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

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 pickling mechanism and demonstrate how to re-load it for prediction.

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

We will be using a T4 GPU instance in Colab.

Random Forest vs Random Forest cuML

Import packages

```
# load cuDF GPU extension for Pandas and import supporting capabilities
%load_ext cudf.pandas
import pandas as pd
import pickle

# Import CPU libraries
import numpy as np
from sklearn.ensemble import RandomForestClassifier as rfc_skl
from sklearn.datasets import make_classification as make_classification_skl
from sklearn.model_selection import train_test_split as train_test_split_skl

# Import GPU Libraries
import cupy as cp
import cuml
from cuml.ensemble import RandomForestClassifier as rfc_cuml
from cuml.datasets.classification  import make_classification as make_classification_cuml
from cuml import train_test_split as train_test_split_cuml
```

Setting parameters

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

n_samples = 2**12 # use for faster CPU processing or for smaller GPUs (<12GB)
# n_samples = 2**17 # use for larger GPUs (>=12GB)
n_features = 399
n_info = 300
random_state = 123
n_classes = 2
data_type = np.float32
```


Random Forest vs Random Forest cuML

Generating Data with SK Learn

```
%%time
X,y = make_classification_skl(n_samples=n_samples,
                              n_features=n_features,
                              n_informative=n_info,
                              random_state=random_state,
                              n_classes=n_classes
                              )

X = pd.DataFrame(X.astype(data_type))
y = pd.Series(y.astype(np.int32))

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

```
CPU times: user 5.78 s, sys: 3.22 s, total: 9 s
Wall time: 9.98 s
```

Generating Data with cuML

```
%%time
X,y = make_classification_cuml(n_samples=n_samples,
                               n_features=n_features,
                               n_informative=n_info,
                               random_state=random_state,
                               n_classes=n_classes
                               )

X = pd.DataFrame(X.astype(data_type)) # takes advantage of GPU acceleration
# cuML Random Forest Classifier requires the labels to be integers
y = pd.Series(y.astype(cp.int32)) # takes advantage of GPU acceleration

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

```
CPU times: user 1.31 s, sys: 69.8 ms, total: 1.38 s
Wall time: 1.58 s
```

Random Forest vs Random Forest cuML

Random Forest with SK Learn

Fit

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

sk_model.fit(X_train, y_train)
```

Time

n_samples = 2^{12}
```
CPU times: user 57.4 s, sys: 211 ms, total: 57.6 s
Wall time: 1min 8s
```

n_samples = 2^{17}
```
CPU times: user 48min 49s, sys: 3.2 s, total: 48min 52s
Wall time: 49min 16s
```

Predict

```
%%time
sk_predict = sk_model.predict(X_test)
```

Time

n_samples = 2^{12}
```
CPU times: user 14.7 ms, sys: 1.96 ms, total: 16.7 ms
Wall time: 18.3 ms
```

n_samples = 2^{17}
```
CPU times: user 388 ms, sys: 2 ms, total: 390 ms
Wall time: 388 ms
```

Evaluate

```
%%time
sk_acc = accuracy_score(y_test, sk_predict)
```

Time

n_samples = 2^{12}
```
CPU times: user 2.21 ms, sys: 4 µs, total: 2.21 ms
Wall time: 2.07 ms
```

n_samples = 2^{17}
```
CPU times: user 6.07 ms, sys: 0 ns, total: 6.07 ms
Wall time: 6.65 ms
```

Random Forest with cuML

Fit

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

cuml_model.fit(X_train, y_train)
```

Time

n_samples = 2^{12}
```
CPU times: user 663 ms, sys: 285 ms, total: 948 ms
Wall time: 542 ms
```

n_samples = 2^{17}
```
CPU times: user 9.23 s, sys: 6.1 s, total: 15.3 s
Wall time: 9.03 s
```

Predict

```
%%time
cuml_predict = cuml_model.predict(X_test)
```

Time

n_samples = 2^{12}
```
CPU times: user 174 ms, sys: 59.3 ms, total: 233 ms
Wall time: 188 ms
```

n_samples = 2^{17}
```
CPU times: user 370 ms, sys: 69.9 ms, total: 440 ms
Wall time: 369 ms
```

Evaluate

```
%%time
cuml_acc = accuracy_score(y_test, cuml_predict)
```

Time

n_samples = 2^{12}
```
CPU times: user 1.87 ms, sys: 0 ns, total: 1.87 ms
Wall time: 1.87 ms
```

n_samples = 2^{17}
```
CPU times: user 3.49 ms, sys: 872 µs, total: 4.36 ms
Wall time: 4.29 ms
```

Random Forest vs Random Forest cuML

Pickle the cuML random forest classification model (using n_samples = 2^{12} dataset)

```
filename = "cuml_rf_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
```

Predict using the pickled model

```
# load the previously saved cuml model from a file
pickled_cuml_model = pickle.load(open(filename, 'rb'))


%%time
pred_after_pickling = pickled_cuml_model.predict(X_test)

fil_acc_after_pickling = accuracy_score_cuml(y_test, pred_after_pickling)
CPU times: user 108 ms, sys: 45.3 ms, total: 154 ms
Wall time: 121 ms
```

Random Forest vs Random Forest cuML

Compare Results

```
print("cuML accuracy of the RF model before pickling is: %s" % cuml_acc)
print("cuML accuracy of the RF model after pickling is: %s" % fil_acc_after_pickling)
```

```
cuML accuracy of the RF model before pickling is: 0.7130647301673889
cuML accuracy of the RF model after pickling is: 0.7130647301673889
```

```
print("SKL accuracy: %s" % sk_acc)
print("CUML accuracy: %s" % cuml_acc)
```

```
SKL accuracy: 0.6926829268292682
CUML accuracy: 0.7130647301673889
```

DLI Accelerated Data Science Teaching Kit

Thank You