2023-03-07

Converting Scikit-learn Models to ONNX and Performing Inference

Machine Learning ONNX sklearn

Introduction

In this article, we will walk you through the process of converting a Scikit-learn model to ONNX format, and performing inference using the ONNX model.

Prepare Your Scikit-learn Model

In this chapter, we will discuss how to create and train a Scikit-learn model and save it for further use. We will use the Iris dataset and create a Logistic Regression model as an example.

Create and Train a Scikit-learn Model

Before converting the model to ONNX, you need to create and train a Scikit-learn model. In this example, we will create a simple Logistic Regression model using the Iris dataset.

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

By using the `train_test_split` function, we have split the dataset into training and testing sets. Then, we create an instance of the `LogisticRegression` class and fit the model using the training data.

Save the Scikit-learn Model

Once the model is trained, it is essential to save it so that it can be used later for conversion or other purposes. To save the trained Scikit-learn model, you can use the `joblib` library, which is efficient for storing large NumPy arrays, such as Scikit-learn models.

```python
import joblib

# Save the model to a file
joblib.dump(model, 'logistic_regression_model.pkl')
```

This code snippet saves the trained Logistic Regression model to a file named `logistic_regression_model.pkl`.

Convert Your Scikit-learn Model to ONNX

In this chapter, we will discuss the steps to convert a Scikit-learn model to an ONNX model. This will allow you to take advantage of the benefits of ONNX, such as cross-platform support and interoperability with various deep learning frameworks.

Install ONNXMLTools

To convert your Scikit-learn model to ONNX, you need to install the ONNXMLTools package, which provides utilities to convert models from various machine learning frameworks to ONNX format. For Scikit-learn, we will use the `skl2onnx` package, which is part of ONNXMLTools:

```bash
```

```
$ pip install skl2onnx
```

Convert the Model

With the `skl2onnx` package installed, you can now convert the Scikit-learn model to an ONNX model. To do this, you will need to use the `convert_sklearn` function provided by `skl2onnx`. Additionally, you will need to specify the input data types for your model.

In our example, we will convert the previously saved Logistic Regression model.

python

```python
import onnxmltools
from skl2onnx import convert_sklearn
from skl2onnx.common.data_types import FloatTensorType

# Load the Scikit-learn model from the saved file
model = joblib.load('logistic_regression_model.pkl')

# Convert the Scikit-learn model to ONNX
onnx_model = convert_sklearn(model, 'logistic_regression_onnx', [('input', FloatTensorType([None, 4]))])

# Save the ONNX model to a file
onnxmltools.utils.save_model(onnx_model, 'logistic_regression_model.onnx')
```

In this code snippet, we first load the saved Scikit-learn model using `joblib.load`. Then, we call the `convert_sklearn` function, passing the model, a name for the ONNX model, and the input data types. In this case, we use a `FloatTensorType` with a dynamic

batch size (indicated by `None`) and a fixed feature size of 4. Finally, we save the converted ONNX model to a file named `logistic_regression_model.onnx`.

Inference Using the Converted ONNX Model

In this chapter, we will demonstrate how to perform inference using the converted ONNX model. This involves installing the ONNX Runtime, loading the ONNX model, and making predictions using the model.

Install ONNX Runtime

To perform inference with the ONNX model, you need to install the ONNX Runtime, a high-performance inference engine for ONNX models. The ONNX Runtime can be installed using pip:

bash

```
$ pip install onnxruntime
```

Load the ONNX Model

Once the ONNX Runtime is installed, you can load the ONNX model using the `InferenceSession` class provided by the ONNX Runtime:

python

```
import onnxruntime as rt

# Load the ONNX model
sess = rt.InferenceSession('logistic_regression_model.onnx')
```

Make Predictions

Now that the ONNX model is loaded, you can make predictions using the model. To do this, you need to prepare the input data and call the `run` method on the `InferenceSession` object.

In our example, we will use the testing data from the Iris dataset:

python

```
import numpy as np

# Prepare the input data
input_data = np.array(X_test, dtype=np.float32)
input_name = sess.get_inputs()[0].name

# Make predictions using the ONNX model
output_name = sess.get_outputs()[0].name
predictions = sess.run([output_name], {input_name: input_data})[0]
```

In this code snippet, we first convert the input data (i.e., `X_test`) to a NumPy array of type `float32`. Then, we retrieve the input and output names of the model using the `get_inputs` and `get_outputs` methods, respectively. Finally, we call the `run` method on the

`InferenceSession object, passing the output name and a dictionary that maps the input name to the input data. The predictions are returned as a NumPy array.

References

onnx / sklearn-onnx

Convert scikit-learn models and pipelines to ONNX

☆ 429 86 </> Python

« TensorFlow Model Conversion and Inference with ONNX

Converting LightGBM Models to ONNX and Performing Inference »

Ryusei Kakujo

Data science, data engineering, transportation engineering

Bench Press 100kg!

Amazon Cognito Amazon QuickSight Amazon RDS Amazon Redshift Amazon S3 API AWS
AWS API Gateway AWS Chalice AWS Control Tower AWS Lambda BERT Chrome Extension CircleCI
Classification Cloud Functions Cloud IAM Cloud Run Cloud Storage Clustering Data Modeling
Data Platform dbt Decision Tree Deep Learning Descriptive Statistics Dimensionality Reduction
Discrete Choice Model Docker Economics FastAPI Firebase git GitHub GitHub Actions
Google Cloud Hugging Face Hypothesis Testing Inferential Statistics Interval Estimation JavaScript Jinja
Kedro LightGBM Linux LLM Machine Learning MLflow MLOps MySQL NLP Nodejs ONNX
Optuna Pandas Pinecone Probability Distribution Python PyTorch R Regression Singer sklearn
Slack Snowflake Software Development SQL Statistical Model Statistics Tabular TensorFlow
Terraform Transportation TypeScript Vector DB Vertex AI XGBoost