Problems

Problem 1. Let $f(x,y) = ye^x$. Find the Taylor Expansion at $(0,0)$ up to the second order.

Problem 2. Find $x(t)$ that satisfies the following differential equation.

$$\frac{dy}{dt} = \sin(t)y(t) \text{ and } y(0) = 1$$

Problem 3. Consider three data points $(x_1, y_1), (x_2, y_2)$, and (x_3, y_3) on plane $\mathbb{R}^2$. Find $\alpha, \beta \in \mathbb{R}$ minimising

$$\sum_{i=1}^{3} |\alpha + \beta x_i - y_i|^2$$

Problem 4. Let X and Y be a random variable. Assume that X is $\mathcal{G}$-measurable for some sigma algebra $\mathcal{G}$. Show that

$$\mathbb{E}[XY|\mathcal{G}] = X\mathbb{E}[Y|\mathcal{G}]$$

Problem 5. Assume that X and Y are independent standard normal random variables. Then show that $X + 2Y$ is a normal random variable. Find its mean and variance.

Problem 6. Write a Python code that uses the gradient descent method to (numerically) find the minimum of the function

$$f(x,y) := x^2 + 4x - y + e^{x+y}.$$

Solutions

Solution to Problem 1.

$$\begin{aligned} f(x,y) &= \frac{\partial f}{\partial x}(0,0)x + \frac{\partial f}{\partial y}(0,0)y + \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}(0,0)x^2 + \frac{\partial^2 f}{\partial x \partial y}(0,0)xy + \frac{\partial^2 f}{\partial y^2}(0,0)y^2\right) \\ &= y + \frac{1}{2}xy + \mathcal{O}(|x,y|^3) \end{aligned}$$

Solution to Problem 2.

$$\begin{aligned} \frac{dy}{y} &= \sin(t)dt \\ y(t) &= Ce^{\int_0^t \sin(u)du} \end{aligned}$$

By our initial condition, $C = 1$ and $y(t) = e^{\int_0^t \sin(u)du}$.

Solution to Problem 3. Let $f(\alpha, \beta) = \sum_{i=1}^{3} |\alpha + \beta x_i - y_i|^2$. To find the minimising (α, β), let us differentiate.

$$\frac{\partial f}{\partial \alpha} = 2\sum_{i=1}^{3}(\alpha + \beta x_i - y_i) = 0$$
$$\frac{\partial f}{\partial \beta} = 2\sum_{i=1}^{3}(\alpha + \beta x_i - y_i)x_i = 0$$

Then,

$$\begin{pmatrix} 3 & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix}$$

Therefore,

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 3 & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix}$$

Solution to Problem 4. It is clear that the right-hand side is $\mathcal{G}$ measurable by the definition of conditional expectation $\mathbb{E}[Y|\mathcal{G}]$. Therefore, we only need to show, for any $\mathcal{G}$-measurable random variable V, we have

$$\mathbb{E}[VXY] = \mathbb{E}[VX\mathbb{E}[Y|\mathcal{G}]].$$

Since X is $\mathcal{G}$-measurable, we know VX is also $\mathcal{G}$-measurable. By the definition of $\mathbb{E}[Y|\mathcal{G}]$, we have

$$\mathbb{E}[VX\mathbb{E}[Y|\mathcal{G}]] = \mathbb{E}[VXY].$$

Therefore, the claim is proven.

Solution to Problem 5. The moment generating function is

$$\begin{aligned} \mathbb{E}e^{r(X+2Y)} &= \mathbb{E}e^{rX}\mathbb{E}e^{(2r)Y} \\ &= e^{\frac{1}{2}r^2}e^{\frac{1}{2}(2r)^2} = e^{\frac{1}{2}5r^2} \end{aligned}$$

Therefore, $X + 2Y$ is a normal random variable with mean zero and variance 5.

Solution to Problem 5.

```python
import numpy as np

def f(x, y):
    """
    The function to minimize.
    f(x, y) = x^2 + 4x - y + e^(x+y)
    """
    return x**2 + 4*x - y + np.exp(x+y)
```

```
def grad_f(x, y):
  """
  The gradient of the function f(x, y).
  Returns a numpy array [df/dx, df/dy].
  """
  df_dx = 2*x + 4 + np.exp(x+y)
  df_dy = -1 + np.exp(x+y)
  return np.array([df_dx, df_dy])

def gradient_descent(starting_point, learning_rate, n_iterations):
  """
  Performs gradient descent to find the minimum of the function.

  Args:
    starting_point: A numpy array [x, y] for the starting point.
    learning_rate: The learning rate for gradient descent.
    n_iterations: The number of iterations to perform.

  Returns:
    The point [x, y] that minimizes the function.
  """
  point = starting_point
  for i in range(n_iterations):
    grad = grad_f(point[0], point[1])
    point = point - learning_rate * grad
  return point

# --- Hyperparameters ---
starting_point = np.array([0.0, 0.0])
learning_rate = 0.01
n_iterations = 1000

# --- Run Gradient Descent ---
minimum_point = gradient_descent(starting_point, learning_rate, n_iterations)
minimum_value = f(minimum_point[0], minimum_point[1])

# --- Print Results ---
print(f"The minimum is at approximately: ({minimum_point[0]:.4f}, {minimum_point[
print(f"The minimum value of the function is approximately: {minimum_value:.4f}")
```