Junghwan Kim

Professor Tingjian Ge

COMP.6440 Topics in Data Mining

Project Report

Task 1: Build GAT Model

Code File: `/Python/GAT/main.py`

Model File: `/Python/GAT/my_model/saved_model.pb`

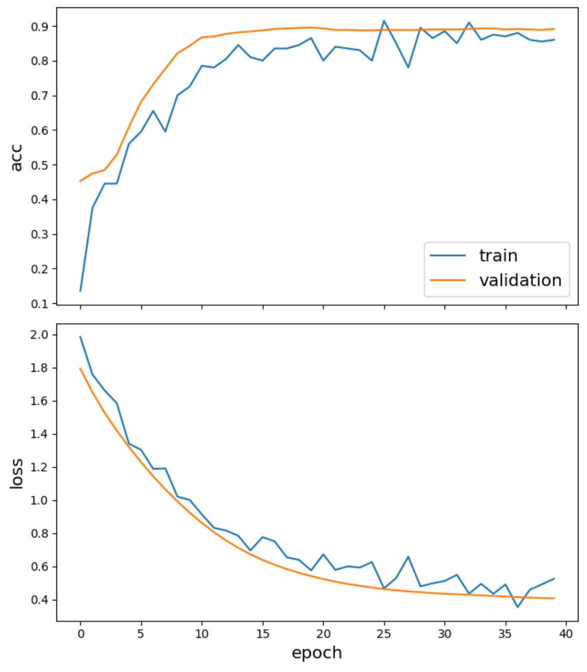
Training Model Parameters

Training/Validation/Testing	200 nodes for training, 700 for validation, and the rest for testing
Epoch Size	50
GAT Layer Size	8
GAT Activations	Softmax
Compile Optimizer	Adam

Training Model Statistics

StellarGraph	Undirected multigraph (Nodes: 2708, Edges: 5429)
Accuracy (After 50 epochs)	0.8761
Loss (After 50 epochs)	0.5645

Output: Evaluation Graph (`/Python/GAT/results/Evaluate.png`)

	Accuracy Graph ----- Train (0.8761) ----- Validation (0.8914) Loss Graph ----- Train (0.5645) ----- Validation (0.4057)

Task 2: Cluster GAT Models using K-means

Code File: `/Python/GAT/main.py`

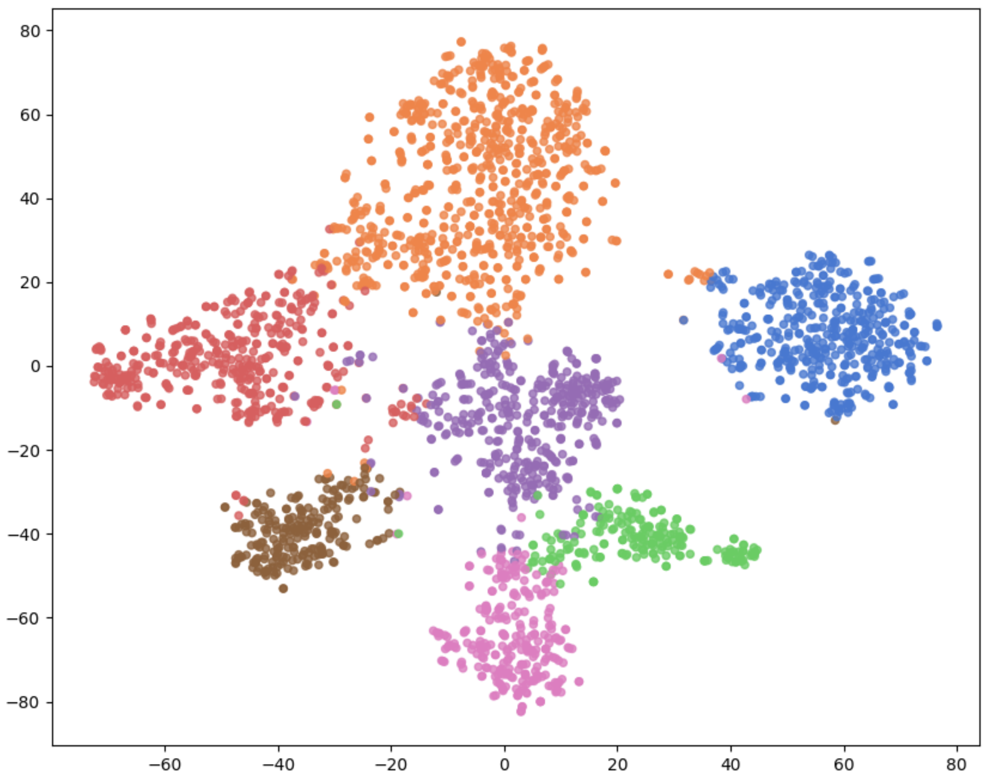
Steps

Step	Task	Run File	Output File
1	Preprocess datasets	`main.py`	
2	Node classification with GAT		
3	Making predictions with the model		
4	Node embeddings		
5	K-means Clustering		`results/K-means Classification.png`

Clustering Model Statistics

Mutual Information	1.334
Normalized Mutual Information	0.725
Adjusted Mutual Information	0.724

Output: K-means Clustering Graph (`/Python/GAT/results/K-means Classification.png`)



K-means Clustering Graph shows 7 clusters for classification.

Task 3: Build LSTM Model

Model File: `/Python/LSTM/my_model/saved_model.pb`

Steps

Step	Task	Run File	Output File
1	**Preprocess ny.csv File** 1) Load dataset 2) Set columns 3) Set index 4) Drop rows which contains NULL 5) Output preprocessed CSV file	`step1_preprocess.py`	`data/ny_preprocessed.csv`
2	**Build LSTM Model** 1) Load preprocessed dataset 2) Check NULL value 3) Use MinMaxScaler to normalize 4) Split train/test dataset 5) For LSTM, shift "distance to target" value into one row 6) Drop rows which contains NULL 7) Reshape to three dimensions 8) Build LSTM model 9) Evaluate model 10) Output evaluation graph and score	`step2_train.py`	`data/ny_X.csv` `results/Evaluate.png`
3	**Predict LSTM Model** 1) Load MinMaxScaler parameters for restoring normalized values 2) Load line numbers to split dataset by taxi ID 3) Predict "distance to target" by timestamp by taxi ID 4) Output final result	`step3_predict.py`	`results/ny_output.csv`

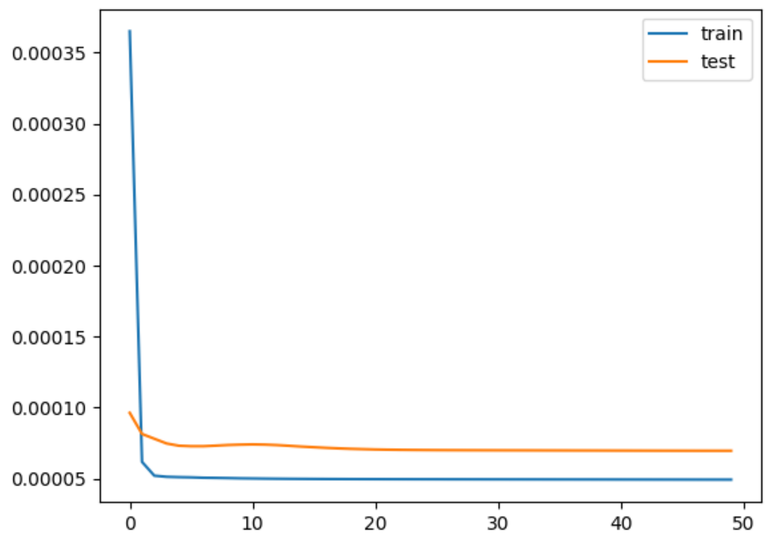
Training Model Parameters

Training/Testing	50 taxis as training data and the rest for testing
Epoch Size	50
Batch Size	32
Compile Optimizer	Adam
Compile Loss	Mean squared error

Training Model Statistics

Evaluation Score	0.00006961073813727126
Loss (After 50 epochs)	4.7173e-05
Validation Loss (After 50 epochs)	6.9611e-05

Evaluation Graph

	Loss Graph ----- Train (0.00005) ----- Validation (0.0006)

Final Output (`/Python/LSTM/results/ny_output.csv`)

My outputs predict how soon a taxi will arrive at the target zone. These are sorted first by taxi ID, and then by timestamp. Please see **`ny_output.csv`** file to view all outputs.

```
id,timestamp,predicted_distance_to_target
1,0,1251.1713
1,1,1247.5228
1,2,1238.1479
1,3,1237.8492
1,4,1229.1423
1,5,1219.4430
1,6,1202.1625
1,7,1176.1140
1,8,1156.8153
1,9,1156.9484
1,10,1149.0439
1,11,1149.7302
1,12,1143.4977
1,13,1140.2195
1,14,1125.1387
1,15,1111.5150
1,16,1105.6411
1,17,1090.4906
1,18,1079.3046
1,19,1068.6392
1,20,1060.4761
1,21,1048.1577
1,22,1037.5543
```

```
1,23,1033.6733
1,24,1018.4643
1,25,1004.5157
1,26,997.0363
1,27,964.7749
1,28,953.8015
1,29,953.0269
1,30,929.9172
1,31,932.9108
1,32,925.2999
1,33,924.0981
1,34,905.2350
1,35,904.8619
1,36,893.9295
1,37,896.5084
1,38,888.6772
1,39,891.2866
1,40,884.4792
1,41,886.6830
1,42,866.0713
1,43,869.2360
1,44,857.4360
1,45,858.1281
1,46,837.5607
1,47,840.2827
1,48,816.9507
1,49,818.7489
1,50,811.4332
1,51,809.8998
1,52,796.8458
1,53,797.6133
1,54,788.9515
1,55,783.4568
1,56,767.0399
1,57,757.1478
1,58,744.2687
1,59,746.9658
1,60,726.3939
1,61,729.3295
1,62,718.1500
1,63,720.9903
1,64,709.2109
1,65,712.2128
1,66,705.6347
1,67,709.8666
1,68,690.3552
1,69,635.7615
1,70,608.2512
```

```
1,71,592.6343
1,72,579.4050
1,73,579.1539
1,74,567.0125
1,75,564.4207
1,76,549.0526
1,77,541.3196
1,78,527.8050
1,79,484.3907
1,80,471.0285
1,81,456.8376
1,82,448.5786
1,83,449.3603
1,84,436.6210
1,85,438.9530
1,86,432.6014
1,87,432.5684
1,88,426.5475
1,89,428.0057
1,90,412.0925
1,91,408.7707
1,92,392.2147
1,93,388.2119
1,94,383.2978
1,95,385.0688
1,96,377.4062
1,97,276.3567
1,98,258.5113
1,99,255.9548
1,100,243.8286
1,101,249.2333
1,102,237.5166
1,103,221.5310
1,104,202.9872
1,105,205.4349
1,106,181.9404
1,107,178.6617
1,108,161.6320
1,109,161.8505
1,110,138.8034
1,111,144.7842
1,112,127.4711
1,113,120.0878
1,114,95.0565
1,115,99.7074
1,116,59.4443
...
```