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 $1, 3, 6 \rightarrow S = \{1, 3, 6\} \rightarrow x_S = \{x_1, x_3, x_6\}$
  - "$-$" 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 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 $x^T y$
  - $C = AB \to C_{i,j}$ 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 unkown, 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 $I_n \in \mathbb{R}^{\ltimes \times \ltimes}$ 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 $A^{-1}A = I_n$

- Solving a system of linear equations:

$$Ax = b \tag{1}$$
$$A^{-1}Ax = A^{-1}b \tag{2}$$
$$I_n x = A^{-1}b \tag{3}$$
$$x = A^{-1}b \tag{4}$$

- inverse matrix is primarily used as a theoretical tool - it's hard to represent at high precision on a digital computer

2.4 Linear Dependence and Span

- for A^{-1} to exist, the system of lin. eqns. must have one solution for every value of b
- it is also possible to have a system with no solutions or infinitely many solutions
- it is not possible to have a system with more than one but less than infinite solutions; if x and y are solutions, then

  $$z = \alpha x + (1 - \alpha)y$$

  is also a solution $\forall \alpha \in \mathbb{R}$
- **Finding Solutions to a System of Linear Equations**:
  - columns of A: different directions that we can travel from the origin
  - how many ways are there to reach b?
  - x how far we travel in each direction:

    $$Ax = \sum_i x_i A_{:,i}$$

  - textbfLinear Combination: multiply a set of vectors by scalars and add the results:

    $$\sum_i c_i v^i$$

  - **Span**: all points obtainable by linear combination of the original vectors
  - to find if the system has a solution, we must check if b is in the span of the columns of A, called the **column space/range** of A.
  - A must have at least m columns to span $\mathbb{R}^m$; if A has fewer than m columns, then it cannot span $\mathbb{R}^m$ and the system has no solutions
  - columns can also be redundant; so this is not a sufficient condition for a solution

  - **Linear Dependence**: if one column can be expressed as a linear combination of other columns
  - **Linear Independence**: no vector in the set is a linear combination of the other vectors
  - for a column space to span $\mathbb{R}^m$, it must have at least m linearly independent columns

- for a matrix to have an inverse, we need to ensure that our system has *atmost* one solution for each value of b; the matrix then must have at most m columns, otherwise one of the columns must be linearly dependent
- therefore, our matrix must be square ($m = n$), and all columns must be linearly independent; this is a **singular** matrix

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