The Hundred-Page LMs Book - Chapter 1 Notes

tags: The Hundred-Page Language Models Book

Chapter 1: Machine Learning Basics - Technical Deep Dive

The chapter introduces the field of **Artificial Intelligence (AI)**, tracing its origins back to a 1955 workshop that aimed to explore how machines could emulate human cognitive abilities such as language use, concept formation, problem-solving, and learning over time. Early approaches in AI heavily relied on **symbolic AI**, also known as **Good Old-Fashioned AI (GOFAI)**. GOFAI systems, exemplified by **expert systems**, used explicitly coded rules and symbols to represent knowledge and perform reasoning within narrowly defined domains. However, a significant limitation of symbolic AI was its struggle with **scalability** and **adaptability** to new situations, contributing to periods of reduced funding and research known as "AI winters." Despite these challenges, techniques like **decision trees**, which recursively partition data based on feature values to make predictions, continued to develop, although they were susceptible to **overfitting**, learning noise in the training data and generalizing poorly to unseen data.

The chapter then formally defines a **model** in machine learning as a **mathematical function** $y = f(x)$. This function maps inputs x from the **domain** to outputs y in the **codomain**, with the fundamental property that each input has exactly one output determined by a specific rule or formula. The core task in machine learning is to use a **dataset** of examples, often consisting of input-output pairs, to build or learn this function f. The goal is for the learned function to provide meaningful predictions or insights when applied to new, unseen input data.

In **supervised learning**, the dataset comprises examples where each input x is associated with a corresponding target y. The objective is to find the optimal set of **parameters** (also referred to as **weights**, often denoted as w) and a **bias** term (b) within the chosen model structure. Even for a simple **linear model** represented as $y = wx + b$, the learning process involves determining the values of w and b that best fit the relationship between the inputs and targets in the training data.

To quantify how well a model's predictions align with the true targets, a **loss function** (or cost function) $J(w, b)$ is defined. The loss function measures the average

prediction error across the training dataset. Different types of models, including **neural networks** and other **non-linear models**, typically aim to achieve lower loss values compared to simpler models on complex datasets.

The chapter outlines the **four-step machine learning process** for supervised learning:

1. **Collect a dataset** of input-output pairs.
2. **Define the model's structure** as a mathematical function.
3. **Define the loss function** to quantify errors.
4. **Minimize the loss function** on the dataset to find the optimal model parameters.

For complex models with a large number of parameters, such as the **billions of parameters** in large language models, manual minimization of the loss function becomes infeasible.

When dealing with inputs that have multiple attributes, these attributes can be represented as a **vector** $\mathbf{x}$, where each element of the vector corresponds to a specific **feature**, **dimension**, or **component**. Vectors are typically denoted by **lowercase bold letters**. Similarly, model parameters can also be organized into vectors and matrices. A **matrix** $\mathbf{A}$ of size $m \times n$ is a rectangular array of numbers, and matrices are represented by **uppercase bold letters**. These mathematical constructs are fundamental for representing data and performing computations efficiently in machine learning.

The chapter introduces **neural networks** as more flexible, **non-linear models** that are crucial for understanding large language models. Neural networks consist of interconnected **neurons** organized in layers, with **weights** associated with the connections and **biases** associated with each neuron. By stacking multiple layers, forming **deep neural networks**, models can learn increasingly complex relationships in the data. **Convolutional neural networks (CNNs)**, a type of **feedforward neural network (FNN)**, are mentioned as being effective for tasks like document classification.

Gradient Descent

Gradient descent is presented as a fundamental optimization algorithm used to minimize the loss function by iteratively adjusting the model's parameters. The algorithm calculates the **gradient** of the loss function with respect to each parameter, which indicates the direction of the steepest increase in the loss. To minimize the loss, the parameters are updated in the opposite direction of the gradient, scaled by a **learning rate** η. For example, a weight $w^{(2)}$ is updated as:

$$w^{(2)} \leftarrow w^{(2)} - \eta \frac{\partial loss}{\partial w^{(2)}}$$

The training process involves multiple **iterations** or **epochs** of updating the parameters based on the gradients calculated on the training data.

Automatic Differentiation

Finally, the chapter introduces **automatic differentiation** (often referred to as **autograd**), a powerful technique that allows for the automatic computation of gradients of complex functions. Frameworks like **PyTorch** provide built-in support for automatic differentiation, which is essential for efficiently training large neural networks.

A key advantage of automatic differentiation is its **flexibility with model switching** within the PyTorch ecosystem. For instance, one can easily replace a logistic regression model with a more complex **feedforward neural network (FNN)** defined using the `nn.Sequential` API, and PyTorch's automatic differentiation engine will handle the gradient computations for the new model without requiring significant code changes. This flexibility allows researchers and practitioners to experiment with different model architectures with relative ease.

The chapter sets the stage for subsequent discussions on how these fundamental machine learning concepts are applied to represent and process text data, starting with techniques like **bag of words** and **word embeddings**.