UniTs - University of Trieste

Faculty of Scientific and Data Intensive Computing

Department of mathematics informatics and geosciences

Deep Learning

Lecturer:
Prof. Alessio Ansuini

Author:
Andrea Spinelli

March 11, 2025

github.com/Spina02 andreaspinelli2002@gmail.com

Preface

As a student of Scientific and Data Intensive Computing, I've created these notes while attending the **Deep Learning** course.

The prerequisites of the course are basic knowledge of:

- Linear Algebra (eigenvalue problems, SVD, etc.)
- Mathematical Analysis (multivariate differential calculus, etc.)
- Probability (chain rule, Bayes theorem, etc.)
- Machine Learning (logistic regression, PCA, etc.)
- Programming (Python, Linux Shell, etc.)

While these notes were primarily created for my personal study, they may serve as a valuable resource for fellow students and professionals interested in Deep Learning.

Contents

1
Introduction

1.1 What is Deep Learning?

Definition: ***Deep Learning***

*"**Deep Learning** is constructing networks of parametrized functional modules and training them from examples using gradient-based optimization."*

~ Yann LeCun

In other words, Deep Learning is a collection of tools to build complex modular differentiable functions. These tools are devoid of meaning, it is pointless to discuss what DL can or cannot do. What gives meaning to it is how it is trained and how the data is fed to it.

Deep learning models usually have an architecture that is composed of multiple layers of functions. These functions are called **modules** or **layers**. Each layer is a function that takes an input and produces an output. The output of one layer is the input of the next layer. The output of the last layer is the output of the model.

Practical Applications of Deep Learning

Deep Learning has revolutionized numerous fields by achieving unprecedented performance on complex tasks. In **computer vision**, convolutional neural networks can recognize objects, detect faces, segment images, and even generate realistic images. **Protein structure prediction** has seen remarkable advances with models like AlphaFold, which can accurately predict 3D protein structures from amino acid sequences, fundamentally changing molecular biology and drug discovery. **Speech recognition and synthesis** systems powered by deep learning can transcribe spoken language with near-human accuracy and generate natural-sounding speech, enabling voice assistants and accessibility tools. Other applications include natural language processing (powering chatbots and translation systems), recommendation systems, anomaly detection in cybersecurity, weather forecasting, and autonomous driving. The versatility of deep learning comes from its ability to learn meaningful representations directly from data, reducing the need for manual feature engineering while achieving superior performance across diverse domains.

1.2 Family of linear functions

Let's consider a simple modell

$$y = \Phi_0 + \Phi_1 x$$

This model is a linear function of x with parameters Φ_0 and Φ_1. The model can be represented as a line in the $x - y$ plane. The parameters Φ_0 and Φ_1 determine the slope and the intercept of the line.

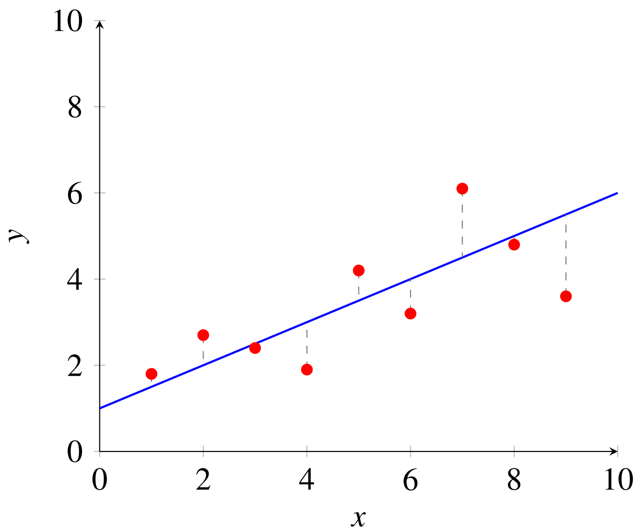

Figure 1.1: Linear model with scatter points and error segments

Loss Function

The loss function for this model is the mean squared error (MSE) between the predicted values and the actual values:

$$L = \frac{1}{N}\sum_{i=1}^{N}(\underbrace{\Phi_0 + \Phi_1 x_i}_{=P_i} - y_i)^2$$

where N is the number of data points, x_i is the i-th input, y_i is the i-th target, and P_i is the predicted value for the i-th data point.

Optimization

The goal of optimization is to find the values of Φ_0 and Φ_1 that minimize the loss function. This is done by computing the gradient of the loss function with respect to the parameters and updating the parameters in the opposite direction of the gradient. The update rule for the parameters is given by: Let's calculate the gradient of the losso function:

$$\nabla_{\{\Phi\}} L = \left[\frac{\partial L}{\partial \Phi_0}, \frac{\partial L}{\partial \Phi_1}\right]$$

Then we can update the parameters as follows:

$$\begin{cases} \Phi_0 \leftarrow \Phi_0 - \lambda \dfrac{\partial L}{\partial \Phi_0} \\ \Phi_1 \leftarrow \Phi_1 - \lambda \dfrac{\partial L}{\partial \Phi_1} \end{cases} \quad \Rightarrow \quad \Phi^{new} = \Phi^{old} - \lambda \nabla_{\{\Phi\}} L$$

where λ is the **learning rate**, a hyperparameter that controls the size of the parameter updates. This is only a step in the optimization process. The optimization algorithm iteratively updates the parameters until the loss converges to a minimum. But when shell we stop?

$$|L^{new} - L^{old}| < \varepsilon$$

where ε is a small positive number that determines the convergence threshold.

Exercises

Let's talk further about the loss function:

$$L = \frac{1}{N}\sum_{i=1}^{N}(\underbrace{\Phi_0 + \Phi_1 x_i}_{=P_i} - y_i)^2$$

The aim is to minimize the loss function. In this case the loss is a paraboloid function of Φ_0 and Φ_1, so it has a single minimum.

Questions

- Calculate the gradient of the Loss function
- Find the minimum of the loss function
- Show that the gradient of L is orthogonal to the level lines
- Estimate the computational complexity of the exact solution of the linear regression problem in the general case (take into account the number of data samples and the number of dimensions/features used)

$$\Phi = (X^\top X)^{-1} X^\top y$$