robot doggo can learn how to walk???

Sky Hong

April 20, 2025

Abstract

something idk, figure out final formatting later

Contents

Chapter 1

Introduction

Chapter 2

Background

2.1 Introduction to Machine Learning

The field of machine learning is fundamentally about finding functions that model data. For example, a speech recognition model takes audio as an input and outputs a text, and AlphaZero takes a chessboard state as an input and outputs a move. These functions may be vastly complex, with far too many parameters and relations for any human to reasonably articulate and program, even if the task comes naturally to our evolved biology. Machine learning offers methods for computers to achieve these tasks, without the need for a human to give explicit instructions on **how** it should be accomplished.

This section will give a very introductory overview of machine learning, the specific details of which will be delved more deeply in following sections. The general framework of creating a machine learning model is as follows:

1. Identification of the properties of a model. What should this model be able to do? What are its inputs and outputs? What kind of model architecture is best suited to its problem?
2. Defining a loss function, some metric to measure how well a model performs.
3. Optimizing the parameters of a model to minimize the loss function and maximize performance.

2.1.1 Parameters

As an example, consider a simple model consisting of a linear relationship between inputs and outputs.

$$f(\mathbf{x}) = b + \sum_i w_i x_i \tag{2.1}$$

Parameters that directly multiply values, w_i, are called **weights**, and parameters that offset

values, b, are called **biases**. This can be generalized to vector outputs as

$$f = \mathbf{b} + W\mathbf{x}, \tag{2.2}$$

where W is now a matrix.

The parameters of a model are usually collectively referred to as a vector θ, with components of the individual weights and biases of the model. For complex models, θ may be millions, billions, or even trillions of parameters long, populated by numerous parameter tensors.

$$\theta = \begin{bmatrix} W_{11} \\ W_{12} \\ \vdots \\ b_1 \\ b_2 \\ \vdots \end{bmatrix} \tag{2.3}$$

For virtually all non-trivial problems, linear relationships are far too reductive to completely capture the complexity of a problem. Such inherent limitations of a model due to its architecture known as **model bias**. Model architectures will be expounded upon in more detail in Section 1.2.

2.1.2 Loss Functions

A **loss function** (also sometimes called a **cost function**), typically denoted $L(\theta)$ or just L, is the measure of how "bad" a set of parameters θ for a model is. A common definition is the deviation between a model's prediction and an actual result. For example, when building a speech recognition model, the loss may be defined as the error rate between its output transcription and the correct transcription. The training data that is provided to the model, in this case, the correct transcriptions, are identified with **labels** that tell the model what it should train towards.

Losses over all labels in the training data are aggregated into an value for the overall loss function, the most common method of which is the **mean square error (MSE)**:

$$L = \frac{1}{N}\sum(y - \hat{y})^2. \tag{2.4}$$

L is associated with an **error surface** that can be understood as the plot of L in a $|\theta|$-dimensional parameter space, though it is usually far too complex to be directly visualized.

2.1.3 Optimization

Since L is continuous, there must exist some set of parameters θ^* that minimize L and model the training data best.

$$\theta^* = \arg\min_{\theta} L \tag{2.5}$$

The process of improving the model by finding values for θ that lower the loss function is what it really means for a model to "learn" or "train".

Gradient descent

Since brute-forcing the error surface of L is infeasible with a large number of parameters, the standard algorithmic approach to find minima of L is **gradient descent**. In gradient descent, L is iteratively lowered by stepping θ against the gradient of L.

$$\theta_{n+1} = \theta_n - \eta \nabla L(\theta_n) \tag{2.6}$$

The **learning rate**, η, determines the scaling for the size of each step.

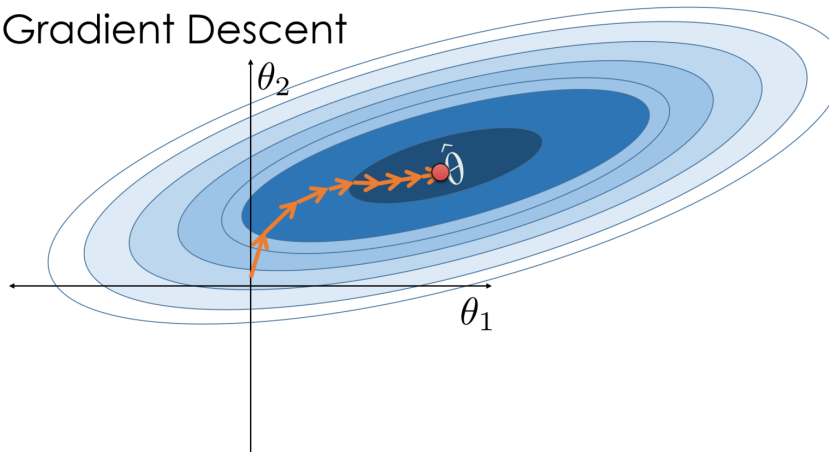

Figure 2.1: Visualization of gradient descent on an error surface with two parameters. Darker color indicates smaller L. From the Cornell University Computational Optimization Open Textbook, will replace with better figure.

In practice, some conditions may be defined to determine when to cease training, such as setting an upper limit on the number of iterations, setting a time limit, or setting a threshold for a satisfactory value of L. These values that modify aspects of a model's learning are called **hyperparameters**. Hyperparameter optimization is itself a complex topic that is beyond the scope of this section.

When θ reaches a critical point ($\nabla L = 0$), the updates to θ vanish, and the algorithm has reached a stable ending point. In an ideal case, this would be the global minima of L, the best that the model could possibly reach given the training data. In reality, reaching the global minima is highly improbable, θ is much more likely to be stuck at a local minima, or a saddle point. We will discuss methods to overcome these challenges in following sections.

2.2 Neural Networks

Artificial neural networks (ANNs), often shortened to **Neural Networks (NNs)**, are one of the largest classes of models used in machine learning.

Activation functions

For virtually all non-trivial problems, a linear relationships are far too reductive to completely capture the complexity of the task at hand. To model more complex relationships, a variety of non-linear **activation functions** can process the inputs.

Sigmoid functions are ubiquitous in this role, though other classes of activation functions will also be mentioned later.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

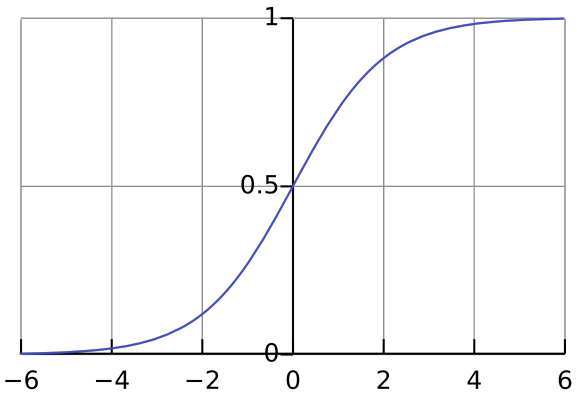

Figure 2.2: replace this with own figure

Any continuous relationship between two variables can be approximated by a sufficiently large linear combination of sigmoid functions. (Note that we define σ to act component wise on vector inputs.)

$$f \approx b + \mathbf{w} \cdot \sigma(\mathbf{b} + W\mathbf{x}). \tag{2.8}$$

This property is formalized more rigorously in the **Universal Approximation Theorem**.[1]

Sigmoid functions also have the advantage that

2.3 Backpropagation

Backpropagation is the textbook algorithm for computing the gradient of the loss function, as in practice, the error surface is almost never computable.

2.4 Overfitting

2.5 Variations on Gradient Descent

2.6 Reinforcement Learning

Chapter 3

Robotic Systems

3.1 ANYmal by ANYbotics

Chapter 4

Literature

Glossary

bias Parameters that add/offset to values in a model. Adds to a value add a node in neural networks. Not to be confused with *model bias*. 4

error surface The surface that a loss function plots in parameter space. 4

hyperparameter Values that specify aspects of a model's inherent architecture or its learning. 5

label The identifiers attached to training data that function as an "answer key" for what it is supposed to be. 4

loss function Function defined to measure the performance of a model, lower is better. 4

model bias Inherent limitations of a model's performance to its architecture. Not to be confused with *bias*. 4

MSE mean square error. 4

weight Parameters that are direct coefficients to values in a model. Determines strength of connections between nodes in neural networks. 3

Bibliography

[1] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.