Module 1: Mathematical Concepts in ML/AI

Learning outcomes

1. Apply linear algebra concepts, specifically vectors, matrices, transformations and eigenvalues, to solve mathematical problems in ML/AI.
2. Analyse key calculus concepts, specifically derivatives, partial differentiation, chain rule and norms.
3. Analyse key optimisation concepts, including gradient descent, local minimum and global minimum, and learning rate.
4. Apply optimisation techniques in Python.

Linear algebra

- **Vectors**
  - Represent features, predictions and model parameters
  - Operations
    - Addition: $a + b = (a_1 + b_1, \dots, a_n + b_n)$
    - Scalar multiplication: $k \cdot v = (kv_1, \dots, kv_2)$
    - Dot product: $a \cdot b = \sum a_i b_i$
- **Matrices**
  - Used for representing data and transformations
  - Matrix multiplication: rows of A and columns of B
  - Identity matrix: I
  - Zero matrix: 0
- **Eigenvalue and eigenvectors**
  - $Av = \lambda v$
  - Used in PCA, data compression and stability analysis
- **Matrix decomposition**
  - SVD: $Av = U \sum V^{Y}$
  - LU: $A = LU$
  - Enables efficient computation and simplification

Calculus

- **Derivatives**
  - Measure the rate of change
  - $f'(x) = \lim_{\text{h}} \rightarrow 0\ [\frac{f(x+h) - f(x)}{g}]$
- **Partial derivatives**
  - Derivatives with respect to one variable at a time
  - $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$

- **Chain rule**
    - Composite functions: $h(x) = f\big(g(x)\big) \to h'(x) = f'(g(x)) \cdot g'(x)$
- **Norms**
    - Measure vector length or distance
        - $L_1: \big||x|\big|_1 = \sum |x_{i|}$
        - $L_2: \big||x|\big|_2 = \sqrt{\sum x_i^2}$
        - $L_\infty: \big||x|\big|\infty = \max\,(|x_i|)$

Probability and statistics

- **Distributions**
    - Model randomness (e.g. binomial, normal)
- **Expectation and variance**
    - $E[X] = \sum x \cdot P(x)$
    - $Var(X) = E[(X - E[X])^2]$
- **Entropy**
    - Measures uncertainty
    - $H(X) = -\sum p(x) \cdot \log_2 p(x)$

Optimisation

- **Gradient descent**
    - Minimises loss
    - $\theta = \theta - \alpha \cdot \nabla J(\theta)$, in which $\alpha =$ learning rate
- **Loss functions**
    - MSE: $\frac{1}{n} \sum (y_i - \hat{y}_i)^2$
    - Cross-entropy: $-\sum y \cdot \log \hat{y}$
- **Activation functions**
    - Sigmoid: $\frac{1}{(1+e^{-x})}$
    - ReLU: $\max\,(0, x)$
- **Regularisation**
    - Prevents overfitting
    - L1: adds $\lambda \sum |w_i|$
    - L2: adds $\lambda \sum w_i^2$