A Journey through Deep Learning Fundamentals with TensorFlow and Keras

Introduction

Welcome to the world of Deep Learning (DL), where we'll embark on an exciting journey to understand some fundamental concepts using popular frameworks like TensorFlow and Keras. In this blog, we will delve into critical concepts such as activation functions, regression, classification, feed-forward propagation, backpropagation, and the implementation of a simple neural network using Keras.

Activation Functions

Activation functions are the soul of a neural network, determining the output of each neuron. They introduce non-linearity, enabling the network to learn complex patterns. Common activation functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU). In our example, we've used the absolute value (`tf.math.abs`) as an activation function.

```
import tensorflow as tf


x = tf.constant(3.0)

y = tf.constant(6.0)

w = tf.Variable(20.0)


with tf.GradientTape() as tape:

    loss = tf.math.abs(w * x - y)

    dx = tape.gradient(loss, w)

    w.assign(w - dx)
```

```
print("Weight is:", w.numpy(), "Descent:", dx.numpy(), "Loss:", loss.numpy())
```

In this snippet, we define a simple linear equation and update the weight using gradient descent to minimize the loss.

Regression and Classification

Deep Learning models can be categorized into regression and classification tasks. Regression predicts continuous values, while classification assigns data points to predefined classes. Our simple example demonstrates a regression task, predicting weight based on height.

Feed-forward Propagation

Feed-forward propagation is the process by which input data is passed through the neural network, layer by layer, to produce an output. Each layer contains neurons with weights that are adjusted during training. Our code snippet showcases a basic feed-forward propagation scenario using TensorFlow.

Backpropagation

Backpropagation is the heart of training a neural network. It involves adjusting weights backward through the network to minimize the difference between predicted and actual outputs. The `GradientTape` in TensorFlow allows us to calculate gradients efficiently. Our example demonstrates backpropagation to update weights and minimize the loss.

Keras: Simplifying Deep Learning

Keras, a high-level neural networks API, simplifies the process of building and training models. In our code snippet, we use Keras to create a Sequential model with a single Dense layer for our regression task. The model is compiled with the mean squared error loss function and optimized using the Adam optimizer.

Hands-On Example with Real Data

To make things more tangible, we apply our knowledge to a real-world dataset. Using the SOCR-HeightWeight dataset, we predict weight based on height. The dataset is loaded using Pandas, and the Keras model is trained with the Adam optimizer.

Regression with Keras

Now, let's dive into real-world data. We'll use Keras, a high-level neural networks API running on top of TensorFlow, to perform regression on a dataset.

```
import pandas as pd

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import Adam


# Load dataset

dataset = pd.read_csv("SOCR-HeightWeight.csv")

X = dataset["Height(Inches)"]

y = dataset["Weight(Pounds)"]


# Build a Sequential model

model = Sequential()

model.add(Dense(units=1, input_dim=1))

model.summary()


# Compile the model
```

```
model.compile(loss="mean_squared_error", optimizer=Adam(learning_rate=0.8))


# Train the model

model.fit(X, y, epochs=20)
```

In this example, we create a simple neural network with one dense layer and compile it using mean squared error loss and the Adam optimizer. The model then undergoes training for 20 epochs.

Conclusion

In this document,, we've journeyed through fundamental concepts of Deep Learning, exploring activation functions, regression, classification, feed-forward propagation, and backpropagation. We've also touched upon the simplicity that Keras brings to the world of DL, providing a practical example with a real dataset.

Understanding these concepts lays a strong foundation for diving deeper into the vast and fascinating realm of Deep Learning. As you continue your exploration, remember that these concepts are building blocks for more complex models that can tackle a wide range of tasks.