data-process

March 18, 2024

[1]:
```
import pandas as pd
import pyarrow.parquet as parquet
first = 'QCDToGGQQ_IMGjet_RH1all_jet0_run0_n36272.test.snappy.parquet'
second = 'QCDToGGQQ_IMGjet_RH1all_jet0_run1_n47540.test.snappy.parquet'
third = 'QCDToGGQQ_IMGjet_RH1all_jet0_run2_n55494.test.snappy.parquet'
```

[2]:
```
parquet_file = parquet.ParquetFile('/content/drive/MyDrive/Sci_data/'+first)

num_row_groups = parquet_file.num_row_groups

print("\Total Number of Row Groups:", num_row_groups)

chunk_size =3000
```

```
\Total Number of Row Groups: 36272
```

Due to computational constraints, the complete dataset cannot be read at once. Instead, chunks of data are read from the dataset and are later saved as .pt files.

[3]:
```
chunk_num = 1
for batch in parquet_file.iter_batches(chunk_size):
    print("RecordBatch")
    batch_df = batch.to_pandas()
    chunk_num -=1
    if chunk_num == 0:
      break
```

```
RecordBatch
```

[4]:
```
batch_df['y'].value_counts()
```

[4]:
```
0.0    1520
1.0    1480
Name: y, dtype: int64
```

0.1 Converting Image Dataset to Graph Dataset

[5]:
```
import numpy as np
img_list = []

limit = chunk_size
for number in range(limit):
  for idx, channels in enumerate(batch_df['X_jets'][number]):
    for i, row in enumerate(channels):
      if i==0:
        img = row
      else:
        img = np.vstack([img, row])
    if idx==0:
      final_img = img
    else:
      final_img = np.dstack([final_img, img])
  img_list.append(final_img)
```

[6]:
```
mini_df = batch_df[['pt', 'm0',         'y']]
del batch_df
```

[7]:
```
x_jet = np.array(img_list)
```

[8]:
```
x_jet.reshape((chunk_size, -1, 3)).shape
```

[8]:
```
(3000, 15625, 3)
```

[9]:
```
X_data = x_jet.reshape((chunk_size, -1, 3))
del img_list
del x_jet
```

[10]:
```
unmasked = np.any(X_data != 0., axis=-1)
```

[11]:
```
unmasked.shape
```

[11]:
```
(3000, 15625)
```

[12]:
```
node_list = []
for i, x in enumerate(X_data):
    node_list.append(x[unmasked[i]])
```

[13]:
```
del unmasked
del X_data
```

[14]:
```
y = mini_df['y'].to_numpy()
y = y.reshape((-1,1))
```

[15]:
```
import torch
from tqdm import tqdm
from sklearn.neighbors import kneighbors_graph
from torch_geometric.data import Data

dataset = []
for i,nodes in enumerate(tqdm(node_list)):
    dummy = mini_df['m0'][i]*np.ones((nodes.shape[0],1))
    dummy2 = mini_df['pt'][i]*np.ones((nodes.shape[0],1))
    nodes = np.concatenate([nodes, dummy, dummy2], axis=-1)
    edges = kneighbors_graph(nodes, 10, mode='connectivity', include_self=True)
    c = edges.tocoo()
    # print(c)
    edge_list = torch.from_numpy(np.vstack((c.row, c.col))).type(torch.long)
    edge_weight = torch.from_numpy(c.data.reshape(-1,1))
    y_g = y[i]
    # print(type(torch.from_numpy(nodes)), type(edge_list), type(edge_weight))
    dataset.append(Data(x=torch.from_numpy(nodes), edge_index=edge_list,␣
 ↪edge_attr=edge_weight, y=torch.from_numpy(y_g).long()))
```

```
100%|      | 3000/3000 [00:17<00:00, 172.52it/s]
```

[16]:
```
dataset[0]
```

[16]:
```
Data(x=[717, 5], edge_index=[2, 7170], edge_attr=[7170, 1], y=[1])
```

[17]:
```
torch.save(dataset, '/content/drive/MyDrive/Sci_data/first.pt')
```

[17]:
```

```