CS-E4075 Special course on Gaussian processes: Session #7

Arno Solin

Aalto University

arno.solin@aalto.fi

Monday February 1, 2021

Roadmap for today

1. Convolutions
2. Convolutional Gaussian Processes
3. Recap
4. Bibliography

Convolutions

- A convolution is an operation between two functions
- It is an example of an integral transform
- And thus related to, e.g., Fourier, Laplace, and other similar transforms (which also play a major role in GP methods)

Definition

Convolutions

In common engineering notational convention, a convolution between two functions $f(\cdot)$ and $g(\cdot)$ can is denoted by

$$f(t) * g(t) := \underbrace{\int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, \mathrm{d}\tau}_{(f*g)(t)}$$

Intuitive interpretation: The convolution formula can be described as a weighted average of the function $f(\tau)$ at the moment t where the weighting is given by $g(-\tau)$ simply shifted by amount t.

Properties

- **Commutativity**

$$f * g = g * f$$

- **Associativity**

$$f * (g * h) = (f * g) * h$$

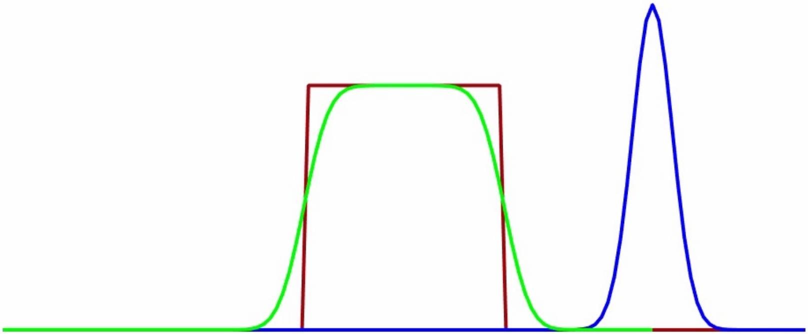
- **Distributivity**

$$f * (g + h) = (f * g) + (f * h)$$

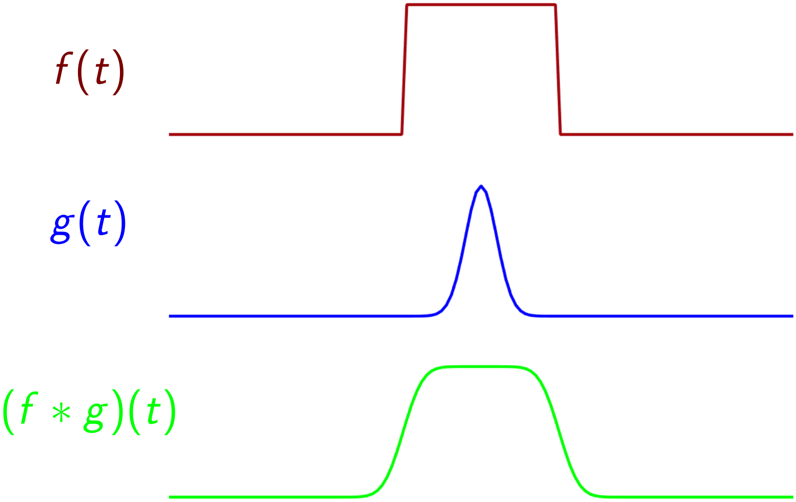
- **Associativity with scalar multiplication**

$$a\,(f * g) = (a\,f) * g$$

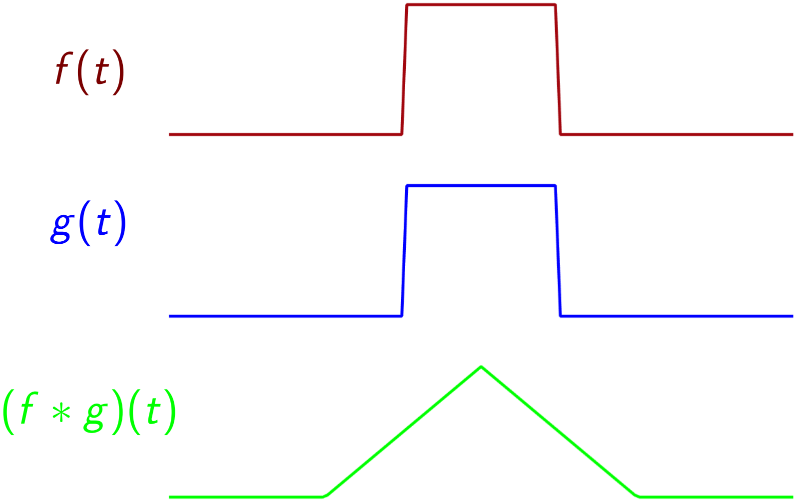
Illustrative example



Example convolutions



Example convolutions

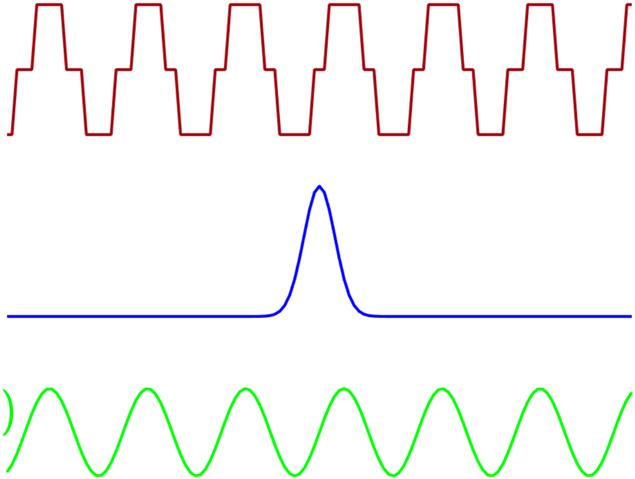


Example convolutions

$f(t)$

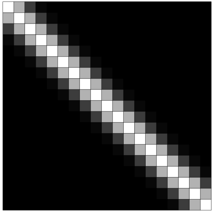
$g(t)$

$(f * g)(t)$



Discrete convolutions

- Depending a bit on what you aim for, you might either consider the (continuous) integral or resort to an approximation by discretization
- In practice, you may replace the the integral with a finite sum
- This also allows for turning the convolution into a linear mapping, where the convolution operation can be turned into a matrix-vector multiplication



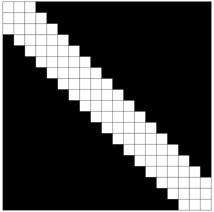


Image filtering

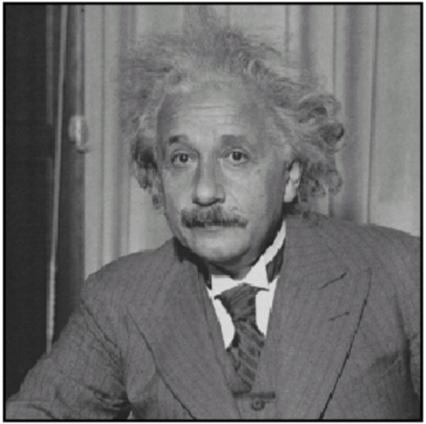
- Image filters in spatial domain
  - A filter is a mathematical operation of a grid of numbers
  - Smoothing, sharpening, edge detection
- Image filters in the frequency domain
  - Filtering is a way to modify the frequencies of images
  - Hybrid images, sampling, image resizing
- Templates and image pyramids
  - Filtering is a way to match a template to the image
  - Detection, coarse-to-fine registration

Image filters

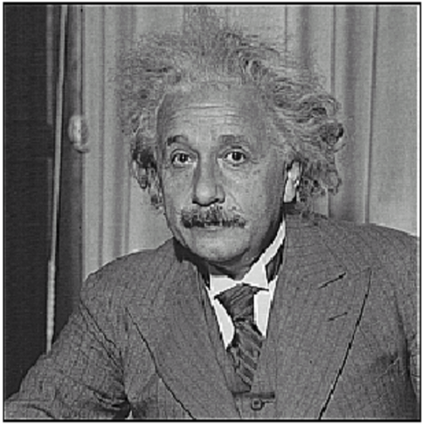
- Image filtering:
  A function of the local neighbourhood at each point in the image.
- The weights for the local neighbourhood are called the filter kernel.
- A sharpening kernel:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Sharpening

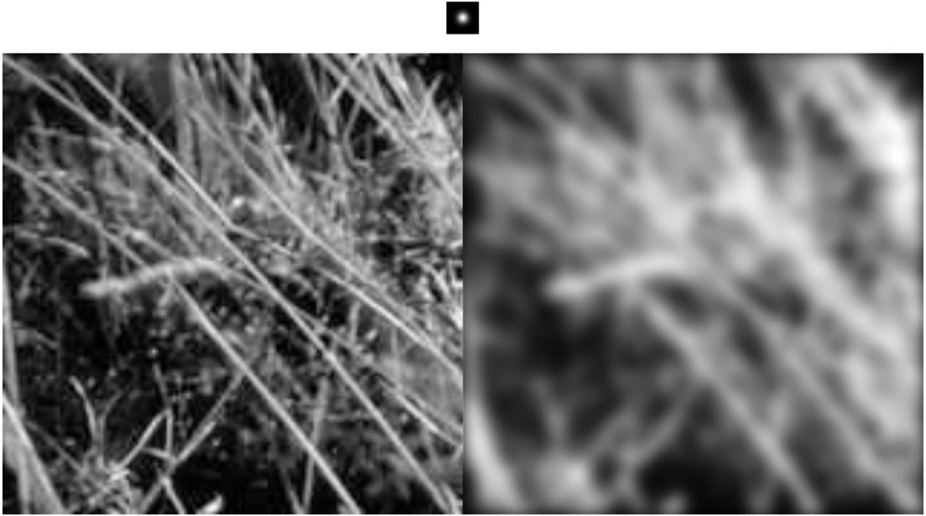


before



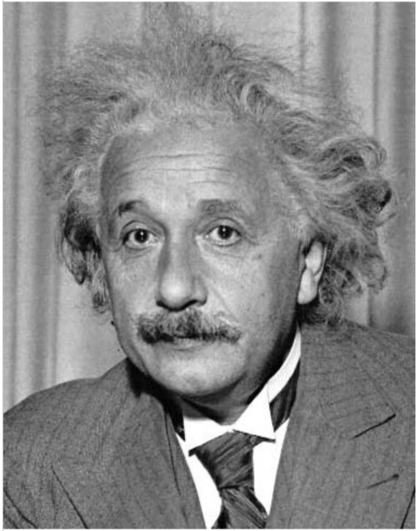
after

Gaussian filter



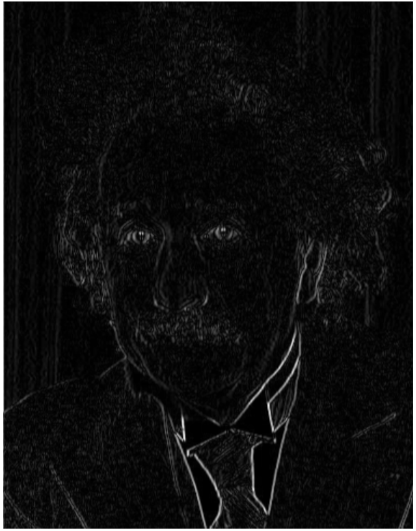
Box filter

Edge detection

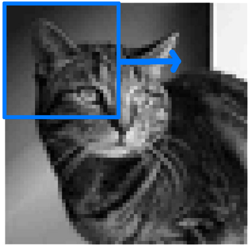


1	0	-1
2	0	-2
1	0	-1

Sobel

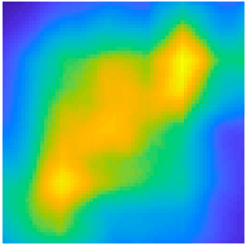


What if the convolution kernel is more complicated?



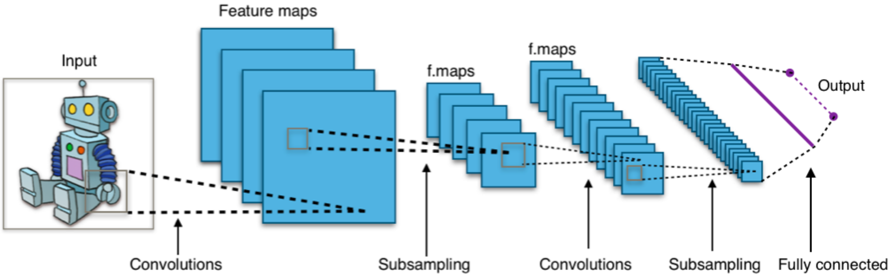
Input image

Convolution kernel
(what to extract)



Output
'cat-likeness'

... which leads us to CNNs



- Convolutional neural networks leverage convolutions
- The convolutions are learned as to best explain the data
- A lot of weights are parameters that need to be ‘learned’

Image: Typical cnn, Wikimedia Commons.

What about GPs then?

$$f(\mathbf{x}) \sim \mathcal{GP}(0, \kappa(\mathbf{x}, \mathbf{x}')) \qquad \text{GP prior}$$

$$\mathbf{y} \mid \mathbf{f} \sim \prod_{i=1}^{n} p(y_i \mid f(\mathbf{x}_i)) \qquad \text{likelihood}$$

Convolutional Gaussian process models

- **Convolutional likelihood models**
  (observations are made through convolutions)
- **Convolutional GP priors**
  (the GP prior itself has a convolutional structure)
- **Convolutions in deep GP models**
  (this is more in the domain of the next lecture)

Convolutional likelihood models

- Consider a GP model where the latent process is observed through a convolution
- As an example, this would be the case for this 1D model:

$$f(t) \sim \mathcal{GP}(0, \kappa(t, t')) \qquad \text{GP prior}$$

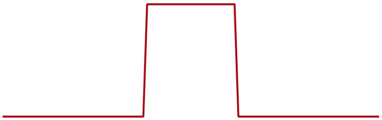
$$y_i = \int_{-\infty}^{\infty} f(\tau)\, g(t_i - \tau)\, \mathrm{d}\tau + \varepsilon_i \qquad \text{likelihood}$$

  for $i = 1, 2, \ldots, n$ and where $g(\cdot)$ is the convolution kernel and $\varepsilon_i \sim \mathrm{N}(0, \sigma_\mathrm{n}^2)$
- This particular problem can be seen as a deconvolution problem (a type of an *inverse* problem)

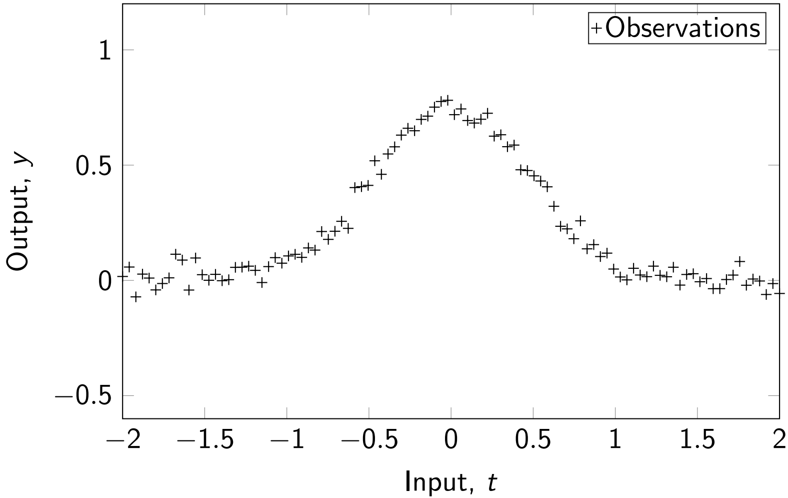
Convolutional likelihood models (example)

- For simplicity, let's assume we know the convolutior ("box filter") kernel:

$$g(t) = \begin{cases} 1 & \text{for } -\frac{1}{2} < t < \frac{1}{2} \\ 0 & \text{elsewhere} \end{cases}$$



Convolutional likelihood models (example)



(remember that we observe the 'true' process through a convolution now)

Convolutional likelihood models

- The likelihood model looks tricky

$$f(t) \sim \mathcal{GP}(0, \kappa(t, t')) \qquad \text{GP prior}$$

$$y_i = \int_{-\infty}^{\infty} f(\tau)\, g(t_i - \tau)\, \mathrm{d}\tau + \varepsilon_i \qquad \text{likelihood}$$

- But let's define an operator $\mathcal{C}$

$$(\mathcal{C}\, f)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t - \tau)\, \mathrm{d}\tau$$

- Which now allows us to re-write the observation model as (with some misuse of notation, sorry)

$$y_i = \mathcal{C}\, f(t_i) + \varepsilon_i$$

Convolutional likelihood models

- The next step hinges on the simple, but very powerful realization that Gaussians are closed under linear transformations:

  If $\boldsymbol{x}$ is Gaussian, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is also Gaussian

- The same principle generalizes to Gaussian processes in the sense that GPs are closed under linear operations:

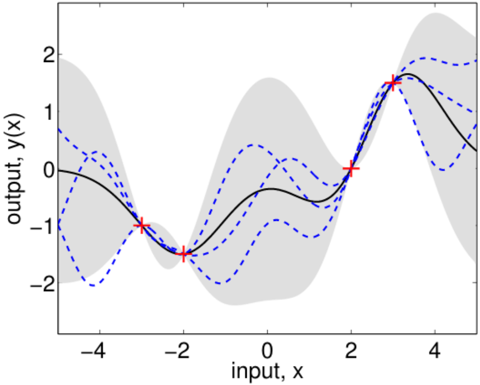
  If $f(\boldsymbol{x})$ is a Gaussian process, $h(\boldsymbol{x}) = \mathcal{A}\, f(\boldsymbol{x})$ is also a Gaussian process (where $\mathcal{A}$ is a linear operator)

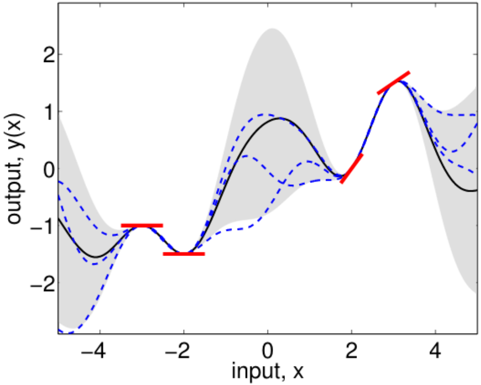
Detour: Derivative observations

- This property is perhaps better understood by considering another linear operation: derivatives of the GP
- Since differentiation is a linear operator, the derivative of a Gaussian process is another Gaussian process
- We can make inference based on the joint Gaussian distribution of function values and partial derivatives
- We get the following (mixed) covariance between function values and partial derivatives, and between partial derivatives

$$\mathrm{cov}\left(f, \frac{\partial f}{\partial t'}\right) = \frac{\partial \kappa(t, t')}{\partial t'} \qquad \text{and} \qquad \mathrm{cov}\left(\frac{\partial f}{\partial t}, \frac{\partial f}{\partial t'}\right) = \frac{\partial^2 \kappa(t, t')}{\partial t \partial t'}$$

Detour: Derivative observations





From Rasmussen and Williams (2006), Sec. 9.4

Convolutional likelihood models

- The realization from before (and understanding that the convolution operator $\mathcal{C}$ is linear) tells us that the convolution likelihood model is actually a Gaussian observation model (conjugate likelihood!).
- Thus the problem is actually pretty close to vanilla Gaussian process regression:

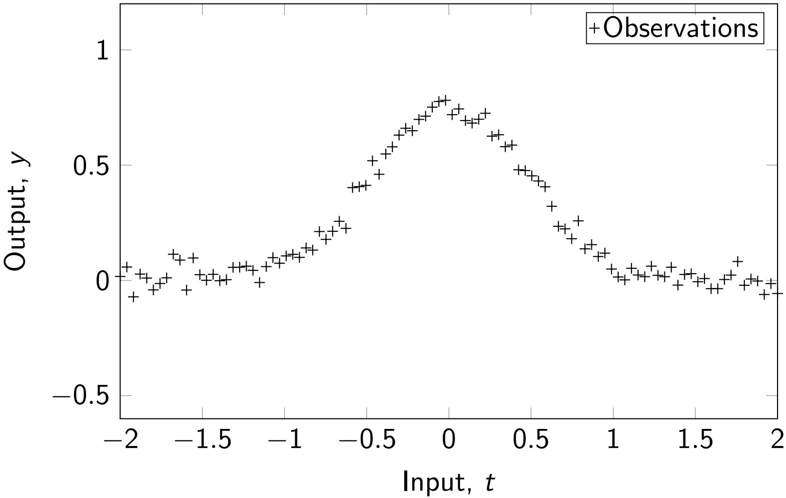
$$\mathrm{E}[f(t_*) \mid \mathcal{D}] = \boldsymbol{k}_*(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y},$$
$$\mathrm{V}[f(t_*) \mid \mathcal{D}] = \kappa(t_*, t_*) - \boldsymbol{k}_*(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_*^\top,$$

- where

$$[\boldsymbol{k}_*]_i = \kappa(t_*, t)\mathcal{C}^* \mid_{t=t_i} \qquad \text{and} \qquad \boldsymbol{K}_{ij} = \mathcal{C}\kappa(t, t')\mathcal{C}^* \mid_{t=t_i, t'=t_j}$$

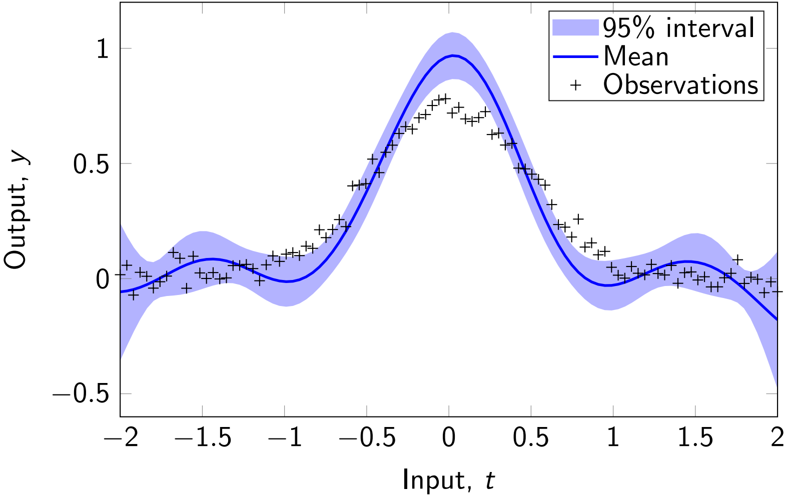
for $i, j = 1, 2, \ldots, n$ and $\mathcal{C}^*$ denotes an operator adjoint.

Convolutional likelihood models (example cont.)



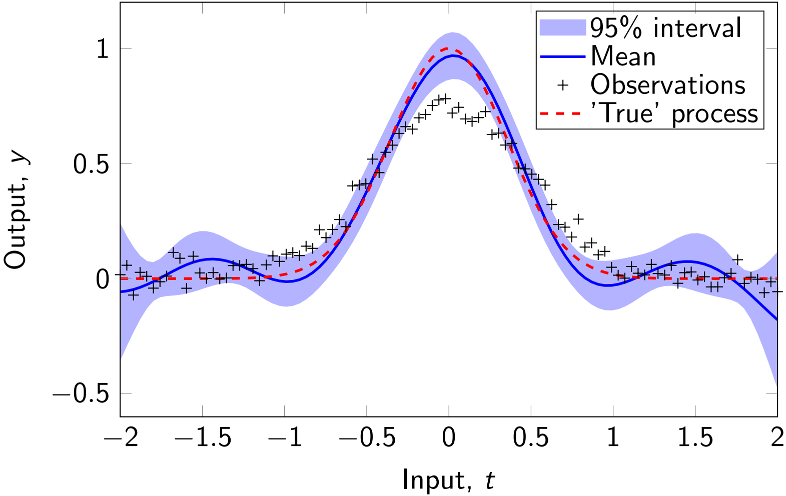
(remember that we observe the 'true' process through a convolution now)

Convolutional likelihood models (example cont.)



(remember that we observe the 'true' process through a convolution now)

Convolutional likelihood models (example cont.)



(remember that we observe the 'true' process through a convolution now)

Convolutional Gaussian process models

- **Convolutional likelihood models**
  (observations are made through convolutions)
- **Convolutional GP priors**
  (the GP prior itself has a convolutional structure)
- **Convolutions in deep GP models**
  (this is more in the domain of the next lecture)

Convolutional GP priors

- For a GP, the ability to generalise in a specific problem, are fully encoded by its covariance function (kernel)
- Most common kernel functions rely on rather rudimentary and local metrics for generalisation, like the Euclidean distance
- What kind of non-local generalisation structures can be encoded in shallow structures like kernels, while preserving the elegant properties of GPs?

Convolutional GP priors

- van der Wilk et al. (2017) presented an approach for capturing non-local structures with GP priors in a convolutional patch fashion
- The work builds upon combining principles we have already looked into on this lecture
- They leverage an additive structure and so called inter-domain GPs, where the inducing variables are constructed using a weighted integral of the GP (this fits well with other integral transformations as well)

$$u_m = \int \phi(\boldsymbol{x}, \boldsymbol{z}_m) \, f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

- Instead of relying on an integral transformation of the GP, they construct the inducing variables $\boldsymbol{u}$ alongside the new kernel such that the effective basis functions contain a convolution operation

Convolutional GP priors

- The model of van der Wilk et al. (2017) takes the following form:

$$g \sim \mathcal{GP}(0, \kappa_{\mathrm{g}}(\boldsymbol{z}, \boldsymbol{z}')), \qquad f(\boldsymbol{x}) = \sum_{p} g(\boldsymbol{x}^{[p]}),$$

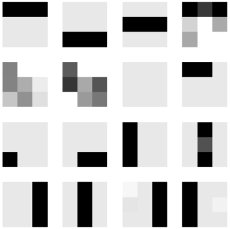
  where $g(\cdot)$ is a 'patch response function' and $f(\cdot)$ simply a sum over all patch responses
- If $g(\cdot)$ is given a GP prior, a GP prior will also be induced on $f(\cdot)$:

$$f \sim \mathcal{GP}\left(0, \sum_{p=1}^{P} \sum_{p'=1}^{P} \kappa_{\mathrm{g}}(\boldsymbol{x}^{[p]}, \boldsymbol{x}'^{[p']})\right),$$

  where $\boldsymbol{x}^{[p]}$ indicates the p$^{\text{th}}$ patch of the vector $\boldsymbol{x}$

Convolutional GP priors

- The basic structure of the model is not that different from a convnet
- The model itself is complicated and does not scale well with massive training data
- The solution is considering it as a variational sparse GP and learn the inducing points (inducing patches)
- You get a chance to play with this model in the exercises



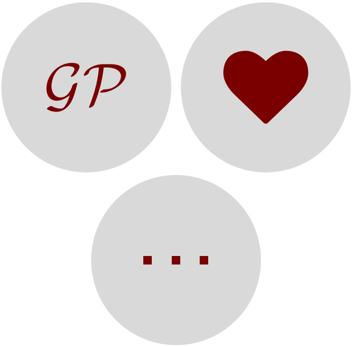
Inducing patches for the squares data

Inducing patches for the MNIST 0–1 data

Images from van der Wilk (2017)

Recap

- GPs provide a plug&play machinery for statistical inference and learning
- This lecture has tried to open your eyes in seeing how GPs can act as building blocks in slightly more versatile models beyond standard GP regression and classification
- We have covered two aspects of how convolutions can be used in association with Gaussian processes showing how GPs can appear both in the likelihood model and the GP prior itself



Bibliography

- Carl Edward Rasmussen and Christopher K. I. Williams (2006). Gaussian Processes for Machine Learning. MIT Press.
- Mark van der Wilk, Carl Edward Rasmussen, and James Hensman (2017). Convolutional Gaussian Processes. *Advances in Neural Information Processing Systems 30* (NIPS).
- Daniela Calvetti and Erkki Somersalo (2007). An Introduction to Bayesian Scientific Computing: Ten Lectures on Subjective Computing. Springer.
- Mark van der Wilk, Matthias Bauer, ST John, and James Hensman (2018). Learning Invariances using the Marginal Likelihood. *Advances in Neural Information Processing Systems 31* (NeurIPS).