ViT Mideval Project Report: Deep Learning Fundamentals

Introduction

This report provides a comprehensive overview of fundamental deep learning concepts. It aims to consolidate knowledge on regression, neural network architecture, learning algorithms, convolutional neural networks (CNNs), and modern recurrent neural networks (RNNs). Deep learning, a subfield of machine learning, has revolutionized various domains, including computer vision, natural language processing, and predictive analytics, by enabling models to learn hierarchical representations of data.

Week 1: Foundations of Regression

Week 1 introduced the foundational concepts of **regression**, a statistical method used for modeling the relationship between a dependent variable and one or more independent variables. The core idea is to define a function y = f(x, some constants) + error and then adjust these constants and the error term to best fit the observed data.

Two types of regression that were covered:

- **Linear Regression**: This is applied when the function f is a linear combination of the input features. It seeks to establish a linear relationship between the input variables and the output. Linear regression is widely used for predicting continuous values.
- **Logistic Regression**: Unlike linear regression, logistic regression is employed when the output is a class or a probability. After calculating the linear expression we pass it through the sigmoid function.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Central to the learning process in regression models is **optimization**. This involves defining a **loss function** (or cost function) that quantifies the discrepancy between the model's predictions and the actual values. The goal is to minimize this loss function. **Gradient Descent** was introduced as a primary optimization algorithm, iteratively adjusting model parameters (weights and biases) in the direction opposite to the gradient of the loss function to minimize the error.

Week 2: Neural Network Architecture and Learning

In Week 2 we did learning mechanisms of **neural networks**, drawing insights from the 3Blue1Brown series. At its most fundamental, a neural network is composed of interconnected **neurons**, each acting as a unit that holds a numerical value, referred to as its "activation," typically between 0.0 and 1.0.

Neural Network Structure

The architecture of a typical neural network, specifically a Multilayer Perceptron (MLP), consists of several layers:

- **Input Layer**: This layer receives the raw data. For instance, in digit recognition, an input layer might have 784 neurons, corresponding to the 28x28 pixels of an image.
- **Hidden Layers**: These are intermediate layers between the input and output layers. They are responsible for extracting increasingly complex features from the input data. The number of hidden layers and neurons within them can vary, influencing the network's capacity to learn intricate patterns.
- **Output Layer**: This layer produces the network's final prediction. For a digit recognition task, it would typically have 10 neurons, each representing a digit from 0 to 9, with their activations indicating the probability of the input belonging to that digit.

The Learning Process

The learning process in neural networks is driven by minimizing a **cost function** that quantifies the error between the network's predictions and the true labels. This minimization is achieved through **Gradient Descent**, an iterative optimization algorithm. The gradient indicates the direction of the steepest increase in the cost function, so by moving in the opposite direction (negative gradient), the algorithm gradually adjusts the weights and biases to reduce the error.

Backpropagation is the cornerstone algorithm that enables efficient training of neural networks. It determines how each weight and bias in the network should be adjusted based on its contribution to the overall error for a given training example. This process involves two main steps:

1. **Forward Pass**: Input data is fed through the network to compute the output and the corresponding error.
2. **Backward Pass**: The error is propagated backward through the network, layer by layer. Algorithm computes the gradient of the cost function with respect to each weight and bias, allowing for their precise adjustment.

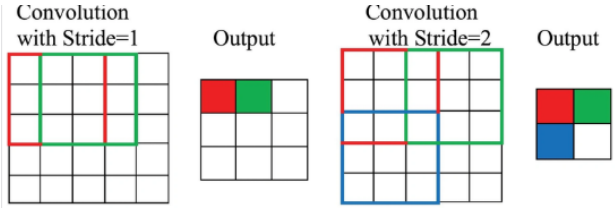
Week 3: Convolutional Neural Networks (CNN)

Week 3 introduced **Convolutional Neural Networks (CNNs)**, a specialized class of neural networks particularly effective for processing data with a known grid-like topology, such as images. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input data, making them highly successful in computer vision tasks.

Core Operations

The effectiveness of CNNs stems from several key operations:

- **Convolution**: This operation involves sliding a small filter (or kernel) over the input data, performing element-wise multiplications, and summing the results to produce a feature map. This process helps in detecting local features such as edges, textures, or patterns .
- **Cross-Correlation**: Often used interchangeably with convolution in deep learning literature, cross-correlation is a similar operation where the filter is slid over the input without flipping. Both operations achieve feature extraction by identifying patterns in the input data.
- **Padding**: To preserve the spatial dimensions of the input and prevent information loss at the borders, padding adds extra pixels (typically zeros) around the input image before convolution.
- **Stride**: Stride defines the step size with which the filter moves across the input. A larger stride reduces the spatial dimensions of the output feature map, effectively downsampling the data.



Layer Types and Architecture

CNN architectures typically comprise several types of layers:

- **Convolutional Layers**: These are the primary building blocks where convolution operations are performed. Each convolutional layer learns multiple filters, allowing it to detect various features at different locations in the input.
- **Pooling Layers**: These layers reduce the spatial dimensions (width and height) of the feature maps, thereby reducing the number of parameters and computational cost, and helping to control overfitting. Common pooling operations include **Max Pooling** (taking the maximum value in each window) and **Average Pooling** (taking the average value).
- **Fully Connected Layers**: After several convolutional and pooling layers, the high-level features extracted are typically fed into one or more fully connected layers, similar to those in traditional MLPs, for classification or regression tasks.

Week 4: Introduction to Recurrent Neural Networks (RNN)

Unlike traditional models that treat inputs as independent vectors, sequence modeling addresses data where the order and dependencies between elements are crucial. Examples include words in a document or a patient's medical history. The objective is to predict targets based on these sequentially structured inputs.

Sequence Modeling Approaches

- **Auto Regressive Models**: These models predict future values based on past observations, such as forecasting stock prices.

- **Latent Autoregressive Models**: These models extend the autoregressive concept by maintaining and continuously updating a "hidden state" that summarizes all past observations. This hidden state is then used to predict the current output.

Language Models and Perplexity

In the context of natural language processing, **Language Models** aim to estimate the joint probability of an entire sequence of discrete tokens (words). To process textual data, **Tokenization** breaks down the sequence into individual tokens, and a **Vocabulary** maps these tokens to unique numerical indices.

Perplexity serves as a crucial metric for evaluating the quality of a language model. A lower perplexity indicates a better model, as it implies higher accuracy in predicting the next token in a sequence and thus requires fewer bits to compress the sequence. It is mathematically defined as the exponential of the cross-entropy loss, averaged over all tokens in a sequence.

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of neural networks specifically designed for sequence modeling, characterized by their internal "hidden states" that capture information from previous steps in the sequence. The core equations governing an RNN's operation are:

- $\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{\text{xh}} + \mathbf{H}_{t-1} \mathbf{W}_{\text{hh}} + \mathbf{b}_{\text{h}})$: This equation describes how the current hidden state (H_t) is computed based on the current input (X_t), the previous hidden state (H_{t-1}), and learned weight matrices (W_xh, W_hh) and bias (b_h), with φ being an activation function.
- $\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{\text{hq}} + \mathbf{b}_{\text{q}}$: This equation shows how the output (O_t) is derived from the current hidden state (H_t) using another weight matrix (W_hq) and bias (b_q).

For representing tokens as inputs to RNNs, **One-Hot Encoding** is commonly used. This method converts each token into a vector of the vocabulary size (N), where all entries are zero except for a single '1' at the index corresponding to that specific token.

We can compute the full gradient through chain rule, but this is very slow and gradients can blow up, so we just calculate the sum upto T steps i.e. truncate the time steps.

> **Gradient Clipping**: This technique addresses exploding gradients by scaling down the gradients if their norm exceeds a predefined threshold (θ). This prevents the gradients from becoming too large and destabilizing the training process.

Week 5: Modern Recurrent Neural Networks (RNN)

Week 4 focused on **Recurrent Neural Networks (RNNs)**, which are specifically designed to process sequential data, where the output from the previous step is fed as input to the current step. This characteristic makes RNNs suitable for tasks involving time series, natural language, and speech recognition.

Traditional or "plain" RNNs face significant challenges, primarily the **vanishing and exploding gradient problems**.

Advanced Architectures

To address these limitations, several advanced RNN architectures have been developed:

- **Gated Recurrent Units (GRU)**: GRUs are a simpler variant of LSTMs that use gating mechanisms to control the flow of information. They have two gates: an update gate and a reset gate, which help in retaining relevant information and forgetting irrelevant information over long sequences.
- **Long Short-Term Memory (LSTM)**: LSTMs are a more sophisticated type of RNN that effectively mitigate the vanishing gradient problem. They achieve this through a complex internal mechanism comprising three gates: the input gate, the forget gate, and the output gate. These gates regulate the flow of information into and out of the cell state, allowing LSTMs to learn and remember long-term dependencies.
- **Deep RNNs**: These involve stacking multiple RNN layers, allowing the network to learn hierarchical representations of sequential data, similar to how deep feedforward networks learn hierarchical features in static data.
- **Bidirectional RNNs**: Unlike unidirectional RNNs that process sequences in one direction, bidirectional RNNs process the sequence in both forward and backward directions. This allows the network to capture context from both past and future elements in the sequence, which is particularly useful in tasks like natural language understanding.
- **Sequence-to-Sequence Models**: These models consist of an encoder RNN that processes the input sequence into a fixed-context vector and a decoder RNN that generates an output sequence from this context vector. They are widely used in machine translation and text summarization.

Conclusion

This report has covered the fundamental concepts of deep learning, starting with regression as a basic modeling technique and progressing to the intricate architectures of neural networks. We explored the mechanics of how neural networks learn through gradient descent and backpropagation, and then specialized into CNNs for image processing and RNNs for sequential data. The evolution from plain RNNs to advanced architectures like LSTMs and GRUs highlights the continuous innovation in the field to overcome challenges and enhance model capabilities. These foundational topics provide a strong basis for understanding more advanced deep learning models and their applications in artificial intelligence.