Deep Learning Book Notes

July 28, 2025

Deep Learning Book Notes Peter Kruse

Contents

1 Introduction

- **Deep Learning** building hierarchical graph of concepts with many layers
  - representations are expressed in terms of other, simpler representations
  - **MLP**: function that maps input to output; composition of many simpler functions
- **Knowledge Base** Apporach: hard-code knowledge or rules in formal language
- **Machine Learning**: the ability to extract ("learn") patterns from raw data
- **Representation Learning**: using machine learning to derive a representation (extract features); ex: autoencoders

1.1 Who Should Read This Book?

1.2 Historical Trends in Deep Learning

- **Cybernetics** (1940s-50s): aimed to computationally model the brain, very theoretical, very little learning mechanism
  - **MCP Neuron**: first model of a neuron, inspired by human brain; used propositional logic, no learning mechanism
  - **Perceptron**: first learning algorithm, used for binary classification; limited to linearly separable data
  - **ADALINE**: special case of SGD
- **Connectionism** (1980s-90s): introduced backpropagation, focus on MLPs and CNNs for automatic feature extraction on basic learning tasks
  - **Backprop**: discovered independently in the 70s/80s by multiple groups; popularized by Rumelhart, Hinton, and Williams, efficient and scalable learning mechansim
  - **MLP**: multi-layer perceptron; used for supervised learning; feature differentiable and continuous nonlinearities, which worked with backprop; universal approximator
  - **CNN**: convolutional neural networks; used for image processing, introduced by LeCun et al. in 1989; uses local connectivity and weight sharing
- **Deep Learning** (2000s-present): focus on large datasets, deeper models, new architectures, and computational power

  - **GPU Computing**: use of graphics processing units to accelerate deep learning training
  - **Transfer Learning**: leveraging pre-trained models on new tasks with limited data
  - **Generative Models**: models that can generate new data samples, e.g., GANs and VAEs

- Models became more useful as data sizes increased; performance increased despite very little difference in architecture

- Models became more complex with infrastructure improvements
  - faster CPUs, general purpose GPUs
  - software libraries like TensorFlow, PyTorch, and JAX

2 Linear Algebra Basics

2.1 Scalars, Vectors, Matrices, and Tensors

- **Scalar**: single number, specified by type $\mathbb{R}$, $\mathbb{N}$, $\mathbb{Z}$
- **Vector**: an array of numbers arranged in a single row or column
  - First element of $\mathbf{x}$ is x_1, second is x_2, and so on: $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$
  - must specify the type of numbers stored, i.e., $\mathbf{x} \in \mathbb{R}^n$, where n is the number of elements/dimensionality
  - can think of a vector as identifying a point in space; each element gives a coordinate along a different axis
  - can index vectors with a set
    * indices $1, 3, 6 \to S = \{1, 3, 6\} \to x_S = \{x_1, x_3, x_6\}$
  - "$-$" indicates the complement of a set; $x_{-1} \to$ all elements except -1
- **Matrix**: 2-d array of numbers
  - each element is specified by two indices (row, col) instead of one
  - $A_{m,n}$: entry at row m, col n
  - $A_{i,:}$: all entries in the i_{th} row of A
  - $A_{:,j}$: all entries in the j_{th} column of A
- **Tensor**: Array with more than two axes
  - $A_{i,j,k}$
- **Transpose**: mirror image of a matrix across its main diagonal
  - $(A^T)_{i,j} = A_{j,i}$
  - row-column swap
- **Matrix Addition**: element-wise addition of two matrices of the same size
- Scalar times matrix: $D = a \cdot B + c \to D_{i,j} = a \cdot B_{i,j} + c$
- Matrix-Vector Addition: $C = A + \mathbf{b} \to C_{i,j} = A_{i,j} + b_j$
  - vector b is added to each row of matrix A
  - **Broadcasting**: the copying of a vector to match the dimensions of a matrix