DLI Accelerated Data Science Teaching Kt

Lecture 15.3 - RAPIDS Acceleration: KMeans

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 KMeans

KMeans is a basic but powerful clustering method which is optimized via Expectation Maximization.

It randomly selects K data points in X, and computes which samples are close to these points.

- For every cluster of points, a mean is computed, and this becomes the new centroid.

cuML's KMeans supports the scalable KMeans++ intialization method.

- This method is more stable than randomnly selecting K points.

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.

KMeans vs Kmeans_cuML

Import packages

```
import cudf
import cupy
import matplotlib.pyplot as plt
from cuml.cluster import KMeans as cuKMeans
from cuml.datasets import make_blobs
from sklearn.cluster import KMeans as skKMeans
from sklearn.metrics import adjusted_rand_score

%matplotlib inline
```

Setting parameters

```
n_samples = 100000
n_features = 2

n_clusters = 5
random_state = 0
```

KMeans vs Kmeans_cuML

Generating Data

```
device_data, device_labels = make_blobs(n_samples=n_samples,
                                        n_features=n_features,
                                        centers=n_clusters,
                                        random_state=random_state,
                                        cluster_std=0.1)

device_data = cudf.DataFrame(device_data)
device_labels = cudf.Series(device_labels)
```

```
# Copy dataset from GPU memory to host memory.
# This is done to later compare CPU and GPU results.
host_data = device_data.to_pandas()
host_labels = device_labels.to_pandas()
```

KMeans vs Kmeans_cuML

Sklearn KMeans

Fit

```
%%time
kmeans_sk = skKMeans(init="k-means++",
                     n_clusters=n_clusters,
                     n_jobs=-1,
                    random_state=random_state)

kmeans_sk.fit(host_data)
```

```
CPU times: user 891 ms, sys: 13.2 ms, total: 905 ms
Wall time: 518 ms
```

cuML

Fit

```
%%time
kmeans_cuml = cuKMeans(init="k-means||",
                       n_clusters=n_clusters,
                       oversampling_factor=40,
                       random_state=random_state)

kmeans_cuml.fit(device_data)
```

```
CPU times: user 532 ms, sys: 84.1 ms, total: 616 ms
Wall time: 623 ms
```

KMeans vs Kmeans_cuML

Visualize Centroids

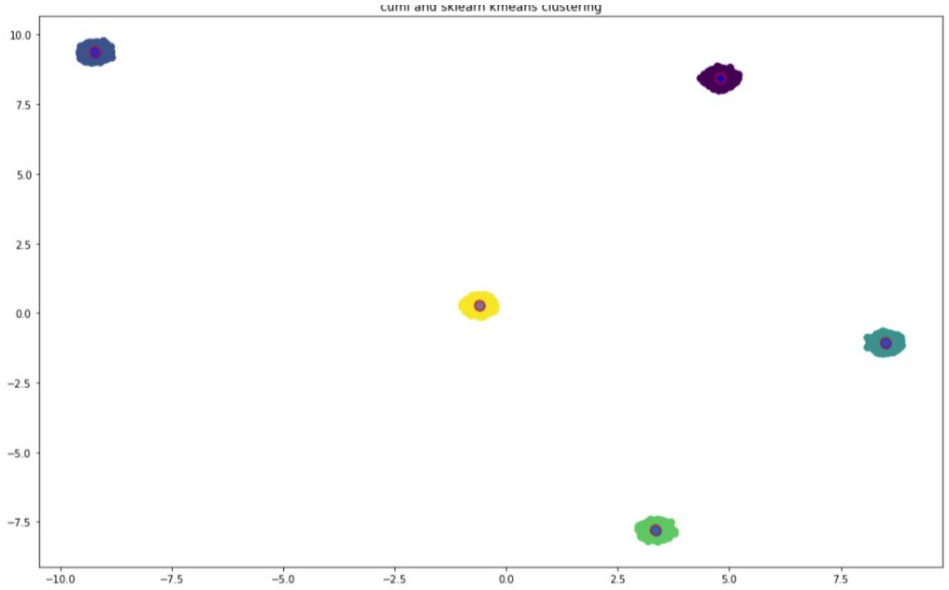
```
fig = plt.figure(figsize=(16, 10))
plt.scatter(host_data.iloc[:, 0], host_data.iloc[:, 1], c=host_labels, s=50, cmap='viridis')

#plot the sklearn kmeans centers with blue filled circles
centers_sk = kmeans_sk.cluster_centers_
plt.scatter(centers_sk[:,0], centers_sk[:,1], c='blue', s=100, alpha=.5)

#plot the cuml kmeans centers with red circle outlines
centers_cuml = kmeans_cuml.cluster_centers_
plt.scatter(cupy.asnumpy(centers_cuml[0].values),
            cupy.asnumpy(centers_cuml[1].values),
            facecolors = 'none', edgecolors='red', s=100)

plt.title('cuml and sklearn kmeans clustering')

plt.show()
```



KMeans vs Kmeans_cuML

Compare Results

```
%%time
cuml_score = adjusted_rand_score(host_labels, kmeans_cuml.labels_.to_array())
sk_score = adjusted_rand_score(host_labels, kmeans_sk.labels_)
```

```
CPU times: user 39.9 ms, sys: 450 µs, total: 40.4 ms
Wall time: 51 ms
```

```
threshold = 1e-4

passed = (cuml_score - sk_score) < threshold
print('compare kmeans: cuml vs sklearn labels_ are ' + ('equal' if passed else 'NOT equal'))
```

```
compare kmeans: cuml vs sklearn labels_ are equal
```

DEEP
LEARNING
INSTITUTE

PRAIRIE VIEW
A&M UNIVERSITY

DLI Accelerated Data Science Teaching Kit

Thank You