FIRST EDITION

MATHEMATICAL FOUNDATIONS OF NEURAL NETWORKS

From CNNs to PINNs, QINNs, and Deep Reinforcement Learning

RAPHAEL NORIEGA

Preface

For as long as I can remember, I have dreamed of writing a book where no step is skipped, where every concept finds its place in a logical chain, and where the reader is never left with the feeling that something essential has been hidden between the lines. Too often, in mathematics and in the sciences, explanations assume too much, or leave gaps that only those with prior training can fill. This book is my answer to that problem.

When I began teaching and researching neural networks, I was struck by how fragmented the explanations often were. Some texts went straight to the formulas, as if the reader should already know why those symbols mattered. Others remained on the surface, offering analogies without showing the rigorous backbone of mathematics that sustains the theory. The result is that many students, researchers, or professionals are left either overwhelmed by abstraction or undernourished by oversimplification. My conviction is that knowledge should not be fragmented: it should flow as a continuous path where each stone is visible and firmly placed for the next step.

That is the spirit of this book. Here, I aim to construct a narrative where the reader walks hand in hand with the mathematics, never jumping over a void of reasoning. Concepts are not presented as isolated facts, but as elements of a larger structure, built slowly with logic as mortar. The aim is not just to show *what* neural networks are, but to reveal *why* they are built the way they are, and how mathematics—linear algebra, calculus, probability, optimization—becomes the invisible skeleton that gives them life.

This is a book for a wide audience. It is written for the undergraduate student taking their first steps in artificial intelligence, for the graduate researcher who needs a rigorous reference, and for professionals in the industry who want to understand the deeper principles behind the tools they use. My hope is that this book will serve as both a guide and a reference: a text you can study systematically from beginning to end, but also one you can revisit to clarify details or to build new insights.

Writing this book has also been a personal journey. I wanted to produce something that stands apart from textbooks that either rush through derivations or leave entire arguments as an exercise for the reader. This will not be a book of shortcuts. It will demand effort. There will be formulas, derivations, and proofs. But I promise that every line is there for a reason. Nothing is left to chance, and nothing is presented without context. If you walk with me through these pages, you will not only learn what neural networks are and how they work—you will also discover the rhythm and poetry of mathematics itself, the logic that binds the abstract to the real.

Finally, this book is also a gesture of trust. Trust in the reader's intelligence, in their patience, and in their hunger for depth. I have chosen not to underestimate you, whoever you are, because I believe that clarity is not about removing complexity but about guiding through it.

May these pages serve as a bridge: between mathematics and intuition, between logic and imagination, and between the academic world and real-world applications.

Contents

Part I.

Mathematical Preliminaries

Introduction 1.

1.1. Historical Context of AI and Neural Networks

The story of neural networks is inseparable from the broader history of artificial intelligence. In the mid-twentieth century, pioneers began to ask a radical question: could machines learn, reason, and perhaps even think?

Early models of computation and the brain appeared in the 1940s and 1950s. Warren McCulloch and Walter Pitts proposed a mathematical model of the neuron, reducing it to a binary threshold unit. Their work suggested that networks of such units could, in principle, compute any logical function. Around the same time, Alan Turing speculated about "learning machines," planting seeds that would later grow into the foundations of AI.

The perceptron era began in 1957, when Frank Rosenblatt introduced a trainable model capable of learning linear decision boundaries. It captured the imagination of both scientists and the public, sparking optimism that machines could soon replicate the brain's capacity for learning. Yet mathematics also revealed the perceptron's limits. In the late 1960s, Marvin Minsky and Seymour Papert proved that single-layer perceptrons could not solve even simple nonlinear problems such as XOR. This was a sobering reminder that without mathematical rigor, bold claims collapse under scrutiny.

These early attempts reveal an important lesson: science moves in cycles of enthusiasm and skepticism. Each generation rediscovers that true progress requires a marriage between creative vision and mathematical clarity.

1.1.1. Early visions of artificial intelligence

1.1.2. Symbolic AI and rule-based reasoning (1950s–1970s)

1.1.3. Cybernetics and control theory foundations

1.1.4. Birth of the perceptron and connectionism

1.1.5. Shift from handcrafted rules to learning from data

1.2. AI Winters and the Deep Learning Revolution

The collapse of optimism after the perceptron marked the first "AI winter" of the 1970s. Funding dried up, and public interest waned. Limited computing power, scarce data, and inflated promises led many to dismiss neural networks as a dead end. A second AI winter followed in the late

1980s, as symbolic methods, once thought to be the future of AI, also struggled to deliver.

Yet beneath the surface, mathematics was preparing a renaissance. In the 1980s, the backpropagation algorithm was formalized and popularized, allowing multilayer perceptrons to model complex nonlinear functions. Still, adoption was slow, because hardware had not yet caught up with theory. Neural networks were powerful on paper, but impractical in real-world applications.

The deep learning explosion of the 2010s changed everything. With the rise of GPUs, massive datasets, and architectures such as convolutional and recurrent networks, machines suddenly outperformed classical methods in vision, language, and speech. Soon after, transformers redefined the field altogether, enabling large-scale models that blurred the line between statistics and creativity. At the core of these breakthroughs was not magic, but mathematics: linear algebra for representation, probability for modeling uncertainty, and optimization theory for training vast networks.

The history of neural networks is therefore not merely technical—it is also the history of human patience. What once looked like failure was in fact a pause, waiting for mathematics and technology to converge.

1.2.1. The first AI winter: funding collapse in the 1970s

1.2.2. The second AI winter: overpromises and underdelivery

1.2.3. The comeback: backpropagation and multilayer perceptrons

1.2.4. The ImageNet moment and deep convolutional networks

1.2.5. Acceleration through GPUs, big data, and open-source ecosystems

1.3. Innovation and Impact on the World

1.3.1. Transformation of computer vision and natural language processing

1.3.2. AI-driven automation in industry and science

1.3.3. Deep learning in healthcare, robotics, and materials discovery

1.3.4. Societal, ethical, and economic implications

1.3.5. AI as a new general-purpose technology

1.4. Why Mathematics Matters in Deep Learning

If philosophy gave us the first questions about intelligence, mathematics gave us the tools to answer them. Galileo once wrote that the universe "is written in the language of mathematics, and its characters are triangles, circles, and other geometrical figures." In the same spirit, deep learning is written in the language of vectors, matrices, and functions. Every model is a translation from the world's complexity into mathematical form, and every training process is an attempt to solve an equation that nature has posed.

Linear algebra is the grammar of representation, calculus is the machinery of change, probability is the measure of uncertainty, and optimization is the path to improvement. Without them, neural networks would be shapeless intuitions. With them, they become structured systems capable of learning patterns from the world.

Mathematics is therefore not a peripheral tool but the very skeleton of deep learning. It gives rigor to vision, coherence to creativity, and structure to intuition. Just as the Greeks once sought logic to discipline thought, we now rely on mathematics to discipline learning.

This book begins here—at the intersection of history, philosophy, and mathematics—because to understand neural networks is not merely to know how they work, but to see them as part of humanity's timeless attempt to comprehend intelligence itself.

1.4.1. Mathematics as the foundation of generalization

1.4.2. Linear algebra, calculus, and probability as core pillars

1.4.3. Optimization theory behind learning algorithms

1.4.4. Mathematical guarantees vs empirical success

1.4.5. Bridging theory, practice, and innovation

Linear Algebra Essentials 2.

2.1. Vector spaces and inner products

2.1.1. Vector spaces and subspaces

2.1.2. Basis and dimension

2.1.3. Linear independence

2.1.4. Inner product and norms

2.1.5. Orthogonality and orthonormal bases

2.2. Eigenvalues, eigenvectors, diagonalization

2.2.1. Eigenvalues and eigenvectors

2.2.2. Characteristic polynomial

2.2.3. Diagonalization of matrices

2.2.4. Spectral theorem

2.2.5. Applications to stability and data analysis

2.3. Singular value decomposition (SVD)

2.3.1. Matrix factorization concepts

2.3.2. Singular values and orthogonal matrices

2.3.3. Rank and null space

2.3.4. Low-rank approximations

2.3.5. PCA and compression

2.4. Tensor notation and operations

2.4.1. Scalars, vectors, matrices, tensors

2.4.2. Tensor indexing and ranks

2.4.3. Tensor contraction and Einstein summation

2.4.4. Broadcasting and reshaping

2.4.5. Tensors in backpropagation and GPU computation

3. Multivariable Calculus and Analysis

3.1. Gradients, Jacobians, Hessians

3.1.1. Partial derivatives and notation

3.1.2. Gradient vectors and directional derivatives

3.1.3. Jacobian matrices for vector-valued functions

3.1.4. Hessian matrices and second-order derivatives

3.1.5. Applications in optimization and backpropagation

3.2. Taylor expansions in multiple variables

3.2.1. Multivariate Taylor series formula

3.2.2. Linear and quadratic approximations

3.2.3. Error terms and convergence

3.2.4. Applications to loss landscapes

3.3. Divergence, curl, and Laplacians

3.3.1. Vector fields and notation

3.3.2. Divergence and its interpretation

3.3.3. Curl and rotational fields

3.3.4. Laplacian operator and harmonic functions

3.3.5. Applications to PDEs and diffusion

3.4. Variational principles

3.4.1. Functional derivatives

3.4.2. Euler–Lagrange equations

3.4.3. Variational methods for optimization

3.4.4. Applications in physics-informed neural networks (PINNs)

4. Probability, Statistics, and Information Theory

4.1. Random variables and distributions

4.1.1. Sample spaces and events

4.1.2. Discrete and continuous random variables

4.1.3. Probability mass and density functions (PMF, PDF)

4.1.4. Cumulative distribution functions (CDF)

4.1.5. Joint, marginal, and conditional distributions

4.2. Expectation, variance, covariance

4.2.1. Expected value of random variables

4.2.2. Variance and standard deviation

4.2.3. Covariance and correlation

4.2.4. Covariance matrices

4.2.5. Law of large numbers and central limit theorem

4.3. Gaussian and exponential families

4.3.1. Gaussian (normal) distribution properties

4.3.2. Multivariate Gaussian distribution

4.3.3. Exponential family definition and structure

4.3.4. Bernoulli, Binomial, Poisson, and Exponential as exponential-family members

4.3.5. Maximum likelihood under exponential families

4.4. Entropy, KL divergence, mutual information

4.4.1. Shannon entropy and information content

4.4.2. Cross-entropy and its link to loss functions

4.4.3. Kullback–Leibler divergence

4.4.4. Mutual information

4.4.5. Applications in machine learning (regularization, uncertainty, feature selection)

5. Optimization Theory

5.1. Convexity, duality, and Lagrangians

5.1.1. Convex sets and convex functions

5.1.2. First- and second-order conditions for convexity

5.1.3. Lagrange multipliers

5.1.4. Karush–Kuhn–Tucker (KKT) conditions

5.1.5. Strong and weak duality

5.2. Gradient descent and its variants

5.2.1. Steepest descent method

5.2.2. Learning rate selection and scheduling

5.2.3. Momentum-based methods

5.2.4. Nesterov accelerated gradient (NAG)

5.2.5. Adaptive methods (AdaGrad, RMSProp, Adam)

5.3. Stochastic optimization and convergence

5.3.1. Stochastic gradient descent (SGD)

5.3.2. Mini-batch training

5.3.3. Convergence criteria and analysis

5.3.4. Variance reduction techniques

5.3.5. Generalization vs optimization trade-offs

5.4. Newton and quasi-Newton methods

5.4.1. Newton's method for optimization

5.4.2. Hessian-based updates

5.4.3. BFGS and L-BFGS algorithms

5.4.4. Trust-region methods

5.4.5. Comparisons with first-order methods

5.5. Variational Principles in Optimization and

Functional Analysis Foundations 6.

6.1. Normed spaces, Banach and Hilbert spaces

6.1.1. Normed vector spaces and examples

6.1.2. Completeness and Banach spaces

6.1.3. Inner product spaces

6.1.4. Hilbert spaces and orthonormal bases

6.1.5. Riesz representation theorem

6.2. Orthogonal polynomials (Hermite, Laguerre, Legendre)

6.2.1. Orthogonality conditions and weight functions

6.2.2. Hermite polynomials and Gaussian weighting

6.2.3. Laguerre polynomials and exponential weighting

6.2.4. Legendre polynomials and spherical harmonics

6.2.5. Applications in spectral methods

6.3. Special functions (Gamma, Beta, Bessel)

6.3.1. Gamma and Beta functions definitions and properties

6.3.2. Relationship with factorials and combinatorics

6.3.3. Bessel functions and their differential equations

6.3.4. Asymptotic behavior and orthogonality

6.3.5. Applications in physics and PDEs

6.4. Operator theory foundations for PINNs and QINNs

6.4.1. Linear operators and boundedness

6.4.2. Self-adjoint, unitary, and compact operators

6.4.3. Spectral theory of operators

6.4.4. Differential operators in Hilbert spaces

Part II.

Classical Machine Learning

7. Foundations of Machine Learning

7.1. What is Machine Learning?

7.1.1. Definitions and scope

7.1.2. Learning from data vs explicit programming

7.1.3. Types of tasks: classification, regression, clustering, control

7.1.4. Data-driven modeling and generalization

7.2. Historical Background and Paradigms

7.2.1. Origins in statistics and computer science

7.2.1.1. Early statistical learning and pattern recognition

7.2.1.2. Symbolic AI and expert systems

7.2.1.3. Connectionism and the perceptron era

7.2.2. Supervised, unsupervised, and reinforcement learning

7.2.2.1. Supervised learning paradigm

7.2.2.2. Unsupervised learning paradigm

7.2.2.3. Reinforcement learning paradigm

7.2.2.4. Key differences and overlaps among paradigms

Core Algorithms of Classical ML | 8.

8.1. Linear and Logistic Regression

8.1.1. Linear regression model formulation

8.1.2. Least squares estimation

8.1.3. Regularization (Ridge and Lasso)

8.1.4. Logistic regression for classification

8.1.5. Decision boundaries and sigmoid function

8.1.6. Evaluation metrics (accuracy, ROC, AUC)

8.2. Support Vector Machines (SVMs) and Kernels

8.2.1. Maximum margin classifiers

8.2.2. Soft margin SVMs

8.2.3. Kernel trick and feature mapping

8.2.4. Common kernels (linear, polynomial, RBF)

8.2.5. Support vector regression (SVR)

8.2.6. Computational considerations

8.3. Decision Trees, Random Forests, and Boosting

8.3.1. Decision tree construction (ID3, CART)

8.3.2. Overfitting and pruning

8.3.3. Bagging and random forests

8.3.4. Boosting algorithms (AdaBoost, Gradient Boosting, XGBoost)

8.3.5. Feature importance and interpretability

Probabilistic Models and Bayesian Learning | 9.

9.1. Naïve Bayes Classifiers

9.1.1. Bayes' theorem recap

9.1.2. Conditional independence assumption

9.1.3. Training and parameter estimation

9.1.4. Multinomial and Gaussian Naïve Bayes

9.1.5. Applications and limitations

9.2. Gaussian Mixture Models (GMMs) and the EM Algorithm

9.2.1. Mixture models and latent variables

9.2.2. Expectation-Maximization (EM) algorithm

9.2.3. E-step and M-step derivations

9.2.4. Convergence and initialization strategies

9.2.5. Applications to clustering and density estimation

9.3. Bayesian Inference: Priors, Posteriors, and Evidence

9.3.1. Prior distributions and subjective knowledge

9.3.2. Likelihood functions

9.3.3. Posterior distributions via Bayes' rule

9.3.4. Evidence (marginal likelihood) and model comparison

9.3.5. Bayesian updating and sequential inference

10 Dimensionality Reduction and Unsupervised Learning

10.1. Principal Component Analysis (PCA)

10.1.1. Covariance matrix and eigendecomposition

10.1.2. Dimensionality reduction via projection

10.1.3. Explained variance and component selection

10.1.4. Whitening and data preprocessing

10.1.5. Applications in visualization and noise reduction

10.2. Kernel PCA and Nonlinear Manifolds

10.2.1. Limitations of linear PCA

10.2.2. Kernel trick for nonlinear feature mapping

10.2.3. Kernel matrix (Gram matrix) construction

10.2.4. Eigen decomposition in feature space

10.2.5. Manifold interpretation and applications

10.3. Clustering Algorithms

10.3.1. k-means

10.3.1.1. Centroid initialization and updates

10.3.1.2. Convergence and limitations

10.3.2. Hierarchical Clustering

10.3.2.1. Agglomerative vs divisive approaches

10.3.2.2. Linkage criteria and dendrograms

10.3.3. DBSCAN

10.3.3.1. Density-based clustering principle

10.3.3.2. Parameters: ε and minPts

10.3.3.3. Handling noise and arbitrary-shaped clusters

Learning Theory and Generalization 11.

11.1. VC Dimension and Shattering

11.1.1. Hypothesis classes and expressiveness

11.1.2. Shattering sets of points

11.1.3. Definition and examples of VC dimension

11.1.4. VC dimension and overfitting risk

11.2. PAC Learning Framework

11.2.1. Probably Approximately Correct (PAC) definition

11.2.2. Sample complexity bounds

11.2.3. Realizable vs agnostic PAC learning

11.2.4. PAC learnability and generalization guarantees

11.3. Rademacher Complexity

11.3.1. Definition and intuition

11.3.2. Empirical Rademacher complexity

11.3.3. Relation to uniform convergence

11.3.4. Bounds on generalization error

11.4. Bias–Variance Trade-Off

11.4.1. Decomposing expected error

11.4.2. High bias (underfitting) vs high variance (overfitting)

11.4.3. Regularization and model complexity

11.4.4. Implications for model selection

12. Optimization in Machine Learning

12.1. Least Squares and Maximum Likelihood

12.1.1. Ordinary least squares formulation

12.1.2. Normal equations and closed-form solution

12.1.3. Maximum likelihood estimation (MLE)

12.1.4. Connection between least squares and Gaussian likelihood

12.1.5. Log-likelihood and optimization objectives

12.2. Convex vs. Non-Convex Optimization

12.2.1. Convex sets and convex functions

12.2.2. Properties of convex optimization problems

12.2.3. Challenges of non-convex landscapes

12.2.4. Local minima vs global minima

12.2.5. Saddle points and flat regions

12.3. Gradient Descent and Variants

12.3.1. Batch gradient descent

12.3.2. Stochastic gradient descent (SGD)

12.3.3. Mini-batch gradient descent

12.3.4. Learning rate scheduling

12.3.5. Momentum, RMSProp, and Adam optimizers

13. Transition to Neural Networks

13.1. Limitations of Classical ML

13.1.1. Dependence on feature engineering

13.1.2. Scalability issues with high-dimensional data

13.1.3. Difficulty handling unstructured data (images, text, audio)

13.1.4. Limited capacity for hierarchical representations

13.1.5. Performance saturation with more data

13.2. Feature Engineering vs. Representation Learning

13.2.1. Manual feature extraction approaches

13.2.2. Representation learning definition

13.2.3. Advantages of learned hierarchical features

13.2.4. Examples in computer vision and NLP

13.2.5. Shift from hand-crafted to learned representations

13.3. The Bridge to Perceptrons and Deep Learning

13.3.1. Introduction to artificial neurons

13.3.2. Perceptron architecture and learning rule

13.3.3. Multilayer perceptrons (MLPs) and hidden layers

13.3.4. From shallow to deep networks

13.3.5. The deep learning paradigm shift

Part III.

Classical Neural Networks

14. Foundations of Neural Networks

14.1. What is a Neural Network?

14.2. Biological Inspiration vs. Mathematical Abstraction

14.3. Basic Structure: Neurons, Layers, and Activations

14.4. From Perceptrons to Modern Architectures (preview)

14.5. Backpropagation and Automatic Differentiation

14.5.1. Computational Graphs and the Chain Rule

14.5.2. Forward-Mode vs. Reverse-Mode AD

14.5.3. Backpropagation Algorithm

14.5.4. Hessian–Vector Products

15. The Perceptron and Linear Models

15.1. McCulloch–Pitts Neurons

15.1.1. Biological inspiration and abstraction

15.1.2. Binary threshold units

15.1.3. Logical operations with MP neurons

15.1.4. Limitations and historical impact

15.2. Rosenblatt's Perceptron and Linear Separability

15.2.1. Perceptron model architecture

15.2.2. Learning rule and weight updates

15.2.3. Geometric interpretation of linear separability

15.2.4. Perceptron convergence theorem

15.2.5. Limitations: XOR problem and need for hidden layers

15.3. Logistic Regression as Probabilistic Perceptron

15.3.1. Sigmoid activation and probabilistic outputs

15.3.2. Log-likelihood formulation

15.3.3. Gradient-based training

15.3.4. Decision boundaries and interpretation

15.3.5. Connection to neural networks

Feedforward Networks (MLPs) 16.

17 Convolutional Neural Networks (CNNs)

17.1. Mathematical Basis of Convolution

17.1.1. Discrete convolution definition

17.1.2. Cross-correlation vs convolution

17.1.3. Properties of convolution (linearity, shift-invariance)

17.1.4. 2D convolution for image data

17.1.5. Backpropagation through convolutional layers

17.2. Feature Maps, Receptive Fields, Pooling

17.2.1. Concept of receptive field

17.2.2. Feature map construction and interpretation

17.2.3. Stride, padding, and dilation

17.2.4. Pooling operations (max, average, global)

17.2.5. Effect on translation invariance and dimensionality reduction

17.3. Modern CNN Architectures (AlexNet, VGG, ResNet)

17.3.1. AlexNet: deep convolution revival

17.3.2. VGG: simplicity and depth

17.3.3. ResNet: residual connections and very deep networks

17.3.4. Other influential models (Inception, DenseNet)

17.3.5. Trends in modern convolutional design

17.4. Applications in Vision, Audio, and Physics

17.4.1. Image classification and object detection

17.4.2. Semantic segmentation and medical imaging

17.4.3. Audio spectrogram analysis

Recurrent Neural Networks (RNNs) | 18

18.1. Sequences as Dynamical Systems

18.1.1. Sequence modeling motivation

18.1.2. Recurrent computation and state updates

18.1.3. Unrolling through time

18.1.4. Training with backpropagation through time (BPTT)

18.1.5. Limitations of basic RNNs

18.2. Gradient Vanishing and Exploding

18.2.1. Analysis of gradient propagation over time steps

18.2.2. Vanishing gradients and long-term dependencies

18.2.3. Exploding gradients and instability

18.2.4. Gradient clipping techniques

18.2.5. Initialization strategies to mitigate gradient issues

18.3. LSTMs and GRUs

18.3.1. Long Short-Term Memory (LSTM) architecture

18.3.2. Input, forget, and output gates

18.3.3. Gated Recurrent Unit (GRU) architecture

18.3.4. Comparison of LSTMs and GRUs

18.3.5. Handling long-term dependencies

18.4. Applications in NLP, Speech, Time-Series

18.4.1. Language modeling and text generation

18.4.2. Machine translation

18.4.3. Speech recognition and synthesis

18.4.4. Time-series forecasting

18.4.5. Sensor data and sequence classification

Autoencoders and Representation Learning 19

19.1. Linear Autoencoders and PCA

19.1.1. Autoencoder architecture and objective

19.1.2. Relation between linear autoencoders and PCA

19.1.3. Reconstruction error minimization

19.1.4. Dimensionality reduction and compression

19.2. Nonlinear Autoencoders

19.2.1. Nonlinear activation functions

19.2.2. Deep autoencoder architectures

19.2.3. Training strategies and regularization

19.2.4. Denoising and sparse autoencoders

19.2.5. Applications in feature learning

19.3. Variational Autoencoders (VAEs)

19.3.1. Latent variable models

19.3.2. Probabilistic encoder and decoder networks

19.3.3. Reparameterization trick

19.3.4. Evidence lower bound (ELBO)

19.3.5. Generative modeling and sampling

19.4. Latent Space Geometry

19.4.1. Structure of learned latent spaces

19.4.2. Interpolation and disentanglement

19.4.3. Manifold hypothesis

19.4.4. Visualization of latent spaces (t-SNE, UMAP)

19.4.5. Applications in generative and representation learning

Graph Neural Networks (GNNs) 20

20.1. Graph Laplacians and Spectral Methods

20.1.1. Graphs: nodes, edges, adjacency and degree matrices

20.1.2. Definition of the graph Laplacian

20.1.3. Normalized Laplacian and its properties

20.1.4. Spectral graph theory basics

20.1.5. Graph Fourier transform and convolution

20.2. Message Passing Frameworks

20.2.1. Message passing neural network (MPNN) paradigm

20.2.2. Aggregation and update functions

20.2.3. Neighborhood sampling strategies

20.2.4. Graph convolutional networks (GCNs)

20.2.5. Graph attention networks (GATs)

20.3. Applications in Chemistry, Materials, Biology

20.3.1. Molecular property prediction

20.3.2. Protein structure and interaction networks

20.3.3. Materials informatics and crystal graphs

20.3.4. Drug discovery and bioinformatics

20.3.5. Social and knowledge graph analysis

20.4. Spectral Methods and Eigen-Decomposition in Graphs

20.4.1. Eigen-decomposition of the Laplacian

20.4.2. Graph embedding via spectral methods

20.4.3. Chebyshev polynomials for fast spectral filtering

Part IV.

Neural Networks for Differential Equations

Mathematical Methods for Differential Equations 21

21.1. Classification of ODEs and PDEs

21.1.1. Ordinary vs partial differential equations

21.1.2. Linear vs nonlinear equations

21.1.3. Order and degree of a differential equation

21.1.4. Initial value vs boundary value problems

21.1.5. Well-posedness and existence theorems

21.2. Boundary and Initial Conditions

21.2.1. Dirichlet and Neumann boundary conditions

21.2.2. Robin (mixed) boundary conditions

21.2.3. Initial conditions in time-dependent problems

21.2.4. Physical interpretations of boundary data

21.3. Separation of Variables

21.3.1. Method for solving linear PDEs

21.3.2. Eigenfunction expansions

21.3.3. Application to heat, wave, and Laplace equations

21.4. Sturm–Liouville Problems and Orthogonal Expansions

21.4.1. Sturm–Liouville operator form

21.4.2. Orthogonality of eigenfunctions

21.4.3. Weight functions and inner products

21.4.4. Fourier series as a Sturm–Liouville system

21.5. Fourier and Laplace Transforms

21.5.1. Fourier series and Fourier transform

21.5.2. Laplace transform and inverse transform

21.5.3. Solving ODEs and PDEs with transforms

Physics-Informed Neural Networks (PINNs) 22

22.1. Embedding PDEs into loss functions

22.2. Collocation and weak formulations

22.3. Elliptic, parabolic, and hyperbolic PDEs

22.4. Applications: fluids, electromagnetism, quantum mechanics

22.5. Extensions: XPINNs, VPINNs, Bayesian PINNs

22.6. Inverse problems (intro to iPINNs)

22.7. Quantum Foundations for PINNs

22.7.1. Postulates of quantum mechanics and Hilbert spaces

22.7.2. Operators, commutators, and observables

22.7.3. Dirac notation and basis projections

22.8. The Schrödinger Equation

22.8.1. TISE and TDSE: formulation and nondimensionalization

22.8.2. Boundary conditions and wavefunction normalization

22.8.3. Variational and Hellmann–Feynman theorems

22.9. Model Problems for Quantum PINNs

22.9.1. 1D infinite well

22.9.2. 1D harmonic oscillator

22.9.3. Hydrogen atom: spherical separation and special functions

22.10. Loss Function Design for Schrödinger PINNs

Inverse Physics-Informed Neural Networks (iPINNs) 23.

24 Quantum Neural Networks (QINNs)

24.1. Hilbert spaces and Dirac notation

24.2. Quantum perceptron and gates as layers

24.3. Variational quantum circuits (VQE, QAOA)

24.4. Quantum Boltzmann Machines, QCNNs, Quantum Reservoirs

24.5. Parameter-shift rule for gradients

24.6. Challenges: barren plateaus, NISQ hardware

24.7. Applications in optimization, chemistry, cryptography

24.7.1. Inverse Quantum Neural Networks (iQINNs)

24.7.1.1. Hamiltonian parameter estimation

24.7.1.2. Quantum state tomography with QNNs

24.7.1.3. Physics-informed cost functions and unitarity constraints

24.7.2. Training QINNs for Schrödinger Problems

24.7.2.1. Parameter-shift gradients, expectation and variance estimation

24.7.2.2. Variational losses (VQE) for bound states

24.7.2.3. QAOA/ansätze for spin and tight-binding models

24.7.3. Canonical Test Problems

24.7.3.1. Harmonic oscillator, infinite well, double well

24.7.3.2. Hydrogen/Helium effective models with discrete bases

24.7.3.3. Scaling to multi-electron systems

24.7.4. Robustness and Barren Plateaus

24.7.4.1. Hardware-efficient vs. problem-inspired ansätze

24.7.4.2. Noise mitigation and measurement averaging on NISQ

Neural Operators and DeepONets 25

25.1. Learning operators between function spaces

25.1.1. Motivation and concept of operator learning

25.1.2. Function spaces, Banach/Hilbert foundations

25.1.3. Universal approximation of operators

25.2. Comparison with PINNs, iPINNs, FEM

25.2.1. Data requirements and generalization

25.2.2. Computational complexity and scalability

25.2.3. Accuracy on out-of-distribution regimes

25.2.4. Hybrid models: combining operators and physics-informed loss

25.3. DeepONet Architectures and Training

25.3.1. Branch and trunk networks

25.3.2. Basis function representation and integration layers

25.3.3. Loss functions and operator regression

25.3.4. Generalization bounds and convergence theory

25.4. Applications in PDEs and scientific computing

25.4.1. Elliptic, parabolic, and hyperbolic PDEs

25.4.2. Parametric PDE families and meta-learning

25.4.3. Multiscale and multiphysics problems

25.4.4. Real-time surrogate modeling in engineering and physics

25.5. Extensions and Future Directions

25.5.1. Fourier Neural Operators (FNOs)

25.5.2. Graph Neural Operators (GNOs)

Part V.

Reinforcement Learning

26 Classical Reinforcement Learning

26.1. Agents, Environments, States, Actions, Rewards

26.1.1. Reinforcement learning problem setup

26.1.2. Agent-environment interaction loop

26.1.3. Definition of states, actions, and rewards

26.1.4. Exploration vs exploitation

26.1.5. Episodic vs continuing tasks

26.2. Markov Decision Processes (MDPs)

26.2.1. Markov property and state transitions

26.2.2. Transition probability matrices

26.2.3. Reward functions and discount factors

26.2.4. Policy definition and representation

26.2.5. Formulating RL as an MDP

26.3. Value Functions and Bellman Equations

26.3.1. State-value and action-value functions

26.3.2. Bellman expectation equations

26.3.3. Bellman optimality equations

26.3.4. Relationship between value functions and policies

26.3.5. Policy evaluation and improvement

26.4. Tabular Methods: SARSA, Q-Learning

26.4.1. On-policy learning with SARSA

26.4.2. Off-policy learning with Q-learning

26.4.3. Temporal difference (TD) learning

26.4.4. Exploration strategies (epsilon-greedy, softmax)

26.4.5. Convergence properties of tabular methods

Deep Reinforcement Learning | 27

27.1. Deep Q-Networks (DQN)

27.1.1. Limitations of tabular Q-learning

27.1.2. Neural network function approximation for Q-values

27.1.3. Experience replay and target networks

27.1.4. Training stability techniques

27.1.5. Extensions: Double DQN, Dueling DQN, Prioritized Replay

27.2. Policy Gradient Methods (REINFORCE, PPO)

27.2.1. Policy-based vs value-based approaches

27.2.2. REINFORCE algorithm and derivation

27.2.3. High variance and variance reduction techniques

27.2.4. Proximal Policy Optimization (PPO) algorithm

27.2.5. Trust region methods and clipping objectives

27.3. Actor–Critic Architectures (A2C, A3C)

27.3.1. Concept of actor and critic networks

27.3.2. Advantage functions and baseline subtraction

27.3.3. Asynchronous advantage actor-critic (A3C)

27.3.4. Synchronous advantage actor-critic (A2C)

27.3.5. Sample efficiency and stability considerations

27.4. Landmark Systems: AlphaGo, AlphaZero, MuZero

27.4.1. AlphaGo: combining deep neural networks and MCTS

27.4.2. AlphaZero: self-play reinforcement learning

27.4.3. MuZero: learning dynamics models from scratch

Part VI.

Modern Architectures

Generative Models 28

28.1. GANs and Minimax Optimization

28.1.1. Generative vs discriminative models

28.1.2. Adversarial training framework

28.1.3. Minimax optimization objective

28.1.4. Training instability and mode collapse

28.1.5. Evaluation metrics (FID, IS)

28.2. Wasserstein GANs, StyleGAN

28.2.1. Wasserstein distance and Earth Mover's metric

28.2.2. WGAN training stability improvements

28.2.3. Gradient penalty techniques

28.2.4. Style-based generator architecture (StyleGAN)

28.2.5. High-resolution image synthesis

28.3. Diffusion Models and Stochastic Processes

28.3.1. Denoising diffusion probabilistic models (DDPMs)

28.3.2. Forward and reverse diffusion processes

28.3.3. Score-based generative modeling

28.3.4. Sampling strategies and efficiency

28.3.5. Comparisons with GANs and VAEs

28.4. Applications in Synthesis and Design

28.4.1. Image and video generation

28.4.2. Text-to-image and multimodal synthesis

28.4.3. Molecule and material design

28.4.4. Creative content and art generation

28.4.5. Data augmentation for machine learning

29 Transformers and Attention Mechanisms

29.1. Self-Attention: Queries, Keys, Values

29.1.1. Motivation for attention over recurrence

29.1.2. Query-key-value formulation

29.1.3. Scaled dot-product attention

29.1.4. Attention weights and softmax normalization

29.1.5. Computational complexity considerations

29.2. Multi-Head Attention

29.2.1. Parallel attention heads

29.2.2. Linear projections and concatenation

29.2.3. Benefits of multi-head structure

29.2.4. Implementation details

29.2.5. Visualization and interpretability

29.3. Positional Encodings

29.3.1. Need for positional information in sequences

29.3.2. Sinusoidal positional encodings

29.3.3. Learned positional embeddings

29.3.4. Incorporation into transformer architecture

29.3.5. Impact on long-range dependencies

29.4. Transformer Architectures: BERT, GPT, Multimodal

29.4.1. Encoder-decoder structure of the original Transformer

29.4.2. BERT: bidirectional encoder representations

29.4.3. GPT: autoregressive decoder-only models

29.4.4. Vision transformers (ViTs) and multimodal transformers

Part VII.

Advanced Topics

30. Optimization Beyond Gradient Descent

30.1. Variational Inference

30.1.1. Bayesian inference challenges and motivation

30.1.2. Evidence lower bound (ELBO)

30.1.3. Mean-field approximation

30.1.4. Coordinate ascent variational inference (CAVI)

30.1.5. Applications in probabilistic deep models

30.2. Expectation–Maximization (EM)

30.2.1. Latent variable models and incomplete data

30.2.2. E-step and M-step derivations

30.2.3. Convergence properties of EM

30.2.4. EM for Gaussian mixture models

30.2.5. Generalizations and variants of EM

30.3. Federated Optimization Challenges

30.3.1. Federated learning paradigm and architecture

30.3.2. Data heterogeneity and non-iid distributions

30.3.3. Communication constraints and efficiency

30.3.4. Privacy and security considerations

30.3.5. Optimization algorithms for federated settings (FedAvg, FedProx)

31 Mathematical Frontiers of Neural Networks

31.1. Neural Tangent Kernels (NTK)

31.1.1. Definition and derivation of NTK

31.1.2. Linearization of neural networks at initialization

31.1.3. Connection between NTK and gradient descent dynamics

31.1.4. Applications of NTK to generalization analysis

31.1.5. Limitations and current research directions

31.2. Infinite-Width Limits and Mean-Field Theory

31.2.1. Neural networks as infinite-width Gaussian processes

31.2.2. Mean-field limit of gradient descent dynamics

31.2.3. Law of large numbers in parameter distributions

31.2.4. Implications for training dynamics and convergence

31.2.5. Bridging finite- and infinite-width behaviors

31.3. Geometry of Loss Landscapes

31.3.1. Critical points and saddle points

31.3.2. Flat vs sharp minima

31.3.3. Hessian spectrum analysis

31.3.4. Mode connectivity and loss basins

31.3.5. Implications for optimization and generalization

31.4. Generalization Bounds and Capacity

31.4.1. Capacity measures: VC dimension, Rademacher complexity

31.4.2. Norm-based generalization bounds

31.4.3. PAC-Bayesian bounds for deep networks

32. Meta-Learning and Transfer Learning

32.1. Few-Shot Learning

32.1.1. Problem formulation and motivation

32.1.2. Metric-based meta-learning (Siamese networks, prototypical networks)

32.1.3. Optimization-based meta-learning (MAML)

32.1.4. Memory-augmented meta-learning models

32.1.5. Evaluation benchmarks for few-shot learning

32.2. Pretraining and Fine-Tuning

32.2.1. Transfer learning paradigm

32.2.2. Feature extraction vs full fine-tuning

32.2.3. Domain adaptation techniques

32.2.4. Multitask and multi-domain pretraining

32.2.5. Scaling laws and large pretrained models

32.3. Continual Learning

32.3.1. Catastrophic forgetting problem

32.3.2. Regularization-based methods (EWC, SI)

32.3.3. Replay and rehearsal strategies

32.3.4. Dynamic architecture approaches

32.3.5. Applications in lifelong learning systems

Explainability and Interpretability 33.

33.1. Saliency Maps and Grad-CAM

33.1.1. Saliency maps for visualizing input sensitivity

33.1.2. Gradient-based attribution methods

33.1.3. Grad-CAM (Gradient-weighted Class Activation Mapping)

33.1.4. Guided backpropagation and integrated gradients

33.1.5. Limitations and reliability concerns

33.2. SHAP and LIME

33.2.1. Model-agnostic interpretability approaches

33.2.2. Local Interpretable Model-agnostic Explanations (LIME)

33.2.3. SHAP values and Shapley game-theoretic foundation

33.2.4. Comparing SHAP vs LIME performance and stability

33.2.5. Use cases in tabular, text, and vision models

33.3. Interpretable PINNs and DRL Policies

33.3.1. Physics-informed constraints as interpretability tools

33.3.2. Sensitivity analysis in PINNs

33.3.3. Policy visualization and feature attribution in DRL

33.3.4. Reward decomposition and causal interpretability

33.3.5. Bridging performance with scientific understanding

Ethical and Societal Aspects 34.

34.1. Bias and Fairness in AI

34.1.1. Sources of bias in data and models

34.1.2. Fairness definitions and metrics

34.1.3. Mitigation strategies (reweighting, debiasing, adversarial methods)

34.1.4. Societal impact of biased AI systems

34.1.5. Case studies and ethical dilemmas

34.2. Privacy and Security

34.2.1. Data privacy regulations (GDPR, CCPA)

34.2.2. Differential privacy techniques

34.2.3. Federated learning and privacy-preserving methods

34.2.4. Adversarial attacks on AI models

34.2.5. Robustness, security, and safe deployment

34.3. AI Regulation and Governance

34.3.1. AI governance frameworks and standards

34.3.2. Ethical guidelines from international organizations

34.3.3. Risk assessment and accountability mechanisms

34.3.4. Transparency and explainability as regulatory goals

34.3.5. Future challenges in global AI governance

Part VIII.

Practical Implementation

Computational Frameworks 35.

35.1. PyTorch Fundamentals

35.1.1. Tensors and automatic differentiation

35.1.2. Building and training neural networks

35.1.3. Data loaders and dataset utilities

35.1.4. GPU acceleration with CUDA

35.1.5. Debugging and model inspection

35.2. TensorFlow and Keras

35.2.1. TensorFlow computation graphs and eager execution

35.2.2. High-level API with Keras

35.2.3. Model definition (Sequential vs Functional API)

35.2.4. Training loops and callbacks

35.2.5. Deployment and TensorFlow Serving

35.3. JAX and Differentiable Programming

35.3.1. JAX NumPy and XLA compilation

35.3.2. Automatic differentiation with `grad` and `jit`

35.3.3. Vectorization with `vmap` and parallelization with `pmap`

35.3.4. Composable function transformations

35.3.5. Applications in scientific machine learning

Efficient Training and Scaling | 36.

36.1. Hardware Acceleration: GPUs, TPUs

36.1.1. GPU architecture and parallel computation

36.1.2. Tensor cores and mixed-precision training

36.1.3. TPU architecture and matrix units

36.1.4. Framework integration with accelerators

36.1.5. Energy efficiency considerations

36.2. Parallelization and Distributed Training

36.2.1. Data parallelism strategies

36.2.2. Model and pipeline parallelism

36.2.3. Parameter servers and communication overhead

36.2.4. Synchronous vs asynchronous training

36.2.5. Fault tolerance and scalability

36.3. Memory-Efficient Backpropagation

36.3.1. Memory bottlenecks in deep networks

36.3.2. Gradient checkpointing techniques

36.3.3. Recomputation and activation offloading

36.3.4. Quantization and low-precision training

36.3.5. Sparse training and pruning approaches

37 Case Studies in Scientific Machine Learning

37.1. Navier–Stokes with PINNs

37.1.1. Formulating PDE residuals for fluid dynamics

37.1.2. Boundary and initial condition encoding

37.1.3. Physics-informed loss design

37.1.4. Handling turbulence and high Reynolds numbers

37.1.5. Benchmark results and limitations

37.2. QINNs for Quantum Chemistry

37.2.1. Quantum-inspired neural architectures

37.2.2. Encoding wavefunctions and Hamiltonians

37.2.3. Variational quantum eigensolver (VQE)-style training

37.2.4. Predicting molecular energies and properties

37.2.5. Scalability and hybrid quantum-classical methods

37.3. DRL for Robotics and Control

37.3.1. Formulating control tasks as MDPs

37.3.2. Reward shaping and curriculum learning

37.3.3. Simulation-to-reality transfer (sim2real)

37.3.4. Safety and sample efficiency challenges

37.3.5. Applications in manipulation and locomotion

37.4. CNNs/GNNs for Materials Science

37.4.1. Image-based microstructure characterization with CNNs

37.4.2. Graph representations of crystal structures

37.4.3. Predicting mechanical and electronic properties

37.4.4. Materials discovery and inverse design

37.4.5. Integrating experimental and simulation data

Mathematical Notation and Symbols A.

Linear Algebra Toolbox | B.

Probability Distributions C.

Special Functions (Gamma, Beta, Bessel, etc.) D.

E. Implementations in PyTorch and TensorFlow