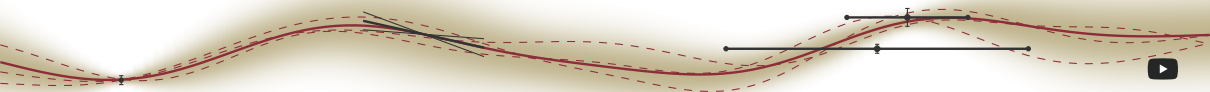
Probabilistic Inference and Learning
Lecture 09
Gaussian Processes

Philipp Hennig

18 May 2020

Eberhard Karls
Universität
Tübingen

Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning



#	date	content	Ex	#	date	content	Ex
1	20.04.	Introduction	1	14	09.06.	Logistic Regression	8
2	21.04.	Reasoning under Uncertainty		15	15.06.	Exponential Families	
3	27.04.	Continuous Variables	2	16	16.06.	Graphical Models	9
4	28.04.	Monte Carlo		17	22.06.	Factor Graphs	
5	04.05.	Markov Chain Monte Carlo	3	18	23.06.	The Sum-Product Algorithm	10
6	05.05.	Gaussian Distributions		19	29.06.	Example: Topic Models	
7	11.05.	Parametric Regression	4	20	30.06.	Mixture Models	11
8	12.05.	Learning Representations		21	06.07.	EM	
9	18.05.	**Gaussian Processes**	5	22	07.07.	Variational Inference	12
10	19.05.	An Example for GP Regression		23	13.07.	Example: Topic Models	
11	25.05.	Understanding Kernels	6	24	14.07.	Example: Inferring Topics	13
12	26.05.	Gauss-Markov Models		25	20.07.	Example: Kernel Topic Models	
13	08.06.	GP Classification	7	26	21.07.	Revision	

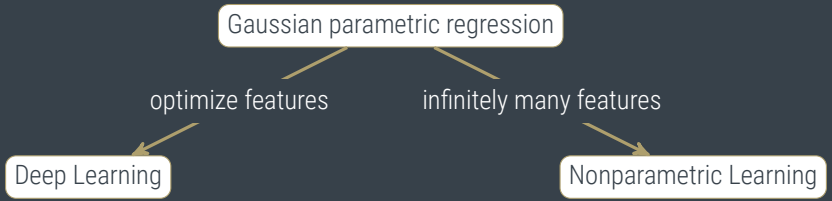
Summary:

- Gaussian distributions can be used to learn functions
- Analytical inference is possible using general linear models

$$f(x) = \phi(x)^\intercal \boldsymbol{w} = \phi_x^\intercal \boldsymbol{w}$$

- The choice of features $\phi : \mathbb{X} \to \mathbb{R}$ is essentially unconstrained
- with parametrized feature families, hierarchical inference allows **fitting** features
- Gaussian linear regressors are **single-layer neural networks** with quadratic loss

aka. least-square regression (cf. next week)



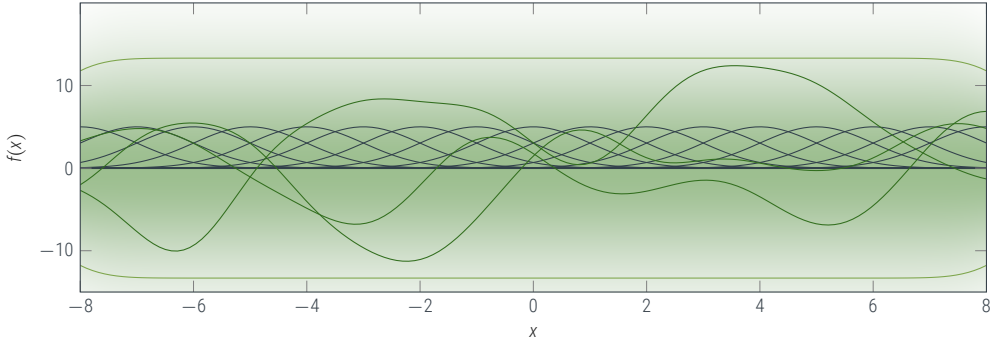
Reminder: Where we left off last time

Probabilistic Parametric Regression

$$f(x) := \phi_x^\intercal \boldsymbol{w} \qquad p(w) = \mathcal{N}(w; \mu, \Sigma) \qquad p(\boldsymbol{y} \mid f) = \mathcal{N}(\boldsymbol{y}; f_X, \sigma^2 I)$$

$$p(w \mid \boldsymbol{y}, \phi_X) = \mathcal{N}\left(w; \mu + \Sigma\phi_X(\phi_X^\intercal\Sigma\phi_X + \sigma^2 I)^{-1}(\boldsymbol{y} - \phi_X^\intercal\mu), \Sigma - \Sigma\phi_X(\phi_X^\intercal\Sigma\phi_X + \sigma^2 I)^{-1}\phi_X^\intercal\Sigma\right)$$

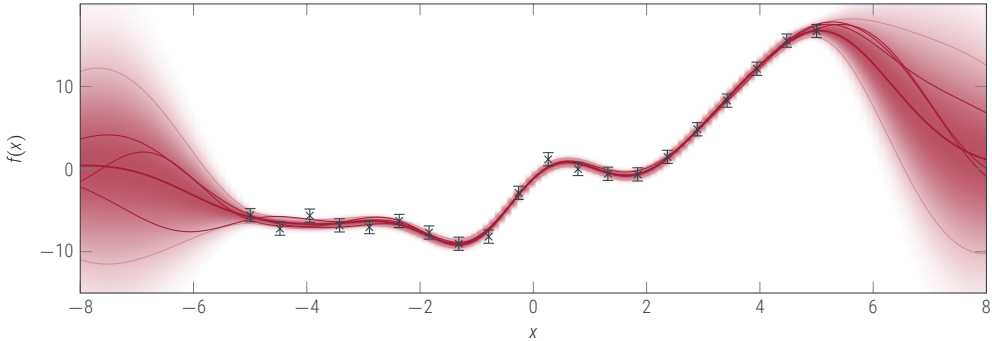
$$p(f_x \mid \boldsymbol{y}, \phi_X) = \mathcal{N}\left(f_x; \phi_x^\intercal\mu + \phi_x^\intercal\Sigma\phi_X(\phi_X^\intercal\Sigma\phi_X + \sigma^2 I)^{-1}(y - \phi_X^\intercal\mu), \phi_x^\intercal\left(\Sigma - ^\intercal\ \Sigma\phi_X(\phi_X^\intercal\Sigma\phi_X + \sigma^2 I)^{-1}\phi_X^\intercal\Sigma\right)\phi_x\right)$$



Reminder: Where we left off last time

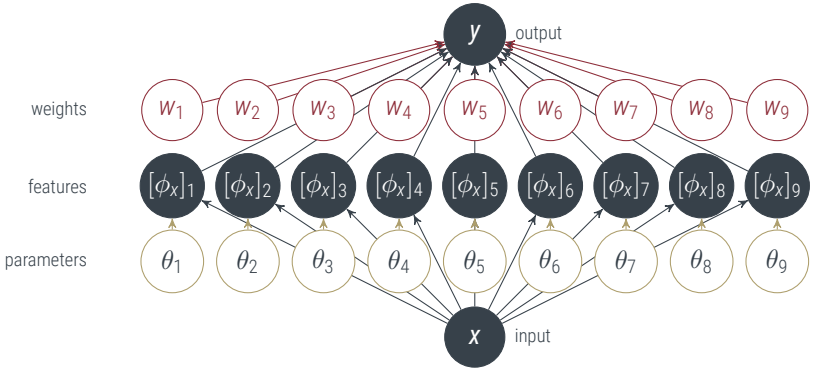
Probabilistic Parametric Regression

$$\phi(x) = \begin{bmatrix} e^{-\frac{1}{2}(x-8)^2} & e^{-\frac{1}{2}(x-7)^2} & e^{-\frac{1}{2}(x-6)^2} & \dots \end{bmatrix}^\intercal$$
$$= \left[e^{-(x-c_i)^2/(2\lambda^2)} \right]_{i=1,\dots,F}$$



A Structural Observation

Graphical Model



A linear Gaussian regressor is a **single hidden layer** neural network, with quadratic output loss, and fixed input layer. Hyperparameter-fitting corresponds to training the input layer. The usual way to train such network, however, does not include the Occam factor.

What are we actually doing with those features?

let's look at that algebra again

$$
\begin{aligned}
p(f_x \mid y, \phi_X) &= \mathcal{N}(f_x; \phi_x^\intercal \mu + \phi_x^\intercal \Sigma \phi_X (\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}(y - \phi_X^\intercal \mu), \\
&\qquad \phi_x^\intercal \Sigma \phi_x - \phi_x^\intercal \Sigma \phi_X (\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1} \phi_X^\intercal \Sigma \phi_x) \\
&= \mathcal{N}(f_x; m_x + k_{xX}(k_{XX} + \sigma^2 I)^{-1}(y - m_X), \\
&\qquad k_x x - k_{xX}(k_{XX} + \sigma^2 I)^{-1} k_{Xx})
\end{aligned}
$$

using the abstraction / encapsulation

$m_x := \phi_x^\intercal \mu$	$m : \mathbb{X} \to \mathbb{R}$	mean function
$k_{ab} := \phi_a^\intercal \Sigma \phi_b$	$k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$	covariance function, aka. **kernel**

Code

Gaussian_Process_Regression.ipynb

Sometimes, **more** features make things **cheaper**

an example

[DJC MacKay, 1998]

- For simplicity, let's fix $\Sigma = \frac{\sigma^2(c_{\max}-c_{\min})}{F} I$

$$\text{thus:} \quad \phi(x_i)^\intercal \Sigma \phi(x_j) = \frac{\sigma^2(c_{\max} - c_{\min})}{F} \sum_{\ell=1}^{F} \phi_\ell(x_i)\phi_\ell(x_j)$$

- especially, for $\phi_\ell(x) = \exp\left(-\frac{(x - c_\ell)^2}{2\lambda^2}\right)$

$$\begin{aligned}\phi(x_i)^\intercal \Sigma \phi(x_j) &= \frac{\sigma^2(c_{\max} - c_{\min})}{F} \sum_{\ell=1}^{F} \exp\left(-\frac{(x_i - c_\ell)^2}{2\lambda^2}\right) \exp\left(-\frac{(x_j - c_\ell)^2}{2\lambda^2}\right) \\ &= \frac{\sigma^2(c_{\max} - c_{\min})}{F} \exp\left(-\frac{(x_i - x_j)^2}{4\lambda^2}\right) \sum_{\ell}^{F} \exp\left(-\frac{(c_\ell - \frac{1}{2}(x_i + x_j))^2}{\lambda^2}\right)\end{aligned}$$

Sometimes, **more** features make things **cheaper**

an example

[DJC MacKay, 1998]

$$\phi(x_i)^\intercal \Sigma \phi(x_j) = \frac{\sigma^2(c_{\max} - c_{\min})}{F} \exp\left(-\frac{(x_i - x_j)^2}{4\lambda^2}\right) \sum_{\ell}^{F} \exp\left(-\frac{(c_\ell - \frac{1}{2}(x_i + x_j))^2}{\lambda^2}\right)$$

now increase F so # of features in δc approaches $\frac{F \cdot \delta c}{(c_{\max} - c_{\min})}$

$$\phi(x_i)^\intercal \Sigma \phi(x_j) \to \sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{4\lambda^2}\right) \int_{c_{\min}}^{c_{\max}} \exp\left(-\frac{(c - \frac{1}{2}(x_i + x_j))^2}{\lambda^2}\right) dc$$

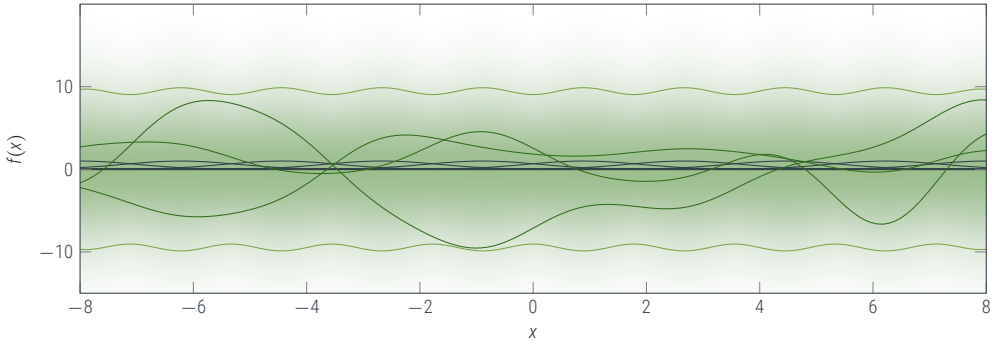
let $c_{\min} \to -\infty$, $c_{\max} \to \infty$

$$k(x_i, x_j) := \phi(x_i)^\intercal \Sigma \phi(x_j) \to \sqrt{\pi}\lambda\sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{4\lambda^2}\right)$$

$$k_{x_i x_j} = \sqrt{\pi}\lambda\sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{4\lambda^2}\right)$$

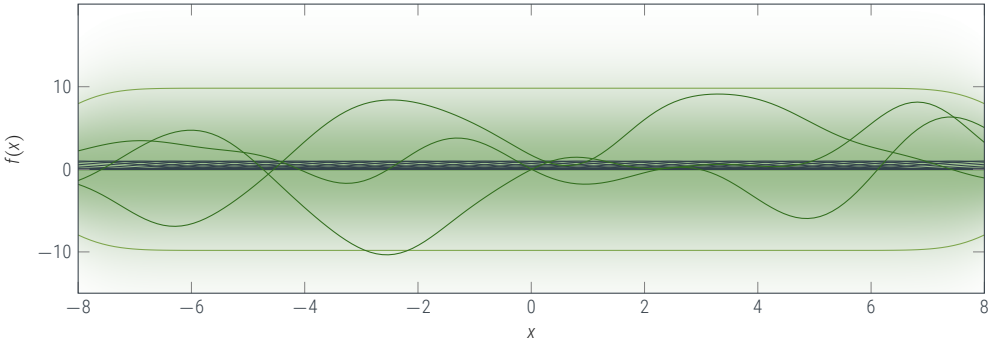
In pictures

convergence to Gaussian kernel



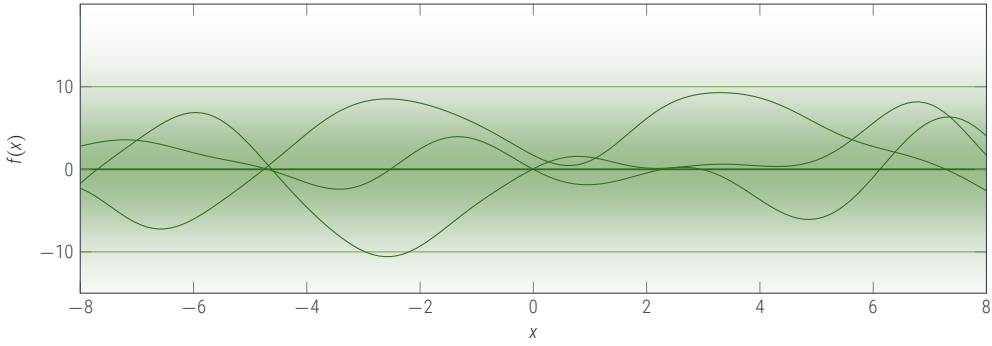
In pictures

convergence to Gaussian kernel



In pictures

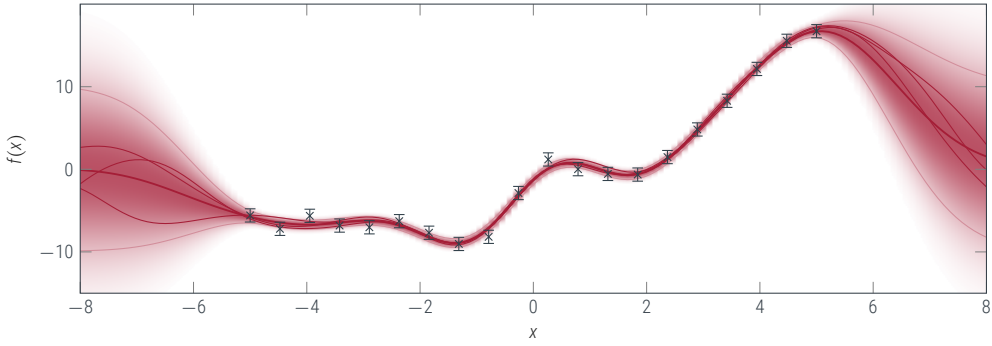
convergence to Gaussian kernel



- aka. radial basis function, square(d)-exponential kernel

In pictures

convergence to Gaussian kernel



- aka. radial basis function, square(d)-exponential kernel

Code

Gaussian_Process_Regression.ipynb

Definition (kernel)

$k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ is a (Mercer / positive definite) **kernel** if, for any finite collection $X = [x_1, \ldots, x_N]$, the matrix $k_{XX} \in \mathbb{R}^{N \times N}$ with $[k_{XX}]_{ij} = k(x_i, x_j)$ is **positive semidefinite**.

```
def kernel (f) : λ (a,b) -> [[f(a[i],b[j]) for j=1:length(b)] for i=1:length(a) ]
```

actually, in python:

```
def kernel (f) : return lambda a,b : np.array([ [np.float64(f(a[i],b[j])) for j in range(b.size) ] for i in range(a.size) ])
```

Definition (positive definite matrix)

A symmetric matrix $A \in \mathbb{R}^{N \times N}$ is called **positive (semi-) definite** if

$$v^\intercal A v \geq 0 \; \forall \, v \in \mathbb{R}^N.$$

Equivalently:

- All eigenvalues of the symmetric matrix A are non-negative
- A is a Gram matrix — the outer product of N vectors $[\phi_i]_{i=1,\ldots N}$

Lemma:

k is a Mercer kernel if it can be written as (assuming $\mathcal{L}$ is positive set for ν)

$$k(x, x') = \sum_{\ell \in \mathcal{L}} \phi_\ell(x)\phi_\ell(x') \quad \text{or} \quad = \int_{\mathcal{L}} \phi_\ell(x)\phi_\ell(x')\, d\nu(\ell)$$

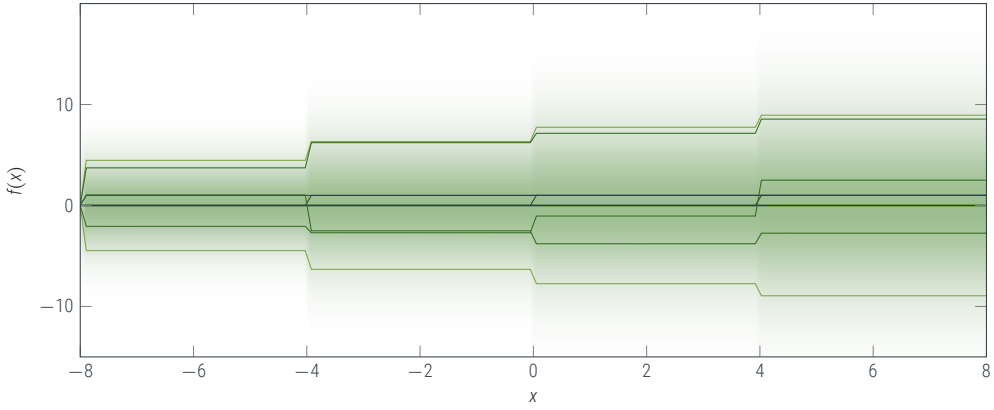
Proof: $\forall X \in \mathbb{X}^N, v \in \mathbb{R}^N : v^\intercal k_{XX} v = \sum_{\ell} \sum_{i}^{N} v_i \phi_\ell(x_i) \sum_{j}^{N} v_j \phi_\ell(x_j)\, d\nu(\ell) = \sum_{\ell} \left[\sum_i v_i \phi_\ell(x_i)\right]^2 d\nu(\ell) \geq 0$ □

Definition

Let $\mu : \mathbb{X} \to \mathbb{R}$ be any function, $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ be a Mercer kernel. A **Gaussian process** $p(f) = \mathcal{GP}(f; \mu, k)$ is a probability distribution over the function $f : \mathbb{X} \to \mathbb{R}$, such that every finite restriction to function values $f_X := [f_{x_1}, \ldots, f_{x_N}]$ is a Gaussian distribution $p(f_X) = \mathcal{N}(f_X; \mu_X, k_{XX})$.

Another kernel

The Wiener process



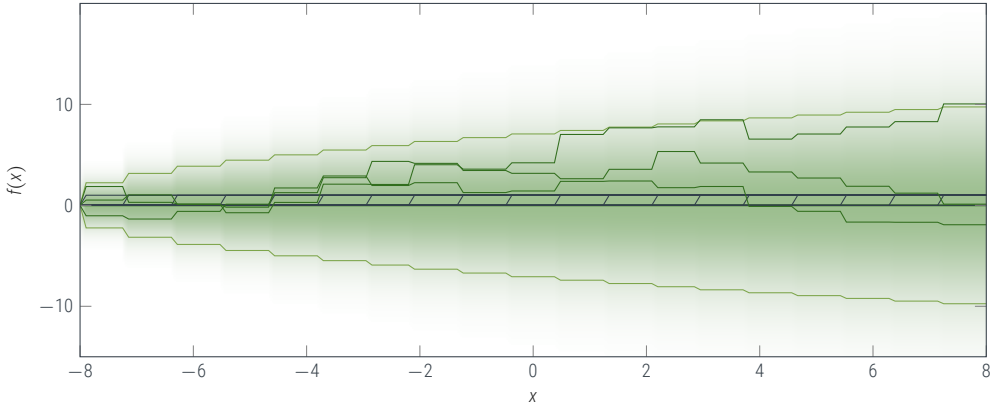
$$\phi_i(x) = \theta(x - c_i) = \begin{cases} 1 & \text{if } x \geq c_i \\ 0 & \text{else} \end{cases}$$

$$\phi_i(x) = \theta(x - c_i) = \begin{cases} 1 & \text{if } x \geq c_i \\ 0 & \text{else} \end{cases} \qquad \Sigma = \sigma^2 \frac{c_{\max} - c_0}{F}$$

$$\Rightarrow \quad k(x_i, x_j) = \phi(x_i)^\intercal \Sigma \phi(x_j) = \frac{\sigma^2 (c_{\max} - c_0)}{F} \sum_{\ell=1}^{F} \theta(\min(x_i, x_j) - c_\ell)$$

now increase F so # of features in δc approaches $\frac{F \cdot \delta c}{(c_{\max} - c_0)}$

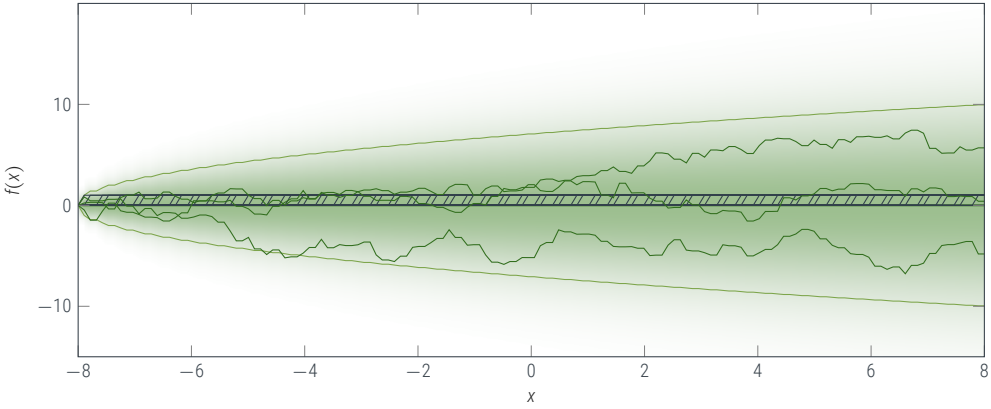
$$\begin{aligned} k(x_i, x_j) &\twoheadrightarrow \sigma^2 \int_{c_0}^{c_{\max}} \theta(\min(x_i, x_j) - c)\, dc = \sigma^2 \, [1]_{c_0}^{\min(x_i, x_j)} \\ &= \sigma^2 (\min(x_i, x_j) - c_0) \end{aligned}$$



$$k(x_i, x_k) = \sigma^2(\min(x_i, x_j) - c_0) \qquad \text{var}\, f(x) = \sigma^2(x - c_0)$$

Graphical View

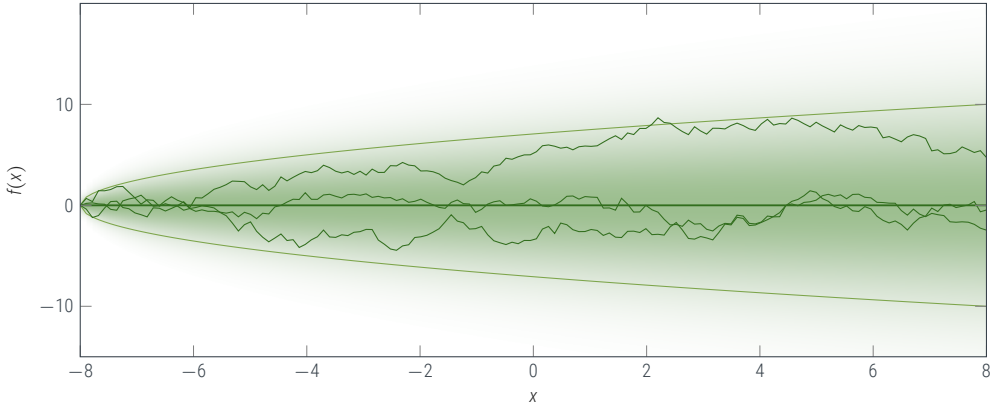
Convergence to Wiener process



$$k(x_i, x_k) = \sigma^2(\min(x_i, x_j) - c_0) \qquad \text{var}\, f(x) = \sigma^2(x - c_0)$$

Graphical View

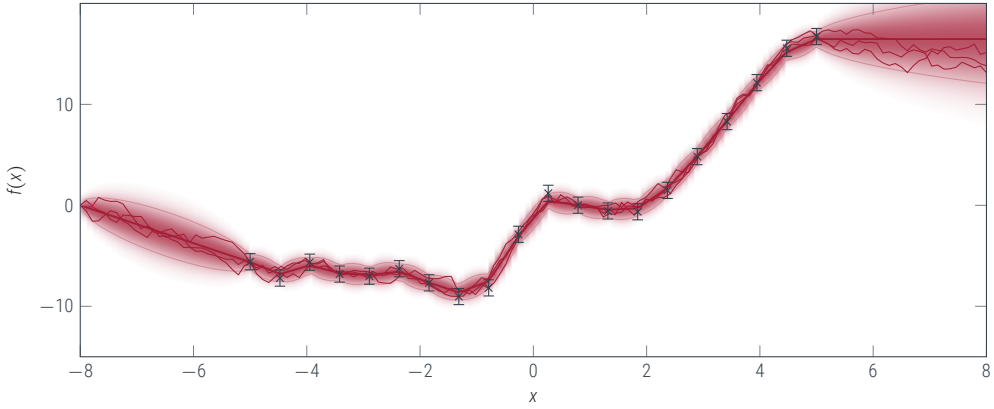
Convergence to Wiener process



$$k(x_i, x_k) = \sigma^2(\min(x_i, x_j) - c_0) \qquad \text{var}\, f(x) = \sigma^2(x - c_0)$$

Graphical View

Convergence to Wiener process

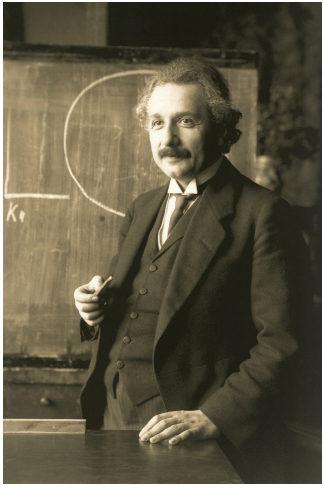


$$k(x_i, x_k) = \sigma^2(\min(x_i, x_j) - c_0) \qquad \text{var}\, f(x) = \sigma^2(x - c_0)$$

sein. Das Problem, welches mit dem Problem der Diffusion von einem Punkte aus (unter Vernachlässigung der Wechselwirkung der diffundierenden Teilchen) übereinstimmt, ist nun mathematisch vollkommen bestimmt; seine Lösung ist:

$$f(x, t) = \frac{n}{\sqrt{4\pi D}} \frac{e^{-\frac{x^2}{4Dt}}}{\sqrt{t}}.$$

[A. Einstein, *Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen*, Ann. d. Physik, 1905]



[image: F. Schmutzer, 1921, Austrian National Library]

Just one more!

Splines

[G. Wahba, *Spline Models for Observational Data*, 1990]

$$\text{remember Wiener: } \phi_i(x) = \theta(x - c_i) \qquad f(x) = \phi(x)^\intercal \boldsymbol{w}$$

$$\text{consider: } \tilde{\phi}_i(x) = \int_{-\infty}^{x} \theta(\tilde{x} - c_i)\, d\tilde{x} \quad \text{“ReLU”}$$

$$\text{equivalent to: } \tilde{f}(x) = \int_{-\infty}^{x} f(\tilde{x})\, d\tilde{x}$$

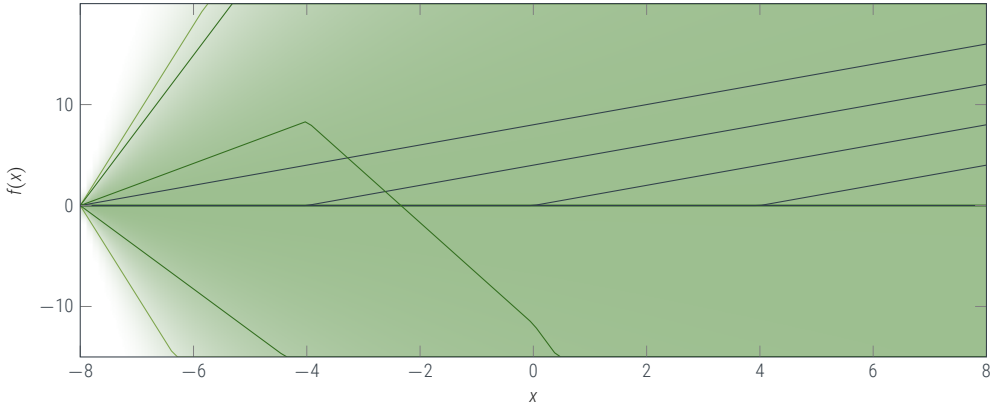
$$\implies \quad \tilde{k}(x_i, x_j) = \tilde{\phi}(x_i)^\intercal \Sigma \tilde{\phi}(x_j) = \iint_{\infty}^{x_i, x_j} k(\tilde{x}_i, \tilde{x}_j)\, d\tilde{x}_i\, d\tilde{x}_j$$

$$= \sigma^2 \left(\frac{1}{3} \min^3(x_i - x_0, x_j - x_0)^3 + \frac{1}{2} |x_i - x_j| \min^2(x_i - x_0, x_j - x_0) \right)$$

Nb: The integral of a Gaussian process is also a Gaussian process!

Cubic Splines

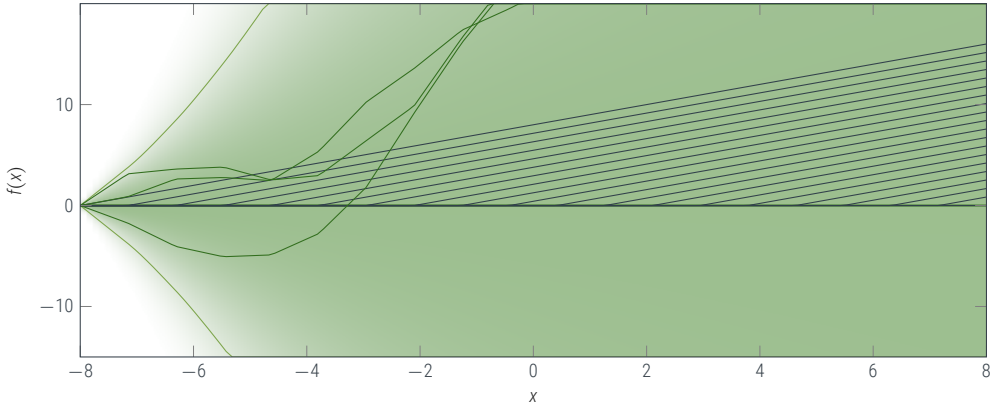
from rectified linear units



$$k(x_i, x_k) = \sigma^2 \left(\frac{1}{3} \min^3(x_i - x_0, x_j - x_0)^3 + \frac{1}{2} |x_i - x_j| \min^2(x_i - x_0, x_j - x_0) \right)$$

Cubic Splines

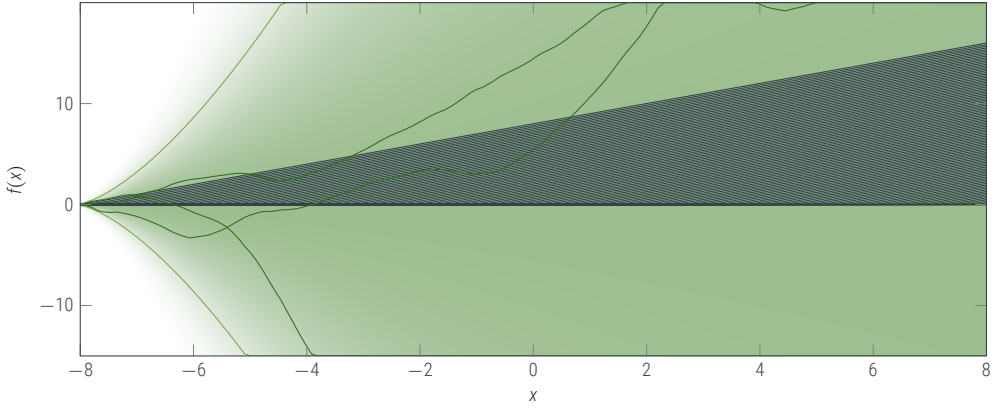
from rectified linear units



$$k(x_i, x_k) = \sigma^2 \left(\frac{1}{3} \min^3(x_i - x_0, x_j - x_0)^3 + \frac{1}{2} |x_i - x_j| \min^2(x_i - x_0, x_j - x_0) \right)$$

Cubic Splines

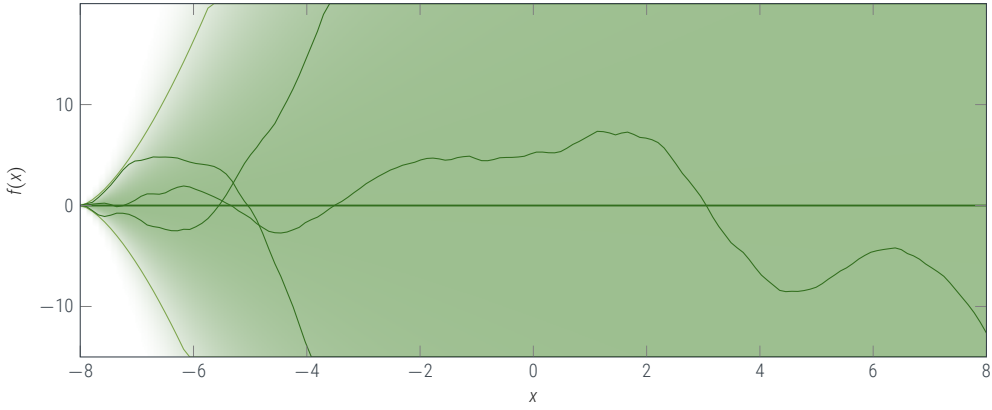
from rectified linear units



$$k(x_i, x_k) = \sigma^2 \left(\frac{1}{3} \min^3(x_i - x_0, x_j - x_0)^3 + \frac{1}{2} |x_i - x_j| \min^2(x_i - x_0, x_j - x_0) \right)$$

Cubic Splines

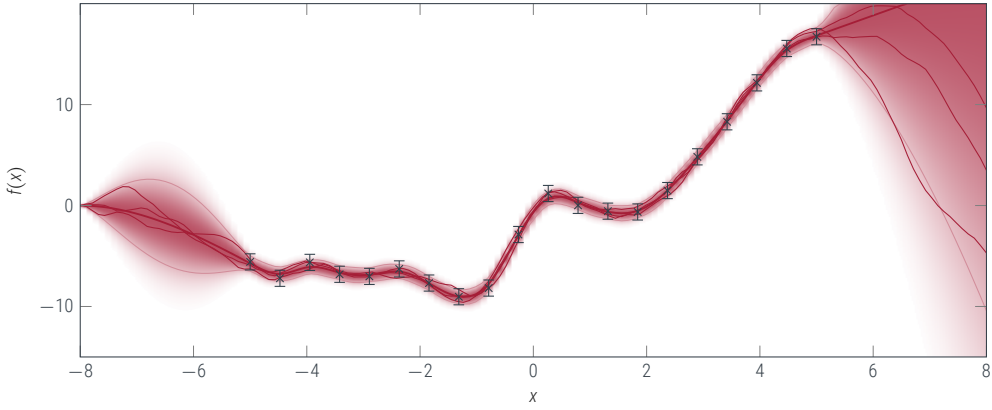
from rectified linear units



$$k(x_i, x_k) = \sigma^2\left(\frac{1}{3}\min^3(x_i - x_0, x_j - x_0)^3 + \frac{1}{2}|x_i - x_j|\min^2(x_i - x_0, x_j - x_0)\right)$$

Cubic Splines

from rectified linear units



$$k(x_i, x_k) = \sigma^2\Big(\frac{1}{3}\min^3(x_i - x_0, x_j - x_0)^3 + \frac{1}{2}|x_i - x_j|\min^2(x_i - x_0, x_j - x_0)\Big)$$

- Sometimes it is possible to consider **infinitely many** features at once, by extending from a sum to an integral. This requires some regularity assumption about the features' locations, shape, etc.
- The resulting **nonparametric model** is known as a **Gaussian process**
- Inference in GPs is tractable (though at polynomial cost $\mathcal{O}(N^3)$ in the number N of datapoints)
- There is no unique kernel. In fact, there are quite a few! E.g.

$$k(a,b) = \exp(-(a-b)^2) \qquad \text{Gaussian / Square Exponential / RBF kernel}$$

$$k(a,b) = \min(a - t_0, b - t_0) \qquad \text{Wiener process}$$

$$k(a,b) = \frac{1}{3}\min^3(a - t_0, b - t_0) + \frac{1}{2}|a-b| \cdot \min^2(a - t_0, b - t_0) \qquad \text{cubic spline kernel}$$

$$k(a,b) = \frac{2}{\pi}\sin^{-1}\left(\frac{2a^\intercal b}{\sqrt{(1+2a^\intercal a)(1+2b^\intercal b)}}\right) \qquad \text{Neural Network kernel (Williams, 1998)}$$

How many kernels are there?

A Lot!

because the space of kernels is large

Theorem:

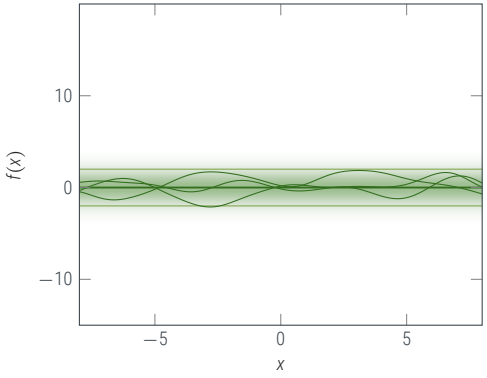
Let $\mathbb{X}, \mathbb{Y}$ be index sets and $\phi : \mathbb{Y} \to \mathbb{X}$. If $k_1, k_2 : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ are Mercer kernels, then the following functions are also Mercer kernels (up to minor regularity assumptions)

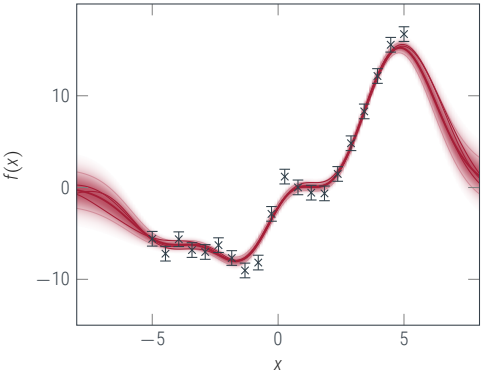
- $\alpha \cdot k_1(a, b)$ for $\alpha \in \mathbb{R}_+$ (proof: trivial)
- $k_1(\phi(c), \phi(d))$ for $c, d \in \mathbb{Y}$ (proof: by **Mercer's theorem**, next lecture)
- $k_1(a, b) + k_2(a, b)$ (proof: trivial)
- $k_1(a, b) \cdot k_2(a, b)$ **Schur product theorem** (proof involved. E.g. Bapat, 1997. Million, 2007)

Scaled Kernels are Kernels

(this affects the effect of noise)

$$k(a,b) = 1^2 \cdot \exp\left(-\frac{(a-b)^2}{2}\right)$$

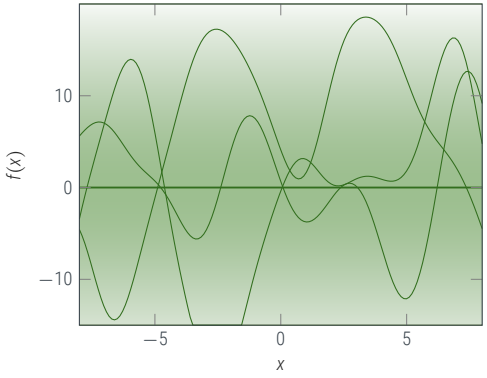


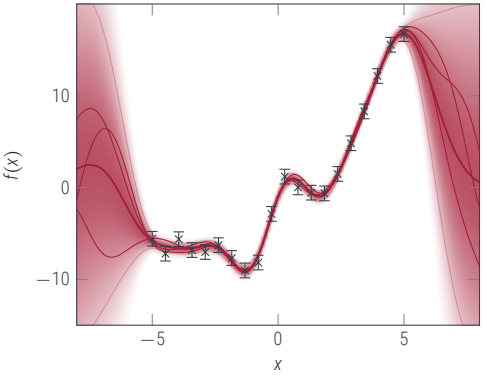


Scaled Kernels are Kernels

(this affects the effect of noise)

$$k(a,b) = 10^2 \cdot \exp\left(-\frac{(a-b)^2}{2}\right)$$

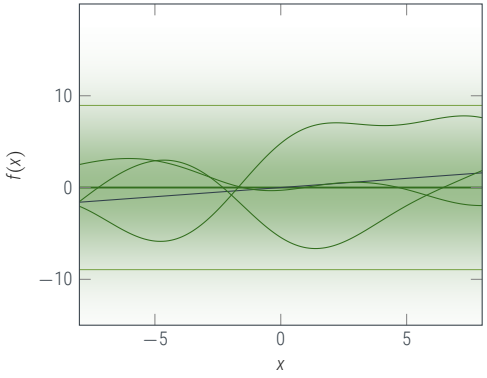


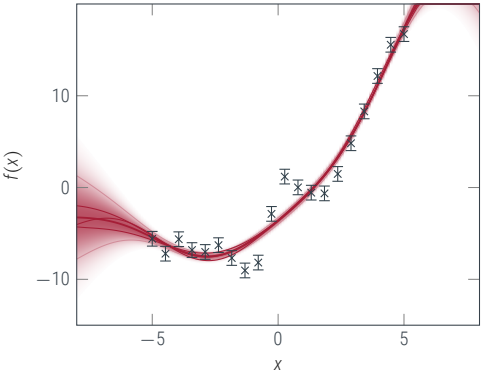


Kernels over Transformed Inputs are Kernels

this can be used to define length-scales

$$k(a,b) = 20 \cdot \exp\left(-\frac{(a-b)^2}{2 \cdot 5^2}\right)$$

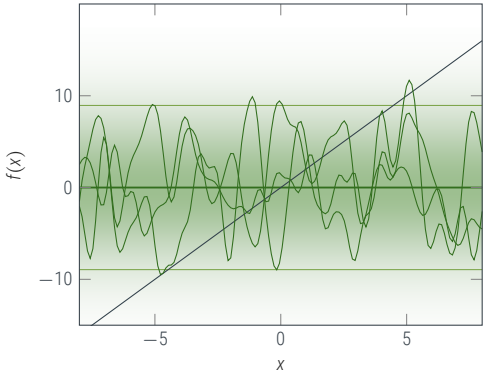


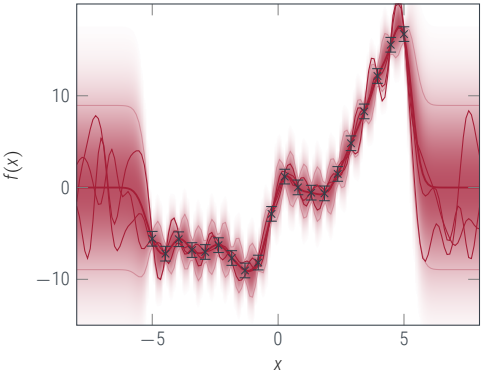


Kernels over Transformed Inputs are Kernels

this can be used to define length-scales

$$k(a,b) = 20 \cdot \exp\left(-\frac{(a-b)^2}{2 \cdot 0.5^2}\right)$$

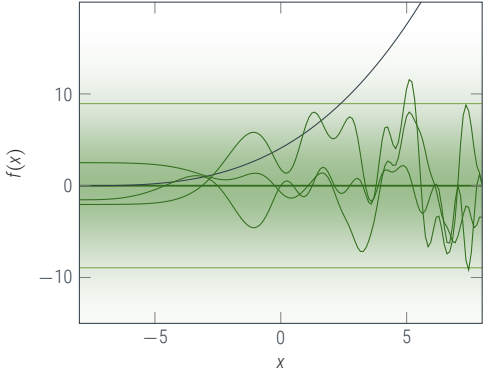


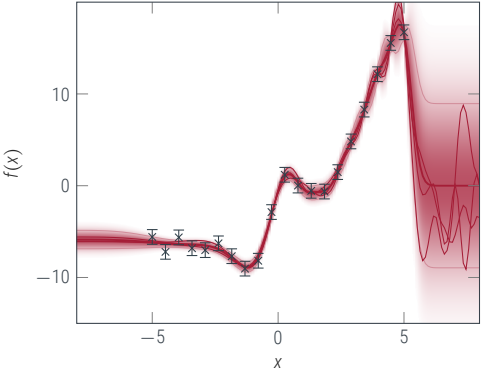


Kernels over Transformed Inputs are Kernels

nonlinear transformations (also at the heart of deep learning)

$$k(a,b) = 20 \cdot \exp\left(-\frac{(\phi(a)-\phi(b))^2}{2}\right) \qquad \phi(a) = \left(\frac{a+8}{5}\right)^3$$

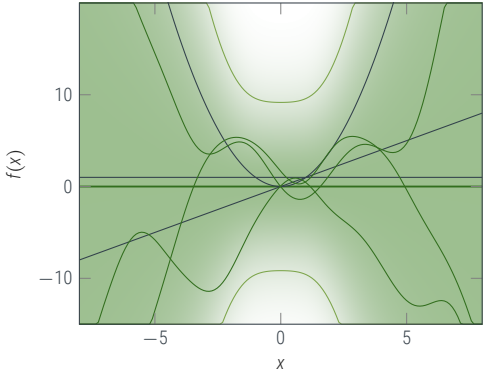


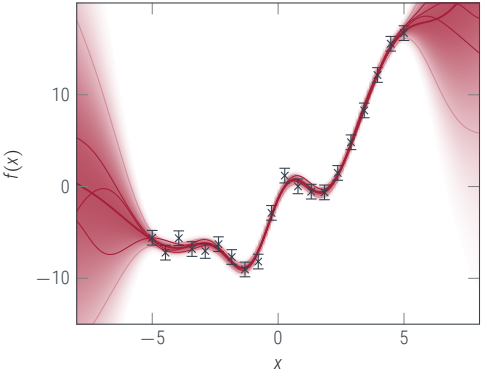


Sums of Kernels are Kernels

$f \sim \mathcal{GP}(0, k_1)$ and $g \sim \mathcal{GP}(0, k_2) \quad \Leftrightarrow \quad h = f + g \sim \mathcal{GP}(0, k_1 + k_2)$

$$k(a,b) = 20 \cdot \exp\left(-\frac{(a-b)^2}{2}\right) + \phi(a)^\intercal \phi(b) \quad \phi(x) := [1, x, x^2]$$

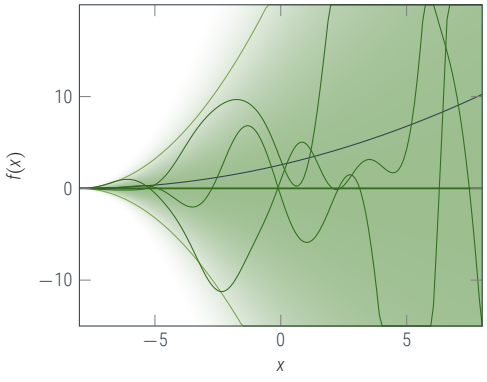


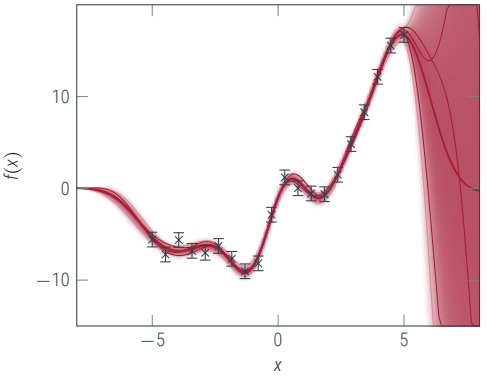


Products of Kernels are Kernels

Schur product theorem

$$k(a,b) = 20 \cdot \exp\left(-\frac{(a-b)^2}{2}\right) \cdot \phi(a)^\intercal \phi(b) \quad \phi(x) := \left(\frac{x+8}{5}\right)^2$$





If you think your model has no parameters, you just haven't found them yet!

$$p(f \mid \theta) = \mathcal{GP}(f; m_\theta, k_\theta) \qquad \text{e.g.} \quad m_\theta(x) = \phi(x)^\intercal \boldsymbol{\theta}, \text{ and/or } k_\theta(a,b) = \theta_1 \exp(-\frac{(a-b)^2}{2\theta_2^2}).$$

- recall from last lecture: the **evidence** in Bayes' theorem is the **marginal likelihood for the model**

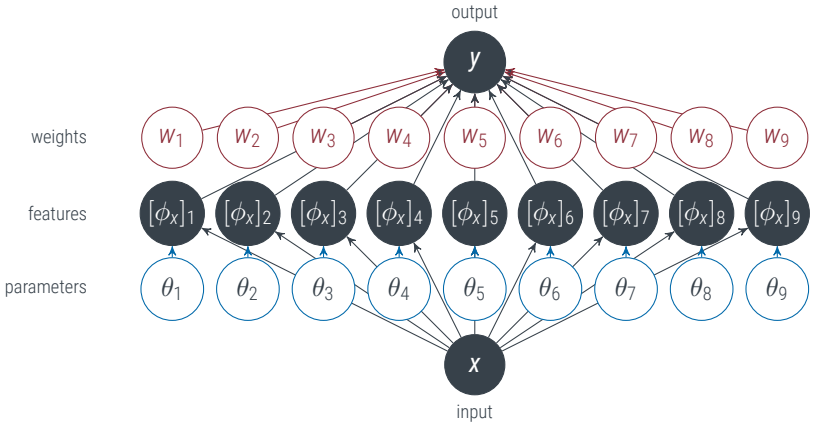
$$p(f \mid \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})\, df} = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})}$$

- For Gaussians, die evidence has **analytic form**:

$$\underbrace{\mathcal{N}(\boldsymbol{y}; \phi_X^{\boldsymbol{\theta}\intercal} w + \boldsymbol{b}, \Lambda)}_{p(\boldsymbol{y}|f,\boldsymbol{x},\boldsymbol{\theta})} \cdot \underbrace{\mathcal{N}(w, \mu, \Sigma)}_{p(f)} = \underbrace{\mathcal{N}(w; m^{\boldsymbol{\theta}}_{\text{post}}, V^{\boldsymbol{\theta}}_{\text{post}})}_{p(f|\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta})} \cdot \underbrace{\mathcal{N}(\boldsymbol{y}; \phi_X^{\boldsymbol{\theta}\intercal} \mu + b, \phi_X^{\boldsymbol{\theta}\intercal} \Sigma \phi_X^{\boldsymbol{\theta}} + \Lambda)}_{p(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{x})}$$

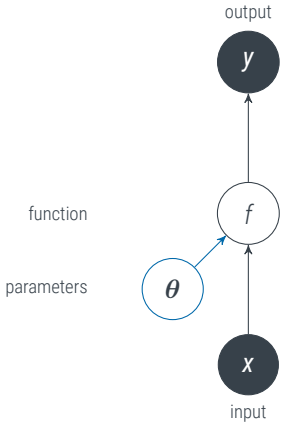
Kernel Learning is Fitting an Entire Population of Features

The Graphical Model View



Kernel Learning is Fitting an Entire Population of Features

The Graphical Model View



Summary: Gaussian Processes

- Sometimes it is possible to consider **infinitely many** features at once, by extending from a sum to an integral.
- The resulting **nonparametric model** is known as a **Gaussian process**
- Inference in GPs is tractable (though at polynomial cost $\mathcal{O}(N^3)$ in the number N of datapoints)
- There is an entire semi-ring of kernels: If k_1, k_2 are kernels, ϕ any function, then so are
  - $\alpha \cdot k_1(a, b)$ for $\alpha \in \mathbb{R}_+$
  - $k_1(\phi(c), \phi(d))$ for $c, d \in \mathbb{Y}$
  - $k_1(a, b) + k_2(a, b)$
  - $k_1(a, b) \cdot k_2(a, b)$
- Kernels can be **learned** just like features, e.g. by type-II MAP/ML or numerical integration

Gaussian processes are an extremely important basic model for supervised regression. They have been used in many pratical applications, but also provide a theoretical foundation for more complicated models, e.g. in classification, unsupervised learning, deep learning. More later.

Next lecture: Some **theory** for Gaussian processes. If you don't like pretty pictures, wait for it!

The Toolbox

Framework:

$$\int p(x_1, x_2)\, dx_2 = p(x_1) \qquad p(x_1, x_2) = p(x_1 \mid x_2) p(x_2) \qquad p(x \mid y) = \frac{p(y \mid x) p(x)}{p(y)}$$

Modelling:

- Directed Graphical Models
- Gaussian Distributions
- Kernels
-
-
-

Computation:

- Monte Carlo
- Linear algebra / Gaussian inference
- maximum likelihood / MAP
-
-