Machine Learning

Summer Semester 2019, Homework 1

Prof. Dr. J. Peters, H. Abdulsamad, S. Stark, D.Koert

TECHNISCHE UNIVERSITÄT DARMSTADT

Total points: 90 + 10 bonus
Due date: Sunday, 26 May 2019 (before midnight)
Hand in a PDF over Moodle **and** a printed version to the postbox at (S2/02 | E315)

Name, Surname, ID Number **Steffen Schäfer, 2635897** **Peter Nickl, 1941346**

Problem 1.1 Linear Algebra Refresher [20 Points]

a) **Matrix Properties [5 Points]**

A colleague of yours suggests matrix addition and multiplication are similar to scalars, thus commutative, distributive and associative properties can be applied. Prove if matrix addition and multiplication are commutative and associative analytically or give counterexamples. Is matrix multiplication distributive with respect to matrix addition? Again, prove it analytically or give a counterexample. Considering three matrices A, B, C of size $n \times n$.

To answer the questions examples calculated by following matrices will be used:

$$A = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 4 & 3 \end{bmatrix} \quad C = \begin{bmatrix} 2 & 2 \\ 2 & 2. \end{bmatrix}$$

The commutative property for matrix addition states: $A + B = B + A$.

$$A + B = \begin{pmatrix} 3 & -1 \\ 4 & 4 \end{pmatrix} \tag{5}$$

$$B + A = \begin{pmatrix} 3 & -1 \\ 4 & 4 \end{pmatrix} = A + B \tag{6}$$

The commutative property for matrix multiplacation states: $A \cdot B = B \cdot A$

$$A \cdot B = \begin{bmatrix} -2 & -3 \\ 4 & 3 \end{bmatrix} \tag{7}$$

$$B \cdot A = \begin{bmatrix} 2 & -1 \\ 8 & -1 \end{bmatrix} = A \cdot B \tag{8}$$

Thus $A \cdot B \neq B \cdot A$

The distributiv property for matrices states: $A \cdot B + A \cdot C = A(B + C)$

$$\begin{bmatrix} -2 & -3 \\ 04 & 3 \end{bmatrix} + \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 \\ 6 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 \\ 6 & 5 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 6 & 5 \end{bmatrix}$$

b) **Matrix Inversion [6 Points]**

Given the following matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 4 \\ 1 & 4 & 5 \end{pmatrix}$$

analytically compute its inverse A^{-1} and illustrate the steps.
If we change the matrix in

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 4 \\ 1 & 2 & 5 \end{pmatrix}$$

is it still invertible? Why?

First the determinant of the matrix is calculated using the Rule of Sarrus

$$det\,(a) = 10 + 8 + 12 - 16 - 10 - 6 = -2 \neq 0 \tag{22}$$

After that we can calculate the Hauptminors of the matrix

$$det\left(A_{1,1}\right) = -6 \tag{23}$$

$$det\left(A_{1,2}\right) = -1 \tag{24}$$

$$det\left(A_{1,3}\right) = 2 \tag{25}$$

$$det\left(A_{2,1}\right) = -2 \tag{26}$$

$$det\left(A_{2,2}\right) = 2 \tag{27}$$

$$det\left(A_{2,3}\right) = 2 \tag{28}$$

$$det\left(A_{3,1}\right) = 2 \tag{29}$$

$$det\left(A_{3,2}\right) = 1 \tag{30}$$

$$det\left(A_{3,3}\right) = 0 \tag{31}$$

Using the Rule of Cramer

$$x_i = \frac{det\left(A_i\right)}{det\,(A)} \forall i \tag{32}$$

We calculate:

$$A^{-1} = \begin{bmatrix} 3 & -1 & -1 \\ 1/2 & -1 & 1/2 \\ -1 & 1 & 0 \end{bmatrix} \tag{33}$$

If we now change the matrix, than it's not invertible anymore, because

$$det(A) = 10 + 8 + 6 - 8 - 10 - 6 = 24 - 24 = 0 \tag{34}$$

c) **Matrix Pseudoinverse [3 Points]**

Write the definition of the right and left Moore-Penrose pseudoinverse of a generic matrix $A \in \mathbb{R}^{n \times m}$.
Given $A \in \mathbb{R}^{2 \times 3}$, which one does exist? Write down the equation for computing it, specifying the dimensionality of the matrices in the intermediate steps.

Definition of the left Pseudoinverse:

$$J^* J = \left(J^T J\right)^- 1 J^T \cdot J = I_m \tag{39}$$

Definition of the right Pseudoinverse:

$$J J^* = J \cdot J^T \left(J J^T\right)^- 1 \cdot = I_m \tag{40}$$

Given: $A \in \mathfrak{R}^{2x3}$ with $m > n$ which implies full row rank we use the right Pseudoinverse

$$A^* = \underbrace{\underbrace{A^T}_{3x2} \cdot \underbrace{\left(JJ^T\right)^{-1}}_{2x2}}_{3x2} \tag{41}$$

$$\implies A^* \in \mathfrak{R}^{3x2} \tag{42}$$

d) **Eigenvectors & Eigenvalues [6 Points]**

What are eigenvectors and eigenvalues of a matrix A? Briefly explain why they are important in Machine Learning.

In general for Eigenvektors the following equation is true:

$$A \cdot v = \lambda \cdot v \tag{48}$$

Example for calculating the Eigenvalues. We start be calculating the characistic Polynom of the matrix.

$$det\,(A - E \cdot \lambda = 0) \tag{49}$$

$$det \begin{bmatrix} 1-\lambda & 2 & 3 \\ 1 & 2-\lambda 4 & \\ 1 & 4 & 5-\lambda \end{bmatrix} = -\lambda^3 + 8\lambda^2 + 4\lambda - 2 \tag{50}$$

This equation can now be solved to get the Eigenvalues. If the Eigenvalues are now placed back into the Matrix it is possible to calculate the corresponding Eigenvectors.

Eigenvectors and values are Vectors and Scalars for which

$$W \cdot v = \lambda v \tag{51}$$

holds true.

They're defined individually for each Transformation Matrix W. If we now want to change the length of an Eigenvector we don't need to multiply it with W, instead we just multiply it with a corresponding Eigenvalue. Since this is a scalar instead of a Matrix, it saves computational power fo this calculation. Actually any transformation W applied to an vector can be seen as a lineare combination of eigenvectors.

$$\boldsymbol{u} = W v = c_1 \lambda_1 \cdot v_1 + ... + c_n \lambda_n \cdot v_n \tag{52}$$

This again is a huge time saver.

Eigenvalues also tell us about the numerical stability of a Matrix transformation. To achieve a vanishing matrix Eigenvalues of $\leq$ 1are needed. If the Eigenvalues are bigger than one, than the matrix will explode. Ideally the Eigenvalues are all equal to one. The Markov Matrix is one of these matrices and thus used in machine learning.

This knowledge is also used in the orthogonal weight initialization method for Neural Networks.

Problem 1.2 Statistics Refresher [25 Points]

a) **Expectation and Variance [8 Points]**

Let Ω be a finite set and $P : \Omega \to \mathbb{R}$ a probability measure that (by definition) satisfies $P(\omega) \geq 0$ for all $\omega \in \Omega$ and $\sum_{\omega \in \Omega} P(\omega) = 1$. Let $f : \Omega \to \mathbb{R}$ be an arbitrary function on Ω.

1) Write the definition of expectation and variance of f and discuss if they are linear operators.

The expectation is defined as follows:

$$\mathbb{E}_{x\sim p(\mathbf{x})}[f(x)] = \mathbb{E}_x[f] = \mathbb{E}[f] = \begin{cases} \sum_x p(x) f(x) & \textit{discrete case} \\ \int p(x) f(x)\,\mathrm{d}x & \textit{continuous case} \end{cases} \tag{57}$$

The expectation is a linear operator, because for a scalar a and the random variables $\mathbf{x}$ *and* $\mathbf{y}$ *the following statements hold:*

$$\begin{aligned} \mathbb{E}[a\mathbf{x}] &= a\mathbb{E}[\mathbf{x}] \\ \mathbb{E}[\mathbf{x}+\mathbf{y}] &= \mathbb{E}[\mathbf{x}] + \mathbb{E}[\mathbf{y}] \end{aligned} \tag{58}$$

The variance of f is defined as follows:

$$\mathrm{var}[x] = \mathbb{E}\Big[\big(x - \mathbb{E}[x]\big)^2\Big] = \mathbb{E}\big[x^2\big] - \mathbb{E}[x]^2 \tag{59}$$

The variance is not a linear operator, as can be shown by using the definition of the variance:

$$\mathrm{var}[ax] = \mathbb{E}\Big[\big(ax - \mathbb{E}[ax]\big)^2\Big] = \mathbb{E}\Big[(ax)^2\Big] - \mathbb{E}[ax]^2 = a^2\mathbb{E}\big[x^2\big] - a^2\mathbb{E}[x]^2 = a^2\big[\mathrm{var}[x]\big] \tag{60}$$

2) You are given a set of three dices $\{A, B, C\}$. The following table describes the outcome of six rollouts for these dices, where each column shows the outcome of the respective dice. (Note: assume the dices are standard six-sided dices with values between 1-6)

A	4	4	2	4	1	1
B	3	6	3	3	4	3
C	5	5	2	1	1	1

Estimate the expectation and the variance for each dice using unbiased estimators. (Show your computations).

For using the formula for the expectation in the discrete case, one needs to calculate the probabilities of the six outcomes, that each of the three dices can take. This is done by dividing the frequencies of the outcomes by the overall number of rolls, which equals six for all three dices.

	p_1	p_2	p_3	p_4	p_5	p_6
A	$\frac{2}{6}$	$\frac{1}{6}$	0	$\frac{3}{6}$	0	0
B	0	0	$\frac{4}{6}$	$\frac{1}{6}$	0	$\frac{1}{6}$
C	$\frac{3}{6}$	$\frac{1}{6}$	0	0	$\frac{2}{6}$	0

The expectations is calculated with the formula above (discrete case). The following formula for the sample variance is used, which leads to an unbiased estimator:

$$\text{var}[x] = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \mathbb{E}[x]\right)^2 \tag{62}$$

Dice A:

$\mathbb{E}[\mathbf{x}] = \frac{2}{6}\cdot 1 + \frac{1}{6}\cdot 2 + \frac{3}{6}\cdot 4 \approx 2.67$
$\text{var}[x] = \frac{1}{5}[3\cdot\left(4-\frac{8}{3}\right)^2 + \left(2-\frac{8}{3}\right)^2 + 2\cdot\left(1-\frac{8}{3}\right)^2] \approx 2.26$

Dice B:

$\mathbb{E}[\mathbf{x}] = \frac{4}{6}\cdot 3 + \frac{1}{6}\cdot 4 + \frac{1}{6}\cdot 6 \approx 3.67$
$\text{var}[x] = \frac{1}{5}[4\cdot\left(3-\frac{11}{3}\right)^2 + \left(6-\frac{11}{3}\right)^2 + \left(4-\frac{11}{3}\right)^2] \approx 1.46$

Dice C:

$\mathbb{E}[\mathbf{x}] = \frac{3}{6}\cdot 1 + \frac{1}{6}\cdot 2 + \frac{2}{6}\cdot 5 = 2.5$
$\text{var}[x] = \frac{1}{5}[3\cdot\left(1-\frac{15}{6}\right)^2 + \left(2-\frac{15}{6}\right)^2 + 2\cdot\left(5-\frac{15}{6}\right)^2] \approx 3.9$

3) According to the data, which of them is the "most rigged"? Why?

One can use the entropy to calculate the information content of a probability distribution. The information content of a probability distribution is low, if most of the probability mass is centered around a few values. We consider the extreme case, that we sample a value from a distribution, where the probability mass is centered at exactly one value. If we sample from such a distribution, the outcome is already known before sampling, the information content and thus the entropy are low (zero in this example).
On the other hand, if we consider a uniform distribution, the outcome of a sampling procedure is very uncertain. The probability distribution has a high content of information and thus has a high entropy. In the case of a perfect dice, we would expect the entropy to be very high, since all six outcome occur exactly with the same probability (uniformly distributed) and the outcome of one sample is highly uncertain.
Based upon these thoughts we can evaluate, which dice is "most rigged" by simply calculating the entropies of the empirical distribution of the three dices. The dice with the highest entropy is closest to a perfect dice. The dice with the lowest entropy can be considered as "most rigged".

One criticism of this procedure is, that we only have six outcomes for each of the dices, which is a very small sample size. Nevertheless, we will approach the problem as discussed.
The formular for the entropy is:

$$H\left(p\right) = \mathbb{E}[h] = \sum_i p_i h\left(p_i\right) = -\sum_i p_i \log_2 p_i \tag{64}$$

The results for the entropy of the three dices A, B and C are:
$H_A\left(p\right) = -\frac{2}{6}\log_2\left(\frac{2}{6}\right) - \frac{1}{6}\log_2\left(\frac{1}{6}\right) - \frac{3}{6}\log_2\left(\frac{3}{6}\right) \approx 1.4591$
$H_B\left(p\right) = -\frac{4}{6}\log_2\left(\frac{4}{6}\right) - \frac{1}{6}\log_2\left(\frac{1}{6}\right) - \frac{1}{6}\log_2\left(\frac{1}{6}\right) \approx 1.252$
$H_C\left(p\right) = -\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{1}{6}\log_2\left(\frac{1}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right) \approx 1.4591$

The entropy of dice B is the lowest. For that reason we can conclude, that based on the sample dice B is deviating the most from the uniform distribution of a perfect dice and thus can be seen as "most rigged".

b) **It is a Cold World [7 Points]**

Consider the following three statements:
a) A person with a cold has backpain 24% of the time.
b) 5% of the world population has a cold.
c) 12% of those who do not have a cold, still have backpain.

1) Identify random variables from the statements above and define a unique symbol for each of them.

A Random Variable is a set of possible values from a random experiment. There are two random variables in this exercise, which are binary variables, that only can take two different values.
The first random variable is $C =$ "having a cold".
The second random variable is $B =$ "having backpain".

2) Define the domain of each random variable.

As both random variables are binary variables, their domain is limited to two values, which we define as 0 *and* 1. *The outcome* 1 *stands for "having a cold" and "having backpain", respectively. The outcome* 0 *stands for "not having a cold" and "not having backpain". More formally the domains are defined as follows:*

$dom(C) = \{0, 1\}$
$dom(B) = \{0, 1\}$

3) Represent the three statements above with your random variables.

Statement a) $P\big(B = 1 \mid C = 1\big) = 0.24$
Statement b) $P\big(C = 1\big) = 0.05$
Statement c) $P\big(B = 1 \mid C = 0\big) = 0.12$

4) If you suffer from backpain, what are the chances that you suffer from a cold? (Show all the intermediate steps.)

To answer this question we use Bayes' formula.

$$P\big(C = 1 \mid B = 1\big) = \frac{P\big(B = 1 \mid C = 1\big)P\big(C = 1\big)}{P\big(B = 1\big)} = \frac{P\big(B = 1 \mid C = 1\big)P\big(C = 1\big)}{P\big(B = 1 \mid C = 1\big)P\big(C = 1\big) + P\big(B = 1 \mid C = 0\big)P\big(C = 0\big)} =$$
$$= \frac{P\big(B = 1 \mid C = 1\big)P\big(C = 1\big)}{P\big(B = 1 \mid C = 1\big)P\big(C = 1\big) + P\big(B = 1 \mid C = 0\big)\big(1 - P\big(C = 1\big)\big)} =$$
$$= \frac{0.24 \cdot 0.05}{0.24 \cdot 0.05 + 0.12 \cdot (1 - 0.05)} = \frac{2}{21} \approx 0.0952$$

c) **Journey to THX1138 [10 Points]**

After the success of the Rosetta mission, ESA decided to send a spaceship to rendezvous with the comet THX1138. This spacecraft consists of four independent subsystems A, B, C, D. Each subsystem has a probability of failing during the journey equal to $1/3$.
1) What is the probability of the spacecraft S to be in working condition (i.e., all subsystems are operational at the same time) at the rendezvous?

There are five random variables in this exercise, which are binary variables, that only can take two different values.
The first random variable is A = "subsystem A operational".
The second random variable is B = "subsystem B operational".
The third random variable is C = "subsystem C operational".
The forth random variable is D = "subsystem D operational".
The fifth random variable is S = "spacecraft S operational".
$dom(A) = dom(B) = dom(C) = dom(D) = dom(S) = \{0, 1\}$*, whereas 0 stands for "not operational" and 1 stands for "operational".*

From the exercise it is given, that $P(A=0) = P(B=0) = P(C=0) = P(D=0) = \frac{1}{3}$
As probalities have to sum to 1, we can calculate the complementary probability easily: $P(A=1) = P(B=1) = P(C=1) = P(D=1) = 1 - \frac{1}{3} = \frac{2}{3}$

By using the independece of the four subsystems A, B, C and D we can calculate $P(S=1)$ *as follows:*
$P(S=1) = P(A=1, B=1, C=1, D=1) = P(A=1)P(B=1)P(C=1)P(D=1) = \left(\frac{2}{3}\right)^4 = \frac{16}{81} \approx 19.75\%$

2) Given that the spacecraft S is not operating properly, compute analytically the probability that **only** subsystem A has failed.

By using definition of conditional dependency, the chain rule of probabilities, the independency assumption of the subsystems A,B,C and D and the known complementary probability of $P(S=0)$ *we can calculate the probability of the event that only subsystem A has failed given that the spacecraft S is not operating properly. From the text we know that the spacecraft is not operating properly, if at least one subsystem is failing, so:*
$P\left(S=0 \mid A=0, B=1, C=1, D=1\right) = P\left(S=0 \mid A=1, B=0, C=1, D=1\right) =$
$= P\left(S=0 \mid A=1, B=1, C=0, D=1\right) = P\left(S=0 \mid A=1, B=1, C=1, D=0\right) = 1$

Using all the information we can obtain:

$$P\left(A=0, B=1, C=1, D=1 \mid S=0\right) = \frac{P(A=0, B=1, C=1, D=1, S=0)}{P(S=0)} =$$
$$= \frac{P\left(S=0 \mid A=0, B=1, C=1, D=1\right) P(A=0, B=1, C=1, D=1)}{1 - P(S=1)} =$$
$$= \frac{1 \cdot P(A=0)P(B=1)P(C=1)P(D=1)}{1 - P(S=1)} =$$
$$= \frac{1 \cdot \frac{1}{3} \cdot \left(\frac{2}{3}\right)^3}{1 - \frac{16}{81}} = \frac{8}{65} \approx 0.123$$

3) Instead of computing the probability analytically, do a simple simulation experiment and compare the result to the previous solution. Include a snippet of your code.

If we set the number of simulation iterations sufficiently high (e.g. 100000) our Monte Carlo simulation yields the same result as analytically calculating the probability, with an accuracy of three decimals. We can reproduce a probability of $P\left(A=0, B=1, C=1, D=1 \mid S=0\right) \approx 0.123$.

```
import numpy as np

iterations = 100000
count_a_failing = 0
count_none_failing = 0

# simulate joint probability A = S = 0, B = C = D = 1
for i in range(iterations):

    subsystem_a = np.random.binomial(1, [2/3.], size=1)
    subsystem_b = np.random.binomial(1, [2/3.], size=1)
    subsystem_c = np.random.binomial(1, [2/3.], size=1)
    subsystem_d = np.random.binomial(1, [2/3.], size=1)

    if subsystem_a != 1 & subsystem_b == 1 & subsystem_c == 1 & subsystem_d == 1:
        count_a_failing += 1

value_a = (count_a_failing / iterations) # joint probability A = S = 0, B = C = D = 1

# simulate probability of S = 1
for i in range(iterations):

    subsystem_a = np.random.binomial(1, [2/3.], size=1)
    subsystem_b = np.random.binomial(1, [2/3.], size=1)
    subsystem_c = np.random.binomial(1, [2/3.], size=1)
    subsystem_d = np.random.binomial(1, [2/3.], size=1)

    if subsystem_a == 1 & subsystem_b == 1 & subsystem_c == 1 & subsystem_d == 1:
        count_none_failing += 1

value_b = (count_none_failing / iterations) # probability of S = 1

# calculate probability of A = 0, B = C = D = 1 given S = 0
value_c = value_a / ( 1 - value_b )
print('Probability of A = 0, B = C = D = 1 given S = 0:', value_c)
```

4) An improved spacecraft version has been designed. The new spacecraft fails if the critical subsystem A fails, or any two subsystems of the remaining B, C, D fail. What is the probability that **only** subsystem A has failed, given that the spacecraft S is failing?

We use the definition of conditional dependency, the chain rule of probabilities, the independency assumption of the subsystems A,B,C and D to calculate the probability of the event that only subsystem A has failed given that the spacecraft S is not operating properly.

The calculation steps are similar to task 2), but we need to calculate a new value for P(S=1) for the improved spacecraft design.

$$P\left(A=0,B=1,C=1,D=1 \mid S=0\right)=\frac{P\left(A=0,B=1,C=1,D=1,S=0\right)}{P\left(S=0\right)}=$$
$$=\frac{P\left(S=0 \mid A=0,B=1,C=1,D=1\right)P\left(A=0,B=1,C=1,D=1\right)}{1-P\left(S=1\right)}=$$
$$=\frac{1\cdot P\left(A=0\right)P\left(B=1\right)P\left(C=1\right)P\left(D=1\right)}{1-P\left(S=1\right)}$$

From the task description we know that the improved spacecraft is operational, given that A does not fail and at most one of the subsystems B, C or D fail:
$P\left(S=1 \mid A=1,B=1,C=1,D=1\right)=P\left(S=1 \mid A=1,B=0,C=1,D=1\right)=P\left(S=1 \mid A=1,B=1,C=0,D=1\right)=$
$P\left(S=1 \mid A=1,B=1,C=1,D=0\right)=1.$

The new probability P(S=1) for the improved spacecraft design being operational can be calculated as follows:

$$\begin{aligned}P\left(S=1\right)= \quad & P\left(S=1,A=1,B=1,C=1,D=1\right)+\\ & +P\left(S=1,A=1,B=0,C=1,D=1\right)+\\ & +P\left(S=1,A=1,B=1,C=0,D=1\right)+\\ & +P\left(S=1,A=1,B=1,C=1,D=0\right)=\\ & =P\left(S=1 \mid A=1,B=1,C=1,D=1\right)P\left(A=1,B=1,C=1,D=1\right)+\\ & +P\left(S=1 \mid A=1,B=0,C=1,D=1\right)P\left(A=1,B=1,C=1,D=1\right)+\\ & +P\left(S=1 \mid A=1,B=1,C=0,D=1\right)P\left(A=1,B=1,C=0,D=1\right)+\\ & +P\left(S=1 \mid A=1,B=1,C=1,D=0\right)P\left(A=1,B=1,C=1,D=0\right)=\\ & =1\cdot P\left(A=1\right)P\left(B=1\right)P\left(C=1\right)P\left(D=1\right)+\\ & +1\cdot P\left(A=1\right)P\left(B=0\right)P\left(C=1\right)P\left(D=1\right)+\\ & +1\cdot P\left(A=1\right)P\left(B=1\right)P\left(C=0\right)P\left(D=1\right)+\\ & +1\cdot P\left(A=1\right)P\left(B=1\right)P\left(C=1\right)P\left(D=0\right)=\\ & =1\cdot\left(\frac{2}{3}\right)^4+3\cdot 1\cdot\frac{1}{3}\cdot\left(\frac{2}{3}\right)^3=\frac{40}{81}\end{aligned}$$

We obtain an overall solution of:

$$P\left(A=0,B=1,C=1,D=1 \mid S=0\right)=\frac{1\cdot\frac{1}{3}\cdot\left(\frac{2}{3}\right)^3}{1-\frac{40}{81}}=\frac{8}{41}\approx 0.1951$$

Problem 1.3 Optimization and Information Theory [45 Points + 10 Bonus]

a) **Entropy [5 Points]**

You work for a telecommunication company that uses a system to transmit four different symbols S_1, S_2, S_3, S_4 through time. In the current system, each symbol has a probability to occur according to the following table

	S_1	S_2	S_3	S_4
p_i	0.05	0.61	0.27	0.07

Compute the entropy of the system and write the minimum number of bits requires for transmission.

The minimum number of bits for transmitting the four different symbols is $N = \lceil \log_2 4 \rceil = 2$ *bits.*
We calculate the entropy of the system to be:

$$H(p) = \mathbb{E}[h] = \sum_i p_i h(p_i) = -\sum_i p_i \log_2 p_i =$$
$$= -0.05 \cdot \log_2(0.05) - 0.61 \cdot \log_2(0.61) - 0.27 \cdot \log_2(0.27) - 0.07 \cdot \log_2(0.07) \approx 1.43$$

b) **Constrained Optimization [25 Points]**

After an upgrade of the system, your boss asks you to change the probabilities of transmission in order to maximize the entropy. However, the new system has the following constraint

$$4 = \sum_{i=1}^{4} 2 p_i i.$$

1) Formulate it as a constrained optimization problem. Do you need to include additional constrains beside the one above?

The constrained optimization problem can be formulated as follows:

$$\begin{aligned} Max\ H\left(p_i\right) = & \sum_{i=1}^{4} -p_i \log_2\left(p_i\right) \\ s.t.\ & \sum_{i=1}^{4} 2 p_i i - 4 = 0 \\ & \sum_{i=1}^{4} p_i - 1 = 0 \\ & p_i \geq 0 \qquad \forall i = 1, ..., 4 \end{aligned}$$

The last twi constraints are added, in order to enforce the basic rules of probability. The four probabilities have to sum to 1 and each individual probability needs to be greater or equal zero.

2) Write down the Lagrangian of the problem. Use one Lagrangian multiplier per constraint.

$$\mathscr{L}\left(p_i, \lambda_1, \lambda_2\right) = \sum_{i=1}^{4} -p_i \log_2\left(p_i\right) + \lambda_1 \left(\sum_{i=1}^{4} 2 p_i i - 4 \right) + \lambda_2 \left(\sum_{i=1}^{4} p_i - 1 \right)$$

The non-negativity constraint $p_i \geq 0 \quad \forall i = 1, ..., 4$ is not included in the Lagrangian, because the the $\log_2\left(p_i\right)$ can not take a negative argument for p_i. The non-negativity is enforced by the objective function.

3) Compute the partial derivatives of the Lagrangian above for each multiplier and the objective variable. Is it easy to solve it analytically?

$$\begin{aligned} & \frac{\partial \mathscr{L}}{\partial p_i} = \log_2\left(p_i\right) - \frac{1}{\log(2)} + 2 i \lambda_1 + \lambda_2 = 0 \\ & \rightarrow p_i^* = e^{-1} 2^{2\lambda_1 i + \lambda_2} \end{aligned}$$

$$\begin{aligned} & \frac{\partial \mathscr{L}}{\partial \lambda_1} = \sum_{i=1}^{4} 2 p_i i - 4 = 0 \\ & \frac{\partial \mathscr{L}}{\partial \lambda_2} = \sum_{i=1}^{4} p_i - 1 = 0 \end{aligned}$$

The problem is hard to solve analytically. We need to use the dual function to solve it.

4) Formulate the dual function of this constrained optimization problem. Solve it analytically.

The dual function is obtainned by plugging p_i^* into the Lagrangian. We use the fact, that $\sum_{i=1}^4 p_i^* = 1$ for simplifying the dual function.

$$\begin{aligned}
\mathcal{G}\left(p_i^*, \lambda_1, \lambda_2\right) &= \sum_{i=1}^{4} -p_i^* \log_2\left(p_i^*\right) + \lambda_1\left(\sum_{i=1}^{4} 2p_i^* i - 4\right) + \lambda_2\left(\sum_{i=1}^{4} p_i^* - 1\right) = \\
&= \sum_{i=1}^{4} -p_i^*[-1 + 2i\lambda_1 + \lambda_2] + \lambda_1\left(\sum_{i=1}^{4} 2p_i i - 4\right) = \\
&= \sum_{i=1}^{4} -p_i^*[-1 + \lambda_2] - 4\lambda_1 = \\
&= \sum_{i=1}^{4} p_i^* - \lambda_2 - 4\lambda_1 = \\
&= \sum_{i=1}^{4} e^{-1} 2^{\lambda_2} 2^{2\lambda_1 i} - \lambda_2 - 4\lambda_1
\end{aligned}$$

We take the partial derivatives of the dual function, in order to calculate the two Langrangian multipliers and subsequently plugging them back into p_i^*.

$$\begin{aligned}
\frac{\partial \mathcal{G}}{\partial \lambda_1} &= \frac{2}{e} 2^{\lambda_2} \sum_{i=1}^{4} i 2^{2\lambda_1 i} - 4 = 0 \quad \rightarrow \quad a\sum_{i=1}^{4} i b^i - 2 = 0 \\
\frac{\partial \mathcal{G}}{\partial \lambda_2} &= \frac{1}{e} 2^{\lambda_2} \sum_{i=1}^{4} 2^{2\lambda_1 i} - 1 = 0 \quad \rightarrow \quad a\sum_{i=1}^{4} b^i - 1 = 0
\end{aligned}$$

We here used the substitutions $a = \frac{1}{e} 2^{\lambda_2}$ and $b = 2^{2\lambda_1}$.
by writing out the sums we obtain the following equation system for a and b:

$$\begin{aligned}
a\left(b + 2b^2 + 3b^3 + 4b^4\right) - 2 = 0 \\
a\left(b + b^2 + b^3 + b^4\right) - 1 = 0
\end{aligned}$$

Solving the equation system leads to a cubic polynomial for b:

$$-1 + b^2 + 2b^3 = 0$$

This polynomial can be solved analytically by using the following general formula for cubic equations of the form $ax^3 + bx^2 + c^x + d = 0$. (see figure 1)
By solving for the two unknowns we obtain the values of $a \approx 0.6410$ and $b \approx 0.6573$.
With a and b we can calculate the Langrangian multipliers to $\lambda_1 = -0.3027$ and $\lambda_2 = 0.8011$. Plugging into $p_i^* = e^{-1} 2^{2\lambda_1 i + \lambda_2}$ we can finally obtain the result to:

$p_1^* = e^{-1} 2^{0.8011 - 2 \cdot 1 \cdot 0.3027} = 0.4213$
$p_2^* = e^{-1} 2^{0.8011 - 2 \cdot 2 \cdot 0.3027} = 0.2770$
$p_3^* = e^{-1} 2^{0.8011 - 2 \cdot 3 \cdot 0.3027} = 0.1820$
$p_4^* = e^{-1} 2^{0.8011 - 2 \cdot 4 \cdot 0.3027} = 0.1197$

5) Name one technique for numerically solve these problems and briefly describe it.

$$x = \sqrt[3]{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)+\sqrt{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)^2+\left(\frac{c}{3a}-\frac{b^2}{9a^2}\right)^3}} + \sqrt[3]{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)-\sqrt{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)^2+\left(\frac{c}{3a}-\frac{b^2}{9a^2}\right)^3}} - \frac{b}{3a}$$

Figure 1: Formula for calculating the solutions of a cubic equation. [Source: https://math.vanderbilt.edu/schectex/courses/cubic/]

One method to solve constrained optimization problems numerically is the penalty method, which uses a penalty function. The goal of penalty functions is to convert constrained problems into unconstrained problems by introducing an artificial penalty for violating the constraint. This artificial penalty is added to the objective function. Now a method for unconstrained problems, like gradient descent, can be used for optimization.

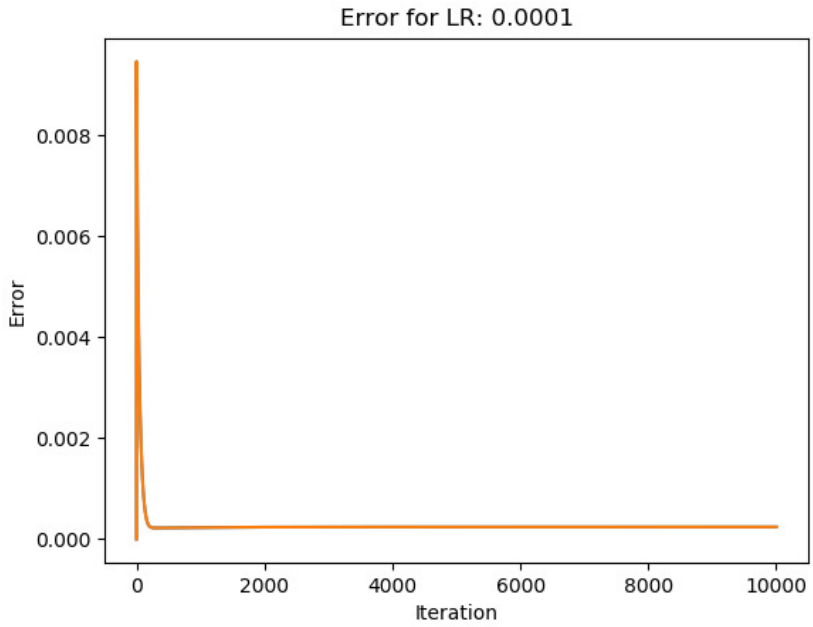

Figure 2: best learning rate was 0.0001

c) **Numerical Optimization [10 Points]**

Rosenbrock's function (to be minimized) is defined as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right].$$

Write in Python a simple gradient descent algorithm and simulate it for 10,000 steps on Rosenbrock's function with $n = 20$. Attach a snippet of your algorithm, discuss the effects of the learning rate and attach a plot of your learning curve with your best learning rate.

Choosing the right learning rate is tricky. A learning rate, which is too high, results in numerical instability and exploding gradients. A learning rate, which is too low, slows down convergence siginifigantly. From all tested cases, a learning rate of 0.0001 worked best for the given Rosenbrock function and out initialization (see figure for the learning curve). The initializations of all x were drawn from a gaussian with mean 0 *and variance* 0.1. *Every learning rate was tested with many initializations, in order to average out over the intializations and to find the overall best learning rate.*

```
import numpy as np
import matplotlib.pyplot as plt

def rosen(x):
    return sum(100.0*(x[1:]-x[:-1]**2.0)**2.0 + (1-x[:-1])**2.0)

def rosen_der(x):
    """
        Gradient of the Rosenbrock function of for gradient descent.
    """
    xm = x[1:-1]
    xm_m1 = x[:-2]
    xm_p1 = x[2:]
    der = np.zeros_like(x)
    der[1:-1] = 200*(xm-xm_m1**2) - 400*(xm_p1 - xm**2)*xm - 2*(1-xm)
    der[0] = -400*x[0]*(x[1]-x[0]**2) - 2*(1-x[0])
    der[-1] = 200*(x[-1]-x[-2]**2)
    return der

cur_x = np.random.normal(0, 0.1, 20)  # Start point randomly drawn from gausian mu = 0, sigma
learning_rate = 0.0001  # Learning rate

max_iteration = 10000  # maximum number of iterations
history = np.zeros((max_iteration+1, 2))

for k in range(max_iteration):
    prev_x = cur_x  # Store current x value in prev_x
    cur_x = prev_x - learning_rate * rosen_der(prev_x)  # Grad descent
    error = abs(cur_x - prev_x)
    k = k + 1  # iteration count

    history[k, :] = np.linalg.norm(error)

    # print("Iteration", iteration, "\nX value is", cur_x)  # Print iterations
    print("␣Iteration:␣", k, "␣\n")
    print("␣Gradient:␣", rosen_der(prev_x), "\n")
    print("␣Error:␣", error, "\n\n")

print("The␣local␣minimum␣occurs␣at", cur_x)


plt.plot(history)
plt.title("Error␣for␣LR:␣0.0001")
plt.ylabel("Error")
plt.xlabel("Iteration")
plt.show()
```

d) **Gradient Descent Variants [5 Points]**

Throughout this class we have seen that gradient descent is one of the most used optimization techniques in Machine Learning. This question asks you to deepen the topic by conducting some research by yourself.

1) There are several variants of gradient descent, namely *batch, stochastic* and *mini-batch*. Each variant differs in how much data we use to compute the gradient of the objective function. Discuss the differences among them, pointing out pros and cons of each one.

2) Many gradient descent optimization algorithms use the so-called *momentum* to improve convergence. What is it? Is it always useful?

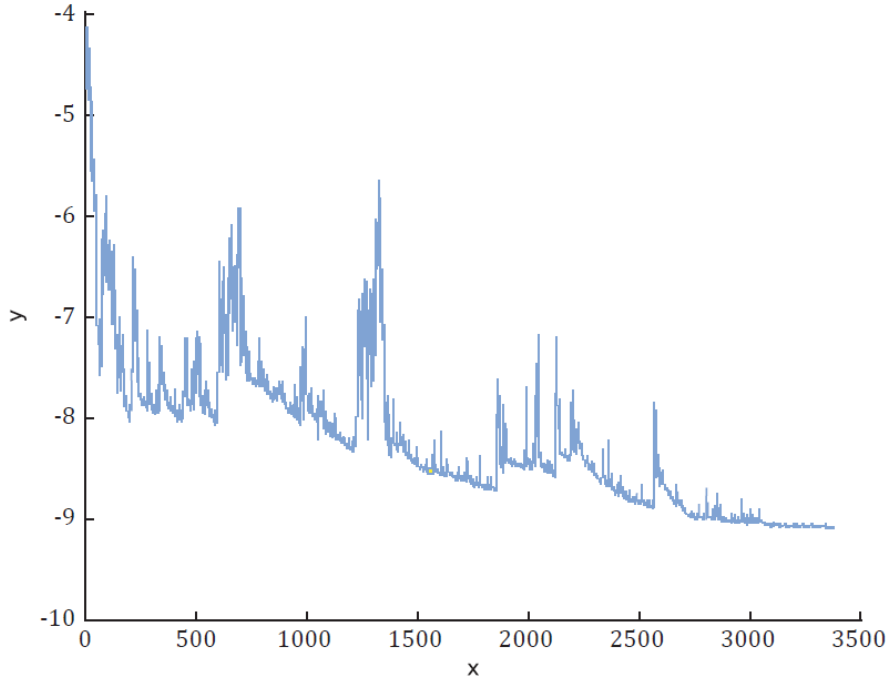

Figure 3: Example for fluctuating cost function for a neural network when using SGD [Source: Me]

Batch gradient descent *in machine learning calculates the gradient using the whole dataset. For this the computer needs to have the whole dataset in memory, which is not always doable. Because of that, its also very time consuming. It converts against the global minima of convex functions and local minima of non convex functions.*

Stochastic gradient descent *calculates the gradient for a single training example. Because of this stochastic gradient descent learns a lot faster than batch gradient descent, however its variance is very high, resulting in a highly fluctuating cost function.*

Mini-batch gradient descent *can be seen as a combination of stochastic gradient descent and batch gradient descent, using mini-batches containing n training examples. This results in reduced variance and more stable conversion rate. When mini-batch gradient descent is calculated on G-PUS, it is possible to reduce the workload by using matrix operations.*

Momentum *is often used to help the gradient descent algorithm to find the global Minimum.*

Without Momentum there is a high possibility, that gradient descent starts oscillating around a local minimum. In order to reach the global minimum, this local one needs to be "jumped over". This can be interpeted as a physical impuls giving a downhill rolling sphere an impuls of energy, to help it to overcome obstacles. This "impuls" is usually calculated by looking at the last gradient and adding a part of it to the current gradient. This means, that if the last update to our parameters was big, we guess that the next one will be big, too.

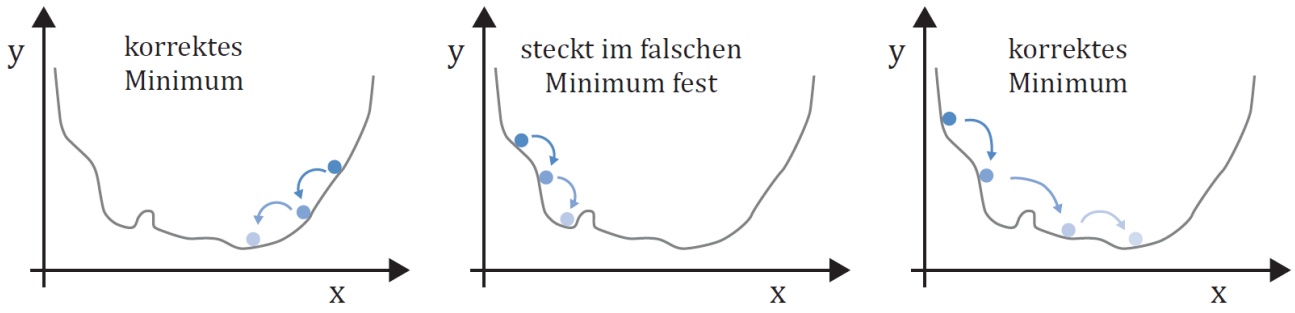

Figure 4: Physically interpretation of Momentum [Source: Me]

e) **Natural Gradient [10 Bonus Points]**

Let $\theta \in \mathbb{R}^n$ be a parameter vector and $J : \mathbb{R}^n \to \mathbb{R}$ a cost function. The negative gradient $-\nabla J(\theta)$ is sometimes called the *steepest descent direction*. But is it really? To be able to claim that it is *the* steepest descent direction, we should compare it to other descent directions and pinpoint what is so unique about the negative gradient direction.

Covariant gradient. A fair way to compare descent directions is to make a small step of fixed length, say ε, in every direction $\Delta\theta$ and check which direction leads to the greatest decrease in $J(\theta)$. Since we assume that the step size is small, we can evaluate the decrease in $J(\theta)$ using its first-order Taylor approximation

$$J(\theta + \Delta\theta) - J(\theta) \approx \nabla J(\theta)^T \Delta\theta.$$

To make precise what we mean by *small* step size, we need to introduce a norm (or a distance) in the space of parameters θ. A good choice, that among other advantages captures the intuition that some parameters may influence the objective function more than others, is the generic quadratic norm

$$\|\Delta\theta\|^2 = \frac{1}{2}\Delta\theta^T F(\theta)\Delta\theta$$

with a positive-definite matrix $F(\theta)$; note that in general F may depend on θ.

1) Find the direction $\Delta\theta$ that yields the largest decrease in the linear approximation of $J(\theta)$ for a fixed step size ε. Does this direction coincide with $-\nabla J(\theta)$? The direction that you found is known as the negative covariant gradient direction.

We first fomalize the optimization problem:

$$\begin{aligned} \max J(\theta + \Delta\theta) - J(\theta) &= \nabla J(\theta)^T \Delta\theta \\ s.t. \frac{1}{2}\Delta\theta^T F(\theta)\Delta\theta &= \varepsilon \end{aligned}$$

For solving this constrained optimization problem we again use the Lagrange method. We set up the Lagrangian, calculate the partial derivative of $\mathcal{L}$ for $\Delta(\theta)$ and solve for $\Delta(\theta)^*$. The last step for solving for $\Delta(\theta)^*$ is due to the symmetry of $F(\theta)$, which follows from positive definiteness of the matrix.

$$\begin{aligned} \mathcal{L}(\Delta\theta, \lambda) &= \Delta J(\theta)^T \Delta\theta + \lambda\left(\frac{1}{2}\Delta\theta^T F(\theta)\Delta\theta - \epsilon\right) \\ \frac{\partial \mathcal{L}}{\partial \Delta\theta} &= \Delta J(\theta) + \frac{1}{2}\lambda\left(F(\theta) + F^T(\theta)\right)\Delta\theta = 0 \\ \rightarrow \quad \Delta\theta^* &= -\frac{1}{\lambda}F(\theta)^{-1}\Delta J(\theta) \end{aligned}$$

We now can formulate the dual problem, by plugging $\Delta\theta^*$ into the Lagrangian:

$$\begin{aligned}\mathcal{G}(\Delta\theta^*,\lambda) &= \Delta J(\theta)^T\left(-\frac{1}{\lambda}F(\theta)^{-1}\Delta J\right)+\lambda\left[\frac{1}{2}\left(\frac{1}{\lambda}F(\theta)^{-1}\Delta J(\theta)\right)^T F(\theta)\left(\frac{1}{\lambda}F(\theta)^{-1}\Delta J(\theta)\right)-\epsilon\right]\\ &= -\lambda\epsilon-\frac{2}{\lambda}\left(\Delta J(\theta)^T F(\theta)^{-1}\Delta J(\theta)\right)\end{aligned}$$

By taking the partial derivative with respect to λ and setting the equation to zero, we can obtain λ:

$$\begin{aligned}\frac{\partial\mathcal{G}}{\partial\lambda} &= -\epsilon+\frac{1}{2\lambda}\left(\Delta J(\theta)^T F(\theta)^{-1}\Delta J(\theta)\right)=0\\ &\rightarrow \lambda=\sqrt{\frac{\Delta J(\theta)^T F(\theta)^{-1}\Delta J(\theta)}{2\epsilon}}\end{aligned}$$

For $\Delta\theta$ we finally obtain:

$$\Delta\theta=-\sqrt{\frac{\epsilon}{\Delta J(\theta)^T F(\theta)^{-1}\Delta J(\theta)}}F(\theta)^{-1}\Delta J(\theta)$$

In this formula the term below the square root is the learning rate. The term $-F(\theta)^{-1}\Delta J(\theta)$ determines the direction of the descent. Depending on the matrix $F(\theta)^{-1}$ is not necessarily is the same direction as $-\Delta J(\theta)$, because this matrix can turn the vector to another direction.

Natural gradient. In statistical models, parameter vector θ often contains parameters of a probability density function $p(x;\theta)$ (for example, mean and covariance of a Gaussian density); thus, the cost function J depends on θ indirectly through $p(x;\theta)$. This two-level structure gives a strong hint to what matrix F to pick for measuring the distance in the parameter space in the most *natural* way. Namely, one can carry over the notion of 'distance' between probability distributions $p(x;\theta+\Delta\theta)$ and $p(x;\theta)$ (which is known from information theory to be well captured by the Kullback-Leibler divergence) to the distance between the corresponding parameter vectors $\theta+\Delta\theta$ and θ.

2) Obtain the quadratic Taylor approximation of the KL divergence from $p(x;\theta)$ to $p(x;\theta+\Delta\theta)$ in the form

$$KL\left(p(x;\theta+\Delta\theta)\,||\,p(x;\theta)\right)\approx\frac{1}{2}\Delta\theta^T F(\theta)\Delta\theta.$$

Covariant gradient with the matrix $F(\theta)$ that you found is known as the natural gradient.