KNN

In this notebook you will use GPU-accelerated k-nearest neighbors to identify the nearest road nodes to hospitals.

Objectives

By the time you complete this notebook you will be able to:

- Use GPU-accelerated k-nearest neighbors using a single GPU

Imports

In [ ]:
```
import cudf
import cuml
```

Load Data

Road Nodes

We begin by reading our road nodes data.

In [ ]:
```
road_nodes = cudf.read_csv('./data/road_nodes_2-06.csv', dtype=['str', 'f
```

In [ ]:
```
road_nodes.dtypes
```

In [ ]:
```
road_nodes.shape
```

In [ ]:
```
road_nodes.head()
```

Hospitals

Next we load the hospital data.

In [ ]:
```
hospitals = cudf.read_csv('./data/hospitals_2-06.csv')
```

In [ ]:
```
hospitals.dtypes
```

In [ ]:
```
hospitals.shape
```

In [ ]:
```
hospitals.head()
```

K-Nearest Neighbors

We are going to use the k-nearest neighbors algorithm to find the nearest *k* road nodes for every hospital. We will need to fit a KNN model with road data, and then give our trained model hospital locations so that it can return the nearest roads.

Exercise: Prep the KNN Model

Create a k-nearest neighbors model `knn` by using the `cuml.NearestNeighbors` constructor, passing it the named argument `n_neighbors` set to 3.

In [ ]:

```
```

Solution

In [ ]:

```
%load solutions/prep_knn
```

Exercise: Fit the KNN Model

Create a new dataframe `road_locs` using the `road_nodes` columns `east` and `north`. The order of the columns doesn't matter, except that we will need them to remain consistent over multiple operations, so please use the ordering `['east', 'north']`.

Fit the `knn` model with `road_locs` using the `knn.fit` method.

In [ ]:

```
```

Solution

In [ ]:

```
%load solutions/fit_knn
```

Exercise: Road Nodes Closest to Each Hospital

Use the `knn.kneighbors` method to find the 3 closest road nodes to each hospital. `knn.kneighbors` expects 2 arguments: `X`, for which you should use the `easting` and `northing` columns of `hospitals` (remember to retain the same column order as when you fit the `knn` model above), and `n_neighbors`, the number of neighbors to search for--in this case, 3.

`knn.kneighbors` will return 2 cudf Dataframes, which you should name `distances` and `indices` respectively.

In [ ]:

```
```

Solution

In [ ]:

```
%load solutions/k_closest_nodes
```

Viewing a Specific Hospital

We can now use `indices`, `hospitals`, and `road_nodes` to derive information specific to a given hospital. Here we will examine the hospital at index `10`. First we view the hospital's grid coordinates:

In [ ]:
```
SELECTED_RESULT = 10
print('hospital coordinates:\n', hospitals.loc[SELECTED_RESULT, ['easting
```

Now we view the road node IDs for the 3 closest road nodes:

In [ ]:
```
nearest_road_nodes = indices.iloc[SELECTED_RESULT, 0:3]
print('node_id:\n', nearest_road_nodes, sep='')
```

And finally the grid coordinates for the 3 nearest road nodes, which we can confirm are located in order of increasing distance from the hospital:

In [ ]:
```
print('road_node coordinates:\n', road_nodes.loc[nearest_road_nodes, ['ea
```

Please Restart the Kernel

In [ ]:
```
import IPython
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```

Next

In the next notebook, you will return to the K-means algorithm, but this time using a multi-node, multi-GPU Dask version that can scale to production size.