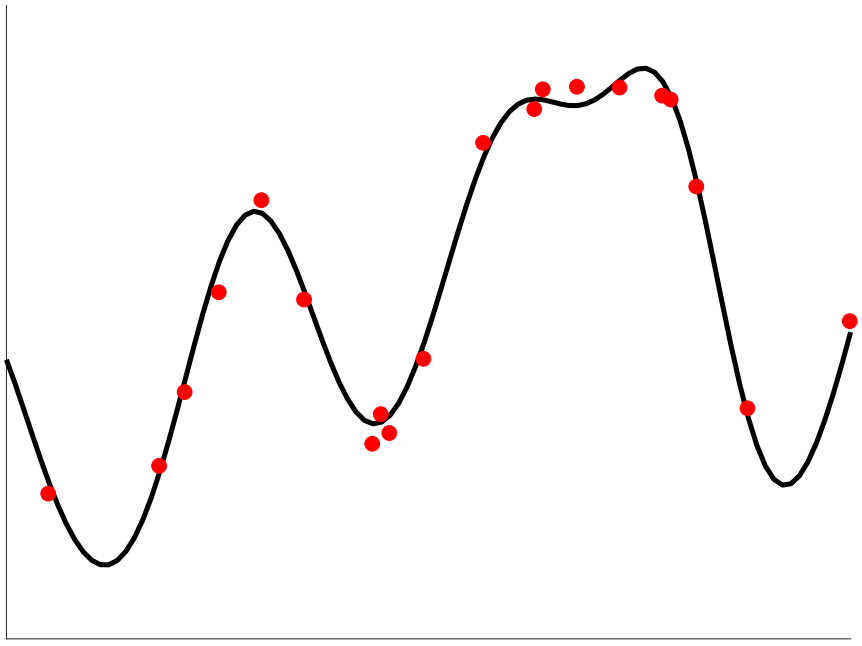
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\mathcal{D} = \{(\mathbf{x}_i, y(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} y(\mathbf{x}_1) \\ \vdots \\ y(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \right)$$

$$\mathbf{y} \qquad\qquad \mathbf{K} \qquad +$$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

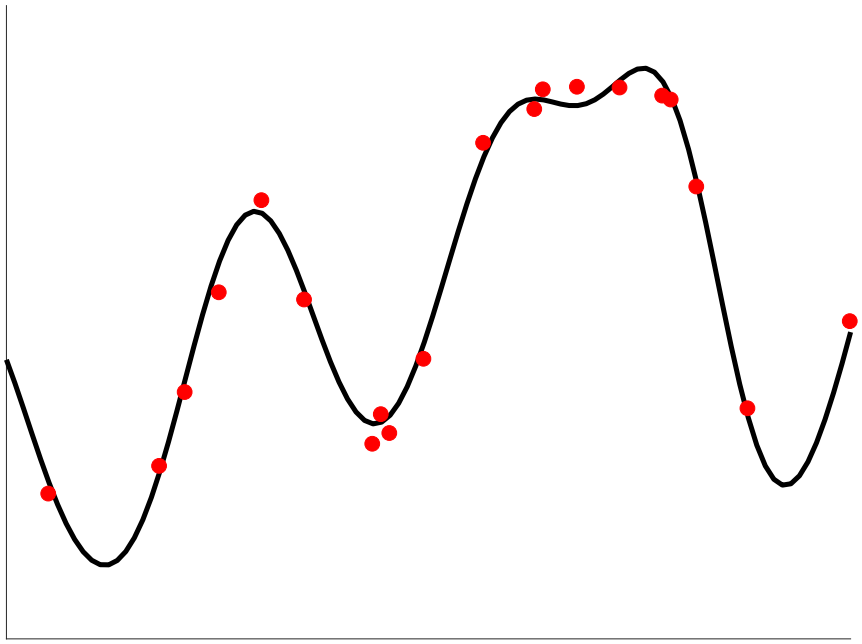
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\mathcal{D} = \{(\mathbf{x}_i, y(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} y(\mathbf{x}_1) \\ \vdots \\ y(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \right)$$

$$\mathbf{y} \qquad\qquad \mathbf{K} \qquad + \qquad \sigma^2 \mathbf{I}$$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

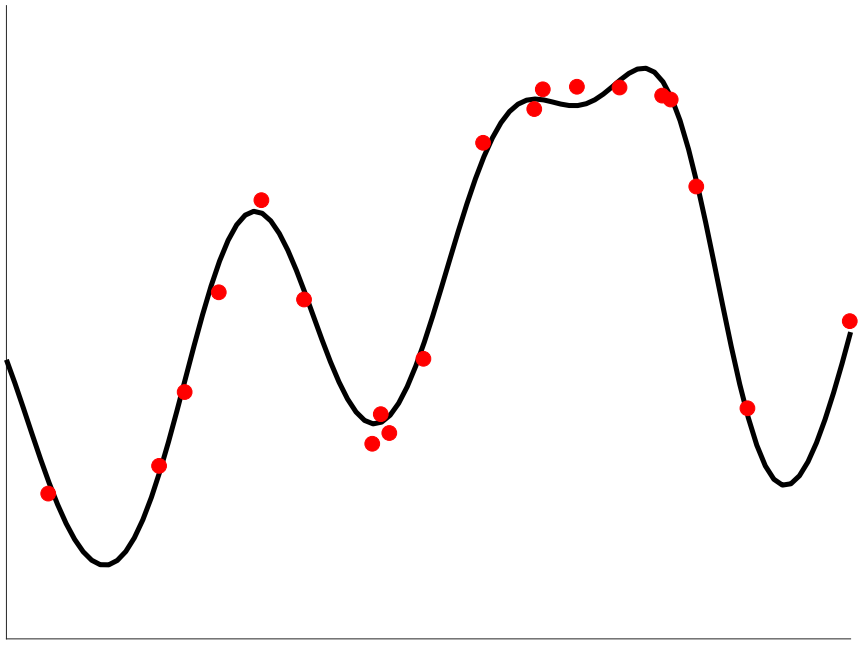
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\mathcal{D} = \{(\mathbf{x}_i, y(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} y(\mathbf{x}_1) \\ \vdots \\ y(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \right)$$

$$\mathbf{y} \qquad \mathbf{0} \qquad \mathbf{K} \qquad + \qquad \sigma^2\mathbf{I}$$

Gaussian processes



$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\mathcal{D} = \{(\mathbf{x}_i, y(\mathbf{x}_i)) | i = 1, \ldots, N\}$$

$$\begin{bmatrix} y(\mathbf{x}_1) \\ \vdots \\ y(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \right)$$

$$\mathbf{y} \qquad \mathbf{0} \qquad \mathbf{K} \qquad + \qquad \sigma^2\mathbf{I}$$

For prediction: $p(f(\mathbf{x}_*)|\mathbf{y})$

Deep Gaussian Processes

Deep Gaussian Processes

Andreas C. Damianou **Neil D. Lawrence**
Dept. of Computer Science & Sheffield Institute for Translational Neuroscience,
University of Sheffield, UK

Abstract

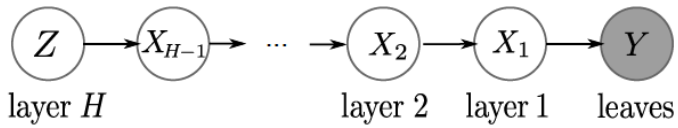
In this paper we introduce deep Gaussian process (GP) models. Deep GPs are a deep belief network based on Gaussian process mappings. The data is modeled as the output of a multivariate

the question as to whether deep structures and the learning of abstract structure can be undertaken in *smaller* data sets. For smaller data sets, questions of generalization arise: to demonstrate such structures are justified it is useful to have an objective measure of the model's applicability.

The traditional approach to deep learning is based around

Published at AISTATS, 2013.

Deep GPs: model



- The leaf nodes $\mathbf{Y} \in \mathbb{R}^{N \times D}$ (observations)
- Intermediate latent spaces $\mathbf{X}_h \in \mathbb{R}^{N \times Q_h}$, with $h = 1, \ldots, H-1$, H number of hidden layers.
- Parent latent node $\mathbf{Z} = \mathbf{X}_H \in \mathbb{R}^{N \times Q_Z}$ (unobserved or inputs).
- Example generative process with two layers:

$$x_{nq} = f_q^X(\mathbf{z}_n) + \epsilon_{nq}^X, \quad q = 1, \ldots, Q \quad \mathbf{z}_n \in \mathbb{R}^{Q_Z}$$
$$y_{nd} = f_d^Y(\mathbf{x}_n) + \epsilon_{nd}^Y, \quad d = 1, \ldots, D \quad \mathbf{x}_n \in \mathbb{R}^{Q}$$

- The functions $f^X \sim \mathcal{GP}(0, k^X(\mathbf{Z}, \mathbf{Z}))$ and $f^Y \sim \mathcal{GP}(0, k^Y(\mathbf{X}, \mathbf{X}))$.

Deep GPs: inference

- Variational inference requires the optimisation of

$$\log p(\mathbf{Y}) = \log \int_{\mathbf{X},\mathbf{Z}} p(\mathbf{Y}|\mathbf{X}) p(\mathbf{X}|\mathbf{Z}) p(\mathbf{Z}) d\mathbf{X} d\mathbf{Z}$$

- Intractability since $\mathbf{X}$ and $\mathbf{Z}$ appear inside the kernel functions.
- A variational lower bound can be found using

$$\mathcal{F} = \int_{\mathbf{X},\mathbf{Z},\mathbf{F}^Y,\mathbf{F}^X} \mathcal{Q} \log \frac{p(\mathbf{Y},\mathbf{F}^Y,\mathbf{X},\mathbf{F}^X,\mathbf{Z})}{\mathcal{Q}} d\mathbf{X} d\mathbf{Z} d\mathbf{F}^Y d\mathbf{F}^X,$$

  where $\mathcal{Q}$ is an approximated posterior to be defined.
- The complete likelihood follows as

$$p(\mathbf{Y},\mathbf{F}^Y,\mathbf{X},\mathbf{F}^X,\mathbf{Z}) = p(\mathbf{Y}|\mathbf{F}^Y) p(\mathbf{F}^Y|\mathbf{X}) p(\mathbf{X}|\mathbf{F}^X) p(\mathbf{F}^X|\mathbf{Z}) p(\mathbf{Z}).$$

Deep GPs: inference

- The key trick for inference: augment the probability space above with K auxiliary pseudo-inputs $\widetilde{\mathbf{X}} \in \mathbb{R}^{K \times Q}$ and $\widetilde{\mathbf{Z}} \in \mathbb{R}^{K \times Q_z}$.
- These auxiliary pseudo-inputs correspond to function values $\mathbf{U}^Y \in \mathbb{R}^{K \times D}$ and $\mathbf{U}^X \in \mathbb{R}^{K \times Q}$.
- The augmented probability space is given as

$$p(\mathbf{Y}, \mathbf{F}^Y, \mathbf{X}, \mathbf{F}^X, \mathbf{Z}, \mathbf{U}^Y, \mathbf{U}^X, \widetilde{\mathbf{X}}, \widetilde{\mathbf{Z}}) = p(\mathbf{Y}|\mathbf{F}^Y)p(\mathbf{F}^Y|\mathbf{U}^Y, \mathbf{X})p(\mathbf{U}^Y|\widetilde{\mathbf{X}}) \times p(\mathbf{X}|\mathbf{F}^X)p(\mathbf{F}^X|\mathbf{U}^X, \mathbf{Z})p(\mathbf{U}^X|\widetilde{\mathbf{Z}})p(\mathbf{Z}).$$

Deep GPs: inference

- The posterior distribution is chosen as

$$\mathcal{Q} = p(\mathbf{F}^Y|\mathbf{U}^Y, \mathbf{X})q(\mathbf{U}^Y)q(\mathbf{X})p(\mathbf{F}^X|\mathbf{U}^X, \mathbf{Z})q(\mathbf{U}^X)q(\mathbf{Z}),$$

  where $q(\mathbf{U}^Y)$ and $q(\mathbf{U}^X)$ are allowed a free-form and

$$q(\mathbf{X}) = \prod_{q=1}^{Q} \mathcal{N}(\boldsymbol{\mu}_q^X, \mathbf{S}_q^X), \qquad q(\mathbf{Z}) = \prod_{q=1}^{Q_Z} \mathcal{N}(\boldsymbol{\mu}_q^Z, \mathbf{S}_q^Z).$$

- With these choices, the new lower bound is

$$\mathcal{F} = \int \mathcal{Q} \log \frac{p(\mathbf{Y}|\mathbf{F}^Y)p(\mathbf{U}^Y)p(\mathbf{X}|\mathbf{F}^X)p(\mathbf{U}^X)p(\mathbf{Z})}{\mathcal{Q}'} d\mathbf{X}d\mathbf{Z}d\mathbf{F}^Y d\mathbf{F}^X \mathbf{U}^Y d\mathbf{U}^X,$$

  where $Q' = q(\mathbf{U}^Y)q(\mathbf{X})q(\mathbf{U}^X)q(\mathbf{Z})$.

- The above lower bound can be expressed analytically.

Recent advances in Deep GPs

Deep Gaussian Processes for Regression using Approximate Expectation Propagation

Thang D. Bui[1] TDB40@CAM.AC.UK
José Miguel Hernández-Lobato[2] JMH@SEAS.HARVARD.EDU
Daniel Hernández-Lobato[3] DANIEL.HERNANDEZ@UAM.ES
Yingzhen Li[1] YL494@CAM.AC.UK
Richard E. Turner[1] RET26@CAM.AC.UK
[1]University of Cambridge, [2]Harvard University, [3]Universidad Autónoma de Madrid

Abstract

Deep Gaussian processes (DGPs) are multi-layer hierarchical generalisations of Gaussian processes (GPs) and are formally equivalent to neural networks with multiple, infinitely wide hidden layers. DGPs are nonparametric proba-

study a multi-layer hierarchical generalisation of GPs or deep Gaussian Processes (DGPs) (Damianou & Lawrence, 2013) for supervised learning tasks. A GP is equivalent to an infinitely wide neural network with single hidden layer and similarly a DGP is a multi-layer neural network with multiple infinitely wide hidden layers (Neal, 1995). The mapping between layers in this type of network is parame-

Published at ICML, 2016.

Recent advances in Deep GPs

Sequential Inference for Deep Gaussian Process

Yali Wang
Chinese Academy of Sciences
yl.wang@siat.ac.cn

Marcus Brubaker
University of Toronto
mbrubake@cs.toronto.edu

Brahim Chaib-draa
Laval University
chaib@ift.ulaval.ca

Raquel Urtasun
University of Toronto
urtasun@cs.toronto.edu

Abstract

A deep Gaussian process (DGP) is a deep network in which each layer is modelled with a Gaussian process (GP). It is a flexible model that can capture highly-nonlinear functions for complex data sets. However,

work as the number of hidden units goes to infinity yields a Gaussian process. Inspired by this and the success of multi-layer neural networks, Damianou et al. [11] proposed deep GPs (DGPs), where each layer of a deep network structure is modelled as a GP. DGPs can address both input-dependent non-stationarity and multi-output modeling via its flexible deep structure. More importantly, it allows us to learn

Published at AISTATS, 2016.

Recent advances in Deep GPs

Random Feature Expansions for Deep Gaussian Processes

Kurt Cutajar [1] **Edwin V. Bonilla** [2] **Pietro Michiardi** [1] **Maurizio Filippone** [1]

Abstract

The composition of multiple Gaussian Processes as a Deep Gaussian Process (DGP) enables a deep probabilistic nonparametric approach to flexibly tackle complex machine learning problems with sound quantification of uncertainty. Existing inference approaches for DGP models have limited scalability and are notoriously cumbersome to construct. In this work we introduce a novel formulation of DGPs based on random feature expansions that we train using stochastic variational inference. This yields a practical learn-

2006) such that the output of each layer of GPs forms the input to the GPs at the next layer, effectively implementing a deep probabilistic nonparametric model for compositions of functions (Neal, 1996; Duvenaud et al., 2014).

Because of their probabilistic formulation, it is natural to approach the learning of DGPs through Bayesian inference techniques; however, the application of such techniques to learn DGPs leads to various forms of intractability. A number of contributions have been proposed to recover tractability, extending or building upon the literature on approximate methods for GPs. Nevertheless, only few works leverage one of the key features that arguably make DNNs

Published at ICML, 2017.

Recent advances in Deep GPs

Doubly Stochastic Variational Inference for Deep Gaussian Processes

Hugh Salimbeni
Imperial College London
hrs13@ic.ac.uk

Marc Deisenroth
Imperial College London
m.deisenroth@imperial.ac.uk

Abstract

Gaussian processes (GPs) are a good choice for function approximation as they are flexible, robust to over-fitting, and provide well-calibrated predictive uncertainty. Deep Gaussian processes (DGPs) are multi-layer generalisations of GPs, but inference in these models has proved challenging. Existing approaches to inference in DGP models assume approximate posteriors that force independence between the layers, and do not work well in practice. We present a doubly stochastic variational inference algorithm, which does not force independence between layers. With our method of inference we demonstrate that a DGP model can be used effectively on data ranging in size from hundreds to a billion points. We provide strong empirical evidence that our inference scheme for DGPs works well in practice in both classification and regression.

To appear at NeurIPS, 2017.

Contents

Deep Kernel Learning

Deep Kernel Learning

Andrew Gordon Wilson* CMU
Zhiting Hu* CMU
Ruslan Salakhutdinov University of Toronto
Eric P. Xing CMU

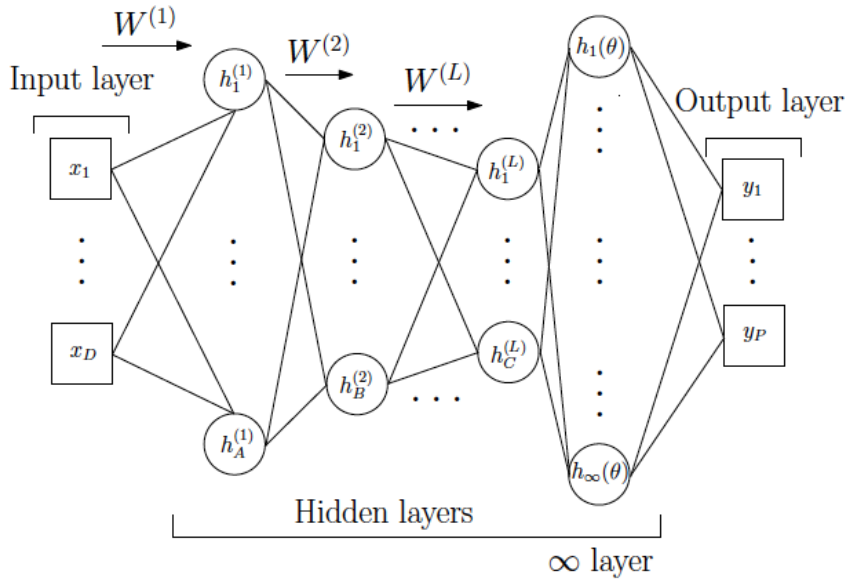
Abstract

We introduce scalable deep kernels, which combine the structural properties of deep learning architectures with the non-parametric flexibility of kernel methods. Specifically, we transform the inputs of a spectral mixture base kernel with a deep architecture, using local kernel interpolation,

(1996), who had shown that Bayesian neural networks with infinitely many hidden units converged to Gaussian processes with a particular kernel (covariance) function. Gaussian processes were subsequently viewed as flexible and interpretable alternatives to neural networks, with straightforward learning procedures. Where neural networks used finitely many highly adaptive basis functions, Gaussian processes typically used infinitely many fixed basis functions. As argued by MacKay (1998), Hinton et al

Published at AISTATS, 2016.

Deep Kernel Learning



- Starting from a base kernel $k(\mathbf{x}_i, \mathbf{x}_j|\theta)$, the inputs are transformed as

$$k(\mathbf{x}_i, \mathbf{x}_j|\theta) \rightarrow k(g(\mathbf{x}_i, \mathbf{w}), g(\mathbf{x}_j, \mathbf{w})|\theta, \mathbf{w}),$$

  where $g(\mathbf{x}, \mathbf{w})$ is a non-linear mapping given by a deep architecture parametrized by weights $\mathbf{w}$.
- For scalability, they use KISS-GP (Kernel Interpolation for Scalable Structured), $K \approx MK_{U,U}^{\text{deep}}M^\top$

Stochastic Variational Deep Kernel Learning

Stochastic Variational Deep Kernel Learning

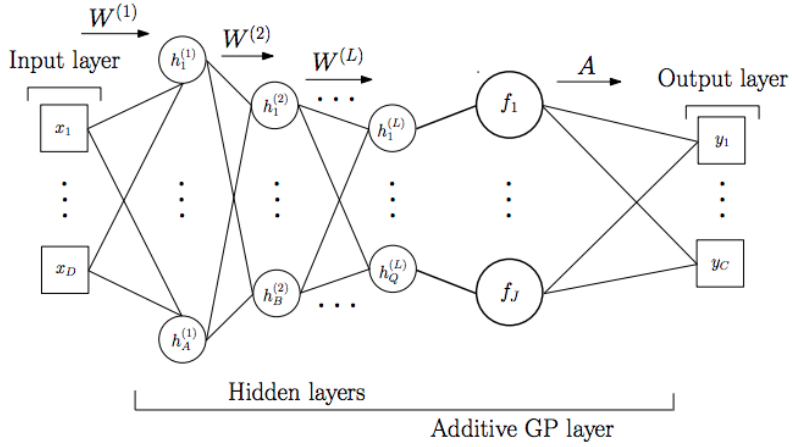
Andrew Gordon Wilson* Cornell University
Zhiting Hu* CMU
Ruslan Salakhutdinov CMU
Eric P. Xing CMU

Abstract

Deep kernel learning combines the non-parametric flexibility of kernel methods with the inductive biases of deep learning architectures. We propose a novel deep kernel learning model and stochastic variational inference procedure which generalizes deep kernel learning approaches to enable classification, multi-task learning, additive covariance structures, and stochastic gradient training. Specifically, we apply additive base kernels to subsets of output features from deep neural architectures, and jointly learn the parameters of the base kernels and deep network through a Gaussian process marginal likelihood objective. Within this framework, we derive an efficient form of stochastic variational inference which leverages local kernel interpolation, inducing points, and structure exploiting algebra. We show improved performance over stand alone deep networks, SVMs, and state of the art scalable Gaussian processes on several classification benchmarks, including an airline delay dataset containing 6 million training points, CIFAR, and ImageNet.

Published at NeurIPS, 2016.

Stochastic Variational Deep Kernel Learning



1. A deep non-linear transformation $\mathbf{h}(\mathbf{x}, \mathbf{w})$ parametrized by $\mathbf{w}$ is applied to the input vector $\mathbf{x}$ to produce Q features at the final layer L, $h_1^L, \ldots, h_Q^L$.
2. J Gaussian processes with base kernels $k_1, \ldots, k_J$, are applied to subsets of these features corresponding to an *additive GP*.
3. These GPs are linearly mixed by a matrix $A \in \mathbb{R}^{C \times J}$, and transformed by an observation model, to produce the output variables $y_1, \ldots, y_C$.