Mathematical Foundations for Machine Learning and Deep Learning

CS 582 - B

Introduction

This document provides a comprehensive overview of the linear algebra and calculus concepts necessary for understanding machine learning and deep learning. It covers scalars, vectors, matrices, derivatives, gradients, optimization, and advanced concepts in deep learning.

Linear Algebraic Objects

1. Scalars, Vectors, Matrices, and Tensors

- **Scalar**: A single real number, denoted as $x \in \mathbb{R}$. *Example*: $x = 5$.
- **Vector**: A one-dimensional array of numbers. A vector in n-dimensional space is denoted as:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \in \mathbb{R}^n$$

  Example: A 3-dimensional vector $\mathbf{v} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$.

- **Matrix**: A two-dimensional array of numbers. A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ has m rows and n columns:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

  Example: A 2x2 matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

- **Tensor**: A generalization of vectors and matrices to higher dimensions. A tensor of rank k is denoted as:

$$\mathbf{T} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_k}$$

  Example: A 3D tensor with dimensions $2 \times 2 \times 2$.

2. Vector and Matrix Operations

- **Vector Addition**:

$$\mathbf{a} + \mathbf{b} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \end{bmatrix}$$

Example:

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

- **Matrix Addition**: For matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$,

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

- **Scalar Multiplication**: For $\alpha \in \mathbb{R}$,

$$\alpha \mathbf{A} = \alpha \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} \\ \alpha a_{21} & \alpha a_{22} \end{bmatrix}$$

Example:

$$3 \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$$

- **Matrix-Vector Multiplication**: If $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{v} \in \mathbb{R}^n$,

$$\mathbf{A}\mathbf{v} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} a_{11}v_1 + a_{12}v_2 \\ a_{21}v_1 + a_{22}v_2 \end{bmatrix}$$

Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \times 5 + 2 \times 6 \\ 3 \times 5 + 4 \times 6 \end{bmatrix} = \begin{bmatrix} 17 \\ 39 \end{bmatrix}$$

- **Dot Product (Inner Product)**: For $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$,

$$\mathbf{a}^T \mathbf{b} = \sum_{i=1}^{n} a_i b_i$$

Example:

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = 1 \times 3 + 2 \times 4 = 11$$

- **Outer Product**: For $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$,

$$\mathbf{a}\mathbf{b}^T = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \begin{bmatrix} b_1 & b_2 \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 \\ a_2 b_1 & a_2 b_2 \end{bmatrix}$$

Example:

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix}$$

3. Matrix Properties and Operations

- **Transpose**: The transpose of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a matrix $\mathbf{A}^T \in \mathbb{R}^{n \times m}$, where:

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix}$$

  Example: If $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, then

$$\mathbf{A}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

- **Inverse**: The inverse of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is $\mathbf{A}^{-1}$ such that:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

  Example: For

$$\mathbf{A} = \begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix}, \quad \mathbf{A}^{-1} = \frac{1}{10} \begin{bmatrix} 6 & -7 \\ -2 & 4 \end{bmatrix}$$

- **Trace**: The trace of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is:

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}$$

  Example:

$$\text{tr}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \right) = 1 + 4 = 5$$

- **Determinant**: The determinant of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, denoted $\det(\mathbf{A})$, is a scalar that represents the volume scaling factor of the linear transformation described by $\mathbf{A}$. *Example*:

$$\det\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \right) = 1 \times 4 - 2 \times 3 = -2$$

- **Norms**:
  - L_2-norm:

$$||\mathbf{v}||_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$$

    Example:

$$|| \begin{bmatrix} 3 \\ 4 \end{bmatrix} ||_2 = \sqrt{3^2 + 4^2} = 5$$

– L_1-norm:

$$||\mathbf{v}||_1 = \sum_{i=1}^{n} |v_i|$$

Example:

$$|| \begin{bmatrix} -1 \\ 2 \end{bmatrix} ||_1 = |-1| + |2| = 3$$

– Frobenius norm:

$$||\mathbf{A}||_F = \sqrt{\sum_{i,j} a_{ij}^2}$$

Example:

$$|| \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} ||_F = \sqrt{1^2 + 2^2 + 3^2 + 4^2} = \sqrt{30} \approx 5.477$$

4. Eigenvalues and Eigenvectors

- **Eigenvalue Equation**: For matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, if there exists a scalar λ and a vector $\mathbf{v} \in \mathbb{R}^n$ such that:

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

then λ is an eigenvalue of $\mathbf{A}$ and $\mathbf{v}$ is the corresponding eigenvector. *Example*: For

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \lambda = 3, \quad \mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

we have:

$$\mathbf{A}\mathbf{v} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

- **Spectral Decomposition**: If $\mathbf{A}$ is a square matrix, it can be decomposed as:

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

where $\mathbf{V}$ is the matrix of eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues. *Example*: For

$$\mathbf{A} = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$$

suppose the eigenvalues are $\lambda_1 = 5$ and $\lambda_2 = 2$, with corresponding eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$. Then:

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}, \quad \mathbf{\Lambda} = \begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}$$

and

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

Calculus Concepts

1. Derivatives and Gradients

- **Derivative of a scalar function**: The derivative of a function $f(x)$ with respect to x is:

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Example: The derivative of $f(x) = x^2$ is:

$$\frac{df}{dx} = 2x$$

- **Partial Derivatives**: For a multivariable function $f(x_1, x_2, \ldots, x_n)$, the partial derivative with respect to x_i is:

$$\frac{\partial f}{\partial x_i}$$

Example: For $f(x, y) = x^2 y + 3y^2$,

$$\frac{\partial f}{\partial x} = 2xy, \quad \frac{\partial f}{\partial y} = x^2 + 6y$$

- **Gradient**: The gradient of a scalar function $f(\mathbf{x})$ with respect to the vector $\mathbf{x}$ is the vector of partial derivatives:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Example: For $f(x, y) = x^2 + y^2$,

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

2. Chain Rule and Hessians

- **Chain Rule**: If $z = f(y)$ and $y = g(x)$, then the chain rule states:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

Example: Let $z = \sin(y)$ and $y = x^2$, then:

$$\frac{dz}{dx} = \cos(y) \cdot 2x = 2x\cos(x^2)$$

- **Hessian Matrix**: The Hessian is a matrix of second-order partial derivatives of a scalar function $f(\mathbf{x})$:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Example: For $f(x, y) = x^3 + y^3$,

$$\mathbf{H} = \begin{bmatrix} 6x & 0 \\ 0 & 6y \end{bmatrix}$$

3. Convexity and Optimization

- **Convex Function**: A function $f(x)$ is convex if for any two points x_1 and x_2,
$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$
for all $\lambda \in [0,1]$. *Example*: $f(x) = x^2$ is convex because for any x_1, x_2 and $\lambda \in [0,1]$,
$$(\lambda x_1 + (1-\lambda)x_2)^2 \leq \lambda x_1^2 + (1-\lambda)x_2^2$$

- **First-order condition for convexity**: If $f(x)$ is convex, then for all x and y:
$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$
Example: For $f(x) = x^2$, the gradient is $\nabla f(x) = 2x$. The first-order condition becomes:
$$y^2 \geq x^2 + 2x(y-x) = 2xy - x^2$$
which simplifies to $y^2 \geq 2xy - x^2$, or $(y-x)^2 \geq 0$, which is always true.

- **Gradient Descent**: Iterative optimization method to minimize a function $f(\mathbf{x})$. The update rule is:
$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f(\mathbf{x})$$
where η is the learning rate. *Example*: To minimize $f(x) = x^2$, starting from $x = 4$ with $\eta = 0.1$,
$$x_{\text{new}} = 4 - 0.1 \times 2 \times 4 = 4 - 0.8 = 3.2$$

4. Taylor Series

- The Taylor series expansion of a function $f(x)$ around a point $x = a$ is:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots$$

Example: For $f(x) = e^x$ around $a = 0$,

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Advanced Concepts for Deep Learning

1. Backpropagation

- Backpropagation computes gradients for each layer of a neural network using the chain rule. The goal is to minimize a loss function L by updating the weights $\mathbf{W}$ and biases $\mathbf{b}$.
- For each layer, the gradients are computed as:

$$\frac{\partial L}{\partial \mathbf{W}}, \quad \frac{\partial L}{\partial \mathbf{b}}$$

Example: In a neural network for image classification, suppose the loss function is $L = ||\mathbf{y} - \hat{\mathbf{y}}||^2$, where $\mathbf{y}$ is the true label and $\hat{\mathbf{y}}$ is the predicted label. Backpropagation calculates the gradient of L with respect to each weight and bias in the network to update them accordingly.

2. Lagrange Multipliers

- To optimize a function $f(\mathbf{x})$ subject to a constraint $g(\mathbf{x}) = 0$, the Lagrangian is defined as:

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$$

Example: Maximize $f(x, y) = xy$ subject to $x + y = 10$.

$$\mathcal{L}(x, y, \lambda) = xy + \lambda(10 - x - y)$$

Taking partial derivatives and setting them to zero:

$$\frac{\partial \mathcal{L}}{\partial x} = y - \lambda = 0 \quad \Rightarrow \quad y = \lambda$$

$$\frac{\partial \mathcal{L}}{\partial y} = x - \lambda = 0 \quad \Rightarrow \quad x = \lambda$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 10 - x - y = 0 \quad \Rightarrow \quad x + y = 10$$

Therefore, $x = y = 5$ maximizes xy under the constraint $x + y = 10$.

3. Gradient Descent Variants

- **Stochastic Gradient Descent (SGD)**: Updates the parameters using a single data point at each step. *Example*: In training a neural network with SGD, at each iteration, a single training example is used to compute the gradient and update the weights:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla L(\mathbf{W}; \mathbf{x}^{(i)}, y^{(i)})$$

  where $(\mathbf{x}^{(i)}, y^{(i)})$ is the i-th training example.

- **Momentum**: Adds a fraction of the previous update to the current update to accelerate convergence. *Example*: The update rule with momentum γ is:

$$\mathbf{v}_{t+1} = \gamma \mathbf{v}_t + \eta \nabla L(\mathbf{W}_t)$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \mathbf{v}_{t+1}$$

- **Adam**: Adaptive learning rate optimization algorithm that uses moving averages of the first and second moments of the gradients. *Example*: The Adam optimizer updates parameters using estimates of first and second moments of gradients:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \nabla L(\mathbf{W}_t)$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) (\nabla L(\mathbf{W}_t))^2$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \frac{\mathbf{m}_t}{\sqrt{\mathbf{v}_t} + \epsilon}$$

  where β_1 and β_2 are decay rates, and ϵ is a small constant to prevent division by zero.

4. Convolution Operations

- **Convolution**: In deep learning, convolutions are used for tasks like image processing and feature extraction. The convolution of two functions f and g is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)\, d\tau$$

  Example: In image processing, f can represent an image and g a filter (kernel) applied to extract certain features such as edges.

- **Discrete Convolution**: In practice, we often work with discrete data (e.g., pixel grids), and the discrete convolution is defined as:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m]$$

  Example: Applying a 3×3 filter to an image patch using the convolution operator to detect vertical edges:

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

  applied to a 3×3 image patch.

- **Convolution in Neural Networks**: In convolutional neural networks (CNNs), convolutions are used to automatically and adaptively learn spatial hierarchies of features. *Example*: In a CNN layer, an input image is convolved with multiple filters to produce feature maps that highlight different aspects such as edges, textures, and shapes.

Advanced Concepts for Deep Learning

5. Regularization Techniques

- **L1 Regularization**: Adds the absolute value of weights to the loss function to encourage sparsity.

$$L_{\text{total}} = L_{\text{original}} + \lambda \sum_i |w_i|$$

  Example: In a linear regression model, L1 regularization can lead to some weights being exactly zero, effectively performing feature selection.

- **L2 Regularization**: Adds the squared value of weights to the loss function to prevent overfitting.

$$L_{\text{total}} = L_{\text{original}} + \lambda \sum_i w_i^2$$

  Example: In a neural network, L2 regularization helps in keeping the weights small, which can improve generalization.

- **Dropout**: Randomly sets a fraction of input units to zero during training to prevent co-adaptation of features. *Example*: In a fully connected layer with 50% dropout, during each training iteration, half of the neurons are randomly ignored.

6. Activation Functions

- **Sigmoid**:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Example: Used in binary classification tasks to map predictions to probabilities between 0 and 1.

- **ReLU (Rectified Linear Unit)**:

$$\text{ReLU}(x) = \max(0, x)$$

Example: Applied in hidden layers of neural networks to introduce non-linearity.

- **Tanh**:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Example: Used in recurrent neural networks for handling sequences.

- **Softmax**:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Example: Used in the output layer of multi-class classification neural networks to represent probability distributions over classes.

7. Loss Functions

- **Mean Squared Error (MSE)**:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Example: Used in regression tasks to measure the average squared difference between actual and predicted values.

- **Cross-Entropy Loss**:

$$\text{Cross-Entropy} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

Example: Used in binary classification tasks to measure the performance of classification models whose output is a probability between 0 and 1.

- **Hinge Loss**:

$$\text{Hinge Loss} = \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i\hat{y}_i)$$

Example: Used in support vector machines for classification tasks to ensure a margin between classes.

Additional Concepts

1. Singular Value Decomposition (SVD)

- SVD decomposes a matrix $\mathbf{A}$ into three matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, and $\mathbf{\Sigma}$ is a diagonal matrix of singular values. *Example*: For

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{U} = \mathbf{V} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{A}$$

2. Principal Component Analysis (PCA)

- PCA is a dimensionality reduction technique that transforms data to a new coordinate system, reducing the number of variables while preserving as much variance as possible.
- The principal components are the eigenvectors of the covariance matrix of the data. *Example*: Given a dataset with high-dimensional features, PCA can reduce the dimensionality to two principal components for visualization:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where the top two singular values in $\mathbf{\Sigma}$ correspond to the directions with the highest variance.

3. Regularization and Overfitting

- **Overfitting**: When a model learns the training data too well, including its noise, leading to poor generalization on unseen data.
- **Regularization**: Techniques to prevent overfitting by adding constraints or penalties to the model. *Example*: Using L2 regularization in a neural network to keep the weights small, thereby simplifying the model and improving generalization.

4. Activation Functions and Non-Linearity

- Activation functions introduce non-linearity into neural networks, enabling them to learn complex patterns.
- Without activation functions, a neural network would be equivalent to a single-layer linear model. *Example*: The ReLU activation function allows the network to model non-linear relationships by activating neurons only when the input is positive.