Principles of Machine Learning: Exercise 3

Alina Pollehn (3197257), Julian Litz (3362592), Manuel Hinz (3334548)
Felix Göhde (3336445), Felix Lehmann (3177181), Caspar Wiswesser (3221493)
Adrian Köring (3347785), Greta Günther (3326765), Linus Mallwitz (3327653)
Niklas Mueller-Goldingen (3363219), Jennifer Kroppen (2783393)

04.12.2023

Implementation of exercise 3.1

outer difference:

```
def diffMatrix(u,v):
    return np.subtract.outer(u,v)
```

outer product:

```
def prodMatrix(u,v):
    return np.multiply.outer(u,v)
```

for general operators:

```
def outer_operator(f,u,v):
    return f(np.expand_dims(u,axis=1), np.expand_dims(v,axis=0))
```

Implementation of exercise 3.2.1

Linear kernel matrix $K(u,v|\alpha) \in \mathbb{R}^{n_u \times n_v}$

$$[K]_{ij} = \alpha u_i v_j$$

```
def linearKernelMatrix(u,v,alpha):
    return alpha*prodMatrix(u,v)
```

Implementation of exercise 3.2.2

gaussian kernel matrix $K(u, v|\alpha, \sigma) \in \mathbb{R}^{n_u \times n_v}$

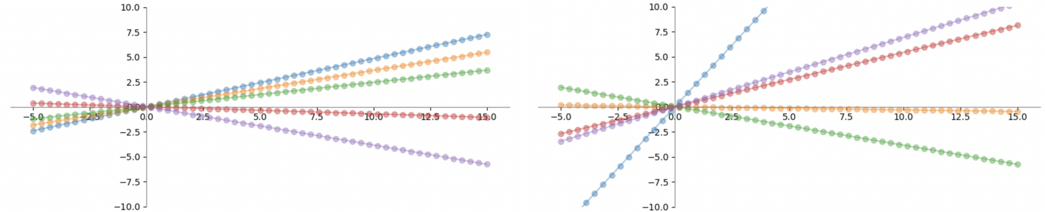
$$[K]_{ij} = \alpha \exp\left(-\frac{(u_i - v_j)^2}{2\sigma^2}\right)$$

```
def gaussKernelMatrix(u,v,alpha,sigma):
    return alpha*(np.exp(-diffMatrix(u,v)**2/(2*sigma**2)))
```

Sampling from a linear kernel matrix

Sampling 5 vectors twice yields

```
y=multivariate_normal(vec0 , linearKernelMatrix(vecX,vecX, 1))
```

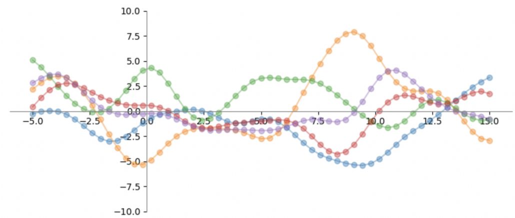


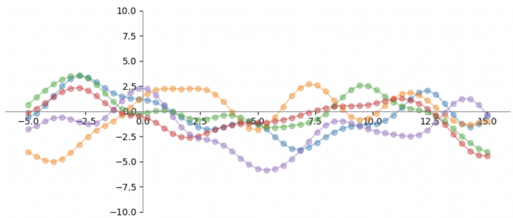
Keep in mind, that these results are random and will look different each time

Sampling from a gaussian kernel matrix

Sampling 5 vectors twice yields

```
y=multivariate_normal(vec0 , gaussKernelMatrix(vecX,vecX, alpha=6, sigma=1.5))
```





Keep in mind, that these results are random and will look different each time

Fitting a gaussian process to the weight data set

- Goal: Given the weight and height data from whData.dat, calculate a gaussian process that fits the data
- Steps:
  1. Remove outliers
  2. Build kernel matrix using diffMatrix and prodMatrix

$$
\begin{aligned}
[K]_{ij} &= \theta_1 \exp\left(-\frac{(x_i - x_j)^2}{2\theta_2^2}\right) + \theta_3 x_i x_j \\
C &= K + \theta_4 I
\end{aligned}
$$

  3. Use Scipy.optimize with appropriate bounds to minimize the negative log-likelihood (that is: maximize the log-likelihood) using

$$
\theta = \begin{pmatrix} 1.0 \\ 20.0 \\ 0.5 \\ 1.0 \end{pmatrix}
$$

- Result: $\theta = (58.318, 12.507, 0.0, 139.343)^T$

Fitting a gaussian process to the weight data set

- Goal: Given the weight and height data from whData.dat, calculate a gaussian process that fits the data
- Approach:
  1. Remove outliers
  2. Build kernel matrix using diffMatrix and prodMatrix

$$[K]_{ij} = \theta_1 \exp\left(-\frac{(x_i - x_j)^2}{2\theta_2^2}\right) + \theta_3 x_i x_j$$
$$C = K + \theta_4 I$$

  3. Use Scipy.optimize with appropriate bounds to minimize the negative log-likelihood (that is: maximize the log-likelihood) using

$$\theta = (1.0, 20.0, 0.5, 1.0)^T$$

```
theta_opt=minimize(lambda theta_prime:negLikelihood(x,y_bar,theta_prime),\
                   x0=theta,bounds=[(0, None),(0, None),(0, None),(0, None)]).x
```

- Result: $\theta = (58.318, 12.507, 0.0, 139.343)^T$

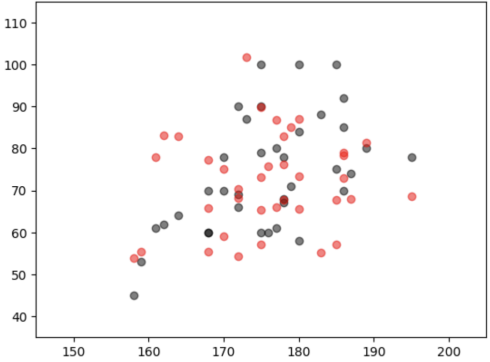
Sampling a fitted Gaussian process model

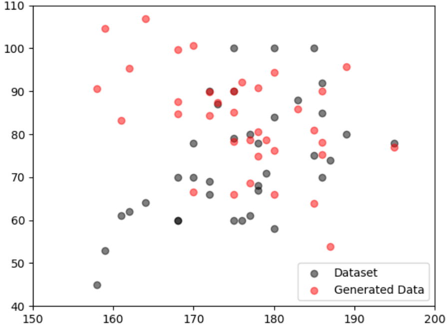
- Now that we have a fitted guassian process, we can use our model to sample random pair according to our distribution.
- Two mathematically equivalent approaches:
  1. Draw $w \sim \overbrace{\mathcal{N}(0, C)}^{=\mathcal{N}(0, K+\theta_4 I)}$
  2. Draw $w \sim \mathcal{N}(0, I)$, calculate a Cholesky factorization $C = LL^T$ and calculate $\overline{y} = Lw$

$$y' = \underbrace{\overline{y}}_{\text{centered sample}} + \underbrace{\frac{1}{n} 1 1^T y}_{\text{mean of weights}}$$

- the second approach results in a faster sampling, because it avoids inverting C

Sampling a fitted Gaussian process model: Results





Predicting with a fitted gaussian model

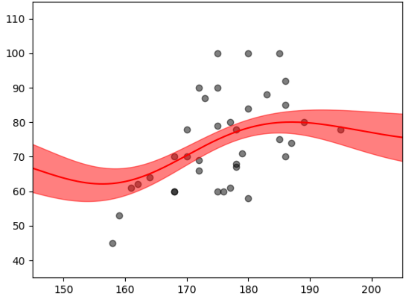
- Goal: Predict weights for heights $x^* \in \mathbb{R}^N$
- Approach:
  1. Given our fitted gaussian with $\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \hat{\theta}_4)^T$
  2. Let

$$
\begin{aligned}
K_{xx} &= K(x, x, \hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3) & K_{x\star} &= K(x, x^*, \hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3), \quad \ldots \\
C &= K_{xx} + \hat{\theta}_4 I & \overline{\mu}^\star &= K_{\star x} C^{-1} \overline{y}, \qquad \Sigma^\star = K_{\star\star} - K_{\star x} C^{-1} K_{x\star} \\
\sigma^\star &= \sqrt{\operatorname{diag}[\Sigma^\star]} & \mu^\star &= \overline{\mu}^\star + \frac{1}{n} 1 1^T y
\end{aligned}
$$

- Our predicted height weight pairs are $(x_j^\star, \mu_j^\star)$
- We also get a one σ-confidence $(x_j^\star, \mu_j^\star \pm \sigma_j^\star)$

Predicting with a fitted gaussian model: Result

We get the following plot:



- Plot as expected:
  1. Thin in the middle (a lot of data)
  2. Greater spread near the lowest / highest weights
- Best model yet, includes with added confidence in each guess.