

Deep Learning Book Notes

July 28, 2025

Deep Learning Book Notes Peter Kruse

Contents

1	Introduction	2
1.1	Who Should Read This Book?	2
1.2	Historical Trends in Deep Learning	2
2	Linear Algebra Basics	4
2.1	Scalars, Vectors, Matrices, and Tensors	4
2.2	Matrix and Vector Multiplication	5
2.3	Identity and Inverse Matrices	5
2.4	Linear Dependence and Span	6
2.5	Norms	6
2.6	Special Kinds of Matrices and Vectors	6
2.7	Eigendecomposition	6
2.8	Singular Value Decomposition	6
2.9	Moore-Penrose Pseudoinverse	6
2.10	Trace Operator	6
2.11	Determinant	6
2.12	Example: PCA	6

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** Approach: 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 mechanism
 - **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 $1, 3, 6 \rightarrow S = \{1, 3, 6\} \rightarrow x_S = \{x_1, x_3, x_6\}$
 - "–" indicates the complement of a set; $x_{-1} \rightarrow$ 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**: $D = a \cdot B + c \rightarrow D_{i,j} = a \cdot B_{i,j} + c$
- **Matrix-Vector Addition**: $C = A + \mathbf{b} \rightarrow C_{i,j} = A_{i,j} + b_j$
 - vector \mathbf{b} is added to each row of matrix A
 - **Broadcasting**: the copying of a vector to match the dimensions of a matrix

2.2 Matrix and Vector Multiplication

- **matrix product:** $C = AB$
 - to be defined, A , must have the same number of columns as B has rows.
 - if A is $m \times n$ and B is $n \times p$, then C is shape $m \times p$
 - $C_{i,j} = \sum_k A_{i,k} B_{k,j}$
- **Hadamard product:** element-wise multiplication of a matrix, denoted $A \circ B$
- **Dot Product:** $x \cdot y$ is the same dimensionality as the matrix product $x^T y$
 - $C = AB \rightarrow C_{i,j}$ is the dot product of row i of A and column j of B
- **Matrix Operation Properties:**
 - distributive: $A(B + C) = AB + AC$
 - associative: $(AB)C = A(BC)$
 - **not** commutative: $AB \neq BA$ in general; however vector dot product is commutative: $x^T y = y^T x$
 - transpose of a matrix product: $(AB)^T = B^T A^T$
- **System of Linear Equations**
 - $Ax = b$, $A \in \mathbb{R}^{m \times n}$ is a known matrix, $b \in \mathbb{R}^m$ is a known vector, and $x \in \mathbb{R}^n$ is unknown, and we would like to solve for it.
 - can rewrite as:

$$A_{1,1}x_1 + A_{1,2}x_2 + \cdots + A_{1,n}x_n = b_1$$

$$A_{2,1}x_1 + A_{2,2}x_2 + \cdots + A_{2,n}x_n = b_2$$

$$\vdots$$

$$A_{m,1}x_1 + A_{m,2}x_2 + \cdots + A_{m,n}x_n = b_m$$

2.3 Identity and Inverse Matrices

- **Matrix Inversion:** analytic solution to the system of linear equations
- **Identity Matrix:** does not change a vector when multiplied by it; denoted I_n , formally $I_n \in \mathbb{R}^{n \times n}$ and $\forall x \in \mathbb{R}^n, I_n x = x$; takes form of 1s along the diagonal
- **Matrix Inverse:** A^{-1} is the inverse of A , defined as the matrix such that $A^{-1}A = I_n$
-

- 2.4 Linear Dependence and Span
- 2.5 Norms
- 2.6 Special Kinds of Matrices and Vectors
- 2.7 Eigendecomposition
- 2.8 Singular Value Decomposition
- 2.9 Moore-Penrose Pseudoinverse
- 2.10 Trace Operator
- 2.11 Determinant
- 2.12 Example: PCA