DLI Accelerated Data Science Teaching Kt

Lecture 14.10 - RAPIDS Acceleration: Random Forest

RAPIDS

The RAPIDS data science framework includes a collection of libraries for executing end-to-end data science pipelines completely in the GPU.

It is designed to have a familiar look and feel to data scientists working in Python.

Features

Hassle-Free Integration Accelerate your Python data science toolchain with minimal code changes and no new tools to learn.	**Top Model Accuracy** Increase machine learning model accuracy by iterating on models faster and deploying them more frequently.
Reduced Training Time Drastically improve your productivity with near-interactive data science.	**Open Source** Customizable, extensible, interoperable - the open-source software is supported by NVIDIA and built on Apache Arrow.

Source: https://rapids.ai/start.html

Speed Up Learning of Random Forest

The Random Forest algorithm is a classification method which builds several decision trees, and aggregates each of their outputs to make a prediction.

We will train a scikit-learn and a cuML Random Forest Classification model.

Then we save the cuML model for future use with Python's picklingmechanism and demonstrate how to re-load it for prediction.

We also compare the results of the scikit-learn, non-pickled and pickled cuML models.

Random Forest vs Random Forest cuML

Import packages

```
import cudf
import numpy as np
import pandas as pd
import pickle

from cuml.ensemble import RandomForestClassifier as curfc
from cuml.metrics import accuracy_score

from sklearn.ensemble import RandomForestClassifier as skrfc
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
```

Setting parameters

```
# The speedup obtained by using cuML'sRandom Forest implementation
# becomes much higher when using larger datasets. Uncomment and use the n_samples
# value provided below to see the difference in the time required to run
# Scikit-learn's vs cuML's implementation with a large dataset.

# n_samples = 2*17
n_samples = 2**12
n_features = 399
n_info = 300
data_type = np.float32
```

Random Forest vs Random Forest cuML

Generating Data

Host

```
%%time
X,y = make_classification(n_samples=n_samples,
                          n_features=n_features,
                          n_informative=n_info,
                          random_state=123, n_classes=2)

X = pd.DataFrame(X.astype(data_type))
# cuML Random Forest Classifier requires the labels to be integers
y = pd.Series(y.astype(np.int32))

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.2,
                                                    random_state=0)
```

GPU

```
%%time
X_cudf_train = cudf.DataFrame.from_pandas(X_train)
X_cudf_test = cudf.DataFrame.from_pandas(X_test)

y_cudf_train = cudf.Series(y_train.values)
```

Random Forest vs Random Forest cuML

Sklearn Random Forest

Fit

```
%%time
sk_model = skrfc(n_estimators=40,
                 max_depth=16,
                 max_features=1.0,
                 random_state=10)

sk_model.fit(X_train, y_train)
```

```
CPU times: user 28.4 s, sys: 0 ns, total: 28.4 s
Wall time: 28.3 s
```

Evaluate

```
%%time
sk_predict = sk_model.predict(X_test)
sk_acc = accuracy_score(y_test, sk_predict)
```

```
CPU times: user 19.7 ms, sys: 0 ns, total: 19.7 ms
Wall time: 21.8 ms
```

cuML

Fit

```
%%time
cuml_model = curfc(n_estimators=40,
                   max_depth=16,
                   max_features=1.0,
                   random_state=10)

cuml_model.fit(X_cudf_train, y_cudf_train)
```

```
CPU times: user 919 ms, sys: 600 ms, total: 1.52 s
Wall time: 842 ms
```

Evaluate

```
%%time
fil_preds_orig = cuml_model.predict(X_cudf_test)

fil_acc_orig = accuracy_score(y_test.to_numpy(), fil_preds_orig)
```

```
CPU times: user 214 ms, sys: 12.6 ms, total: 226 ms
Wall time: 127 ms
```

Random Forest vs Random Forest cuML

Pickle the cuML random forest classification model

```
filename = 'cuml_random_forest_model.sav'
# save the trained cuml model into a file
pickle.dump(cuml_model, open(filename, 'wb'))
# delete the previous model to ensure that there is no leakage of pointers.
# this is not strictly necessary but just included here for demo purposes.
del cuml_model
# load the previously saved cuml model from a file
pickled_cuml_model = pickle.load(open(filename, 'rb'))
```

Predict using the pickled model

```
%%time
pred_after_pickling = pickled_cuml_model.predict(X_cudf_test)

fil_acc_after_pickling = accuracy_score(y_test.to_numpy(), pred_after_pickling)
```

```
CPU times: user 216 ms, sys: 16.7 ms, total: 233 ms
Wall time: 138 ms
```

Random Forest vs Random Forest cuML

Compare Results

```
print("CUML accuracy of the RF model before pickling: %s" % fil_acc_orig)
print("CUML accuracy of the RF model after pickling: %s" % fil_acc_after_pickling)
```

```
CUML accuracy of the RF model before pickling: 0.7329268455505371
CUML accuracy of the RF model after pickling: 0.7329268455505371
```

```
print("SKL accuracy: %s" % sk_acc)
print("CUML accuracy before pickling: %s" % fil_acc_orig)
```

```
SKL accuracy: 0.6926829218864441
CUML accuracy before pickling: 0.7329268455505371
```

DEEP
LEARNING
INSTITUTE

PRAIRIE VIEW
A&M UNIVERSITY

DLI Accelerated Data Science Teaching Kit

Thank You