TU München
Fakultät für Informatik
PD Dr. Rudolph Triebel
John Chiotellis, Maximilian Denninger

Machine Learning for Computer Vision
Winter term 2018

December 6, 2018
Topic: Kernels and Gaussian Processes

Exercise 1: Constructing kernels

During this solution we assume the feature spaces of k_1 and k_2 to have finite dimensions. Thus they can be written as $k_1(x_1, x_2) = \phi_1(x_1)^T \phi_1(x_2)$, $k_2(x_1, x_2) = \phi_2(x_1)^T \phi_2(x_2)$, where $\phi_1(x) \in \mathbb{R}^{n_1}$, $\phi_2(x) \in \mathbb{R}^{n_2}$. Note however that in general feature spaces can be infinite dimensional (e.g. $\phi(x) \in l^2(\mathbb{R})$, see 4.). We now have to define new kernels via a scalarproduct $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$

a) $k(x_1, x_2) = k_1(x_1, x_2) + k_2(x_1, x_2)$

To warm up:
$\phi(x) = \begin{pmatrix} \phi_1(x) \\ \phi_2(x) \end{pmatrix} \in \mathbb{R}^{n_1+n_2}$

b) $k(x_1, x_2) = k_1(x_1, x_2) k_2(x_1, x_2)$

Note that the matrix-products do not commute, so it is a bit of work:

$$
\begin{aligned}
k(x_1, x_2) &= \phi_1(x_1)^T \phi_1(x_2) \phi_2(x_1)^T \phi_2(x_2) \\
&= (\sum_i (\phi_1(x_1))_i (\phi_1(x_2))_i)(\sum_j (\phi_2(x_1))_j (\phi_2(x_2))_j) \\
&= \sum_i \sum_j (\phi_1(x_1))_i (\phi_1(x_2))_i (\phi_2(x_1))_j (\phi_2(x_2))_j \\
&= \underbrace{\sum_i \sum_j}_{\sum_k} \underbrace{(\phi_1(x_1))_i (\phi_2(x_1))_j}_{\phi_k(x_1)} \underbrace{(\phi_1(x_2))_i (\phi_2(x_2))_j}_{\phi_k(x_2)}
\end{aligned}
$$

$$
\Rightarrow \phi(x) = \begin{pmatrix} (\phi_1(x))_1 (\phi_2(x))_1 \\ \vdots \\ (\phi_1(x))_1 (\phi_2(x))_{n_2} \\ (\phi_1(x))_2 (\phi_2(x))_1 \\ \vdots \\ (\phi_1(x))_{n_1} (\phi_2(x))_{n_2} \end{pmatrix} \in \mathbb{R}^{n_1 \cdot n_2}
$$

c) $k(x_1, x_2) = f(x_1) k_1(x_1, x_2) f(x_2)$
$\phi(x) = f(x) \phi_1(x)$

d) $k(x,y) = \exp(k_1(x,y))$

Again we write the scalarproduct as a sum:

$$\begin{aligned}\exp((\phi_1(x))^T\phi(y)) &= \exp(\sum(\phi_1(x))_i(\phi_1(y))_i) \\ &= \prod \exp((\phi_1(x))_i(\phi_1(y))_i)\end{aligned}$$

Since we already know that the product of kernels is again a kernel it remains to show, that $\exp((\phi(x))_i(\phi(y))_i)$ is a kernel for a fixed index i. In the following we will omit i and imagine ϕ_1 to be a scalar-valued function. From the Taylor-expansion of the exponential function, we know that

$$\exp(\phi_1(x))(\phi_1(y)) = \sum_{k=0}^{\infty} \frac{1}{k!}(\phi_1(x))^k(\phi_1(y))^k$$

This is an inner product in $l^2(\mathbb{R})$ with

$$\phi(x) = \begin{pmatrix} \phi_1(x) \\ \frac{1}{\sqrt{2}}\phi_1(x)^2 \\ \frac{1}{\sqrt{6}}\phi_1(x)^3 \\ \vdots \\ \frac{1}{\sqrt{k!}}\phi_1(x)^k \\ \vdots \end{pmatrix}$$

e) $k(x_1,x_2) = x_1^T A x_2$

Since A is symmetric positive-definite, it admits a Cholesky decomposition $A = LL^T$. Therefore, we have $x_1^T A x_2 = x_1^T L L^T x_2 = (L^T x_1)^T (L^T x_2)$. So $\phi(x) = L^T x$.

Exercise 2: Gaussian Regression

a) Implement a simple gaussian regressor. As trainings data you can use the provided code snippet to generate ten points along a sinus curve. Use a fixed *length* param of 3.0, with a $sigma_f$ of 1.0 and $sigma_n$ of 0.5.

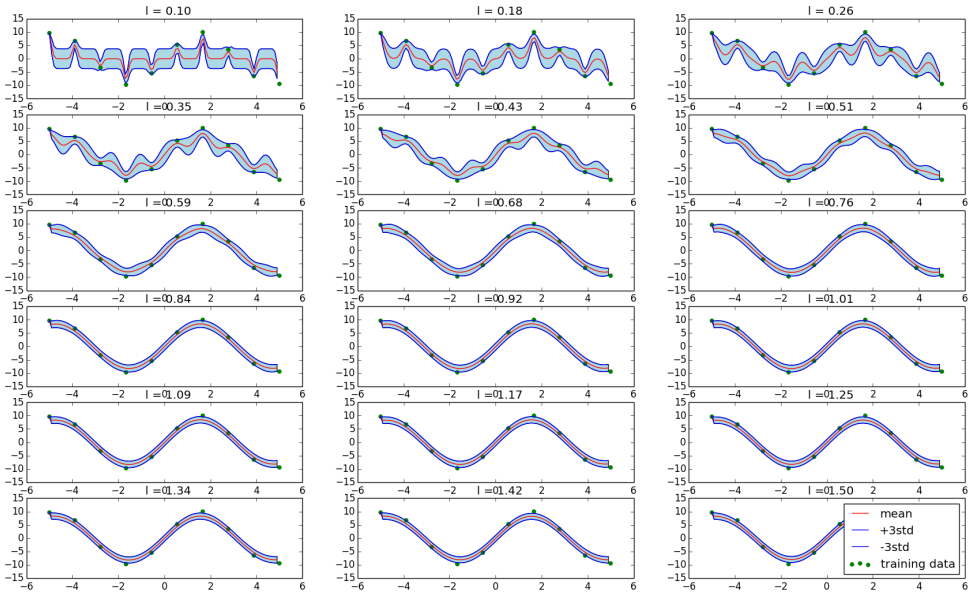
```
import numpy as np
sigma_noise = 0.5
x_min, x_max = -5, 5
X_train = np.linspace(x_min, x_max, num=10)
# Simulate sinusoid with some gaussian noise
Y_train = [10*np.sin(x) + (np.random.rand() - 0.5) * sigma_noise for x in
    X_train]
```

We suggestion you use a kernel function like this:

```
# Kernel function
def rbf_kernel(x, y, l=1.0, sigma_f=1.0, sigma_n=0.5):
    return sigma_f**2 * np.exp(-(x - y)**2 / (2*l**2)) + sigma_n**2*(x==y
    )
```

See code.

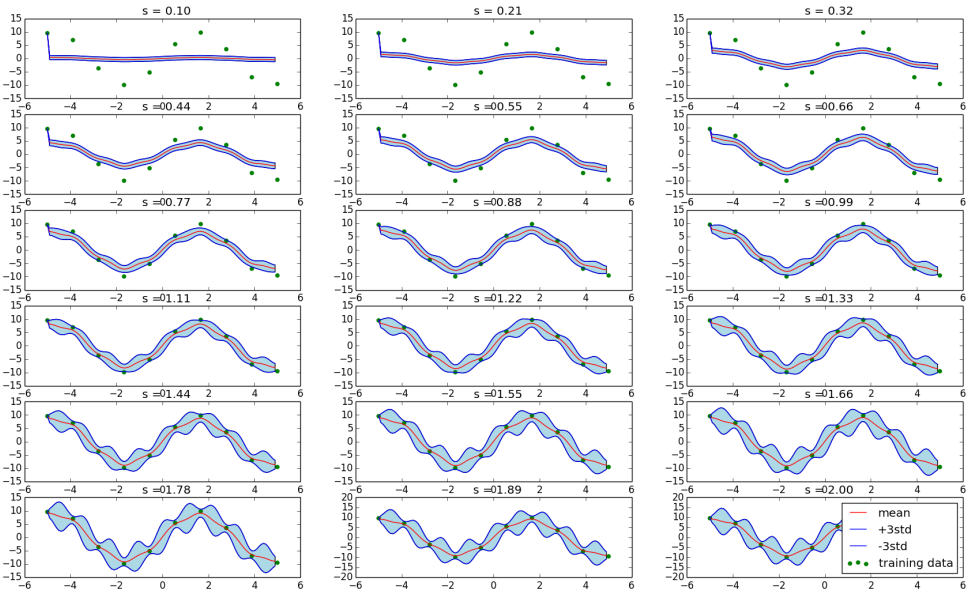
b) Now test different *length* parameter and plot the results and compare them to each other, what do you observe.



An higher *length* parameter incorporates a wider range of data and makes the function smoother, if the range contains new data points. If the value is too small, it only spikes at the data points and else uses the system noise.

c) Do the same for the $sigma_f$ parameter, use a *length* of 0.5. How does it influence the result?

The $sigma_f$ mainly influences the parts where no points are around, an higher value increase the uncertainty in this areas.


s = 0.10
s = 0.21
s = 0.32
s =00.44
s =00.55
s =00.66
s =00.77
s =00.88
s =00.99
s =01.11
s =01.22
s =01.33
s =01.44
s =01.55
s =01.66
s =01.78
s =01.89
s =02.00
mean
+3std
-3std
training data