DLI Accelerated Data Science Teaching Kt

Lecture 14.4 - RAPIDS Acceleration: Linear Regression

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 Linear Regression

Linear Regression is a simple machine learning model where the response y is modelled by a linear combination of the predictors in X.

The model can take array-like objects, either in host as NumPy arrays or in device (as Numba or cuda_array_interface-compliant), as well as cuDF DataFrames as the input.

Linear Regression vs Linear Regression cuML

Import packages

```
import cudf
from cuml import make_regression, train_test_split
from cuml.linear_model import LinearRegression as cuLinearRegression
from cuml.metrics.regression import r2_score
from sklearn.linear_model import LinearRegression as skLinearRegression
```

Setting parameters

```
n_samples = 2**20 #If you are running on a GPU with less than 16GB RAM, please change to 2**19 or you could run out of memory
n_features = 399

random_state = 23
```

Linear Regression vs Linear Regression cuML

Generating Data

```
%%time
X, y = make_regression(n_samples=n_samples, n_features=n_features, random_state=random_state)

X = cudf.DataFrame(X)
y = cudf.DataFrame(y)[0]

X_cudf, X_cudf_test, y_cudf, y_cudf_test = train_test_split(X, y, test_size = 0.2, random_state=random_state)
```

```
# Copy dataset from GPU memory to host memory.
# This is done to later compare CPU and GPU results.
X_train = X_cudf.to_pandas()
X_test = X_cudf_test.to_pandas()
y_train = y_cudf.to_pandas()
y_test = y_cudf_test.to_pandas()
```

Linear Regression vs Linear Regression cuML

Sklearn Linear Regression

Fit

```
%%time
ols_sk = skLinearRegression(fit_intercept=True,
                            normalize=True,
                            n_jobs=-1)

ols_sk.fit(X_train, y_train)
```

```
CPU times: user 29 s, sys: 5.47 s, total: 34.5 s
Wall time: 21.3 s
```

Predict

```
%%time
predict_sk = ols_sk.predict(X_test)
```

```
CPU times: user 125 ms, sys: 490 µs, total: 125 ms
Wall time: 105 ms
```

Evaluate

```
%%time
r2_score_sk = r2_score(y_cudf_test, predict_sk)
```

```
CPU times: user 5.18 ms, sys: 20 µs, total: 5.2 ms
Wall time: 11.2 ms
```

cuML

Fit

```
%%time
ols_cuml = cuLinearRegression(fit_intercept=True,
                              normalize=True,
                              algorithm='eig')

ols_cuml.fit(X_cudf, y_cudf)
```

```
CPU times: user 181 ms, sys: 161 ms, total: 343 ms
Wall time: 454 ms
```

Predict

```
%%time
predict_cuml = ols_cuml.predict(X_cudf_test)
```

```
CPU times: user 29.2 ms, sys: 6.27 ms, total: 35.5 ms
Wall time: 36.4 ms
```

Evaluate

```
%%time
r2_score_cuml = r2_score(y_cudf_test, predict_cuml)
```

```
CPU times: user 1.25 ms, sys: 1.07 ms, total: 2.32 ms
Wall time: 2.33 ms
```

Linear Regression vs Linear Regression cuML

Compare Results

```
print("R^2 score (SKL):  %s" % r2_score_sk)
print("R^2 score (cuML): %s" % r2_score_cuml)
```

```
R^2 score (SKL):  1.0
R^2 score (cuML): 1.0
```

DEEP
LEARNING
INSTITUTE

PRAIRIE VIEW
A&M UNIVERSITY

DLI Accelerated Data Science Teaching Kit

Thank You