OST
Ostschweizer
Fachhochschule

Lab 7

Support Vector Machines (SVM)

FTP MachLe MSE
FS 2022

MSE MachLe
WÜRC

Lernziele/Kompetenzen

- You know the primal form of a *separating hyperplane classifier*:

$$f_{w,b}(x) = \text{sign}(\langle w, x\rangle + b)$$

- You can calculate the *geometric margin* γ_i of a given sample point x_i for a separating hyperplane classifier $f_{w,b}$ with given parameters w and b.

$$\gamma_i := y_i \cdot \left(\frac{\langle w, x_i\rangle + b}{\|w\|}\right)$$

- You can explain the progression of the SVM family from *maximal margin classifier* to *support vector classifier* (SVC) and finally to *support vector machines* (SVM) that use the kernel trick.
- You can use SVM successfully on tutorial style examples, including (cross-validated) parameter *grid search*.
- You know, that SVM use only a subset of training points in the decision function (called *support vectors*), so they are memory efficient. SVM are still effective in cases where number of dimensions d is greater than the number of samples N.
- You know what a *kernel function* $K(x_i, x_j)$ is and how it is related to a *feature transform* $\phi(x)$ (Mercer Theorem). Different kernel functions can be specified for the decision function of a SVM. Common kernels are the *radial basis function* kernel, the *linear* kernel and the *polynomial* kernel.

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle_{\mathcal{H}}$$

- You can explain the effect of the C and γ parameters for a SVM with a radial basis function kernel (rbf-kernel). If you have a lot of noisy observations, C should be small. *Decreasing* C corresponds to *higher bias*, *less variance*, *larger margin*, *higher tolerance for misclassification*, less confidence in the dataset (more noise).

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

- You know that you can always *apply the kernel trick* whenever there appears an scalar product $\langle x_i, x_j\rangle$ of two feature vectors x_i and x_j in a given loss function $\mathcal{L}$. You know that by applying the kernel trick to the dual loss function of the separating hyperplane classifier linearly non-separable classes become separable if the chosen dimension is high enough.

1. Feature Transform ϕ of the Polynomial Kernel [A,I]

a) Wie calculate the feature transform $\phi(\vec{z}) : \mathbb{R}^2 \to \mathbb{R}$ associated with the inhomogeneous quadratic polynomial kernel $K(\vec{z}_1, \vec{z}_2)$ explicitely for two-dimensional features $\vec{z} = \begin{pmatrix} x \\ y \end{pmatrix}$, i.e. we have to find a feature transform $\phi(\vec{z})$ and an associated HILBERT space $\mathcal{H}$ such that the kernel can be expressed as a scalar product of the transformed features $\phi(\vec{z})$.

$$K(\vec{z}_1, \vec{z}_2) = (\vec{z}_1 \cdot \vec{z}_2 + 1)^2 = \langle \phi(\vec{z}_1), \phi(\vec{z}_2) \rangle_{\mathcal{H}}$$

We do this by expanding the square and then write down the kernel function as a *sum of symmetric products*. Then we are sure, that we can represent this as a scalar product in a higher dimensional vector HILBERT space $\mathcal{H}$. This is guaranteed by the MERCER theorem.

$$\begin{aligned}
K(\vec{z}_1, \vec{z}_2) &= (\vec{z}_1 \cdot \vec{z}_2 + 1)^2 \\
&= (\vec{z}_1 \cdot \vec{z}_2)^2 + 2\,(\vec{z}_1 \cdot \vec{z}_2) + 1 \\
&= (x_1 x_2 + y_1 y_2)^2 + 2 \cdot (x_1 x_2 + y_1 y_2) + 1 \\
&= (x_1^2 x_2^2 + 2 \cdot x_1 x_2 y_1 y_2 + y_1^2 y_2^2) + 2 \cdot (x_1 x_2 + y_1 y_2) + 1 \\
&= (x_1^2) \cdot (x_2^2) + (\sqrt{2} x_1 y_1) \cdot (\sqrt{2} x_2 y_2) + (y_1^2) \cdot (y_2^2) \\
&+ (\sqrt{2} x_1) \cdot (\sqrt{2} x_2) + (\sqrt{2} y_1) \cdot (\sqrt{2} y_2) + 1 \cdot 1
\end{aligned}$$

This is a sum of symmetric factors, where the first factor only depends on $\vec{z}_1$ and the second factor only depends $\vec{z}_2$. on By collecting now the first summands of the linearly independent compontens, we can construct the feature transform:

$$\phi(\vec{z}_1) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 y_1 \\ y_1^2 \\ \sqrt{2} x_1 \\ \sqrt{2} y_1 \\ 1 \end{pmatrix}$$

Whe can then write the kernel function $K(\vec{z}_1, \vec{z}_2)$ as a normal scalar product in a 6-dimensional vector HILBERT space.

$$K(\vec{z}_1, \vec{z}_2) = \langle \phi(\vec{z}_1), \phi(\vec{z}_2) \rangle_{\mathcal{H}} = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 y_1 \\ y_1^2 \\ \sqrt{2} x_1 \\ \sqrt{2} y_1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} x_2^2 \\ \sqrt{2} x_2 y_2 \\ y_2^2 \\ \sqrt{2} x_2 \\ \sqrt{2} y_2 \\ 1 \end{pmatrix}$$

b) The dimensionality $\dim(\mathcal{H})$ of the associated Hilbert space, i.e. the dimensionality of the transformed features $\phi(\vec{z})$ is $D = 6$.

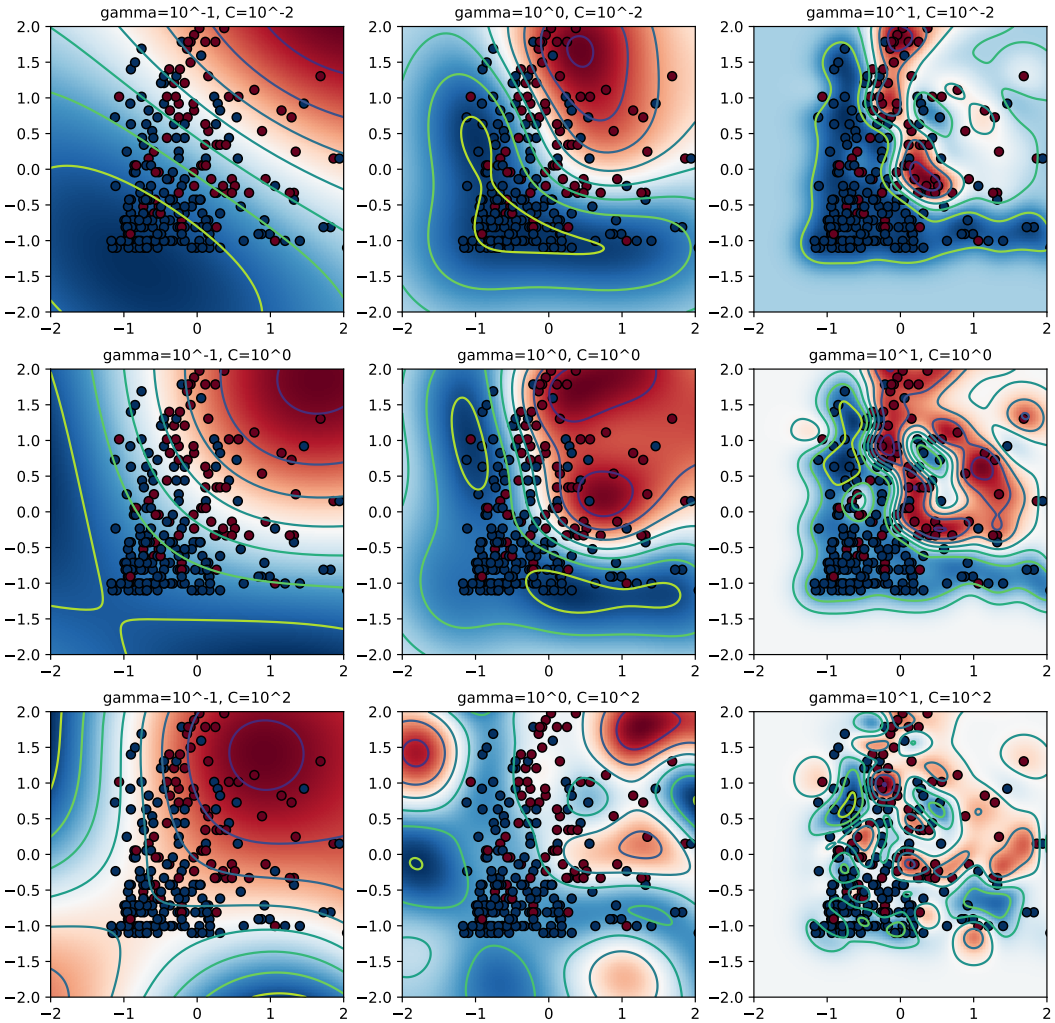
c) The dimensionality $\dim(\mathcal{H})$ of the associated Hilbert space for the radial basis function kernel would be $D = \infty$. We would have to expand the exponential function in an infinite Taylor series to be able to write down the kernel function as a normal dot product.

2. Simple SVM example [A,I]

The Juypter notebook can be found on moodle:
`ML07_A2_SVM_Simple.jpynb`

3. Predicting Diabetes using a SVM [A,II]

The Juypter notebook can be found on moodle:
`ML07_A3_PimaIndians.jpynb`



4. Polynomial Kernel SVM [A,I]
The Juypter notebook can be found on moodle:
`ML07_A4_PolynomialKernel.jpynb`

5. General Questions about SVM [A,I]

a) The fundamental idea behind Support Vector Machines is to fit the widest possible «street»between the classes. In other words, the goal is to have the *largest possible margin* between the decision boundary that separates the two classes and the training instances. When performing soft margin classification, the SVM searches for a compromise between perfectly separating the two classes and having the widest possible street (i.e., a few instances may end up on the street). Another key idea is to use *kernels* when training on nonlinear datasets.

b) After training an SVM, a support vector is any instance located on the «street»(see the previous answer), including its border. The decision boundary is entirely determined by the support vectors. Any instance that is not a support vector (i.e., off the street) has no influence whatsoever; you could remove them, add more instances, or move them around, and as long as they stay off the street they won't affect the decision boundary. Computing the predictions only involves the support vectors, not the whole training set.

c) SVMs try to fit the largest possible «street»between the classes (see the first answer), so if the training set is not scaled, the SVM will tend to neglect small features.

d) An SVM classifier can output the distance between the test instance and the decision boundary, and you can use this as a confidence score. However, this score cannot be directly converted into an estimation of the class probability. If you set `probability=True` when creating an SVM in `Scikit-Learn`, then after training it will calibrate the probabilities using Logistic Regression on the SVM's scores (trained by an additional five-fold cross-validation on the training data). This will add the `predict_proba()` and `predict_log_proba()` methods to the SVM.

e) This question applies only to linear SVMs *since kernelized can only use the dual form.* The computational complexity of the primal form of the SVM problem is proportional to the number of training instances N, while the computational complexity of the dual form is proportional to a number between N^2 and N^3. So if there are millions of instances, you should definitely use the primal form, because the dual form will be much too slow.

f) If an SVM classifier trained with an RBF kernel *underfits* the training set, there might be too much regularization. To decrease it, you need to *increase* γ or C (or both).

OST
Ostschweizer
Fachhochschule

Lab 8 (solution)

Probabilistic Reasoning - Gaussian distribution and Bayes Theorem

FTP MachLe MSE
FS 2022

Machine Learning
WÜRC

The central *paradigm of probabilistic reasoning* is to identify all relevant variables $x_1, \ldots, x_N$ in the environment,and make a probabilistic model $p(x_1, \ldots, x_N)$ of their interaction. Reasoning (*inference*) is then performed by introducing *evidence* that sets variables in known states, and subsequently computing probabilities of interest, *conditioned on this evidence*. The rules of probability, combined with Bayes' rule make a reasoning system complete.

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 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 *sample* data points from a given multivariate gaussian distribution.
- you can explain the *naïve Bayes classifier* to your classmates and to your teacher.

1. Supervised Bayesian Learining [M,II]

The table below contains the result of a market survey for a promotion for different items such as a magazine, a watch and a life insurance and a credit card insurance. 10 people were interviewed and asked whether they would buy such items.

Use this count table for supervised Bayesian learning. The output attribute is *sex* with possible values `male` and `female`. Consider an individual who has said *no* to the life insurance promotion, *yes* to the magazine promotion, *yes* to the watch promotion and *yes* to the credit card insurance. Use the values in the table together with the Naive Bayes classifier to determine which of a,b,c or d represents the probability that this individual is male. $p(E)$ is the marginal distribution.

	Magazine		Watch		Life Insurance		Credit Card	
	`male`	`female`	`male`	`female`	`male`	`female`	`male`	`female`
yes	4	3	2	2	2	3	2	1
no	2	1	4	2	4	1	4	3

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) $p(\texttt{sex} = \texttt{male}\|...) = \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{2}{6} \cdot \frac{2}{6} \cdot \frac{6}{10} \cdot \frac{1}{p(E)}$	○	●
b) $p(\texttt{sex} = \texttt{male}\|...) = \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{3}{4} \cdot \frac{2}{6} \cdot \frac{3}{4} \cdot \frac{1}{p(E)}$	○	●
c) $p(\texttt{sex} = \texttt{male}\|...) = \frac{4}{6} \cdot \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{2}{6} \cdot \frac{6}{10} \cdot \frac{1}{p(E)}$	●	○
d) $p(\texttt{sex} = \texttt{male}\|...) = \frac{2}{6} \cdot \frac{4}{6} \cdot \frac{4}{6} \cdot \frac{2}{6} \cdot \frac{4}{10} \cdot \frac{1}{p(E)}$	○	●

2. 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:

$$p(\mathsf{KJ}|\mathsf{HE}) = \frac{p(\mathsf{HE},\mathsf{KJ})}{p(\mathsf{HE})} = \frac{p(\mathsf{HE}|\mathsf{KJ}) \cdot p(\mathsf{KJ})}{p(\mathsf{HE})} \tag{1}$$

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

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?

$$p(\mathsf{KJ}|\mathsf{HE}) = \frac{p(\mathsf{HE},\mathsf{KJ})}{p(\mathsf{HE})} = \frac{p(\mathsf{HE}|\mathsf{KJ}) \cdot p(\mathsf{KJ})}{p(\mathsf{HE})} \tag{3}$$

$$= \frac{\frac{9}{10} \cdot \frac{1}{100000}}{\frac{1}{1000}} = \underline{\underline{9 \cdot 10^{-3}}} \approx 1\% \tag{4}$$

3. 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:

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

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

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

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 $p(x_1, \ldots, x_5|y = c)$ 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.

4. Passenger Scanner [A, II]

The detector is such that 95% of all terrorists are identified as terrorists

$$p(\mathsf{label} = true \mid \mathsf{terr} = true) = 0.95 \tag{5}$$

We can infer from the former that only 5% of terrorists got mislabeled as good people (false negative):

$$p(\mathsf{label} = false \mid \mathsf{terr} = true) = 0.05 \tag{6}$$

Furthermore 95% of all upstanding citizens are identified as such (non terrorists). Therefore, the false positive rate is:

$$p(\mathsf{label} = true \mid \mathsf{terr} = false) = 0.05 \tag{7}$$

Assuming that the informant is correct:

$$p(\mathsf{terr} = true) = \frac{1}{100} \tag{8}$$

We can then infer that $\frac{99}{100}$ are not terrorists:

$$p(\mathsf{terr} = false) = \frac{99}{100} \tag{9}$$

We can find out the probability that the person picked is a terrorist using *Bayes' Rule*:

$$p(\mathsf{terr} = true \mid \mathsf{label} = true) = \frac{p(\mathsf{label} = true \mid \mathsf{terr} = true)p(\mathsf{terr} = true)}{p(\mathsf{label} = true)} \tag{10}$$

To find out the denominator, we can *marginalize* over the joint distribution of $p(\mathsf{label} = true)$ and $p(\mathsf{terr})$:

$$\begin{aligned} p(label = true) &= \sum_t p(\mathsf{label} = true, \mathsf{terr} = t) \\ &= \sum_t p(\mathsf{label} = true \mid \mathsf{terr} = t) \cdot p(\mathsf{terr} = t) \end{aligned}$$

We make the summation explicit:

$$\begin{aligned} p(label = true) = p(\mathsf{label} = true \mid \mathsf{terr} = true)p(\mathsf{terr} = true)+ \\ p(\mathsf{label} = true \mid \mathsf{terr} = false)p(\mathsf{terr} = false) \\ = 0.95 \cdot 0.01 + 0.05 \cdot 0.99 = \underline{\underline{0.059}} \end{aligned}$$

Substituting all the values into Bayes' Rule, we find out that even though the detector is 'reliable' we still get a low posterior probability that the person suspected of being a terrorist is actually a terrorist:

$$\begin{aligned} p(\mathsf{terr} = true \mid \mathsf{label} = true) &= \frac{p(\mathsf{label} = true \mid \mathsf{terr} = true)p(\mathsf{terr} = true)}{p(\mathsf{label} = true)} \\ &= \frac{0.95(0.01)}{0.059} = \underline{\underline{0.1610169492}} \end{aligned}$$

5. 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

$$\begin{aligned} \mathbf{x} &= \begin{pmatrix} x_a \\ x_b \end{pmatrix} \\ \boldsymbol{\mu} &= \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\ \boldsymbol{\Sigma} &= \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} = \begin{pmatrix} 2 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 1 \end{pmatrix} \end{aligned}$$

$$|\mathbf{\Sigma}| = \det(\mathbf{\Sigma}) = 2 \cdot 1 - \left(\frac{\sqrt{2}}{2}\right)^2 = \frac{3}{2}$$
$$\sqrt{|\mathbf{\Sigma}|} = \sqrt{\frac{3}{2}}$$

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

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

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

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

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

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

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`.

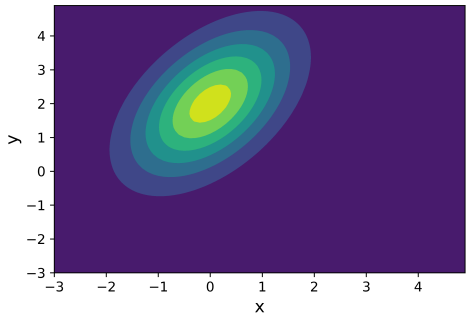
```
import numpy as np
w, v = np.linalg.eig(np.array([[2, np.sqrt(2)/2], [np.sqrt(2)/2, 1]]
                                                  ))
print(w)
print(v)

[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:

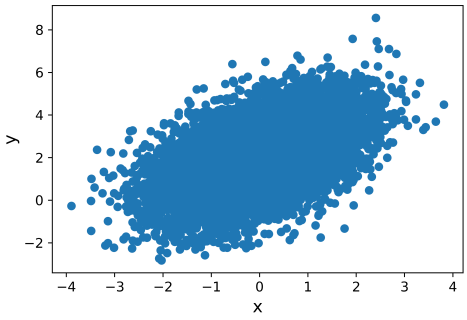
$$(\mathbf{u}_1, \mathbf{u}_1) = \begin{pmatrix} 0.88807383 & -0.45970084 \\ 0.45970084 & 0.88807383 \end{pmatrix}$$

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



(a) bivariate gaussian



(b) 10'0000 samples

f) We calculate the *conditional probability* $p(x_a|x_b)$. In general, the conditional of a multivariate Gaussian distribution is again gaussian with a mean $\boldsymbol{\mu}_{a|b}$ and a covariance matrix $\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Lambda}_{aa}^{-1}$

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

$$\begin{aligned} p(\mathbf{x}_a|\mathbf{x}_b) &= \mathcal{N}\left(\mathbf{x}_a|\boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}\right) \\ \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b) \\ \boldsymbol{\Sigma}_{a|b} &= \boldsymbol{\Lambda}_{aa}^{-1} \end{aligned}$$

$$\begin{aligned} \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b) \\ &= 0 + \frac{\sqrt{2}}{2} \cdot (1)^{-1} (x_b - 2) \\ &= \frac{\sqrt{2}}{2} (x_b - 2) \end{aligned}$$

$$\begin{aligned} \boldsymbol{\Sigma}_{a|b} &= \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba} \\ &= 2 - \frac{\sqrt{2}}{2} \cdot (1)^{-1} \cdot \frac{\sqrt{2}}{2} \\ &= 2 - \frac{1}{2} = \frac{3}{2} \\ &= \boldsymbol{\Lambda}_{aa}^{-1} = \left(\frac{2}{3}\right)^{-1} = \frac{3}{2} \end{aligned}$$

$$\begin{aligned} p(\mathbf{x}_a|\mathbf{x}_b) &= \mathcal{N}\left(\mathbf{x}_a|\boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}\right) \\ &= \mathcal{N}\left(\mathbf{x}_a \middle| \frac{\sqrt{2}}{2}(x_b - 2), \frac{3}{2}\right) \end{aligned}$$

6. 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?

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

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

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

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

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

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

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

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:

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

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

$$p(\mathsf{today} = sun) = \sum_y p(\mathsf{today} = sun, \mathsf{yesterday} = y)$$

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

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

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.

7. Inspector Clouseau [A, II]

a) Using b for the two states of B and m for the two states of M,

$$
p(B|K) = \sum_m p(B, m|K) = \sum_m \frac{p(B, m, K)}{p(K)} = \frac{p(B) \sum_m p(K|B, m) p(m)}{\sum_b p(b) \sum_m p(K|b, m) p(m)}
$$

Plugging in the values we have

$$
\begin{aligned}
p(B = \text{murderer}|\text{knife used}) &= \frac{\frac{6}{10}\left(\frac{2}{10} \cdot \frac{1}{10} + \frac{8}{10} \cdot \frac{6}{10}\right)}{\frac{6}{10}\left(\frac{2}{10} \cdot \frac{1}{10} + \frac{8}{10} \cdot \frac{6}{10}\right) + \frac{4}{10}\left(\frac{2}{10} \cdot \frac{2}{10} + \frac{8}{10} \cdot \frac{3}{10}\right)} \\
&= \frac{300}{412} \approx \underline{\underline{0.73}}
\end{aligned}
$$

Remark: The role of $p(\text{knife used})$ in the Inspector Clouseau example can cause some confusion. In the above, $p(\text{knife used})$ is computed to be 0.412.

$$
p(\text{knife used}) = \sum_b p(b) \cdot \sum_m p(\text{knife used}|b, m) \cdot p(m) \approx \underline{\underline{0.412}} \tag{11}
$$

But surely, $p(\text{knife used}) = 1$, since this is given in the question! Note that the quantity $p(\text{knife used})$ relates to the *prior probability* the model assigns to the knife being used (in the absence of any other information). If we know that the knife is used, then the posterior $p(\text{knife used}) = 1$.

8. Factorization of a multivariate probability distribution [A, II]

A belief network (BN) is a distribution of the form

$$
p(x_1, \ldots, x_N) = \prod_{i=1}^{N} p(x_i|\text{pa}(x_i)) \tag{12}
$$

where pa(x_i) represent the *parental* variables of variable x_i. Represented as a directed graph, with an arrow pointing from a parent variable to child variable, a *belief network* corresponds to a *Directed Acyclic Graph* (DAG), with the i^{th} node in the graph corresponding to the factor $p(x_i|\text{pa}(x_i))$. In general, two different graphs may represent the same independence assumptions. If one wishes to make independence assumptions, then the choice of factorisation becomes significant.
The observation that any distribution may be written in the *cascade form*, gives an algorithm for constructing a BN on variables $x_1, \ldots, x_n$: write down the $n-$node cascade graph; label the nodes with the variables in any order; now each successive independence statement corresponds to deleting one of the edges. More formally, this corresponds to an ordering of the variables which, without loss of generality, we may write as $x_1, \ldots, x_n$. Then, from Bayes' rule, we have

$$p(x_1, \ldots, x_N) = p(x_1|x_2, \ldots, x_N) \cdot p(x_2, \ldots, x_N) \tag{13}$$
$$= p(x_1|x_2, \ldots, x_N) \cdot p(x_2|x_3, \ldots, x_N) \cdot p(x_3, \ldots, x_N) \tag{14}$$
$$= p(x_n) \cdot \prod_{i=1}^{N-1} p(x_i|x_{i+1}, \ldots, x_N) \tag{15}$$

The representation of any BN is therefore a Directed Acyclic Graph (DAG). Every probability distribution can be written as a BN, even though it may correspond to a fully connected 'cascade' DAG. The particular role of a BN is that the structure of the DAG corresponds to a *set of conditional independence assumptions*, namely which ancestral parental variables are sufficient to specify each conditional probability table. Note that this does not mean that non-parental variables have no influence.

9. Conditional distribution of a bivariate Gaussian distribution [A, II]

- We start by calculating the precision matrix $\mathbf{\Lambda} = \mathbf{\Sigma}^{-1}$, the inverse of the covariance matrix $\mathbf{\Sigma}$. We use the fact that the inverse of a 2×2-matrix $\mathbf{A}$ is given by:

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{16}$$
$$\mathbf{A}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \tag{17}$$

  In our case, we have

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \tag{18}$$
$$\mathbf{\Lambda} = \mathbf{\Sigma}^{-1} = \frac{1}{1 - \rho^2} \begin{pmatrix} \frac{1}{\sigma_1^2} & -\frac{\rho}{\sigma_1\sigma_2} \\ -\frac{\rho}{\sigma_1\sigma_2} & \frac{1}{\sigma_2^2} \end{pmatrix} \tag{19}$$

- Now, we concentrate only on the *quadratic term* $\square$ in the exponential without the factor $-\frac{1}{2}$ which is given by:

$$\square = \frac{1}{1 - \rho^2} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix}^T \begin{pmatrix} \frac{1}{\sigma_1^2} & -\frac{\rho}{\sigma_1\sigma_2} \\ -\frac{\rho}{\sigma_1\sigma_2} & \frac{1}{\sigma_2^2} \end{pmatrix} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix} \tag{20}$$

$$= \frac{1}{1-\rho^2} \begin{pmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{pmatrix}^T \cdot \begin{pmatrix} \frac{1}{\sigma_1^2}(x_1-\mu_1) - \frac{\rho}{\sigma_1\sigma_2}(x_2-\mu_2) \\ \frac{1}{\sigma_2^2}(x_2-\mu_2) - \frac{\rho}{\sigma_1\sigma_2}(x_1-\mu_1) \end{pmatrix} \tag{21}$$

$$= \frac{1}{1-\rho^2}\left\{\frac{1}{\sigma_1^2}(x_1^2 - 2\mu_1 x_1 + \mu_1^2) - \frac{\rho}{\sigma_1\sigma_2}(x_1x_2 - x_1\mu_2 - x_2\mu_1 + \mu_1\mu_2)\right\} \tag{22}$$

$$+ \frac{1}{1-\rho^2}\left\{\frac{1}{\sigma_2^2}(x_2^2 - 2\mu_2 x_2 + \mu_2^2) - \frac{\rho}{\sigma_1\sigma_2}(x_1x_2 - x_1\mu_2 - x_2\mu_1 + \mu_1\mu_2)\right\} \tag{23}$$

- Now, we only consider terms quadratic and linear in x_1 and get:

$$\square = \frac{1}{1-\rho^2}\left\{\frac{x_1^2}{\sigma_1^2} - x_1 \cdot \left(\frac{2\mu_1}{\sigma_1^2} + \frac{2\rho}{\sigma_1\sigma_2}(x_2-\mu_2)\right) + \dots\right\} \tag{24}$$

$$= \frac{1}{1-\rho^2}\frac{1}{\sigma_1^2}\left\{x_1^2 - x_1 \cdot \left(2\mu_1 + \frac{2\rho\sigma_1^2}{\sigma_1\sigma_2}(x_2-\mu_2)\right) + \dots\right\} \tag{25}$$

- By completing the square, we finally get:

$$\square = \frac{1}{1-\rho^2}\frac{1}{\sigma_1^2}\left\{\left(x_1 - (\mu_1 + \frac{\rho\sigma_1}{\sigma_2}(x_2-\mu_2))\right)^2 + \dots\right\} \tag{26}$$

- By identifying the coefficient before the quadratic term as the inverse of the variance $\tilde{sigma}$ of the conditional probability $p(x_1|x_2)$ and the shift as the mean $\tilde{\mu}$ of the conditional probability density, we find:

$$\tilde{\mu} = \mu_1 + \rho\frac{\sigma_1}{\sigma_2} \cdot (x_2 - \mu_2) \tag{27}$$

$$\tilde{\sigma} = (1-\rho^2) \cdot \sigma_1^2 \tag{28}$$

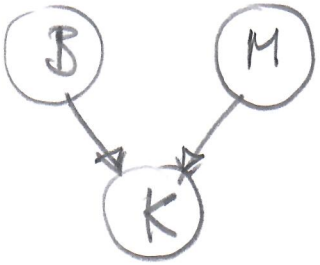
Inspector Clouseau

$p(b) = 0.6$
$p(m) = 0.2$
} priors, independent!

Table of conditionals: (et complements)

$p(k \mid \bar{b}, \bar{m}) = 0.3 \longrightarrow p(\bar{k} \mid \bar{b}, \bar{m}) = 1 - p(k \mid \bar{b}, \bar{m}) = 0.7$
$p(k \mid \bar{b}, m) = 0.2$
$p(k \mid b, \bar{m}) = 0.6$
$p(k \mid b, m) = 0.1$

DAG: Directed Acyclic Graph



"is made by" ↓

$dom(M) = \{m, \bar{m}\}$
$dom(B) = \{b, \bar{b}\}$
$dom(K) = \{k, \bar{k}\}$

$\Rightarrow$ joint probability distribution

$p(K, B, M) = p(K \wedge B \wedge M)$
$= p(K \mid B, M) \cdot \underbrace{p(B) \cdot p(M)}_{\text{independent!}}$

$\Rightarrow$ 3D probability table

$8 = 2^3$ entries

we look for:

$$p(\mathcal{B}=b \mid K=k) = p(b \mid k) = ?$$

↖ no M → need to marginalize m out (average out...)

$$p(b \mid k) \overset{\text{marginalize } m \in \text{dom}(M)}{=} \sum_m p(b, M \mid k)$$

$$= \sum_m \frac{p(b, M, k)}{p(k)} \quad \text{(Bayes, Product rule)}$$

$$= \frac{\sum_m p(k \mid b, M) \cdot p(b) \cdot p(M)}{\underbrace{\sum_m \sum_b p(k \mid B, M) \cdot p(B) \cdot p(M)}_{p(k)}}$$

$$= \frac{p(b) \cdot \sum_m p(k \mid b, M) \cdot p(M)}{\sum_{b \in \text{dom}(B)} p(B) \cdot \sum_{m \in \text{dom}(M)} p(M) \cdot p(k \mid B, M)}$$

☑ Alternatively:

$$p(b \mid k) = \sum_m p(b, M \mid k)$$

$$= \sum_m \frac{p(k \mid b, M)\, p(b)\, p(M)}{p(k)}$$

$$p(k) = \sum_{b \in B} \sum_{m \in M} p(k, b, m) \quad \} \text{ "evidence"}$$

Conditional Gaussian

$$\Delta = -\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})$$

$$\vec{x} = \begin{pmatrix} x_a \\ x_b \end{pmatrix}; \quad \vec{\mu} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

$$\Rightarrow \Delta = -\frac{1}{2}(x_a-\mu_a)^T \Lambda_{aa} (x_a-\mu_a) \quad (1) \text{ quadratic term}$$

$$-\frac{1}{2}(x_b-\mu_b)^T \Lambda_{ba} (x_a-\mu_a) \quad (2)$$

$$-\frac{1}{2}(x_a-\mu_a)^T \Lambda_{ab} (x_b-\mu_b) \quad (3) \quad \Big\} \text{ linear term.}$$

$$- \ldots\ldots \text{ (only dependent on } x_b, \mu_b)$$

$$\Rightarrow \boxed{\Lambda_{aa} = \Sigma_{a|b}^{-1}} \quad \text{or} \quad \boxed{\Sigma_{a|b} = \Lambda_{aa}^{-1}}$$

(2)+(3): $\Rightarrow$ linear in x_a

$$+\frac{1}{2} x_a^T (\Lambda_{aa}\mu_a) + \frac{1}{2}(\mu_a^T \Lambda_{aa} x_a)$$

$$-\frac{1}{2}(x_b-\mu_b)^T \Lambda_{ba} x_a - \frac{1}{2} x_a^T \Lambda_{ab} (x_b-\mu_b)$$

$$= x_a^T \underbrace{[\Lambda_{aa}\mu_a - \Lambda_{ab}(x_b-\mu_b)]}_{\Sigma_{a|b}^{-1}\mu_{a|b} \text{ why?}} \quad (A)$$

$$\frac{1}{2}(x_a - \mu_{a|b})^T \Sigma_{a|b}^{-1} (x_a - \mu_{a|b})$$

$$= \underbrace{x_a^T \Sigma_{a|b}^{-1} x_a}_{\text{quadr.}} - \underbrace{x_a^T \Sigma_{a|b}^{-1} \mu_{a|b}}_{\text{linear}} + \dots$$

$$\Rightarrow \quad \Sigma_{a|b}^{-1} \mu_{a|b} = \Lambda_{aa} \mu_a - \Lambda_{ab}(x_b - \mu_b) \quad (B)$$

$$(A,B) \Rightarrow \boxed{\mu_{a|b} = \Sigma_{a|b}\left[\Lambda_{aa}\mu_a - \Lambda_{ab}(x_b - \mu_b)\right]}$$

OST
Ostschweizer
Fachhochschule

Lab 9 (solution)

FTP MachLe MSE
FS 2022

Gaussian Processes

Machine Learning
WÜRC

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

After this unit, ...

Lernziele/Kompetenzen

- you know the principle of maximum likelihood (`ML`) and and maximum a posteriori probability (`MAP`) and you know their difference.
- you know (K1), that both the *conditionals* $p(x|y)$ and the *marginals* $p(x)$ of a joint Gaussian $p(x, y)$ 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 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.

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

- your are able to formulate *probabilitstic models* that use *priors* to express knowlege (or beliefs) about aspects of the model. You can formulate a *probabilistic model* for a process $f(x, \theta)$ with additive Gaussian noise ε. You can derive the *likelihood function* $p(y|x, \varepsilon, \theta)$ for this model given the parameters θ.
- 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) that the *predictive distribution* which is used for making predictions for unknown data (x^*, y^*) can be calculated by *marginalizing* (integrating or averaging) over the parameter distribution.

$$p(y^*|x^*, x, y) = \int_\theta p(y^*|x^*, x, y, \theta) \cdot p(\theta|x, y) d\theta \tag{2}$$

- 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. Medical Inference (Bayes Theorem) [M,I]

Breast cancer facts:

- 1 % of scanned women have breast cancer
- 80 % of women with breast cancer get positive mammography scans
- 9.6 % of women without breast cancer also get positive mammography scans

Question: A woman gets a scan, and it is positive. what is the probability that she has breast cancer?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) less than 1 %	○	●
b) less than 10 %	●	○
c) around 80 %	○	●
d) around 90 %	○	●

2. Likelihood function, MAP and linear regression [L, II]

a) The emphlikelihood is the probability of each datapoint y_i given the model and its parameters.

$$p(\mathbf{Y}|\mathbf{X}, \theta) \tag{3}$$

The probability of *one* datapoint y_i given the model $\hat{y}(\theta, x_i)$ for a Gaussian noise ε is:

$$p(y_i|x_i, \theta) \sim \mathcal{N}\left(y_i|\hat{y}, \sigma_n^2\right) \tag{4}$$

$$= \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left\{-\frac{1}{2\sigma_n^2}\left(y_i - \hat{y}(x_i, \theta)\right)^2\right\} \tag{5}$$

$$= \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left\{-\frac{1}{2\sigma_n^2}\left(y_i - f(x_i|\theta)\right)^2\right\} \tag{6}$$

The likelihood of all data points is the product of the probabilities of each datapoint (x_i, y_i):

$$p(\mathbf{Y}|\mathbf{X}, \theta) = \prod_{i=1}^{N} p(y_i|x_i, \theta) \tag{7}$$

$$= \frac{1}{(2\pi\sigma_n^2)^{N/2}} \cdot \exp\left\{-\frac{1}{2\sigma_n^2}\sum_{i=1}^{N}\left(y_i - f(x_i|\theta)\right)^2\right\} \tag{8}$$

$$= \frac{1}{(2\pi\sigma_n^2)^{N/2}} \cdot \exp\left\{-\frac{1}{2\sigma_n^2}\left\|\mathbf{Y} - f(\mathbf{X}|\theta)\right\|^2\right\} \tag{9}$$

b) By taking the natural logarithm of (7), the result immediately follows:

$$\log\left[p\left(\mathbf{Y}|\mathbf{X}, \theta\right)\right] = -\frac{1}{2\sigma_n^2}\sum_{i=1}^{N}\left(y_i - f(x_i|\theta)\right)^2 - \frac{N}{2} \cdot \log\left(2\pi\sigma_n^2\right) \tag{10}$$

$$= -\frac{1}{2\sigma_n^2}\left\|\mathbf{Y} - f(\mathbf{X}|\theta)\right\|^2 - \frac{N}{2} \cdot \log\left(2\pi\sigma_n^2\right) \tag{11}$$

$$= -\frac{1}{2\sigma_n^2}\mathrm{SSE}(\theta) - \frac{N}{2} \cdot \log\left(2\pi\sigma_n^2\right) \tag{12}$$

c) In case of a linear model, we can write the sum of the squared error $\mathrm{SSE}(\theta)$ in matrix form:

$$\mathrm{SSE}(\theta) = \left\|\mathbf{Y} - f(\mathbf{X}|\theta)\right\|^2 \tag{13}$$

$$= \left(\mathbf{Y} - f(\mathbf{X}|\theta)\right)^T \cdot \left(\mathbf{Y} - f(\mathbf{X}|\theta)\right) \tag{14}$$

$$= \left(\mathbf{Y} - \Phi \cdot \theta\right)^T \cdot \left(\mathbf{Y} - \Phi \cdot \theta\right) \tag{15}$$

The square norm can always be written in form of a scalar product, so it is sufficient to consider only one term of the scalar product:

$$\left(\mathbf{Y} - f(\mathbf{X}|\theta)\right) = \begin{pmatrix} y_1 - (\theta_1 + \theta_2 x_1) \\ y_2 - (\theta_1 + \theta_2 x_2) \\ \vdots \\ y_N - (\theta_1 + \theta_2 x_N) \end{pmatrix} \tag{16}$$

$$= \left(\mathbf{Y} - [\mathbb{1}_N \, \mathbf{X}] \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \right) \tag{17}$$

$$= (\mathbf{Y} - \Phi \cdot \theta) \tag{18}$$

d) The two following results from matrix calculus are useful. For column vectors $\boldsymbol{\theta}$ and $\mathbf{x}$ of the same length, the following statement is valid:

$$\nabla_{\boldsymbol{\theta}}(\mathbf{a}^T\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^T\mathbf{a}) = \mathbf{a} \tag{19}$$

For a column vector $\boldsymbol{\theta}$ and matrix $\mathbf{A}$, the following identity holds:

$$\nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^T\mathbf{A}\boldsymbol{\theta}) = (\mathbf{A} + \mathbf{A}^T)\boldsymbol{\theta} \tag{20}$$

Especially, if $\mathbf{A}$ is symmetric:

$$\nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^T\mathbf{A}\boldsymbol{\theta}) = 2\mathbf{A}\boldsymbol{\theta} \tag{21}$$

To find the maximum likelihood solution, we set the gradient of the log likelihood function to zero:

$$0 = \nabla_\theta \log\left[p\left(\mathbf{Y}|\mathbf{X},\theta\right)\right] \tag{22}$$

$$= -\frac{1}{2\sigma_n^2}\nabla_{\boldsymbol{\theta}} \left\|\mathbf{Y} - f(\mathbf{X}|\theta)\right\|^2 \tag{23}$$

$$= -\frac{1}{2\sigma_n^2}\nabla_{\boldsymbol{\theta}} \left[(\boldsymbol{\Phi\theta} - \mathbf{Y})^T \cdot (\boldsymbol{\Phi\theta} - \mathbf{Y})\right] \tag{24}$$

$$= -\frac{1}{2\sigma_n^2}\nabla_{\boldsymbol{\theta}} \left[\boldsymbol{\theta^T\Phi^T\Phi\theta} - \boldsymbol{\theta^T\Phi^T}\mathbf{Y} - \mathbf{Y^T\Phi\theta} - \mathbf{Y^TY}\right] \tag{25}$$

$$= -\frac{1}{2\sigma_n^2}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\theta^T\Phi^T\Phi\theta} - 2\nabla_{\boldsymbol{\theta}}\boldsymbol{\theta^T\Phi^T}\mathbf{Y} - 0\right] \tag{26}$$

$$= -\frac{1}{2\sigma_n^2}\left[2\boldsymbol{\Phi^T\Phi\theta} - 2\boldsymbol{\Phi}\mathbf{Y}\right] \tag{27}$$

This leads to the definition of the *Least Squares Normal Equations*:

$$\boldsymbol{\Phi}\mathbf{Y} = \boldsymbol{\Phi^T\Phi\theta} \tag{28}$$

The Least Squares estimate $\hat{\boldsymbol{\theta}}_{\mathrm{ML}}$ of the parameters $\boldsymbol{\theta}$ is then given by:

$$\hat{\boldsymbol{\theta}}_{\mathrm{ML}} = (\boldsymbol{\Phi^T\Phi})^{-1}\boldsymbol{\Phi^T}\mathbf{Y} \tag{29}$$

3. Prior samples and posterior distributions from differnt kernels of a $\mathcal{GP}$ [A,II]
The solution Juypter notebook can be found on moodle:
`Lab9_A3_plot_gpr_prior_posterior.ipynb`

4. Model fitting, prediction and noise estimation using a $\mathcal{GP}$ [A,II]
The solution Juypter notebook can be found on moodle:
`Lab9_A4_FitGPModel_NoiseEstimation_solution.ipynb`

OST
Ostschweizer
Fachhochschule

Lab 10

FTP MachLe MSE
FS 2022

Reinforcement Learning (solutions)

MSE MachLe
WÜRC

Lernziele/Kompetenzen

- You know the definition of a finite *Markov decision process* (MDP) and the definition of a *Markov Reward Process* (MRP).
- You understand the two basic algorithms for iteratively approximating the dynamic programming task for finite MDPs in the special cases of one-step, tabular, modelfree TD (time-difference) methods, namely *value iteration* and *Q-learning*.
- You know the difference between *on-policy* and *off-policy* learning.
- You can explain the difference between *value iteration* and *policy iteration*.
- You know the trade-off between *exploitation* and *exploration*.

---

1. General Questions: Reinforcement Learning [A, II]

a) We can try to think about it in terms of the limitations a finite MDP imposes in a problem definition and whether any approximations could exist within the framework.

(i). *The Markov property*, if not only the previous state and action selected influence which will be the next state. An example of this could be a modified game of chess where the order in which the pieces were moved affects the possible moves and the sequence information wasn't available (or only partially available) at the time of making a decision. Information from the past that can affect the next state would be missing, breaking the Markov property.

(ii). *The action and state sets must be finite*. Any problem with infinite available actions or states would need alternative representations, such as grouping them into subsets and use these sets.

(iii). *Rewards must be numerical*. If the rewards the environment gives back are not numerical we would need to encode them as numbers. This can be a highly difficult task as the rewards may not translate naturally to numbers. For example, if the rewards were your family's verbal feedback on how good the meal you prepared was, it would be difficult to convert it into a number and capture correctly its intensity.

b) This is likely an exploration issue where the agent is unable to find the exit the first time and therefore doesn't know there's anything better than 0 as a reward. Potential solutions include having each non-goal state be worth -1, and/or extending the episode length. This

would mean states the agents visits a lot (particularly around the start) will get worse and worse values so it will want to move away from there and eventually find the goal (essentially reaching the goal stops it being in pain).

c) The code could look like this. Note the inverse indexing.

```
r = [-1, 2, 6, 3, 2]
gamma = 0.5

for g_i in range(len(r)):
  ret = sum([gamma**i * r_i for i, r_i in enumerate(r[g_i:])])
  print("G_"+ str(g_i) + " = " + str(ret))

G_0 = 2.0
G_1 = 6.0
G_2 = 8.0
G_3 = 4.0
G_4 = 2.0
```

d) The discounted cumulative reward can be caluclated as a sum of a geometric sequence using the formula for $|q| < 1$:

$$\sum_{k=0}^{\infty} q^k = \frac{1}{1-q}$$

$$G_0 = 2 + \gamma\ 7 + \gamma^2\ 7 + \cdots = 2 + \gamma\ (\sum_{k=0}^{\infty} \gamma^k\ 7)$$

$$= 2 + \gamma\ 7\ (\sum_{k=0}^{\infty} \gamma^k) = 2 + \gamma\ 7\ (\frac{1}{1-\gamma}) = 2 + 0.9 \times 7 \times 10 = \underline{\underline{65}}$$

$$G_1 = 7 + \gamma\ 7 + \gamma^2\ 7 + \cdots = 7 \cdot (\sum_{k=0}^{\infty} \gamma^k) = 7 \cdot \frac{1}{1-\gamma} = \underline{\underline{70}}$$

e) In general it would play worse. The chance that the correct action for a situation in the long run is the first one that returns a positive reward is pretty small, particularly if there are a large number of actions available. It would also be unable to adapt to opponents that slowly altered behaviour over time.

2. Value Iteration on the Grid-World [A,II]

The Jupyter notebook solution can be found on moodle:
`Lab10_A2_ValueIteration_GridWorld_SOLUTION.jpynb`.

3. Q-Learning on the Grid-World [A,II]

The Jupyter notebook solution can be found on moodle:
`Lab10_A3_QLearning_GridBoard_SOLUTION.jpynb`.

4. Tic Tac Toe (optional) [A,I]

The Jupyter notebook solution can be found on moodle:
`Lab10_A4_TicTacToe.jpynb`.

a) The Jupyter notebook solution can be found on moodle: `Lab10_A3_TicTacToe_solution.jpynb`.

b)

c)

d) For tic-tac-toe it is possible to use 4 axis of symmetry to essentially fold the board down to a quarter of the size This would dramatically increase the speed/reduce the memory required. If the opponent did not take advantage of symmetries then it could result in a worse overall performance, for example, if the opponent always played correct except for 1 corner, then using symmetries would mean you never take advantage of that information. This means symmetrically equivalent positions don't always hold the same value in a multi-player game.

OST
Ostschweizer
Fachhochschule

Lab 11 (solution)

FTP MachLe MSE
FS 2022

Dimensionality Reduction

MSE MachLe
WÜRC

Lernziele/Kompetenzen

- You know the main motivations and applications and drawbacks of *dimensionality reduction* of a high dimensional dataset.
- You can explain by an example what is meant by the term *curse of dimensionality* and the consequences of this fact.
- You can apply *Principal Component Analysis* (**PCA**) to high dimensional datasets and explain how PCA works. You know different versions of PCA like incremental PCA, vanilla PCA, randomized PCA and kernel PCA. You can explain the differences between these methods and know to apply them accordingly using `scikit learn`.
- You know the *kernel trick*, it's advantages and when it can be applied to a given technique. You can list at least four different kernels $k(x, x')$ that are frequently used in Machine Learning.
- You know the four axioms that define a *metrics* and can name four different metrices that are commonly used in Machine Learning.
- You know the manifold hypothesis and different manifold techniques like **MDS** (*Multidimensional scaling*), **LLE** (*local linear embedding*), **t-SNE** (*t-distributed stochastic neighbor embedding*) and **Isomap** (*Isometric mapping*) and can apply them to high dimensional datasets in `scikit learn`.
- You can successfully apply dimensionality reduction techniques for the *visualization* of high dimensional datasets, for *noise reduction* in datasets and for the *elimination of meaningless features used for classification*.

---

1. Applications of dimensionality reduction techniques [A, I]

a) The main motivations for dimensionality reduction are:

- To speed up a subsequent training algorithm (in some cases it may even remove noise and redundant features, making the training algorithm perform better).
- To visualize the data and gain insights on the most important features.
- Simply to save space (compression).
- To reduce noise.

b) The main applications are:

- removal of noise and redundant features for classification tasks
- visualization of high dimensional data
- data compression
- noise reduction, removal

c) The main drawbacks are:

- Some information is lost, possibly degrading the performance of subsequent training algorithms.
- It can be computationally intensive.
- It adds some complexity to your Machine Learning pipelines.
- Transformed features are often hard to interpret.

2. Curse of Dimensionality [A,I]

The *curse of dimensionality* refers to the fact that many problems that do not exist in low-dimensional space arise in high-dimensional space. In Machine Learning, one common manifestation is the fact that randomly sampled highdimensional vectors are generally very sparse, increasing the risk of overfitting and making it very difficult to identify patterns in the data without having plenty of training data.

3. General questions [A,II]

a) Once a dataset's dimensionality has been reduced using one of the algorithms we discussed, it is almost always impossible to perfectly reverse the operation, because some information gets lost during dimensionality reduction. Moreover, while some algorithms (such as PCA) have a simple reverse transformation procedure that can reconstruct a dataset relatively similar to the original, other algorithms (such as t-SNE) do not.

b) For dimensionality reduction of high dimensional, nonlinear datasets, mainfold methods that use the local structure of the dataset should be used:

- *Kernel PCA* is generally well suited in reducing the dimensionality of high dimensional, nonlinear datasets. By applying the kernel trick, a nonlinear mapping is applied to the input data, actually increasing the dimensionality even more. The kernel however can be evaluated in dataspace. The problem complexity is given by the number of data points.
- *Local Linear Embedding* (LLE) reduces dimensionality while trying to preserve the distances between close instances only.
- *Isomap* creates a graph by connecting each instance to its nearest neighbors, then reduces dimensionality while trying to preserve the geodesic distances between the instances.
- *t-Distributed Stochastic Neighbor Embedding* (t-SNE) reduces dimensionality while trying to keep similar instances close and dissimilar instances apart. It is mostly used for visualization, in particular to visualize clusters of instances in high-dimensional space (e.g., to visualize the MNIST images in 2D).

- *Linear Discriminant Analysis* (LDA) is actually a classification algorithm. During training it learns the most discriminative axes between the classes. These axes can be used to define a hyperplane onto which to project the data. The projection will keep classes as far apart as possible, so LDA is a good technique to reduce dimensionality before running another classification algorithm such as an SVM classifier.

c) Intuitively, a dimensionality reduction algorithm performs well if it eliminates a lot of dimensions from the dataset without losing too much information. One way to measure this is to apply the reverse transformation and measure the reconstruction error. However, not all dimensionality reduction algorithms provide a reverse transformation.

Alternatively, if you are using dimensionality reduction as a preprocessing step before another Machine Learning algorithm (e.g., a Random Forest classifier), then you can simply measure the performance of that second algorithm; if dimensionality reduction did not lose too much information, then the algorithm should perform just as well as when using the original dataset.

d) It can absolutely make sense to chain two different dimensionality reduction algorithms. A common example is using PCA to quickly get rid of a large number of useless dimensions, then applying another much slower dimensionality reduction algorithm, such as LLE. This two-step approach will likely yield the same performance as using LLE only, but in a fraction of the time.

e) In which cases would you use incremental PCA, randomized PCA or kernel PCA?

- *Regular PCA* is the default, but it works only if the dataset fits in memory.
- *Incremental PCA* is useful for large datasets that don't fit in memory, but it is slower than regular PCA, so if the dataset fits in memory you should prefer regular PCA. Incremental PCA is also useful for online tasks, when you need to apply PCA on the fly, every time a new instance arrives.
- *Randomized PCA* is useful when you want to considerably reduce dimensionality and the dataset fits in memory; in this case, it is much faster than regular PCA. Finally, Kernel PCA is useful for nonlinear datasets.

4. PCA and scaling importance [A,II]

The Jupyter notebook solution can be found on moodle:
`Lab10_A4_PCA_ScalingImportance_Wine_solution.ipynb`

5. Stochastic Neighbour Embedding [A,II]

The Jupyter notebook solution can be found on moodle:
`Lab11_A5_MNIST_tSNE_solution.ipynb`

6. Feature Engineering using PCA [A,II]

The Jupyter notebook solution can be found on moodle:
`Lab11_A6_MNIST_tSNE_solution.ipynb`

7. Kernels and the Kernel Trick [A, II]

a) The kernel trick avoids the explicit mapping that is needed to get linear learning algorithms to learn a nonlinear function or decision boundary. For all $\mathbf{x}$ and $\mathbf{x}'$ in the input space $\mathcal{X}$, certain functions $k(\mathbf{x}, \mathbf{x}')$ can be expressed as an inner product in another space $\mathcal{V}$. The function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is often referred to as a *kernel* or a *kernel function*. The word 'kernel' is used in mathematics to denote a weighting function for a weighted sum or integral.

Certain problems in machine learning have additional structure than an arbitrary weighting function k. The computation is made much simpler if the kernel can be written in the form of a 'feature map' $\varphi\colon \mathcal{X} \to \mathcal{V}$ which satisfies

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{V}} \tag{1}$$

The key restriction is that $\langle \cdot, \cdot \rangle_{\mathcal{V}}$ must be a proper inner product. On the other hand, an explicit representation for φ is not necessary, as long as $\mathcal{V}$ is an inner product space. The alternative follows from MERCER's theorem: an implicitly defined function φ exists whenever the space $\mathcal{X}$ can be equipped with a suitable measure ensuring the function k satisfies MERCER's condition. MERCER's theorem is similar to a generalization of the result from linear algebra that associates an inner product to any positive-definite matrix.

Some algorithms that depend on arbitrary relationships in the native space $\mathcal{X}$ would, in fact, have a linear interpretation in a different setting: the range space of φ. The linear interpretation gives us insight about the algorithm. Furthermore, there is often no need to compute φ directly during computation, as is the case with *support vector machines*. Some cite this running time shortcut as the primary benefit. Researchers also use it to justify the meanings and properties of existing algorithms.

b) The most common kernels in Machine Learning are:

$$\begin{aligned}
\text{Gaussian}: &\quad k(\mathbf{x}, \mathbf{x}') = \exp\left(-\beta \cdot \|\mathbf{x} - \mathbf{x}'\|^2\right) \\
\text{Laplacian}: &\quad k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \cdot |\mathbf{x} - \mathbf{x}'|_1\right) \\
\text{Polynomial}: &\quad k(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x} \cdot \mathbf{x}^T\right)^p \\
\text{Sigmoid}: &\quad k(\mathbf{x}, \mathbf{x}') = \tanh\left(\alpha \mathbf{x} \cdot \mathbf{x}^T + \delta\right)
\end{aligned}$$

OST
Ostschweizer
Fachhochschule

Lab 12 (solution)

FTP MachLe MSE
FS 2022

Unsupervised Learning: Clustering

Machine Learning
WÜRC

After this unit, ...

Lernziele/Kompetenzen

- you know the *three* clustering algorithms: *k-means*, *dbscan* and a*agglomerative clustering* (using average, complete or Ward linkage).
- you are able to explain the working principle of *k-means*, *dbscan* and *agglomerative clustering*, their advantages and disadvantages and to apply them to data using `scikit-tlearn` in Python.
- you are able to plot the `inertia` and to determine the elbow point of this curve to find an optimum number of clusters as *hyperparameter*.
- you know the way how to *evaluate* a cluster algorithm using *metrics*, namely using `ARI` (*adjusted rand index*), `NMI` (*normalized mutual information*), `SC` (*silhouette score*) and *inertia*.
- you are able to correctly *scale* the data before clustering is applied especially (MinMax, StandardScaler, RobustScaler) or to meaningfully `transform` the data (eg. using `PCA`, `t-SNE` or `NMF`) before a clustering algorithm is applied.
- you know what a *Gaussian Mixture Model* `GMM` is and how the *expectation maximization algorithm* (`EM` algorithm) works. You are able to interpret the `kMeans` algorithm as a form of an `EM` algorithm with an E-step and and M-Step.
- your are able to apply clustering on the faces dataset (agglomerative, k-means and dbscan) to detect and *group* similar faces.
- your are able to apply a *hierarchical cluster analysis* on a voting dataset.

1. Clustering Algorithms [M,I]
This clustering algorithm initially assumes that each data instance represents a single cluster.

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) agglomerative clustering	●	○
b) t-SNE	○	●
c) k-means clustering	○	●
d) expectation maximization	○	●

2. Elbow Curve and `sklearn.cluster.KMeans` [A,I]
The solution Juypter notebook can be found on moodle:
`Lab12_A2_MakeBlobs_kMeans.ipynb`

3. k-Means, Gaussian Mixture Models and the EM algorithm [A,II]
The solution Juypter notebook can be found on moodle:
`Lab12_A3_EM_KMeans_MixtureModels.ipynb`

4. Image compression using `kMeans` [A,II]
The solution Juypter notebook can be found on moodle:
`Lab12_A4_ImageCompressionUsingKMeans.ipynb`

5. Detecting similar faces using `DBSCAN` [A,II]
The solution Juypter notebook can be found on moodle:
`Lab12_A5_DBSCAN_DetectSimilarFaces.ipynb`

OST
Ostschweizer
Fachhochschule

Checkup

FTP MachLe MSE
FS 2022

Exercise Collection (solution)

Machine Learning
WÜRC

1 Short questions (K1)

1. Short Questions [A,II]

a) This is false. A specific classifier (with some fixed model complexity) will be more likely to overfit to noise in the training data when there is less training data, and is therefore more likely to overfit.

b) This is true, if we assume that the data points are i.i.d. A few students pointed out that this might not be the case.

c) This is true. y is linear in α_1, so it can be learned using linear regression.

d) This is false. y is not linear in α_1 and α_2, and no simple transformation will make it linear ($\log\left[x^{\alpha_1} \cdot e^{\alpha_2} + \epsilon_i\right] \neq \alpha_1 \log x_1 + \alpha_2 + \epsilon_i$).

e) In the dual formulation of the SVM, features only appear as dot products which can be represented compactly by kernels.

f) Reject - the *training error* is optimistically biased.

2. Kernel Trick [A,II]

a) The Kernel trick involves kernel functions that can enable in higher-dimension spaces without explicitly calculating the coordinates of points within that dimension: instead, kernel functions compute the inner products between the images of all pairs of data in a feature space. This allows them the very useful attribute of calculating the coordinates of higher dimensions while being computationally cheaper than the explicit calculation of said coordinates. Many algorithms can be expressed in terms of inner products. Using the kernel trick enables us effectively run algorithms in a high-dimensional space with lower-dimensional data.

2 Multiple Choice Questions (K1)

3. Mean and Variance [M,I]

Suppose x is a random variable which is distributed uniformly between 0 and 2. What are the *mean* and *variance* of x?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) $\bar{x} = 1$ and $\mathrm{VAR}(x) = \frac{1}{3}$.	●	○
b) $\bar{x} = 0$ and $\mathrm{VAR}(x) = 1$.	○	●
c) $\bar{x} = 1$ and $\mathrm{VAR}(x) = \frac{\sqrt{3}}{3}$.	○	●
d) $\bar{x} = 1$ and $\mathrm{VAR}(x) = \frac{2}{3}$.	○	●

4. Gaussian Distribution [M,I]

Suppose X and Y are two independent Gaussian random variables. Which of the following variable is Gaussian random variable?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) $4X + 3Y$.	●	○
b) X^2.	○	●
c) $X \cdot Y$.	○	●
d) $$Z = \begin{cases} X & \text{with probability } \frac{1}{2} \\ Y & \text{with probability } \frac{1}{2} \end{cases}$$.	○	●

5. k-fold crossvalidation [M,I]

The numerical complexity of k-fold crossvalidation is ....

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) linear in k.	●	○
b) quadratic in k.	○	●
c) cubic in k.	○	●
d) exponential in k.	○	●

6. Unsupervised Learning [M,I]
Thinking about unsupervised learning, which ones of the following statements are true (multiple choice) ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Differently from partitional graph based algorithms and density estimation algorithms the K-means and the EM algorithm need the number of clusters as an input parameter.	●	○
b) With hierarchical clustering algorithms based on graph theory we obtain a partition of a dataset in K different classes.	●	○
c) The K-Means algorithm obtains a global optimal solution for the partition of a dataset by minimizing the square distance between examples and their nearest centroid.	○	●
d) The EM algorithm assumes that the model of the data comes from a mixture of K n- dimensional probability distributions.	○	●

7. k-NN algorithm, accuracy and computational cost [M,I]
Given a k-NN classifier which ones of the following statements are true (multiple choice) ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) The more examples are used for classifying an example, the higher is the accuracy we obtain.	○	●
b) The more attributes we use to describe the examples the more diffcult it is to obtain high accuracy.	●	○
c) The most costly part of this method is to learn the model.	○	●
d) We can use k-NN for classification and regression.	●	○

8. Overfitting [M,I]
How do you ensure you're not overfitting with a model ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Keep the model simpler: reduce variance by taking into account fewer variables and parameters, thereby removing some of the noise in the training data.	●	○
b) Use cross-validation techniques such as k-fold cross-validation.	○	●
c) Use regularization techniques such as LASSO that penalize certain model parameters if they're likely to cause overfitting.	●	○
d) Generating an ensemble of learners and using a softmax or voting output.	●	○

9. Validation and Learning Curves [M,I]

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) A *validation* curve shows the training error as function of the training data size.	○	●
b) If the *learning* curve shows a large gap between training and test error, the learner most probably suffers from a high *bias* problem.	○	●
c) *Hyperparameter tuning* shoud be done using a *cross-validated grid search* within the inner loop.	●	○
d) In a Bayesian approach, the hyperparameters can be defined by a *prior distribution*. Using Bayes theorem to calculate the posterior, we get a point estimate of the hyperparameters.	○	●

10. Estimation Error [M,I]

Thinking about the estimation error with the training set for a learning method. Which of the following statements are true? (multiple answer)

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) The estimation error of the decision tree built for the training set without pruning on the this trainng data is zero.	●	○
b) The estimation error for the training set for the built 3-KNN classifier with this data is zero.	○	●
c) The estimation error for the training set in the decision tree constructed with this data after the pruning procedure is zero.	○	●
d) The estimation error for the training set for the built 1-KNN classifier with this is zero.	●	○

11. Dimensionality Reduction, Wrappers and Filters [M,I]

Thinking about dimensionality reduction and attribute selection, which ones of the following statements are true (multiple choice) ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) *Wrapper* methods usually provide the best performing feature set for a particular type of model. However, they are computationally intensive since for each subset a new model needs to be trained.	●	○
b) *Filter* methods are independent to the type of predictive model. Common measures are distance metrics, correlation, mutual information, and consistency metrics.	●	○
c) *Wrappers* are feature selection methods that, given a classifer as a performance criteria, search in the space of subset of features (feature combinations) for the minimal one that obtains the higher accuracy.	●	○
d) *Filters* are unsupervised feature selection methods because they use evaluation criteria from the intrinsic connections between features to score a feature subset.	●	○

12. Bias-Variance [M,I]
Adding more basis functions in a linear model ...
(Pick the most probable option.)

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) ... decreases model bias.	●	○
b) ... decreases estimation bias.	○	●
c) ... decreases variance.	○	●
d) ... doesn't affect bias and variance.	○	●

13. Debugging machine learing algorithms [M,I]
Suppose you are using some classifier algorithm on a given training set. The training error is acceptable. However, it makes unacceptably large errors in its predictions on unseen data. What should be tried next?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Get more training samples.	●	○
b) Try larger sets of features, e.g. by generating polynomial features.	○	●
c) Decrease the regularization parameter λ if regularization is used.	○	●
d) Create a meta learner, e.g. a voting or bagging classifier out of several different classifiers or build an ensemble of classifiers.	●	○

14. Regularization for regression [M,I]

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) LASSO regularization uses the L_2 norm of the vector of parameters to be estimated.	○	●
b) LASSO regularization can be used for feature selection, because it forces a spase output where some parameters are set to zero.	●	○
c) Regularization for for any learner can be done by adding a penalty to the cost (loss) function that penalizes too complex models.	●	○
d) The parameter etimates for ridge regression can be calculated directy using an analytical solution.	●	○

15. Regularization [M,I]
Which of the following statements about regularization and the regularization parameter λ are correct ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Using too large a value of λ can cause your hypothesis to underfit the data.	●	○
b) Using too large a value of λ can cause your hypothesis to overfit the data.	○	●
c) Using a very large value of λ cannot hurt the performance of your hypothesis.	○	●
d) None of the above.	○	●

16. Getting struck in local minima [M,I]
How can you prevent K-means algorithm from getting stuck in bad local optima ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Set the same seed value for each run.	○	●
b) Use multiple random initializations.	●	○
c) Taking bootstrap samples of the data and run it several times.	●	○
d) Using K-means++.	●	○

17. Text Features [M,I]
Which of the following techniques can be used for *normalization* in text mining ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) Stemming.	●	○
b) Lemmatization.	●	○
c) Stop Word Removal.	○	●
d) Bag of Words	○	●

18. k-NN classifier [M,I]
Which of the following statements is true for k-NN classifiers ?

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) The classification accuracy is better with larger values of k.	○	●
b) The decision boundary is smoother with smaller values of k.	○	●
c) The decision boundary is linear.	○	●
d) k-NN does not require an explicit training step.	●	○

19. Feature generation for audio signals [M,II]

Welche der folgenden Aussagen sind wahr und welche falsch?	**wahr**	**falsch**
a) MFCC (Mel Frequency Cepstral Coefficients) are generated using filters in the spectral domain with overlapping linearly spaced frequency bands.	○	●
b) The *spectal spread* is the second central moment of the frequency spectrum of a signal.	●	○
c) A time-frequency Fourier transform with overlapping timeframes of approximately 330 ms duration is a good choice for feature generation for speech recognition.	○	●
d) The Fourier or cosine transformed spectrum is called *Cepstrum*. The Cepstrum is calculated in order to decorrelate the features in the MFCC representation.	●	○

3 Exercises (K3)

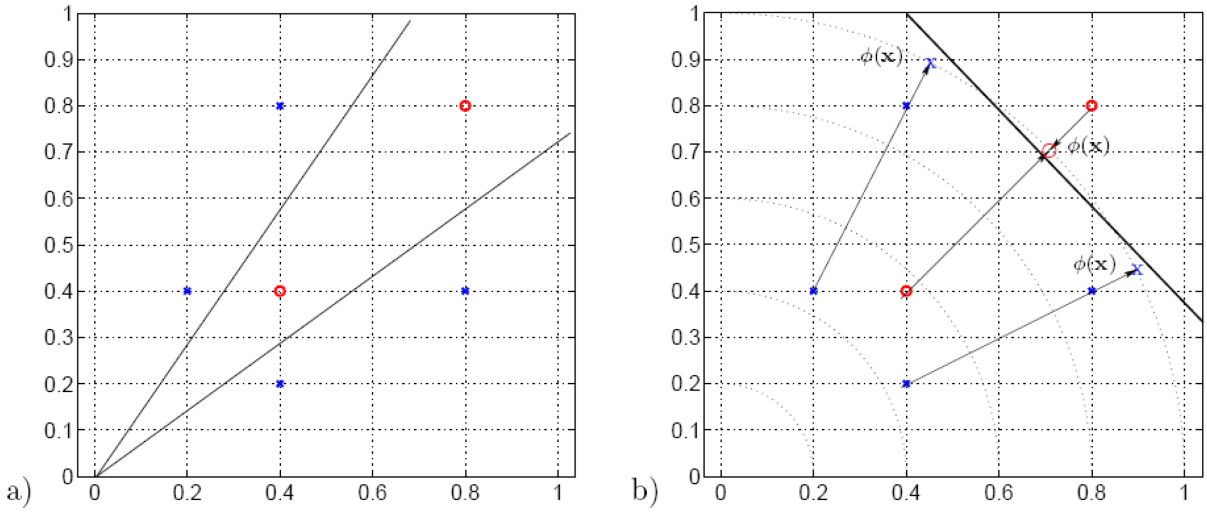
20. Kernel Method [A,II]

a) The feature vector is:

$$\phi(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|} \tag{1}$$

b) The decision boundary is perpendicular to the vector through the two points of class 2. In feature space, this is a straight line with direction $\hat{e} = (-1, 1)^T$. Therefore, the coefficients w_1 and w_2 of the decision boundary are:

$$(w_1, w_2) = (1, 1) \tag{2}$$



c) Three of the five samples act as support vectors - two distinct samples from class 1 and one sample from class 2. After the projection to the unit circle by the feature transform $\phi(x)$, we actually only have three distinct samples which are needed to define the maximum margin hyperplane (in this case of a linear kernel this is a staight line).

d) All data points $\mathbf{x}$ are projected onto the unit circle by $\phi(\mathbf{x})$, i.e. only the unit circle is the feature space. The feature transform ϕ transforms each data point $\mathbf{x}$ to a unit vector. Only the direction of the datapoint is kept. Therefore, a point on the unit circle is transformed back to a line though the origin with the same ploar angle. The only points of the decision boundary $x + y + c = 0$ that are transformed back to the original space are the ones that intersect the unit circle.

21. Manual calculation of the M step for a GMM [A,II]

a) The likelihood is the probability of the data given the model:

$$p(\mathcal{D} \mid \theta) = \prod_{i=1}^{3} \sum_{k=1}^{2} \pi_k \mathcal{N}(x_i \mid \mu_k, \sigma_k)$$

b) The update of the mixing weights is done by a simple mean of the assignments γ_{ik} over all data points for the cluster considered:

$$\pi_k^{(t+1)} = \frac{1}{N} \sum_{i=1}^{3} \gamma_{ik}^{(t)}$$

Therefore, we get for the mixing weights $\pi_k^{(t+1)}$:

$$\pi_1^{(t+1)} = \frac{1}{3} \cdot (1 + \frac{2}{5} + 0) = \frac{1}{3} \cdot \frac{7}{5} = \frac{7}{15}$$
$$\pi_2^{(t+1)} = \frac{1}{3} \cdot (0 + \frac{3}{5} + 1) = \frac{1}{3} \cdot \frac{8}{5} = \frac{8}{15}$$

c) The updated means are

$$\mu_k^{(t+1)} = \frac{\sum_i \gamma_{ik}^{(t)} x_i}{\sum_i \gamma_{ik}^{(t)}}$$

Therefore we get:

$$\mu_1^{(t+1)} = \frac{5}{7} \cdot (1 + \frac{2}{5} \cdot 10 + 20 \cdot 0) = \frac{25}{7}$$
$$\mu_2^{(t+1)} = \frac{5}{8} \cdot (0 + \frac{3}{5} \cdot 10 + 20) = \frac{5}{8} \cdot 26 = \frac{65}{4}$$

22. Box Distribution [A,II]

a) Sketch the PDF of box(L, H) by hand.

b) $\mathbb{E}[X] = \frac{L+H}{2}$

c) $\mathrm{VAR}[X] = \frac{(H-L)^2}{12}$

d) You can't increase L any further because you'd get a log-likelihood of $-\infty$. If you decrease it, you'll just make the height of the box lower and penalize all the log-likelihoods. A similar argument can be used for H.

23. Naive Bayes [A,II]

a) The entropy is:

$$H[5,3] = -\frac{5}{8}\log\frac{5}{8} - \frac{3}{8}\log\frac{3}{8} \tag{3}$$

b) Let us introduce the following abbreviations for the domain of the *binary* features $X = \{X_i\} =$ `IsHeavy, IsSmelly, IsDotted, IsLammelar` and the response y (`IsPoisonous`):

	`true`	`not true`
`IsPoisonous`	y	$\bar{y}$
`IsHeavy`	H	$\bar{H}$
`IsSmelly`	S	$\bar{S}$
`IsDotted`	D	$\bar{D}$
`IsLammelar`	L	$\bar{L}$

To calculate the prediction $p(y \mid X)$ it is best to calculate the *Likelihood Ratio* LR

$$\mathrm{LR} = \frac{p(y \mid X)}{p(\bar{y} \mid X)}$$

If LR > 1 then, according to the *Naive Bayes Classifier*, the mushroom is poisonous, otherwise not. Using the likelihood ratio, we do not need to calculate real probabilities. It is therefore not necessary to calculate the marginal $p(X)$ for the normalization of the likelihood. Assuming that the features X_i are conditionally independent of each other, we can use the Naive Bayes classifier to calculate the probabilities:

$$p(y \mid X) = \frac{1}{p(X)} \cdot \left[\prod_i p(X_i \mid y)\right] \cdot p(y)$$
$$p(\bar{y} \mid X) = \frac{1}{p(X)} \cdot \left[\prod_i p(X_i \mid \bar{y})\right] \cdot p(\bar{y})$$

The best approach is to start with table of the *conditional probabilities* $p(X_i \mid y)$.

$p(H \mid y) = \frac{2}{5}$	$p(\bar{H} \mid y) = \frac{3}{5}$	$p(H \mid \bar{y}) = \frac{1}{3}$	$p(\bar{H} \mid \bar{y}) = \frac{2}{3}$
$p(S \mid y) = \frac{2}{5}$	$p(\bar{S} \mid y) = \frac{3}{5}$	$p(S \mid \bar{y}) = \frac{1}{3}$	$p(\bar{S} \mid \bar{y}) = \frac{2}{3}$
$p(D \mid y) = \frac{2}{5}$	$p(\bar{D} \mid y) = \frac{3}{5}$	$p(D \mid \bar{y}) = \frac{1}{3}$	$p(\bar{D} \mid \bar{y}) = \frac{2}{3}$
$p(L \mid y) = \frac{3}{5}$	$p(\bar{L} \mid y) = \frac{2}{5}$	$p(L \mid \bar{y}) = \frac{1}{3}$	$p(\bar{L} \mid \bar{y}) = \frac{2}{3}$

The priors are $p(y) = \frac{5}{8}$ and $p(\bar{y}) = \frac{3}{8}$.

$$\begin{aligned} p(y \mid U) &= \frac{1}{p(X)} \cdot p(H \mid y) \cdot p(S \mid y) \cdot p(D \mid y) \cdot p(L \mid y) \cdot p(y) \\ &= \frac{1}{p(X)} \cdot \frac{2}{5} \cdot \frac{2}{5} \cdot \frac{2}{5} \cdot \frac{3}{5} \cdot \frac{5}{8} \\ &= \frac{1}{p(X)} \cdot \frac{3}{125} \end{aligned}$$

$$\begin{aligned} p(\bar{y} \mid U) &= \frac{1}{p(X)} \cdot p(H \mid \bar{y}) \cdot p(S \mid \bar{y}) \cdot p(D \mid \bar{y}) \cdot p(L \mid \bar{y}) \cdot p(\bar{y}) \\ &= \frac{1}{p(X)} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{3}{8} \\ &= \frac{1}{p(X)} \cdot \frac{1}{216} \end{aligned}$$

The Likelihood Ratio LR is therefore:

$$\text{LR} = \frac{p(y \mid U)}{p(\bar{y} \mid U)} = \frac{648}{125} \approx 5.184 > 1$$

The prediction $\hat{y}$ for U is `poisonous`.

c)

$$\begin{aligned} p(y \mid V) &= \frac{1}{p(X)} \cdot p(\bar{H} \mid y) \cdot p(S \mid y) \cdot p(\bar{D} \mid y) \cdot p(L \mid y) \cdot p(y) \\ &= \frac{1}{p(X)} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{3}{5} \cdot \frac{3}{5} \cdot \frac{5}{8} \\ &= \frac{1}{p(X)} \cdot \frac{27}{500} \end{aligned}$$

$$\begin{aligned} p(\bar{y} \mid V) &= \frac{1}{p(X)} \cdot p(\bar{H} \mid \bar{y}) \cdot p(S \mid \bar{y}) \cdot p(\bar{D} \mid \bar{y}) \cdot p(L \mid \bar{y}) \cdot p(\bar{y}) \\ &= \frac{1}{p(X)} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{3}{8} \\ &= \frac{1}{p(X)} \cdot \frac{1}{54} \end{aligned}$$

The Likelihood Ratio LR is therefore:

$$\text{LR} = \frac{p(y \mid V)}{p(\bar{y} \mid V)} = \frac{54 \cdot 27}{500} \approx 2.91 > 1$$

The prediction $\hat{y}$ for V is `poisonous`.

$$\begin{aligned} p(y \mid W) &= \frac{1}{p(X)} \cdot p(H \mid y) \cdot p(S \mid y) \cdot p(\bar{D} \mid y) \cdot p(\bar{L} \mid y) \cdot p(y) \\ &= \frac{1}{p(X)} \cdot \frac{2}{5} \cdot \frac{2}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{5}{8} \\ &= \frac{1}{p(X)} \cdot \frac{3}{125} \end{aligned}$$

$$\begin{aligned} p(\bar{y} \mid W) &= \frac{1}{p(X)} \cdot p(H \mid \bar{y}) \cdot p(S \mid \bar{y}) \cdot p(\bar{D} \mid \bar{y}) \cdot p(\bar{L} \mid \bar{y}) \cdot p(\bar{y}) \\ &= \frac{1}{p(X)} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{3}{8} \\ &= \frac{1}{p(X)} \cdot \frac{1}{54} \end{aligned}$$

The Likelihood Ratio LR is therefore:

$$\text{LR} = \frac{p(y \mid W)}{p(\bar{y} \mid W)} = \frac{162}{125} \approx 1.296 > 1$$

The prediction $\hat{y}$ for W is `poisonous`.

d) The root node of a decision tree would be the attribute `IsLammelar`.

24. Inference and Bayesian Networks [A, II]

a)

$$\begin{aligned} P(S) &= \sum_{L_i} \sum_{F_j} P(S \mid L_i, F_j) P(L_i) P(F_j) \\ &= P(S \mid L, F) P(L) P(F) + P(S \mid \bar{L}, F) P(\bar{L}) P(F) \\ &+ P(S \mid L, \bar{F}) P(L) P(\bar{F}) + P(S \mid \bar{L}, \bar{F}) P(\bar{L}) P(\bar{F}) \\ &= 0.4 \cdot 0.6 \cdot 0.8 + 0.6 \cdot 0.6 \cdot 0.5 + 0.4 \cdot 0.4 \cdot 0.6 + 0.6 \cdot 0.4 \cdot 0.3 \\ &= 0.54 \end{aligned}$$

b)

$$\begin{aligned} P(S \mid A) &= \frac{P(S, A)}{P(A)} = \frac{P(A \mid S) P(S)}{P(A \mid S) P(S) + P(A \mid \bar{S}) P(\bar{S})} \\ &= \frac{0.7 \cdot 0.54}{0.7 \cdot 0.54 + 0.3 \cdot (1 - 0.54)} \end{aligned}$$