Lab 9

FTP MachLe MSE
HS 2024

Gaussian Processes

Machine Learning
WÜRC

Essentially, all models are wrong, but some are useful.
George E.T. Box

After this unit, ...

Lernziele/Kompetenzen

- you have repeated the *basic rules of probability theory.*
- you know the difference between a *joint* and a *conditional probability* distribution.
- you know how to apply *Bayes Theorem* to calculate the *posterior* probability distribution for simple discrete examples. You can name the *prior* probability distribution, the *likelihood function*, the *evidence*, and you know how to *marginalize* over a joint probability distribution.
- you know (K1), that every model relies on (explicit or implicit) *assumptions.* We discriminate *knowledge*, *assumptions* and *simplifying assumptions.* In Bayesian reasoning, assumptions are formulated as *prior distribution* $p(\theta)$ over the parameters θ of a model. Using Bayes rule, one can calculate the posterior parameter distribution $p(\theta|x, y)$ given the data (x, y) and the model assumptions.

$$\text{posterior} = p(\theta|x,y) = \frac{p(y|x,\theta)\cdot p(\theta)}{\int_\theta p(y|x,\theta)\cdot p(\theta)d\theta} = \frac{\text{likelihood}\cdot\text{prior}}{\text{marginal}} \tag{1}$$

- you know the basic properties of a *multivariate Gaussian* probability distribution. You can plot a 2D Gaussian probability distribution given the *mean vector* $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$.
- you can explain the *naïve Bayes classifier* to your classmates and to your teacher.
- you know (K1), that both the *conditionals* $p(x|y)$ and the *marginals* $p(x)$ of a joint Gaussian distribution $p(x, y)$ are again Gaussian.
- you know (K1) that a *Gaussian process* $\mathcal{GP}(\mu, k)$ is a generalization of a multivariate Gaussian distribution to infinitely many variables. A Gaussian process is a *prior* over *functions* $p(f)$ which can be used for Bayesian regression. Sampling from a Gaussian process means sampling *functions* (instead of samples of a random variable) out of a pool of functions characterized by a mean function μ and a covariance function $k(x, x')$.

- you are able (K3) to *sample functions* from a Gaussian Process $\mathcal{GP}(\mu, k)$ with given mean $\mu(x)$ and covariance function $k(x, x')$ using the `GaussianProcessRegressor` of the class `sklearn.gaussian_process`.
- you are able (K3) to *fit* n-dimensional data using a Gaussian Process, i.e. you are able to *infer* hyperparameters of the model from given data using the `GaussianProcessRegressor` of the class `sklearn.gaussian_process`.
- you are able (K3) to *make predictions* using the `GaussianProcessRegressor` of the class `sklearn.gaussian_process`.
- you know (K1) the most important covariance functions (kernels) $k(x, x')$, namely the *constant* kernel, the *Gaussian* kernel, the *RBF*-kernel (radial basis function), the *Dot-Product* kernel and the *sine-exponential* kernel.
- you are able (K3) to apply *kernel operations* (namely sum and product) in order to construct a probabilistic model adapted to a given dataset.

---

1. Hamburger and Bayes Rule [A, II]

a) The probability that a hamburger eater HE will have Kreuzfeld-Jacob disease given the prior $p(\mathsf{KJ})$ and the marginal $p(\mathsf{HE}) = \sum_{\mathsf{KJ}} p(\mathsf{HE}|\mathsf{KJ}) \cdot p(\mathsf{KJ})$ is:

$$p(\mathsf{KJ}|\mathsf{HE}) = \frac{p(\mathsf{HE},\mathsf{KJ})}{p(\mathsf{HE})} = \frac{p(\mathsf{HE}|\mathsf{KJ}) \cdot p(\mathsf{KJ})}{p(\mathsf{HE})} \tag{2}$$

$$= \frac{p(\mathsf{HE}|\mathsf{KJ}) \cdot p(\mathsf{KJ})}{\sum_{\mathsf{KJ}} p(\mathsf{HE}|\mathsf{KJ}) \cdot p(\mathsf{KJ})} = \frac{\frac{9}{10} \cdot \frac{1}{100000}}{\frac{1}{2}} \approx \underline{\underline{1.8 \cdot 10^{-5}}} \tag{3}$$

b) If the fraction of people eating hamburgers was rather small, $p(\mathsf{HE}) = 0{,}001$, what is the probability that a regular hamburger eater will have Kreuzfeld-Jacob disease?

$$p(\mathsf{KJ}|\mathsf{HE}) = \frac{p(\mathsf{HE},\mathsf{KJ})}{p(\mathsf{HE})} = \frac{p(\mathsf{HE}|\mathsf{KJ}) \cdot p(\mathsf{KJ})}{p(\mathsf{HE})} \tag{4}$$

$$= \frac{\frac{9}{10} \cdot \frac{1}{100000}}{\frac{1}{1000}} = \underline{\underline{9 \cdot 10^{-3}}} \approx 1\% \tag{5}$$

2. Naïve Bayes Classifier [A, II]

a) Python Code: Naive Bayes prior and likelihoods

```
p_y = 4.0/10; # p(y) = 4/10
# p(xi=1 | y=-1)
p_x1_y0 = 3.0/6;
p_x2_y0 = 5.0/6;
p_x3_y0 = 4.0/6;
p_x4_y0 = 5.0/6;
p_x5_y0 = 2.0/6;
```

```
  # p(xi=1 | y=+1)
  p_x1_y1 = 3.0/4;
  p_x2_y1 = 0.0/4;
  p_x3_y1 = 3.0/4;
  p_x4_y1 = 2.0/4;
  p_x5_y1 = 1.0/4;
```

b) Python Code: Naive Bayes classification decisions

```
f_y1_00000 = p_y*(1-p_x1_y1)*(1-p_x2_y1)*(1-p_x3_y1)*
  (1-p_x4_y1)*(1-p_x5_y1)
print("f_y1_00000 = ",f_y1_00000)

f_y0_00000 = (1-p_y)*(1-p_x1_y0)*(1-p_x2_y0)*(1-p_x3_y0)*
  (1-p_x4_y0)*(1-p_x5_y0)
print("f_y0_00000 = " , f_y0_00000)

if (f_y1_00000 > f_y0_00000):
  print("Predict class +1")
else:
  print ("Predict class -1")
print("\n")

f_y1_11010 = p_y*(p_x1_y1)*(p_x2_y1)*(1-p_x3_y1)*
  (p_x4_y1)*(1-p_x5_y1)
print ("f_y1_11010 = ",f_y1_11010)

f_y0_11010 = (1-p_y)*(p_x1_y0)*(p_x2_y0)*(1-p_x3_y0)*
  (p_x4_y0)*(1-p_x5_y0)
print("f_y0_11010 = ",f_y0_11010)

if (f_y1_11010 > f_y0_11010):
  print("Predict class +1")
else:
  print ("Predict class -1")
```

The numerical solution $x = \{00000\}$ is:

$$\begin{aligned}
&f(y=+1|00000) = 0.009375000000000001\\
&f(y=-1|00000) = 0.0018518518518518515\\
&f(y=+1|00000) > f(y=-1|00000) \implies \hat{y} = +1
\end{aligned}$$

The numerical solution for $x = \{11010\}$ is:

$$\begin{aligned}
&f(y=+1|11010) = 0.0\\
&f(y=-1|11010) = 0.046296296296296315\\
&f(y=+1|11010) < f(y=-1|11010) \implies \hat{y} = -1
\end{aligned}$$

c) Python Code: Naive Bayes posterior probabilities

```
# p(y=1|00000) =
print ("p(y=1|00000) =", f_y1_00000 / (f_y1_00000 + f_y0_00000))

# p(y=1|11010) =
print ("p(y=1|11010) =", f_y1_11010 / (f_y1_11010 + f_y0_11010))
```

The Naive Bayes posterior probabilities are:

$$p(y=1|00000) = 0.8350515463917526$$
$$p(y=1|11010) = 0.0$$

d) A *Bayes classifier using a joint distribution model* for $p(x_1, \dots, x_5|y = c)$ would have $2^5 - 1 = 31$ degrees of freedom (independent probabilities) to estimate. But here, we have only 6 data points from class $y = -1$, and 4 data points from class $y = +1$. Thus these models would assign zero probability to many feature combinations, and *would probably not generalize well* to new data.

e) No, we do not need to re-train the model by estimating new probabilities. Due to the conditional independence assumptions of naïve Bayes, it is optimal to simply ignore $p(x_1|y)$, and use the previously estimated probabilities of the other four features when computing $p(y|x_2, x_3, x_4, x_5)$. If you did recompute $p(x_i|y)$ using the formulas above, it is easy to verify that the values would not change.

3. Weather in London [A, II]

a) Assuming that the prior probability it rained yesterday is 0.5, what is the probability that it was raining yesterday given that it's sunny today?

$$p(\text{yesterday} = rain) = 50\%$$

I infer from this that the probability of being sunny yesterday is also 50%:

$$p(\text{yesterday} = sun) = 50\%$$

$$p(\text{yesterday} = rain \mid \text{today} = sun) = \frac{p(\text{today} = sun \mid \text{yesterday} = rain)p(\text{yesterday} = rain)}{p(\text{today} = sun)}$$

$$\begin{aligned} p(\text{today} = sun) &= \sum_y p(\text{today} = sun, \text{yesterday} = y) \\ &= \sum_y p(\text{today} = sun \mid \text{yesterday} = y)p(\text{yesterday} = y) \\ &= p(\text{today} = sun \mid \text{yesterday} = sun) \cdot p(\text{yesterday} = sun) \\ &+ p(\text{today} = sun \mid \text{yesterday} = rain) \cdot p(\text{yesterday} = rain) \\ &= 0.40 \cdot 0.5 + 0.30 \cdot 0.50 = 0.20 + 0.15 = \underline{\underline{0.35}} \end{aligned}$$

Thus, the probability of raining yesterday given that today is sunny:

$$p(\text{yesterday} = rain \mid \text{today} = sun) = \frac{0.15}{0.35} = \underline{\underline{42.86\%}}$$

b) If the weather follows the same pattern as above, day after day, what is the probability that it will rain on any day (based on an effectively infinite number of days of observing the weather)?
On any day, not considering whether it rained or not the day before, the probability of raining is:

$$\begin{aligned}
p(\mathsf{today} = rain) &= \sum_y p(\mathsf{today} = rain, \mathsf{yesterday} = y) \\
&= \sum_y p(\mathsf{today} = rain \mid \mathsf{yesterday} = y) p(\mathsf{yesterday} = y) \\
&= p(\mathsf{today} = rain \mid \mathsf{yesterday} = rain) p(\mathsf{yesterday} = rain) \\
&+ p(\mathsf{today} = rain \mid \mathsf{yesterday} = sun) p(\mathsf{yesterday} = sun) \\
&= 0.7 \cdot 0.5 + 0.6 \cdot 0.5 = \underline{\underline{0.65}}
\end{aligned}$$

c) Use the result from b) above as a new prior probability of rain yesterday and recompute the probability that it was raining yesterday given that it's sunny today.

$$p(\mathsf{yesterday} = rain) = 0.65$$

Therefore:

$$p(\mathsf{yesterday} = sun) = 0.35$$

$$p(\mathsf{yesterday} = rain \mid \mathsf{today} = sun) = \frac{p(\mathsf{today} = sun \mid \mathsf{yesterday} = rain) p(\mathsf{yesterday} = rain)}{p(\mathsf{today} = sun)}$$

$$\begin{aligned}
p(\mathsf{today} = sun) &= \sum_y p(\mathsf{today} = sun, \mathsf{yesterday} = y) \\
&= \sum_y p(\mathsf{today} = sun \mid \mathsf{yesterday} = y) p(\mathsf{yesterday} = y) \\
&= p(\mathsf{today} = sun \mid \mathsf{yesterday} = sun) p(\mathsf{yesterday} = sun) \\
&+ p(\mathsf{today} = sun \mid \mathsf{yesterday} = rain) p(\mathsf{yesterday} = rain) \\
&= 0.40 \cdot 0.35 + 0.30 \cdot 0.65 \\
&= 0.14 + 0.195 = \underline{\underline{0.335}}
\end{aligned}$$

$$p(\mathsf{yesterday} = rain \mid \mathsf{today} = sun) = \frac{0.30 \cdot 0.65}{0.335} = \underline{\underline{58.21\%}}$$

Very interesting. The probability of raining has increased a little bit after updating the prior probabilities. Since it was likely that it rained yesterday, it's slightly more likely that it will rain today.

4. Bivariate Gaussian Distribution [A,II]

a) The *bivariate normal density* $p(\mathbf{x}) = p(x_a, x_b)$ is defined by

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}\sqrt{\det(\boldsymbol{\Sigma})}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

In this equation, we have $D = 2$ and

$$\begin{aligned}
\mathbf{x} &= \begin{pmatrix} x_a \\ x_b \end{pmatrix} \\
\boldsymbol{\mu} &= \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\
\boldsymbol{\Sigma} &= \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} = \begin{pmatrix} 2 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 1 \end{pmatrix} \\
|\boldsymbol{\Sigma}| &= \det(\boldsymbol{\Sigma}) = 2 \cdot 1 - \left(\frac{\sqrt{2}}{2}\right)^2 = \frac{3}{2} \\
\sqrt{|\boldsymbol{\Sigma}|} &= \sqrt{\frac{3}{2}}
\end{aligned}$$

The inverse matrix of a 2×2-matrix can be calculated by:

$$\begin{aligned}
A &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \\
A^{-1} &= \frac{1}{\det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}
\end{aligned}$$

In this case, we get the following result for the precision matrix $\boldsymbol{\Lambda}$:

$$\boldsymbol{\Lambda} = \frac{2}{3} \begin{pmatrix} 1 & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & 2 \end{pmatrix}$$

b) The squared generalized distance expression, i.e. the *Mahalanobis distance* can be written as:

$$\begin{aligned}
\boldsymbol{\Delta} &= (\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu}) \\
&= \begin{pmatrix} x_a \\ x_b - 2 \end{pmatrix}^T \frac{2}{3} \begin{pmatrix} 1 & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & 2 \end{pmatrix} \begin{pmatrix} x_a \\ x_b - 2 \end{pmatrix} \\
&= \frac{2}{3}\left(x_a^2 - \sqrt{2}x_a(x_b-2) + 2(x_b-2)^2\right)
\end{aligned}$$

as a function of x_a and x_b.

c) The joint probability density is explicitely given by:

$$\begin{aligned}
p(\mathbf{x}) = p(x_a, x_b) &= \frac{1}{(2\pi)^{2/2}\sqrt{3/2}} \cdot \exp\left\{-\frac{1}{2}\begin{pmatrix} x_a \\ x_b - 2 \end{pmatrix}^T \frac{2}{3} \begin{pmatrix} 1 & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & 2 \end{pmatrix} \begin{pmatrix} x_a \\ x_b - 2 \end{pmatrix}\right\} \\
&= \frac{1}{\sqrt{6}\pi} \cdot \exp\left\{-\frac{1}{3}\left(x_a^2 - \sqrt{2}x_a(x_b-2) + 2(x_b-2)^2\right)\right\}
\end{aligned}$$

d) We calculate the *eigenvalues* $\lambda_{1,2}$ and the *eigenvectors* $\mathbf{u}_{1,2}$ of the covariance matrix $\mathbf{\Sigma}$ using `np.linalg.eig`.

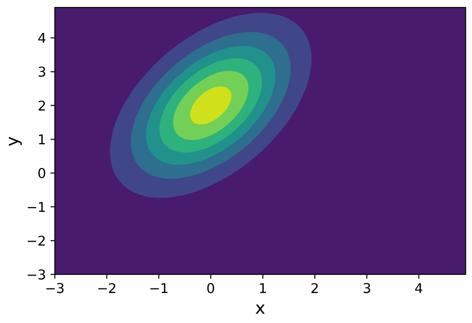
```
    import numpy as np
    w, v = np.linalg.eig(np.array([[2, np.sqrt(2)/2], [np.sqrt(2)/2, 1]]
                                                  ))
    print(w)
    print(v)

    [2.3660254 0.6339746]
    [[ 0.88807383 -0.45970084]
    [ 0.45970084  0.88807383]]
```

The eigenvalues of $\mathbf{\Sigma}$ are $(\lambda_1, \lambda_2) = (2.3660254, 0.6339746)$ with eigenvectors:

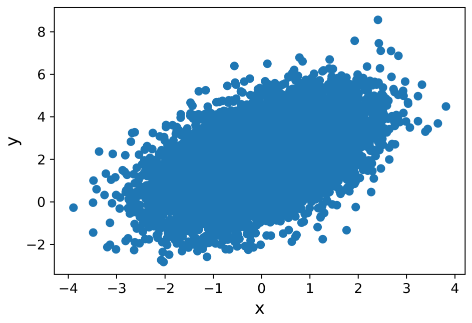
$$(\mathbf{u}_1, \mathbf{u}_1) = \begin{pmatrix} 0.88807383 & -0.45970084 \\ 0.45970084 & 0.88807383 \end{pmatrix}$$

e) The Python code could look like this:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 21 19:52:21 2020
@author: wuersch
"""
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal

# Mean vector and covariance matrix
mu     = np.array([0., 2.])
Sigma  = np.array([[2, np.sqrt(2)/2], [np.sqrt(2)/2, 1]])

F = multivariate_normal(mu, Sigma)

#draw random samples from the multivariate distribution
#and try to reconstruct the gaussian distribution

NSamples=10000
x, y = np.mgrid[-3:5:.1, -3:5:.1]

pos         = np.dstack((x, y))
MVGauss     = multivariate_normal(mu, Sigma)
MVGSamples  = MVGauss.rvs(size=NSamples)
XS          = MVGSamples[:,0]
YS          = MVGSamples[:,1]

fig2 = plt.figure('Using Scikit Learn')
ax2  = fig2.add_subplot(111)
ax2.contourf(x, y, F.pdf(pos))
ax2.set_xlabel("x")
ax2.set_ylabel("y")
plt.savefig('Gauss3D_2.png',dpi=600)

fig3 =plt.figure('Using Scikit Learn to draw random samples')
ax3  = fig3.add_subplot(111)
ax3.scatter(XS,YS)
ax3.set_xlabel("x")
```



(a) bivariate gaussian



(b) 10'0000 samples

```
ax3.set_ylabel("y")
plt.savefig('Gauss3D_3.png',dpi=600)
```

5. Prior samples and posterior distributions from differnt kernels of a $\mathcal{GP}$ [A, II]

The solution Juypter notebook can be found on moodle:
`Lab9_A5_plot_gpr_prior_posterior.ipynb`

6. Model fitting, prediction and noise estimation using a $\mathcal{GP}$ [A, II]

The solution Juypter notebook can be found on moodle:
`Lab9_A6_FitGPModel_NoiseEstimation_solution.ipynb`