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.

RAPIDS

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.

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 randomly 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 pandas or cuDF DataFrames as the input. You can also use the pandas GPU accelerator extension, cuDF.pandas to speed up the processing.

KMeans vs Kmeans_cuML

Import packages

```
# load the cuDF GPU extension for Pandas
%load_ext cudf.pandas
import pandas as pd

# Import CPU based libraries
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans as KMeans_skl
from sklearn.metrics import adjusted_rand_score
from sklearn.datasets import make_blobs as make_blobs_sk

# Import GPU accelerated libraries
from cuml.cluster import KMeans as KMeans_cuml
```

```
# Import GPU accelerated libraries
from cuml.cluster import KMeans as KMeans_cuml
from cuml.datasets import make_blobs as make_blobs_cuml
import cupy
```

Setting parameters

```
n_samples = 10000000
n_features = 2
n_clusters = 5
random_state = 0
```

KMeans vs Kmeans_cuML

Generating Data with SK Learn

```
%%time
data_sk, labels_sl = make_blobs_sk(n_samples=n_samples,
                                   n_features=n_features,
                                   centers=n_clusters,
                                   random_state=random_state,
                                   cluster_std=0.1)
```

```
CPU times: user 3.24 s, sys: 193 ms, total: 3.43 s
Wall time: 3.56 s
```

```
host_data = pd.DataFrame(data_sk)
host_labels = pd.Series(labels_sl)
```

Generating Data with cuML

```
%%time
data, labels = make_blobs_cuml(n_samples=n_samples,
                               n_features=n_features,
                               centers=n_clusters,
                               random_state=random_state,
                               cluster_std=0.1)
```

```
CPU times: user 485 ms, sys: 40.7 ms, total: 526 ms
Wall time: 513 ms
```

```
device_data = pd.DataFrame(data)
device_labels = pd.Series(labels)
```


KMeans vs Kmeans_cuML

SK Learn KMeans

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

kmeans_sk.fit(host_data)
```

```
CPU times: user 3.95 s, sys: 25.1 s, total: 29.1 s
Wall time: 22.9 s
```

cuML KMeans

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

kmeans_cuml.fit(device_data)
```

```
CPU times: user 1.11 s, sys: 110 ms, total: 1.22 s
Wall time: 1.22 s
```

KMeans vs Kmeans_cuML

Visualize Centroids

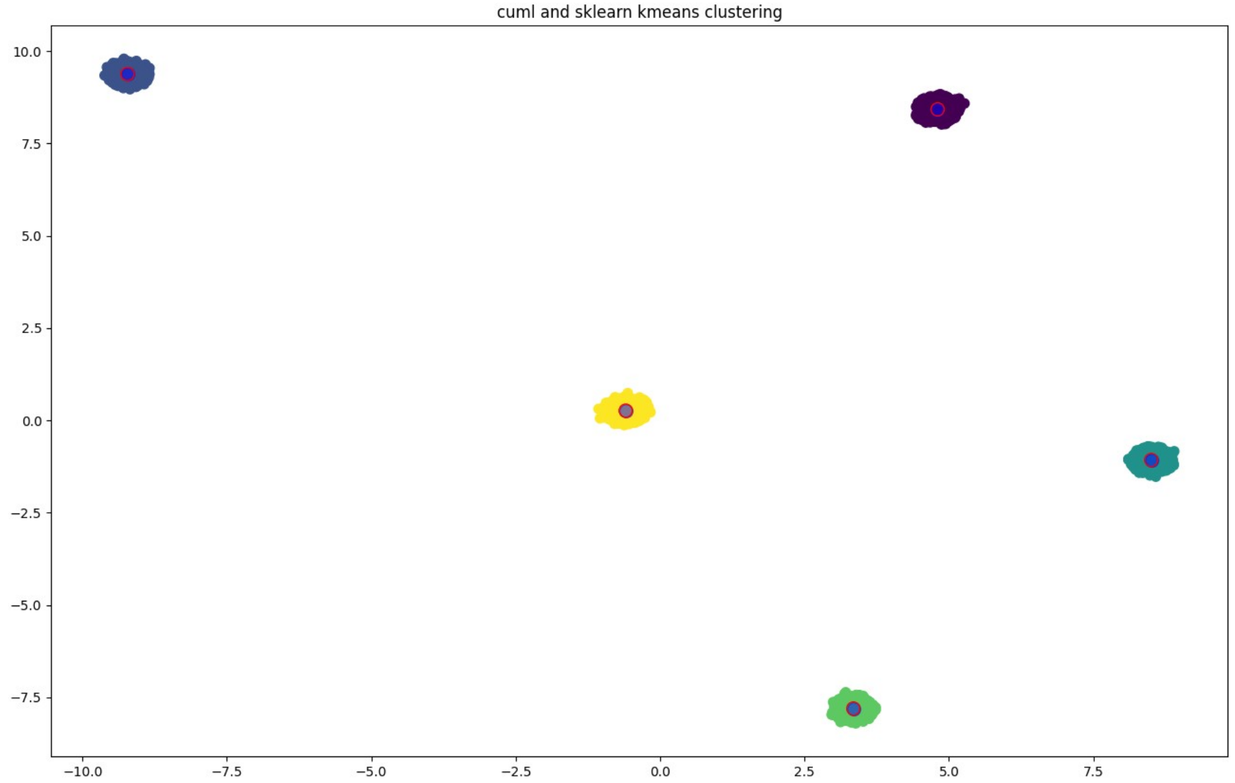
```
fig = plt. figure(figsize=(16, 10))
plt.scatter(host_data.iloc[:, 0], host_data.iloc[:, 1],
c=host_labels, s=50, cmap='viridis')

# 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_.values)
sk_score = adjusted_rand_score(host_labels, kmeans_sk.labels_)
```

```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 7.63 µs
```

```
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
```

DLI Accelerated Data Science Teaching Kit

Thank You