Foundations of AI, Machine Learning, Deep Learning, and Computer Vision

A Mathematical and Conceptual Guide for MSc Students

Author: Bishal Pandey

University of Southampton

Version 1.0 — June 24, 2025

Contents

1 Linear Algebra **1**
- 1.1 Vectors and Scalars . . . 1
- 1.2 Matrix Operations . . . 1
- 1.3 Dot Product and Norms . . . 1
- 1.4 Matrix Inverse and Determinant . . . 2
- 1.5 Linear Independence and Rank . . . 2
- 1.6 Projections . . . 2
- 1.7 Orthogonality and Orthonormality . . . 2
- 1.8 Eigenvalues and Eigenvectors . . . 2
- 1.9 Diagonalization . . . 2
- 1.10 Principal Component Analysis (PCA) . . . 2
- 1.11 Singular Value Decomposition (SVD) . . . 3
- 1.12 Cosine Similarity . . . 3

2 Probability and Statistics **4**
- 2.1 Basic Definitions . . . 4
- 2.2 Random Variables . . . 4
- 2.3 Common Distributions . . . 4
- 2.4 Expectation and Variance . . . 5
- 2.5 Bayes' Theorem . . . 5
- 2.6 Entropy and Information . . . 5
- 2.7 Law of Large Numbers . . . 5
- 2.8 Central Limit Theorem . . . 6
- 2.9 Joint, Marginal, Conditional Probabilities . . . 6
- 2.10 Probability in ML . . . 6

3 Optimization and Calculus **7**
- 3.1 Derivatives and Gradients . . . 7
- 3.2 Chain Rule and Backpropagation . . . 7
- 3.3 Hessian and Jacobian . . . 7
- 3.4 Gradient Descent . . . 7
- 3.5 Convex and Non-Convex Optimization . . . 8
- 3.6 Learning Rate Scheduling . . . 8
- 3.7 Regularization and Constraints . . . 8
- 3.8 Loss Surfaces and Saddle Points . . . 8
- 3.9 Automatic Differentiation . . . 8

4 Machine Learning Algorithms **10**
- 4.1 Types of Learning . . . 10
- 4.2 Linear Regression . . . 10
- 4.3 Logistic Regression . . . 10

4.4 Regularization . . . 11
4.5 Support Vector Machines (SVM) . . . 11
4.6 Decision Trees . . . 11
4.7 K-Nearest Neighbors (KNN) . . . 11
4.8 Naive Bayes . . . 11
4.9 Unsupervised Learning . . . 11
4.10 Evaluation Metrics . . . 11

5 Deep Learning Foundations 13
5.1 Artificial Neural Networks (ANNs) . . . 13
5.2 Activation Functions . . . 13
5.3 Forward Pass . . . 13
5.4 Backpropagation . . . 13
5.5 Loss Functions . . . 14
5.6 Optimization Algorithms . . . 14
5.7 Regularization Techniques . . . 14
5.8 Batch Normalization . . . 14
5.9 Weight Initialization . . . 14
5.10 Deep Network Architectures . . . 14

6 Computer Vision 16
6.1 Images as Matrices . . . 16
6.2 Convolutions and Kernels . . . 16
6.3 Pooling Layers . . . 16
6.4 CNN Architecture . . . 17
6.5 Feature Maps and Filters . . . 17
6.6 Transfer Learning . . . 17
6.7 Image Embeddings and Cosine Similarity . . . 17
6.8 Image Augmentation . . . 17
6.9 Loss Functions for Vision . . . 17
6.10 Advanced Applications . . . 18

7 Advanced Mathematical Topics 19
7.1 Principal Component Analysis (PCA) . . . 19
7.2 Singular Value Decomposition (SVD) . . . 19
7.3 KL Divergence and Cross-Entropy . . . 19
7.4 Fourier Transform (1D and 2D) . . . 20
7.5 Manifold Learning . . . 20
7.6 Information Theory in ML . . . 20
7.7 Bayesian Inference . . . 20
7.8 Lagrange Multipliers . . . 20
7.9 Matrix Calculus . . . 20

8 Extended Topics 22
8.1 Transformers and Attention Mechanism . . . 22
8.2 Gradient Stability Tricks . . . 22
8.3 Explainable AI (XAI) . . . 22
8.4 Generative Models . . . 23
8.5 Reinforcement Learning (Basics) . . . 23
8.6 Hyperparameter Optimization . . . 23
8.7 Time Series Forecasting . . . 23
8.8 Bayesian Inference . . . 23

8.9 Matrix Calculus Essentials 24
8.10 Lagrange Multipliers 24
8.11 Model Evaluation Beyond Accuracy 24

1 Linear Algebra

Overview

Linear Algebra is the backbone of most machine learning, deep learning, and computer vision models. It provides the mathematical framework to represent data (as vectors, matrices), transformations (like convolutions), and optimization (like gradient descent).

1.1 Vectors and Scalars

A scalar is a single number (e.g., temperature = 25). A vector is an ordered array of numbers (e.g., pixel RGB values). Vectors live in a vector space $\mathbb{R}^n$.

Notation

- Scalar: x
- Vector: $\vec{x} = [x_1, x_2, ..., x_n]$
- Matrix: $A \in \mathbb{R}^{m \times n}$

1.2 Matrix Operations

- Addition/Subtraction: Element-wise
- Scalar Multiplication: cA
- Matrix Multiplication: $(AB)_{ij} = \sum_k A_{ik} B_{kj}$
- Transpose: $(A^T)_{ij} = A_{ji}$

1.3 Dot Product and Norms

Dot Product

Given vectors $\vec{a}, \vec{b} \in \mathbb{R}^n$:

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i$$

This measures similarity between two vectors.

Norms

$$\|\vec{x}\|_2 = \sqrt{\sum x_i^2} \quad \text{(Euclidean norm)}$$
$$\|\vec{x}\|_1 = \sum |x_i| \quad \text{(Manhattan norm)}$$

1.4 Matrix Inverse and Determinant

- Inverse: A^{-1} such that $AA^{-1} = I$
- Determinant: A scalar value representing matrix scale

1.5 Linear Independence and Rank

Vectors are linearly independent if no vector is a linear combination of others. The rank of a matrix is the number of linearly independent rows or columns.

1.6 Projections

Projection of vector $\vec{a}$ onto $\vec{b}$:

$$\text{proj}_{\vec{b}}\vec{a} = \frac{\vec{a} \cdot \vec{b}}{\|\vec{b}\|^2}\vec{b}$$

1.7 Orthogonality and Orthonormality

- $\vec{a} \cdot \vec{b} = 0$ vectors are orthogonal
- Orthonormal set: Orthogonal and unit vectors

1.8 Eigenvalues and Eigenvectors

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

Where:

- $\vec{v}$: eigenvector
- λ: eigenvalue

Used in PCA, SVD, dynamics, and neural network analysis.

1.9 Diagonalization

Any symmetric matrix A can be diagonalized as:

$$A = Q\Lambda Q^T$$

Where Q contains eigenvectors, Λ is diagonal of eigenvalues.

1.10 Principal Component Analysis (PCA)

- Compute covariance matrix: $\Sigma = \frac{1}{n}X^T X$
- Find eigenvectors $\vec{v}_i$ and eigenvalues λ_i
- Select top-k eigenvectors to form projection matrix W
- Reduce data: $X_{reduced} = XW$

Used for dimensionality reduction and feature decorrelation.

1.11 Singular Value Decomposition (SVD)

$$A = U\Sigma V^T$$

Where:

- U: left singular vectors (eigenvectors of AA^T)
- Σ: diagonal matrix of singular values
- V: right singular vectors (eigenvectors of $A^T A$)

SVD is more general than diagonalization and is used in PCA, LSA, image compression.

1.12 Cosine Similarity

Used to measure similarity between two embeddings (e.g., image features):

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

Summary

Linear algebra gives the language to:

- Represent and transform data
- Extract features via eigenvectors
- Compress and compare representations

Applications:

- Convolutions in CNNs
- PCA for feature reduction
- Cosine similarity in embeddings

2 Probability and Statistics

Overview

Probability and Statistics are essential for reasoning under uncertainty, modeling randomness, and learning from data. Concepts like distributions, Bayes' theorem, and expectations are used in almost every machine learning algorithm.

2.1 Basic Definitions

- Sample space Ω: the set of all outcomes
- Event A: subset of Ω
- Probability of event A: $P(A) \in [0, 1]$
- Complement: $P(\bar{A}) = 1 - P(A)$
- Union: $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- Conditional: $P(A|B) = \frac{P(A \cap B)}{P(B)}$

2.2 Random Variables

A random variable X maps outcomes to numbers.

Types

- Discrete: finite or countable (e.g., dice)
- Continuous: uncountable (e.g., temperature)

2.3 Common Distributions

Bernoulli Distribution

$$P(X = 1) = p, \quad P(X = 0) = 1 - p$$

Binomial Distribution

$$P(k; n, p) = \binom{n}{k} p^k (1 - p)^{n-k}$$

Uniform Distribution

$$P(x) = \frac{1}{b-a} \quad \text{for } x \in [a, b]$$

Gaussian (Normal) Distribution

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

2.4 Expectation and Variance

Expected Value

$$\mathbb{E}[X] = \sum_x x \cdot P(x) \quad \text{or} \quad \int x \cdot p(x) dx$$

Variance

$$\text{Var}(X) = \mathbb{E}[(X-\mu)^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

2.5 Bayes' Theorem

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Example: Spam filtering, diagnosis.

2.6 Entropy and Information

Entropy

$$H(X) = -\sum_i P(x_i) \log P(x_i)$$

Measures the uncertainty in a random variable.

Kullback-Leibler Divergence (KL Divergence)

$$D_{\text{KL}}(P\|Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Measures how one distribution diverges from another.

Cross-Entropy

$$H(p, q) = -\sum_i p(x_i) \log q(x_i)$$

Used as a loss function in classification tasks.

2.7 Law of Large Numbers

As the number of samples increases, the sample mean converges to the expected value.

2.8 Central Limit Theorem

The sum of many independent random variables tends toward a normal distribution, regardless of their individual distributions.

2.9 Joint, Marginal, Conditional Probabilities

- Joint: $P(X = x, Y = y)$
- Marginal: $P(X = x) = \sum_y P(X = x, Y = y)$
- Conditional: $P(X|Y) = \frac{P(X,Y)}{P(Y)}$

2.10 Probability in ML

- Naive Bayes: assumes feature independence
- Maximum Likelihood Estimation (MLE)
- MAP: includes prior knowledge
- Probabilistic Models: Logistic Regression, GMMs

Summary

- Probability helps model uncertainty
- Distributions help define models
- Entropy and KL divergence help define loss
- Bayes' theorem allows reasoning from data

Applications:

- Classification (Naive Bayes, Softmax)
- Variational Autoencoders
- Uncertainty estimation in DL

3 Optimization and Calculus

Overview

Optimization is at the heart of machine learning and deep learning. Almost every model is trained by minimizing a loss function through optimization techniques. Calculus provides the gradients needed to navigate the loss surface.

3.1 Derivatives and Gradients

- Derivative: Rate of change of a function
- Gradient: Vector of partial derivatives

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right]$$

- Directional Derivative: Rate of change in a specific direction

$$D_{\vec{v}} f(x) = \nabla f(x) \cdot \vec{v}$$

3.2 Chain Rule and Backpropagation

Chain Rule:

$$\frac{d}{dx} f(g(x)) = f'(g(x)) \cdot g'(x)$$

Backpropagation in neural networks uses the chain rule to compute gradients from the output layer back to the input.

3.3 Hessian and Jacobian

- Jacobian: Matrix of all first-order partial derivatives
- Hessian: Matrix of second-order partial derivatives

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

3.4 Gradient Descent

Minimizes a function by taking steps proportional to the negative gradient:

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

Where η is the learning rate.

Variants

- Stochastic Gradient Descent (SGD): uses one sample at a time
- Mini-batch Gradient Descent: uses small batches
- Momentum: adds velocity term
- RMSProp: adapts learning rate using squared gradients
- Adam: combines momentum + adaptive learning rate

3.5 Convex and Non-Convex Optimization

Convex function: $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ Convex functions have a single global minimum.

Deep networks are non-convex, but optimizers like Adam often find good solutions.

3.6 Learning Rate Scheduling

- Step decay
- Exponential decay
- Cosine annealing
- Warm restarts

3.7 Regularization and Constraints

- L1 Regularization (Lasso): adds $\lambda \sum |w|$ (sparsity)
- L2 Regularization (Ridge): adds $\lambda \sum w^2$ (smoothness)
- Early stopping: halts training to prevent overfitting

3.8 Loss Surfaces and Saddle Points

- Saddle point: gradient is zero but not a minimum
- Local minima vs global minima
- Flat minima often generalize better than sharp ones

3.9 Automatic Differentiation

Used by frameworks like TensorFlow and PyTorch to compute gradients through complex networks.

Summary

- Gradients and optimization minimize loss functions
- SGD and its variants scale to large datasets
- Regularization helps avoid overfitting
- Scheduling and initialization improve training

Applications:

- Training deep networks
- Fine-tuning pretrained models
- Hyperparameter tuning

4 Machine Learning Algorithms

Overview

Machine learning is about designing algorithms that improve from data. This section covers supervised and unsupervised learning methods, their math foundations, loss functions, and regularization techniques.

4.1 Types of Learning

- **Supervised Learning:** Learn from labeled data (X, y)
- **Unsupervised Learning:** Discover patterns from unlabeled data X
- **Semi-supervised, Reinforcement Learning:** Hybrid and decision-based learning

4.2 Linear Regression

Predicts a real-valued target:

$$\hat{y} = w^T x + b$$

Loss Function

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

Closed-form solution:

$$\hat{w} = (X^T X)^{-1} X^T y$$

4.3 Logistic Regression

Predicts binary outcomes:

$$\hat{y} = \sigma(w^T x + b), \quad \text{where } \sigma(z) = \frac{1}{1 + e^{-z}}$$

Loss (Binary Cross-Entropy):

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

4.4 Regularization

- **L1 (Lasso):** $\lambda \sum |w_i|$ encourages sparsity
- **L2 (Ridge):** $\lambda \sum w_i^2$ prevents large weights

Used to reduce overfitting.

4.5 Support Vector Machines (SVM)

Finds a hyperplane with the maximum margin between classes.

Hinge loss:

$$\mathcal{L} = \sum \max(0, 1 - y_i(w^T x_i + b))$$

4.6 Decision Trees

Split data by feature values to reduce impurity (Gini or Entropy):

$$\text{Entropy} = -\sum p_i \log(p_i)$$

4.7 K-Nearest Neighbors (KNN)

Classify based on majority label of k closest points. Uses distance metrics (e.g., Euclidean or Cosine).

4.8 Naive Bayes

Assumes conditional independence of features:

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

4.9 Unsupervised Learning

K-Means Clustering

$$\text{Objective: } \min \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

Principal Component Analysis (PCA)

Reduces dimensionality by projecting data onto top eigenvectors of the covariance matrix.

4.10 Evaluation Metrics

- Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$
- Precision: $\frac{TP}{TP+FP}$
- Recall: $\frac{TP}{TP+FN}$

- F1 Score: $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
- ROC-AUC, Log Loss

Summary

- ML algorithms find patterns and make predictions
- Loss functions guide learning
- Regularization and metrics ensure generalization

Applications:

- Credit scoring, spam detection, sentiment analysis
- Dimensionality reduction, anomaly detection

5 Deep Learning Foundations

Overview

Deep learning models are built from layers of neurons that can learn hierarchical features from data. This section covers the structure of neural networks, their training via backpropagation, and essential components like activations, normalization, and dropout.

5.1 Artificial Neural Networks (ANNs)

Each neuron performs:

$$z = w^T x + b \quad , \quad a = \phi(z)$$

Where:

- x: input
- w: weights
- b: bias
- ϕ: activation function (e.g., ReLU)

5.2 Activation Functions

- **Sigmoid:** $\sigma(x) = \frac{1}{1+e^{-x}}$
- **Tanh:** $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- **ReLU:** $f(x) = \max(0, x)$
- **Softmax:** Converts vector to probabilities

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

5.3 Forward Pass

The input passes through each layer:

$$a^{(l)} = \phi(W^{(l)} a^{(l-1)} + b^{(l)})$$

5.4 Backpropagation

Uses chain rule to compute gradient of the loss function:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

Steps

1. Compute loss: $\mathcal{L}$
2. Backpropagate from output to input
3. Update weights using gradients

5.5 Loss Functions

- **MSE (Regression):** $\frac{1}{n}\sum(y_i - \hat{y}_i)^2$
- **Cross-Entropy (Classification):**

$$-\sum y_i \log(\hat{y}_i)$$

5.6 Optimization Algorithms

- **SGD:** Gradient descent using one sample
- **Adam:** Adaptive learning + momentum
- **RMSProp, Adagrad:** Learning rate scaling

5.7 Regularization Techniques

- **L1/L2 Regularization:** Add penalty to loss
- **Dropout:** Randomly zero-out neurons during training
- **Early Stopping:** Stop training on validation error

5.8 Batch Normalization

Normalize intermediate activations:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

Improves stability and convergence speed.

5.9 Weight Initialization

- **Xavier (Glorot):** $\sim \mathcal{N}(0, \frac{1}{n})$
- **He Init (for ReLU):** $\sim \mathcal{N}(0, \frac{2}{n})$

5.10 Deep Network Architectures

- **Feedforward Networks**
- **Convolutional Neural Networks (CNNs)**
- **Recurrent Neural Networks (RNNs)**
- **Transformers**

Summary

- ANNs learn complex functions by stacking layers
- Backpropagation computes gradients efficiently
- Activations, normalization, and dropout boost performance

Applications:

- Image classification, NLP, speech recognition
- Generative models, transfer learning

6 Computer Vision

Overview

Computer Vision enables machines to understand visual data (images, videos). It builds upon linear algebra, optimization, and deep learning to extract patterns, classify objects, detect features, and more.

6.1 Images as Matrices

- Grayscale image: $H \times W$ matrix
- RGB image: $H \times W \times 3$ tensor
- Each pixel is an intensity value (0–255)

6.2 Convolutions and Kernels

Convolution operation:

$$S(i,j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} I(i+m, j+n) \cdot K(m,n)$$

Where I is the image patch and K is the kernel (e.g., edge detector).

Stride, Padding, Dilation

- **Stride:** step size of kernel
- **Padding:** border to preserve spatial size
- **Dilation:** expands receptive field

6.3 Pooling Layers

- **Max Pooling:** outputs maximum value in region
- **Average Pooling:** computes average value
- Reduces spatial dimensions, controls overfitting

6.4 CNN Architecture

- **Input Layer:** image matrix
- **Conv Layers:** apply kernels to extract features
- **Activation:** typically ReLU
- **Pooling Layers:** downsample
- **Fully Connected Layers:** final prediction

6.5 Feature Maps and Filters

Filters learn to detect features like:

- Edges
- Corners
- Textures
- Objects at higher layers

6.6 Transfer Learning

- Use pretrained models (e.g., ResNet, VGG)
- Fine-tune on your dataset
- Saves time and improves performance on small data

6.7 Image Embeddings and Cosine Similarity

- Embeddings = vector representations of images
- Compare embeddings using cosine similarity:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

- Used in image retrieval, face verification, etc.

6.8 Image Augmentation

- Techniques: Rotation, Flip, Crop, Zoom, Brightness
- Helps generalize better by simulating new data

6.9 Loss Functions for Vision

- **Cross-Entropy:** classification
- **Triplet Loss:** embeddings (e.g., FaceNet)
- **IoU Loss:** object detection

6.10 Advanced Applications

- **Object Detection:** YOLO, SSD, Faster R-CNN
- **Semantic Segmentation:** U-Net, DeepLab
- **Image Captioning:** CNN + RNN/Transformer
- **Generative Models:** GANs, Diffusion

Summary

- Images are tensors; convolution extracts patterns
- CNNs learn hierarchical representations
- Embeddings and pretrained models boost accuracy

Applications:

- Medical imaging, autonomous vehicles
- Surveillance, AR/VR, face recognition

7 Advanced Mathematical Topics

Overview

This section includes additional mathematical tools and techniques used in advanced ML, DL, and computer vision applications—especially for optimization, dimensionality reduction, generative models, and signal processing.

7.1 Principal Component Analysis (PCA)

- Reduces dimensionality by projecting data onto top eigenvectors of the covariance matrix.
- Steps:
  1. Center the data: $X \leftarrow X - \mu$
  2. Compute covariance: $\Sigma = \frac{1}{n} X^T X$
  3. Get eigenvectors $\vec{v}_i$ and eigenvalues λ_i
  4. Project onto top-k eigenvectors
- Used in noise reduction, compression, and visualization.

7.2 Singular Value Decomposition (SVD)

$$A = U \Sigma V^T$$

- Generalization of eigen-decomposition
- Used in PCA, Latent Semantic Analysis, image compression
- Σ: diagonal matrix with singular values (importance of features)

7.3 KL Divergence and Cross-Entropy

KL Divergence:

$$D_{\mathrm{KL}}(P \| Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

Cross-Entropy:

$$H(p, q) = -\sum_i p(x_i) \log q(x_i)$$

Used as a loss function in classification and in VAEs.

7.4 Fourier Transform (1D and 2D)

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

- Converts signal from time to frequency domain
- **2D Fourier Transform** used in image processing
- Helps with filtering, edge detection, and frequency analysis

7.5 Manifold Learning

- Data often lies on lower-dimensional manifolds
- Algorithms: t-SNE, UMAP, Isomap
- Good for visualization and nonlinear dimensionality reduction

7.6 Information Theory in ML

- **Entropy:** Uncertainty in a variable
- **Mutual Information:** Shared information between X and Y
- Used in feature selection, attention mechanisms, generative models

7.7 Bayesian Inference

- Updates beliefs using observed data:

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$$

- Used in Bayesian Neural Nets, Variational Inference

7.8 Lagrange Multipliers

Used to solve constrained optimization problems:

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

Useful in SVMs and energy-based models.

7.9 Matrix Calculus

- Essential for backpropagation in deep learning
- Gradients with respect to matrices and vectors:

$$\frac{d}{dX}\text{Tr}(AX) = A^T \quad , \quad \frac{d}{dX}\|X\|^2 = 2X$$

Summary

- PCA and SVD reduce dimensions and find structure
- KL divergence and entropy measure information
- Matrix calculus and Fourier transforms are vital in ML and CV

Applications:

- Embedding models, generative networks, image compression
- Feature extraction, optimization with constraints

8 Extended Topics

Extended Topics in Modern AI

8.1 Transformers and Attention Mechanism

Transformers replace recurrence with self-attention for sequence modeling.

Self-Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Multi-Head Attention

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W^O$$

Positional Encoding

$$\text{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad \text{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

Applications

- NLP: BERT, GPT
- CV: ViT, CLIP
- Multimodal: Image captioning, VQA

8.2 Gradient Stability Tricks

- **Gradient Clipping:** Prevents exploding gradients.
- **Weight Initialization:** (He, Xavier) helps convergence.
- **BatchNorm / LayerNorm:** Normalize activations.

8.3 Explainable AI (XAI)

- **Saliency Maps:** Pixel-level gradients.
- **Grad-CAM:** Heatmaps from final conv layers.
- **SHAP Values:** Based on Shapley value theory.
- **LIME:** Local interpretable approximations.

8.4 Generative Models

VAEs

Latent space sampled as:

$$z \sim \mathcal{N}(\mu, \sigma^2)$$

$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{\mathrm{KL}}(q(z|x)\|p(z))$$

GANs

$$\min_G \max_D \ \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]$$

8.5 Reinforcement Learning (Basics)

- **MDP:** (S, A, P, R, γ)
- **Q-learning:**
$$Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$$
- **Policy Gradients:**
$$\nabla J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) R]$$

8.6 Hyperparameter Optimization

- **Grid Search, Random Search**
- **Bayesian Optimization:** e.g., Gaussian Processes
- Libraries: `Optuna`, `Ray Tune`, `Hyperopt`

8.7 Time Series Forecasting

- **Stationarity:** Constant mean, variance over time
- **ARIMA:** Combines Auto-Regressive (AR), Integrated (I), Moving Average (MA)
- **LSTM, GRU:** Sequence-aware neural nets
- **Cross-Correlation:** Measure lag relationships

8.8 Bayesian Inference

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$$

- Posterior = Likelihood × Prior / Evidence
- Used in Bayesian neural nets and probabilistic programming

8.9 Matrix Calculus Essentials

- $$\frac{d}{dX}\text{Tr}(AX) = A^T \quad , \quad \frac{d}{dX}\|X\|^2 = 2X$$
- Jacobians: First-order derivatives
- Hessians: Second-order derivatives

8.10 Lagrange Multipliers

Used for constrained optimization:

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

- Solves optimization under constraints
- Used in SVMs, energy-based models

8.11 Model Evaluation Beyond Accuracy

- **ROC AUC, PR Curves**
- **Calibration Curves**
- **Confusion Matrix Analysis**
- **Statistical Testing:** t-test, p-values for comparison

Summary

- Modern AI models use attention, generative and probabilistic techniques.
- Optimization and interpretability tools enhance trust and performance.
- Time series, RL, and Bayesian reasoning extend ML's real-world use.