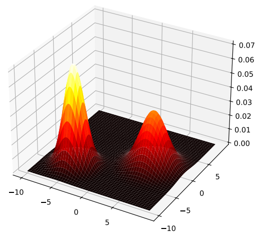
Machine Learning for Computer Security

Exercise Sheet 2

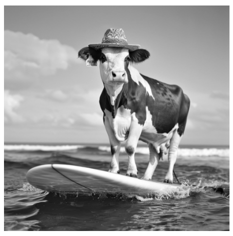
Summer term 2025

Exercise 1 *NumPy and Matplotlib*

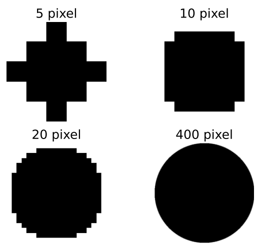
Recreate the plots shown in Figure 1 as closely as possible. Pay attention to every detail.



(a)



(b)



(c)

Figure 1: Recreate these plots as closely as possible.

1. Recreate the plot shown in Figure 1a. To do so, you need to superposition two Gaussian distributions, with a standard deviation of 1.5 and 2.0 respectively. The first Gaussian is centered at $(-5.0, -5.0)$ while the second is centered at $(5.0, 0.0)$.

   Hint: scipy.stats.norm.pdf, matplotlib.pyplot.plot_surface

2. Write a function that splits an image into 4×4 blocks and reorganizes them based on a specific ordering provided as an input to the function. Use the function to solve the puzzle in Figure 1b. You find the required image data on ISIS.

   Hint: matplotlib.pyplot.imread

3. Recreate the plot shown in Figure 1c. Use as few instructions as possible to draw a circle of different diameters. Make sure the circles are perfectly centered. *Only use functions offered by Numpy!*

Exercise 2 *Density estimation*

Density estimation is a classical approach of unsupervised machine learning that is used to construct a simple model for unlabeled data. We assume that a finite set of samples $X = \{x_1, \dots, x_n\} \subset \mathbb{R}$ has been drawn from the unknown probability distribution P. Our goal is to find a good approximation to P solely based on X. These are the necessary steps to follow:

1. We choose P to be a Gaussian distribution with a mean value of 0.2 and a variance of 1. Draw 1000 samples from this distribution as your data set X. Verify that the samples are normally distributed by calculating the mean and variance for X.

2. To approximate P on the interval $[-3; 3]$ generate 1000 equally spaced values in this interval and store the resulting set in a variable named S.

3. For a fixed s from S, implement a function $C_h(s, X)$ to calculate the number of elements from X that are located within a window of size h centered at s:

$$C_h(s, X) = \left| \left\{ x \in X \;\middle|\; s - \frac{h}{2} < x < s + \frac{h}{2} \right\} \right|$$

4. The density estimate at a point s is then obtained by dividing the number of data points in the window by the size of the window and the number of data points, or formally

$$\hat{P}_h(s, X) = \frac{1}{n}\frac{1}{h} \cdot C_h(s, X)$$

   Calculate and plot three estimates of $\hat{P}_h$ using different values for h to illustrate overfitting and underfitting. Label and explain the plots accordingly.

Exercise 3 *PyTorch*

PyTorch is powerful and versatile tool in the research on machine learning security as it allows researchers to quickly prototype attacks and defenses. In this tasks, you will learn how to use PyTorch's automatic differentiation to determine the derivatives of functions.

1. We start with a simple quadratic function:

   $$f(x) = (0.5x + 3)^2 - 2$$

   Implement the function, evaluate it on a PyTorch tensor containing equally spaced values in $[-10, 10]$, and plot the resulting graph with matplotlib.

2. Use `.backward()` to calculate the derivative of f with respect to the input tensor x and plot the derivative alongside the original function.[1]

3. Calculate the derivative $\frac{d}{dx}f(x)$ by hand and check whether your implementation is correct.

4. Repeat the tasks 1-3 for the function $g(x) = \sin(x)$.

5. Newton's method iteratively searches for the root of a function using the sequence

   $$x_{i+1} = x_i - \frac{f(x_i)}{\frac{d}{dx}f(x_i)}$$

   Implement Newton's method in a function `newtons_method(f, x_0, n)` that receives a unary function, a starting value x_0, and the number of iterations n.

[1]Note that `.backward()` expects a scalar value. For a multidimensional input x, you can aggregate the inputs into a single value with `y.sum()` without affecting the gradients.