Statistical Machine Learning: Exercise 1

Group A: Kexin Wang (2540047), Paul Philipp Seitz (2337506)

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Term 2022

Task 1: Machine Learning Introduction (6 Points)

1a) Model Fitting (6 Points)

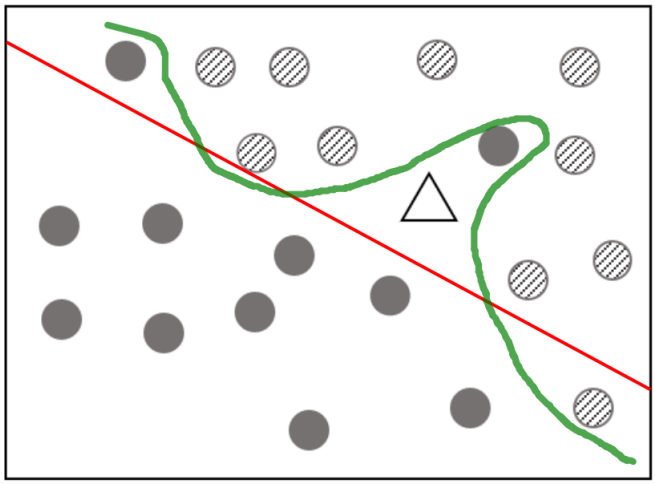


Figure 1: Two different separation lines for the classification problem

As shown in figure 1, the first model is depicted by the red line. It is a simple straight line that does not fit all the training data but correctly classifies the triangle as ‘striped’.

The second model is described by the green line. It is a complex curve that fits the training set perfectly, but does not achieve correct classification of the test data. In this case the triangle would be predicted as ‘filled’.

This example shows that the second model (green line) simply remembers the training data set without generalization, i.e. overfitting, compared to the first model. In general, to achieve generalization we do not need an overly complex model, but rather an optimal trade-off between the accuracy of the training data set and the complexity of the model in order for the model to produce a more accurate output for unknown inputs.

Task 2: Linear Algebra Refresher (20 Points)

2a) Matrix Properties (5 Points)

These statements are not completely correct. In some cases for matrices the addition and multiplication work similarly as for scalars, but in some cases they do not work.

The basic properties of the addition of scalars also apply to the addition of matrices as long as the dimensions of the matrices are the same.

$$A_{m\times n} + B_{m\times n} = B_{m\times n} + A_{m\times n}$$

$$A_{m\times n} + B_{m\times n} + C_{m\times n} = A_{m\times n} + (B_{m\times n} + C_{m\times n})$$

However, the properties of multiplication of scalars do not all generalize to matrices. As the title shown, A, B C are matrices and a,b,c are scalars. The commutative property does not hold for multiplication of matrices, as can be shown by a simple example:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

$$A \times B \neq B \times A$$

The distributive and associative properties hold for multiplication of matrices.
Distributive property:

$$A = [a_{ij}]_{m\times n}, B = [b_{ij}]_{n\times s}, C = [c_{ij}]_{n\times s}$$

$$D = A(B+C) = \left[\sum_{k=1}^{n} a_{ik} \times (b_{kj} + c_{kj})\right]_{m\times s} = AB + AC = \left[\sum_{k=1}^{n} a_{ik} \times b_{kj} + \sum_{k=1}^{n} a_{ik} \times c_{kj}\right]_{m\times s}$$

Associative property:

$$A = [a_{ij}]_{m\times n}, B = [b_{ij}]_{n\times s}, C = [c_{ij}]_{s\times p}$$

$$D = ABC = \left[\sum_{k=1}^{n}\sum_{l=1}^{s} a_{ik} \times b_{kl} \times c_{lj}\right]_{m\times p} = A(BC) = \left[\sum_{k=1}^{n} a_{ik} \times \sum_{l=1}^{s}(b_{kl} \times c_{lj})\right]_{m\times p}$$

2b) Matrix Inversion (7 Points)

$$A^{-1} = \frac{1}{|A|}A^{*}.$$

$$A^{*} = \begin{bmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{bmatrix}$$

A_{xx} is algebraic component, after calculating:

$$A^{*} = \begin{bmatrix} c & -a & ad-bc \\ -1 & 1 & b-d \\ 0 & 0 & c-a \end{bmatrix}$$

$$|A| = c + 0 + 0 - (0 + 0 + a) = c - a$$

Only when $c - a \neq 0$, A is invertible, b can be any value.
So:

$$A^{-1} = \frac{1}{c-a}\begin{bmatrix} c & -a & ad-bc \\ -1 & 1 & b-d \\ 0 & 0 & c-a \end{bmatrix}$$

A matrix is not invertible when its determinant is zero or we can also say, when its rows or columns are linearly dependent.

For matrix

$$A = \begin{bmatrix} 2 & 2 & 3 \\ 0 & 1 & 0 \\ 8 & 3 & 12 \end{bmatrix}$$

$$|A| = 2 \times 1 \times 12 + 0 + 0 - 3 \times 1 \times 8 - 0 - 0 = 0$$

. The $|A| = 0$, so the matrix is not invertible.

2c) Matrix Pseudoinverse (3 Points)

Left Moore-Penrose pseudoinverse: If the matrix A has dimensions $n \times m$ and $rank(A) = m$, then there exists an $m \times n$ matrix $A^{\dagger}_{Left}$ called the left Moore-Penrose pseudoinverse of matrix A such that $A^{\dagger}_{Left}A = I_m$, $A^{\dagger}_{Left} = (A^T A)^{-1}A^T$.

Right Moore-Penrose pseudoinverse: If the matrix A has dimensions $n \times m$ and $rank(A) = n$, then there exists an $m \times n$ matrix $A^{\dagger}_{Right}$ called the right Moore-Penrose pseudoinverse of matrix A such that $AA^{\dagger}_{Right} = I_n$, $A^{\dagger}_{Right} = A^T(AA^T)^{-1}$.

For $A \in \mathbb{R}^{2\times 3}$, $A^T A$ do not have full rank. That means there is no left Moore-Penrose pseudoinverse of matrix A. We can find the right Moore-Penrose pseudoinverse of matrix A: $A^{\dagger}_{Right} = A^T(AA^T)^{-1}$ of dimension 3×2 (for $A^T isofdimension 3 \times 2$ and $(AA^T)^{-1} isofdimension 2 \times 2$).

2d) Eigenvectors Eigenvalues (7 Points)

Eigenvectors: First, thinking about matrix A as linear transformation. And during the linear transformation, some special vectors still remain on its span, meaning the effect that the matrix has on such a vector is just stretch it or squash it, like a scalar. These special vectors are called **Eigenvectors** of the linear transformation (Matrix A).
Eigenvalues: Each eigenvector has associated with it, which called an eigenvalue, which is just the factor by which is stretched or squashed during the transformation. Eigenvectors and eigenvalues can be easily presented as below, $\vec{v}$ is eigenvector and λ is eigenvalue,$\vec{I}$ is identity matrix:

$$A\vec{v} = \lambda\vec{v}$$

$$(A - \lambda I)\vec{v} = 0$$

Eigenvectors and eigenvalues are useful in machine learning, especially in computer vision area. For example, in PCA, we usually use eigenvectors and eigenvalues to represent the data in a lower dimensional space, which means it can reduce the complexity of calculation and calculation cost.
As matrix C is:

$$\begin{bmatrix} 4 & 0 & 1 \\ -2 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix}$$

The eigenvector and eigenvalues are:

$$\begin{bmatrix} 4 & 0 & 1 \\ -2 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \vec{v} = \lambda\vec{v}$$

$$\left(\begin{bmatrix} 4 & 0 & 1 \\ -2 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)\vec{v} = \vec{0}$$

$$\begin{bmatrix} 4-\lambda & 0 & 1 \\ -2 & 1-\lambda & 0 \\ -2 & 0 & 1-\lambda \end{bmatrix} \vec{v} = \vec{0}$$

Because we need a non-zero vector so we need $\det(A - \lambda I) = 0$:

$$\det\left(\begin{bmatrix} 4-\lambda & 0 & 1 \\ -2 & 1-\lambda & 0 \\ -2 & 0 & 1-\lambda \end{bmatrix}\right) = 0$$

$$(4-\lambda)(1-\lambda)(1-\lambda)+0+0-(-2)(1-\lambda)-0-0=0$$

Therefore, we get $\lambda_1 = 1$,$\lambda_2 = 3$ or $\lambda_3 = 2$
Eigenvectors have the property $Av_i = \lambda_i v_i$ and thus $(A-\lambda_i I)v_i = Av_i - \lambda_i I v_i = \lambda_i v_i - \lambda_i v_i = 0$

For $\lambda_1 = 1$:

$$\begin{bmatrix} 4-1 & 0 & 1 \\ -2 & 1-1 & 0 \\ -2 & 0 & 1-1 \end{bmatrix} \vec{v}_1 = \vec{0}$$

then we have:

$$\vec{v}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

For $\lambda_2 = 3$,we calculate the eigenvector as above:

$$\vec{v}_2 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

And for $\lambda_3 = 2$ we calculate the eigenvector:

$$\vec{v}_3 = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}$$

Task 3: Statistics Refresher (29 Points)

3a) Expectation and Variance (8 Points)

1) Expectation, or expected value, is the most basic numerical characteristic that measures the centralized position or average level of the values of a random variable.

$$E(f(\omega)) = \sum_{\omega \in \Omega} P(\omega) f(\omega)$$

Variance is a numerical characteristic that indicates the dispersion of the values of a random variable. The larger the variance, the more uneven the distribution of the values of the random variable and the stronger the variability; the smaller the variance, the more the values of the random variable tend to be close to the mean, i.e., the expected value.

$$Var(f(\omega)) = E((f(\omega) - E(f(\omega)))^2)$$

Expectation is linear operator because it satisfies the superposition theorem but variance not.

2) $E(A) = 3.17, Var(A) = 1.58$ $E(B) = 3.44, Var(B) = 4.14$ $E(C) = 3.67, Var(C) = 2.89$
These are calculated using unbiased estimators.

3) Assum that the discrete distribution of a fair, uniform die is D:

$$KL(A||D) = \sum_{x \in (1,\ldots,6)} A(x) \ln \frac{A(x)}{D(x)} = 0.20583$$

$$KL(B||D) = \sum_{x \in (1,\ldots,6)} B(x) \ln \frac{B(x)}{D(x)} = 0.25377$$

$$KL(C||D) = \sum_{x \in (1,\ldots,6)} C(x) \ln \frac{C(x)}{D(x)} = 0.01888$$

Based on the above results we can learn that C is closest to a fair, uniform die.

3b) It is a Cold World (7 Points)

1) Assume that P(A) is the probability of having a cold, and P(B) is the probability of having a backpain.

2) Domains of the two random variables are the same, that's $0, 1 or true, false$.

3) $P(B|A) = 0.25; P(A) = 0.04; P(B|\tilde{A}) = 0.1.$

4) $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ with $P(B) = P(B|A)P(A) + P(B|\tilde{A})P(\tilde{A})$

$$P(A|B) = 0.09434$$

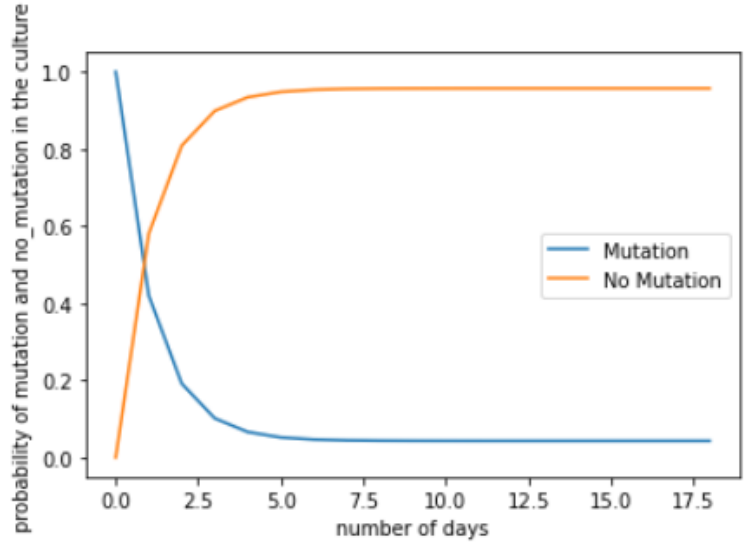
3c) Cure the Virus (14 Points)

1) $\vec{s_t} = [m, \tilde{m}]^T$,
$\vec{s_{t+1}} = [0.42m + 0.026\tilde{m}, 0.58m + 0.974\tilde{m}]^T$

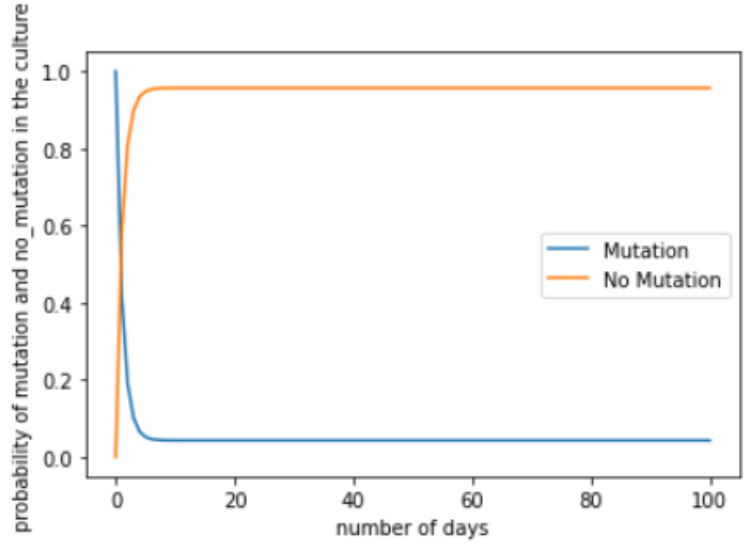
2)

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

# Define a function which uses starte state state0 as input and output the states in the next 18 generations and plot them.
def generations(state0, n):
    # Initialization of the start state.
    state = {}
    state[str(0)] = state0
    mu = [state0[0]]
    no_mu = [state0[1]]

    # Get the data of next n generations.
    for i in range(n):
        state[str(i+1)] = np.array([[0.42, 0.026],[0.58, 0.974]]) @ state[str(i)]
        mu.append(state[str(i+1)][0])
        no_mu.append(state[str(i+1)][1])

    # Plot all the states in the first n generations with time.
    plt.plot(range(n+1), mu, range(n+1), no_mu)
    plt.xlabel('number of days')
    plt.ylabel('probability of mutation and no_mutation in the culture')
    plt.legend(['Mutation', 'No Mutation'], loc='right')
    plt.show()

    return mu, no_mu

# Assum that the start state is [1,0]
state0 = [1,0]
mutation, no_mutation= generations(state0, 18)
print('Probability of mutation after 18 generations is '+ str(mutation[-1]) + '.')

# Task 3c).3)
long_time = 100
mutation, no_mutation= np.array(generations(state0, long_time))
```

```
print('Probability of mutation after ' + str(long_time) + ' generations is '+
    str(mutation[-1]) + '.')
print('No significant change after ' + str(np.where(mutation==mutation[-1])[0][0]) + '
    timesteps.')
```




```
Probability of mutation after 18 generations is 0.042904340534769235.
```




```
Probability of mutation after 100 generations is 0.042904290429042855.
No significant change after 43 timesteps.
```

Figure 2: Probability of mutation and no_mutation in the culture with time

The results are shown in figure 2.

3) As in figure 2, we can see that after 43 timesteps changes the probability of mutation very slightly, nearly zero. For stable probability: $s_{t+1}^{\rightarrow} = Mats_t^{\rightarrow}$ with $Mat = \begin{bmatrix} 0.42 & 0.026 \\ 0.58 & 0.974 \end{bmatrix}$. The final probability converges to the stable probability 0.04.The mathematical expression of stable probability is found below:

$$x + y = 1$$

$$0.58x - 0.026y = 0$$

The result of the equation solution and the result of the simulation are the same.

Task 4: Information Theory (5 Points)

4a) Entropy (5 Points)

1)

$$H(S_1) = -p_1 \log_2 p_1 = 0.1865$$
$$H(S_2) = -p_2 \log_2 p_2 = 0.4796$$
$$H(S_3) = -p_3 \log_2 p_3 = 0.3819$$
$$H(S_4) = -p_4 \log_2 p_4 = 0.2688$$
$$H(S) = H(S_1) + H(S_2) + H(S_3) + H(S_4) = 1.3159$$

2) When $p_i = 0.25, H(S_1) = H(S_2) = H(S_3) = H(S_4) = -0.25 \log_2 0.25 = 0.5$, we can get that $H(s) = 2$ bits. This Distribution with maximum entropy is uniform distribution.

Task 5: Bayesian Decision Theory (20 Points)

5a) Optimal Boundary (4 Points)

Bayesian Decision Theory is decision problem posed in probabilistic terms. It's goal is to minimize the misclassification rate.
Bayes optimal claasification is based on probability distributions $p(x|C_k) \times p(C_k)$. Posterior should be calculated in order to find the optimal decision boundary . We decideC_1, if $p(C_1|x) > p(C_2|x)$, which is equal to:

$$\frac{p(x|C_1)}{p(x|C_2)} > \frac{p(C_2)}{p(C_1)}$$

.

5b) Decision Boundaries (8 Points)

On the decision boundary x^* is $p(C_1|x^*) = p(C_2|x^*)$ valid. From the task description we know that $p(C_1) = p(C_2)$, so we can get $p(x^*|C_1) = p(x^*|C_2)$, which means $(x^* - \mu_1)^2 = (x^* - \mu_2)^2$. Finally we can get $x^* = \frac{\mu_1 + \mu_2}{2}$.

5c) Different Misclassification Costs (8 Points)

The main purpose is to minimize the misclassification rate. That is to say the risk of the two decision are the same on the decision boundary:

$$\lambda(\alpha_2|C_1)p(C_1|x^*) = \lambda(\alpha_1|C_2)p(C_2|x^*)$$

We also know that the misclassification cost of misclassification as C_1 is four times expensive than the oppsite, so we can get:

$$p(C_1|x^*) = 4p(C_2|x^*)$$

which can be written as $p(x^*|C_1) = 4p(x^*|C_2)$ with the fact $p(C_1) = p(C_2)$. Finally we can get:

$$x^* = 1.5\mu_2 + 1.3863\frac{\delta^2}{\mu_2}$$