INTRODUCTION

Transportation systems are infrastructures of cardinal importance in modern societies [1]. As these systems get more and more complex, traffic prediction becomes more essential in urban management and traffic engineering [2]. Over the past decades, various models have been developed to estimate traffic flows, among which the Four-Step Model (FSM), consisting of the steps: trip generation, trip distribution, modal split, and trip assignment, has gained significant popularity [2]. Trip distribution modeling (TDM) is the second step and a crucial component of the Four-Step transportation planning model which provides valuable insights into the spatial allocation of trips from origins to destinations within a given area.

Various models have been developed and examined for predicting trips distributed in transportation networks. TDM mostly utilizes statistical methods such as time-series, regression and the gravity model (GM), which the later is a very popular one [6]. GM derives from the concept of gravitational forces in physics. It distributes trips between zones by considering the relative attractiveness of each zone and a function that incorporates the distances between them. The gravity model, despite its widespread use in TDM, has its own limitations. Overestimation of short trips, underestimation of long trips, and data deformity are some of the drawbacks of GM [9]. GM usually needs large amount of data and high calibration percentage in order to perform well [7]. Also, GM doesn't yield satisfactory results for networks with diverse or heterogeneous characteristics [8].

Machine learning (ML) is a branch of artificial intelligence that focuses on the development of algorithms and models capable of ~~automatically~~ learning from data and making predictions or decisions without being explicitly programmed. It involves the utilization of statistical techniques and optimization algorithms to enable computers to improve their performance on specific tasks through experience and data-driven learning. Various ML models including neural network (NN), random forest (RF), and generalized regression neural network (GRNN) have been compared to traditional models such as GM in previous articles. In [4], GM has shown a poor performance in commuting trip distribution at the census tract level in NYC, compared to RF and ANN in terms of mean square error (MSE) and R^2. Also, the GRNN model adopted in [9] has a lower average Root Mean Square Error (RMSE) and higher R^2 compared to GM.

Summary of comments: Jamali's paper-First Draft_CM.pdf

Page:1

Author: Date: 0000-00-00 00:00:00

intro is too long and too many detail about ML and four-step

you do not need to explain papers in detail

Author: Date: 0000-00-00 00:00:00

what do you mean?

Transportation systems are important infrastructures in modern societies.

Author: Date: 0000-00-00 00:00:00

Author: Date: 0000-00-00 00:00:00

Author: Date: 0000-00-00 00:00:00

stop using too many more

Author: Date: 0000-00-00 00:00:00

which is popular

Author: Date: 0000-00-00 00:00:00

what is automatic?

Author: Date: 0000-00-00 00:00:00

Author: Date: 0000-00-00 00:00:00

R-Squared

Author: Date: 0000-00-00 00:00:00

which is adopted

Author: Date: 0000-00-00 00:00:00

The rise of machine learning (ML) has revolutionized problem-solving approaches and spurred researchers from diverse disciplines to embrace this innovative methodology. ML methods' ability to effectively handle nonlinearities, discontinuities, and polynomial aspects makes them applicable across various domains, including transportation systems [6]. For example, Intelligent Transportation Systems (ITS), have been positively impacted by the growth of machine learning [8]. For example, ML models have been widely used in traffic incidents, since they can learn traffic patterns and then detect and isolate accident conditions from regular conditions [8]. Additionally, ML techniques are crucial in driving advancements in areas such as advanced driver assistance systems (ADAS) [1], traffic prediction [9], and optimization [5] . Deep Learning (DL), a specific type of ML has become popular in transportation systems due to the increased availability of data and advancements in computational techniques [8]. CNN, and RNN have been the most frequently used models in ride-sharing and public transportation [8]. CNN has also been used in h ITS-related visual recognition tasks [8]. Traffic networks often have a graph-like structure, so new deep learning techniques called graph neural networks (GNNs) have been developed to handle graph-structured data [13]. GNNs have demonstrated remarkable performance in diverse applications; for example, traffic congestion forecasting [11], travel demand forecasting [12], transportation safety [13], traffic surveillance [14], and autonomous driving [10],[15]. GNNs are particularly well-suited for traffic forecasting tasks due to their capability to capture spatial dependencies, which are often represented using non-Euclidean graph structures [1]. In the field of TDM, there have been several studies focusing on neural network (NN) approaches. Initially, the results obtained from these NN models were not satisfactory[14]. However, with further research and advancements, the performance of NN models improved over time [3][5][6]. Despite these improvements, there is currently a lack of research specifically exploring the adoption of GNNs in TDM. GNNs possess several features that make them highly suitable for TDM. The spatial modeling capabilities, understanding of location-level and network-level information, and handling of heterogeneous data make GNNs well-suited for TDM. Given the powerful features of GNNs, it is presumed that this model has the potential to outperform conventional methods in this domain. Therefore, the objective of this paper is to investigate the potential benefits of employing GNNs in TDM and evaluate their performance compared to traditional NN and GM models.

Page:2

Author: Date: 0000-00-00 00:00:00

nonsense

Author: Date: 0000-00-00 00:00:00

?

Author: Date: 0000-00-00 00:00:00

models?

Author: Date: 0000-00-00 00:00:00

In TDM, initial indications suggest that GNNs could be highly effective in capturing spatial dependencies between transportation entities, such as origins, destinations, and their connections. By considering the graph structure of the transportation network, GNNs can learn and leverage the relationships between different locations, including their proximity, connectivity, and traffic patterns [7][8][9][10]. Moreover, TDM often requires integration of diverse data sources, such as traffic counts, land use characteristics, socio-economic factors, and transportation infrastructure details. GNNs can handle heterogeneous data by incorporating features associated with each node and edge in the graph representation [11][12]. This allows for the simultaneous consideration of multiple data types, leading to more comprehensive and accurate modeling compared to the GM or traditional NNs. Additionally, GNNs have the advantage of learning end-to-end representations from raw input data. They can automatically discover relevant features and patterns within the transportation network, reducing the need for manual feature engineering [13][14]. This is particularly beneficial in scenarios where the underlying relationships and dependencies are complex and not easily captured using traditional models like GM or NNs.

Literature

In the literature, numerous methods have been utilized to predict travel distribution. Traditional approaches mostly rely on statistical models, including regression analysis [25] and GM [6],[26], which estimate travel patterns based on historical data. The widely used GM method falls short in accurately predicting outcomes on heterogeneous networks, leading Almog et al. [6] to develop an enhanced GM model to address this limitation. This method ~~typically~~ focuses on aggregate-level information, assumes static relationships between variables, and do not handle nonlinearities well [5],[7]. However, with the advent of ML techniques, new approaches have emerged. ML algorithms, including random forests [7], and support vector machines [27], have been applied to TDM. These methods leverage the power of data-driven models to capture complex relationships and patterns in travel behavior. In recent studies, NNs have been widely explored and compared to previously used models, particularly GM, in trip modelling. Tillema et al. [5] developed a NN model that outperformed GM when data was scarce. In [16], the accuracy of the NN model is twice that of the direct demand model.

Page:3

Author: Date: 0000-00-00 00:00:00

too general and no support for it

Author: Date: 0000-00-00 00:00:00

it does not make sense here

Author: Date: 0000-00-00 00:00:00

combine intro and lit

it is too long

Author: Date: 0000-00-00 00:00:00

Author: Date: 0000-00-00 00:00:00

Author: Date: 0000-00-00 00:00:00

Author: Date: 0000-00-00 00:00:00

Author: Date: 0000-00-00 00:00:00

what does the refrence says?

Additionally, deep learning has emerged as a powerful tool in various domains, and its potential in transportation has gathered significant attention. The utilization of deep learning methods in the transportation sector has yielded encouraging outcomes, paving the way for improved transportation systems and services. The papers collectively suggest that deep learning methods, including recurrent neural networks (RNNs) [10], generalized regression neural networks (GRNN) [4], and convolutional neural networks (CNNs) [9],[28], have shown promising results in predicting travel distribution. In [4], the GRNN model slightly outperformed the GM in terms of RMSE and R2. Nguyen et al. [29] provided a review of deep learning applications in transportation, highlighting its potential while also acknowledging limitations in its current utilization. Varghese et al. [30] investigated the impact of spatial and temporal resolution on demand prediction using deep learning models, finding that the granularity of space and time can improve prediction performance. Yao et al. [31] proposed a Deep Multi-View Spatial-Temporal Network (DMVST-Net) framework that considers both spatial and temporal relations for taxi demand prediction, demonstrating its effectiveness over existing methods. Markou et al. [32] explored the combination of time-series data and semantic information using machine learning and deep learning techniques for trip demand prediction in event areas, showing significant error reduction in forecasts.

Recently, GNNs have gained attention for their ability to model the spatial and relational aspects of travel distribution [1],[10],[33]. These methods provide more accurate predictions by considering the interconnections between different locations and their influences on travel patterns. Recent studies suggest that GNNs are promising tools for traffic demand modeling. In their comprehensive survey, Jiang et al. [1] provided a comprehensive survey of recent research using GNNs in various traffic forecasting problems, demonstrating that GNNs have achieved state-of-the-art performance. Diehl et al. [34] specifically evaluated GNNs for modeling traffic participant interaction and found that GNNs can effectively take the interaction between traffic participants into account. In comparison to other modeling techniques, Golshani et al. [35] found that NNs offer better predictions for travel mode and departure time decisions, suggesting that GNNs may also outperform other modeling techniques. However, none of these papers directly compare GNNs to NNs or GMs in trip distribution modeling, so further research is needed to fully address this research gap.

Page:4

Author: Date: 0000-00-00 00:00:00

extra information , no need

Author: Date: 0000-00-00 00:00:00

Author: Date: 0000-00-00 00:00:00

METHOD

1. Trip Distribution

Trip distribution modeling involves the study of spatial interaction, which encompasses the movement of goods, individuals, money, or information across geographical space (Fotheringham and O'Kelly, 1989). At its core, trip distribution involves the examination of travel behavior to discern the likelihood of trips originating from specific zones to terminate at others. For doing this, a better understanding of the pattern of trip making other than productions and attractions is needed. Although productions and attractions provide a general idea of the level of trip making in a study area, they are often inadequate for modeling and decision-making purposes. It is necessary to have a better understanding of the pattern of trip making in order to create an accurate and usable travel demand model. An intriguing challenge arises when information is available on the number of trips originating and ending in each zone. In such cases, the total number of trips produced in a zone should correspond to the total interaction flows exiting that zone. Spatial interaction models are categorized based on the constraints imposed on the predicted trip matrix, which captures prior knowledge about the total interaction flows entering and/or exiting specific zones.

When information is available on the number of trips originating and ending in each zone, an important consideration arises. The total number of trips originating from a particular zone, referred to as trip production, should be equal to the total number of interaction flows exiting that zone:

$$\sum_j T_{ij} = O_i, \forall_i$$

Similarly, the total number of trips ending in a particular zone, known as trip attraction, should be equal to the total number of interaction flows entering that zone:

$$\sum_j T_{ij} = D_i, \forall_i$$

Page:5

Author: Date: 0000-00-00 00:00:00

format of the method is not acceptable

Author: Date: 0000-00-00 00:00:00

Author: Date: 0000-00-00 00:00:00

studying spatial interaction of what?

Author: Date: 0000-00-00 00:00:00

use same citing format

Author: Date: 0000-00-00 00:00:00

what do you mean?

Author: Date: 0000-00-00 00:00:00

it is not clear what you want to say

Author: Date: 0000-00-00 00:00:00

what do you mean by pattern of trip making ?

is it demand pattern?
trip pattern?
trip distribution?

Author: Date: 0000-00-00 00:00:00

on limited number you mean?

what is the challenge?

Author: Date: 0000-00-00 00:00:00

why it is a problem?
why it arise consideration?

Author: Date: 0000-00-00 00:00:00

equations should have number and explain them

When both Equation 1 and Equation 2 are satisfied, the model is referred to as a doubly constrained gravity model:

$$T_{ij} = A_i O_i B_j D_j f(c_{ij})$$

Here, A_i and B_j are balancing factors, c_{ij} represents the travel impedance, O_i and D_j represent production and attraction, $f(c_{ij})$ denotes the distribution function, and T_{ij} represents the estimated flow or interaction between origin zone i and destination zone j.

In trip distribution modeling, the balancing factors A_i and B_j, along with the travel impedance c_{ij} and the distribution function $f(c_{ij})$, play significant roles. Various constraints can be applied to model trip distribution. Studies have demonstrated that the doubly constrained gravity model provides the most accurate results for estimating spatial interaction (Ortuzar and Willumsen, 2001). Therefore, in this article, the doubly constrained model is used as the baseline.

2. Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are a type of neural network designed to operate on graph-structured data. In GNNs, each node in the graph is associated with a feature vector that represents the node's attributes, while each edge is associated with a weight representing the strength of the connection between two nodes.

To work with GNNs, the first step is to construct a graph network. In this research, we divide the studied city into zones, which are then represented as nodes in the graph. Each zone is associated with its respective attractions and productions as node features. The source and destination of travel are used as the graph's edge index, while travel times serve as edge weights.

Page:6

Author: Date: 0000-00-00 00:00:00

where is it ?

Author: Date: 0000-00-00 00:00:00

use equation

Author: Date: 0000-00-00 00:00:00

you need compelet paragraphs and right format

Author: Date: 0000-00-00 00:00:00

what significant role?!

Author: Date: 0000-00-00 00:00:00

what is doubly constrained model? you need to explain it

During training, GNNs utilize message passing to propagate information between nodes in the graph. In each iteration of message passing, each node aggregates information from its neighbors and updates its own feature vector accordingly. The updated feature vectors are then used in the subsequent iteration of message passing.

3.1 Calibration of Gravity Model

The calibration process of the gravity model follows a general form discussed in [1], which can be expressed using Bayes' theorem as:

$$p_{i,j} = a_i b_j f\left(\mu, C_{i,j}\right) \quad (1)$$

In Equation (1) a_i, b_j, and μ represent the model parameters, while $C_{i,j}$ represents constants. The function f is a specified smooth function.

Different decay functions, such as the exponential model and power model, can be utilized. The decay functions are defined as follows:

For the exponential model: $f\left(\mu, C_{ij}\right) = \exp\left(-\mu C_{ij}\right)$ (2)

For the power model: $f\left(\mu, C_{ij}\right) = {C_{ij}}^{-\mu}$ (3)

According to [1], using Equations (1) and (2), the exponential model can be expressed as:

$$T_{ij} = \beta P_i A_j \exp\left(-\mu C_{ij}\right) \quad (4)$$

Similarly, utilizing Equations (1) and (3), the power model is given by:

$$T_{ij} = \beta P_i A_j {C_{ij}}^{-\mu} \quad (5)$$

Here, P and A represent production and attraction, while C denotes travel impedance. The objective of calibration is to determine the optimal values for the β and μ parameters in the above equations.

To facilitate calibration, Equations (4) and (5) can be transformed into a linear regression model in the form of y = mx + b.(Least Square Method)
For the exponential model, we have:

Page:7

Author: Date: 0000-00-00 00:00:00

section numbers are random?!

use paragraphs

you can use tables to show different models and then explain them

Author: Date: 0000-00-00 00:00:00

use table

Author: Date: 0000-00-00 00:00:00

put all models in a table together

be clear and use equation

$$Ln\left(\frac{T_{i,j}}{P_i A_j}\right) = Ln(\beta) - \mu C_{ij} \rightarrow m = -\mu \,\&\, b = Ln(\beta) \qquad (6)$$

For the power model, we have:

$$\text{Ln}\left(\frac{T_{ij}}{P_i A_j}\right) = Ln(\beta) - \mu \times Ln(C_{ij}) \rightarrow m = -\mu \,\&\, b = Ln(\beta) \qquad (7)$$

By fitting a line through the data obtained from Equations (6) or (7), we can calculate the slope (m) and intercept (b) of the line. These values correspond to the parameters μ and Ln(β), respectively.

Hence, the gravity model can be calibrated by estimating the values of m and b through the linear regression process.

3.2 Neural Network

The process of modeling with a neural network consists of two main parts. The first part involves choosing a proper neural network architecture based on the available data. In trip distribution problems, three distinct inputs and one output are typically used to construct the neural network's structure. However, in cases where the inputs lack discriminatory information, the number of inputs may decrease to two or even one.

Input: trip attraction, trip production, impedance

Output: trip distribution

The second part of building a neural network is specifying hyperparameters, including the number of layers, the number of units in each layer, activation functions, epoch number, batch size, and learning rate, among others.

For optimal results, we propose a neural network with 2 hidden layers, each containing 8 units. These numbers are determined through careful examination, allowing the model to learn faster and better while preventing overfitting and underfitting. We assume 500 epochs with an early-stopping strategy to stop the model earlier and prevent overfitting. Each epoch comprises several batches, with

Page:8

Author: Date: 0000-00-00 00:00:00

use table

Author: Date: 0000-00-00 00:00:00

these information are trivial

just say what you did , what is the layer number and ...

Author: Date: 0000-00-00 00:00:00

write a compelet paragraph and follow the format

Author: Date: 0000-00-00 00:00:00

you said that in the next part as well
delet the extra and just keep one part

Author: Date: 0000-00-00 00:00:00

repeating

each batch containing 512 data points. Batch sizes are usually set to 32 or 16, but in this case, as the region is large with hundreds of thousands of training data, a higher batch size yields excellent results. A learning rate of 0.0001 is chosen to ensure a gradual and stable learning process.

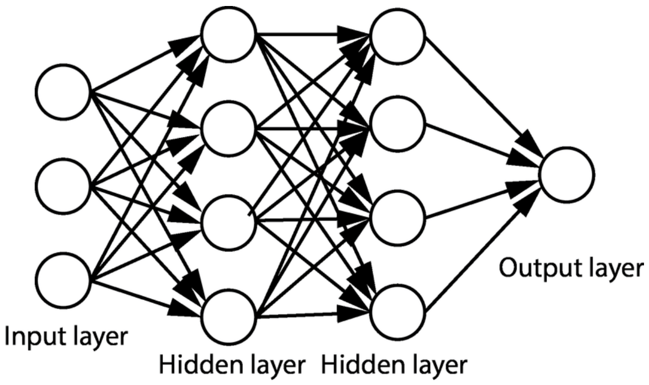

Figure 1: Neural Network Architecture

The neural network architecture can be described mathematically as follows:

1. Input Matrices: Let X be the input matrix with dimensions (m, n), where m represents the number of data points and n represents the number of features. The input matrix is defined as:
$$X = [x_1, x_2, \dots, x_m]$$
where x_i represents the feature vector for the ith data point
2. Hidden Layers: The neural network consists of 2 hidden layers, each containing 8 units. The activation function used in the hidden layers is the rectified linear unit (ReLU), which introduces non-linearity and helps the model learn complex patterns. The output of the first hidden layer can be represented as:
$$H_1 = ReLU(W_1 \times X + b_1)$$
where W_1 represents the weight matrix of dimensions (8, n), b_1 represents the bias vector of dimensions (8, 1), and the ReLU function applies element-wise non-linearity.

Similarly, the output of the second hidden layer can be represented as:

$$H_2 = ReLU(W_2 \times X + b_2)$$

where W_2 represents the weight matrix of dimensions (8, 8), b_2 represents the bias vector of dimensions (8, 1).

No Comments.

3. Output Layer: The output layer is a linear layer that produces the predicted trip distribution. It can be represented as:

$$Y_{pred} = W_{output} \times H_2 + b_{output}$$

where W_{output} represents the weight matrix of dimensions (output_dim, 8), b_{output} represents the bias vector of dimensions (output_dim, 1), and Y_{pred} represents the predicted trip distribution.

During training, the neural network optimizes the mean squared error (MSE) loss function, which measures the average squared difference between the predicted trip distribution and the ground truth values. The Adam optimizer is employed to update the network's weights and biases, utilizing adaptive learning rates for each parameter.

The training process involves feeding the input data (trip attraction, trip production, and impedance) into the neural network and obtaining the predicted trip distribution as the output. The predicted values are then compared to the ground truth trip distribution, and the model's parameters are adjusted through backpropagation and gradient descent to minimize the MSE loss

By defining the matrices for the input data, specifying the layers and activation functions, and outlining the training process, we establish a comprehensive understanding of the mathematical operations occurring within the neural network architecture.

3.3 GNN:

In this article, we employed two Graph Neural Network (GNN) models: a two-layer model for smaller data and a single-layer model for larger data. The input of both models consists of test, train, and validation matrices along with the travel time matrix and production and attraction values for each zone. The output of both models is the predicted values for travel values that are not available. Additionally, Mean Squared Error (MSE) was used to calculate the loss value for both models. In the two-layer model, the hyperparameter values are as follows: learning rate equal to 0.015, weight decay equal to 0.000001, hidden layer equal to 4, hidden channels equal to 64, and node embedding size equal to 2. Also, in the single-layer model used for larger data, the hyperparameter values are as follows: learning rate equal to

Page:10

Author: Date: 0000-00-00 00:00:00

you talked about GNN earlier
combine them

0.015, weight decay equal to 0.0000001, hidden channels equal to 64, and node embedding size equal to 8.

A Graph Convolutional Neural Network (GCN) is a type of neural network that operates directly on graphs and takes advantage of their structural information. It consists of stacked GCN layers, where the input is the ordered nodes and their features. In each GCN layer, the neighboring nodes exchange their feature vectors before performing a standard MLP step. The output of a GCN is also on the same graph as the input data, but with processed features, such as a bucket classification.

A GCN with n layers is designed to allow any two nodes within n hops over the graph to exchange their information. The number of layers selected for the entire network determines the maximum graph size that can be effectively used for transport planning. It can be formulated like below:

$$n_i^j = \sigma\left(\sum_{k \in n(i) \cup \{i\}} n_k^{j-1} \times W^j)\right)$$

n_i^j as the feature of node i in GCN layer j, σ the activation function, W^j the weight matrix of layer j and k as all direct neighbors. It should be noted that the W^j is shared in a single GCN layer. The GCN training process involves differentiable operations, which means that backpropagation can be used to calculate the gradient of all weights. Finally, the Adam optimizer is applied to optimize the weights using stochastic gradient descent.

The above can be defined in the following steps:

1. At first, if this is the first GCN layer, the input is taken. If not, the previous layer's output is taken. It is also should be noted that each node is processed by the network because every node of the network has a feature vector.

Page:11

Author: Date: 0000-00-00 00:00:00

step one: ...

2. Then, by using a shared weight matrix, a linear transformation would be applied to each node.
3. The feature vectors of a node's incoming neighbors are transformed and added to the node's own feature vector.
4. At last, the activation function is applied to all nodes and features element-wise.

The project utilized a two-layer model for the smaller data with 4 hidden layers and 64 hidden channels to enhance the results. These values were meticulously selected to obtain the best possible outcome. Additionally, an early-stopping strategy was employed to prevent overfitting, similar to the neural network section, with 800 epochs. Furthermore, a node embedding size of 2 was chosen, which resulted in the best predicted results with a learning rate of 0.015 and weight decay of 0.000001. In the single-layer model, which was used for larger data, 64 hidden channels and a learning rate of 0.015 were used, similar to the two-layer model. The weight decay value of one-tenth of the single-layer model, which is equal to 0.0000001, was used to obtain the best results. In this model, due to the large size of the data, node embedding 8 was used. Also, an early-stopping strategy was used after 800 epochs, similar to the smaller data model.

The neighbor aggregation function in the GCN assigns equal weight to all neighbors, but not all neighbors are equally important. A problem would be arising in this way when neighbors have different weight. To avoid that, Graph Attention Networks (GAT) has been introduced. In this model a calculation of set of weights for each neighbor of a node will be used to determine attention. Every weights are between 0 and 1, and the summing of all weights is equal to one. During summing up the transformed feature vectors of a node's neighbors, the vectors are multiplied by the comparative weights. These weights would be calculated by applying a two layer MLP. The first layer is a Linear layer with $2 * n\_node\_features + 1$ input

Page:12

Author: Date: 0000-00-00 00:00:00

this study

this work

Author: Date: 0000-00-00 00:00:00

use same tone

features and hidden_channels output features. The second layer is also a Linear layer with hidden_channels input features and 1 output feature.

The GAT architecture can be described mathematically as follows

$$n_i^j = \sigma\left(\sum_{k \in n(i) \cup \{i\}} \alpha_{ik}. n_k^{j-1} \times W^j)\right),$$

$$\alpha_{ik} = \alpha_{softmax}\left(N_{ff}\left(\left[n_k^{j-1}; n_i^{j-1}\right]\right)\right)_k,$$

with $\sum_k \alpha_{ik} = 1$, and $\alpha_{ik} \in [0,1]$.

Just like GCN, n_i^j as the feature of node i in GCN layer j, σ the activation function, W^j the weight matrix of layer j and k as all direct neighbors. α_{ik} is the attention weight of node i for neighbor k. The sum up of the α_{ik} is equal to 1 for all of a given nodes neighbors, and as it was mentioned, each weight is between 0 and 1. The N_{ff} is a two layer MLP as defined above and the $[;]$ operator concatenates the input vector. Also the $\alpha_{softmax}$ would work as an activation function and would force all outputs to be between 0 and 1, and also normalize them so the sum up would be 1.

$$\alpha_{softmax}(x)_k = \frac{e^{x_k}}{\sum_l e^{x_l}}$$

The above can be defined as follows:

No Comments.

1. At first, if this is the first GAT layer, the input is taken. If not, the previous layer's output is taken. It is also should be noted that every node of the network has a feature vector.
2. After that, the attention weights are calculated for each neighbor of a node by a two layer MPL.
3. Then, by using a shared weight matrix, a linear transformation would be applied to each node.
4.

3.4 Real-world OD Matrix:

Our dataset originates from the Chicago region, which has a population of approximately 2.7 million. Chicago is divided into seventy-seven community areas. The OD matrix in our data consists of 387 columns and rows, representing travel flows throughout the Chicago region. Additionally, we prepared two more datasets, one from Sioux Falls and the other from Anaheim, to compare the differences between small and large datasets. The Sioux Falls OD matrix has 24 columns and rows, while the Anaheim OD matrix has 38 columns and rows.

3.5 Data Preparation:

To ensure standardized data analysis, we scaled all the data to range(0,1).

Metrics:

To compare the performance of different methods, we employed the following three metrics:

Page:14

Author: Date: 0000-00-00 00:00:00

!!!!!!!!!!!!!!!!!!!!!!!!

1. Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted values and the actual values. It provides an indication of the model's accuracy.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Where y_i and x_i are prediction and true value and n is the total number of data points.

2. Root Mean Squared Error (RMSE): The RMSE quantifies the average squared difference between the predicted values and the actual values. It is particularly useful for evaluating the magnitude of errors. The formula for calculating RMSE is as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2}{N}}$$

Where x_i is actual observations time series and $\hat{x}_i$ is estimated time series, N is the total number of data points.

3. R-squared (R2): R2 measures the proportion of the variance in the dependent variable that can be explained by the independent variables. It indicates how well the model fits the data.

$$R2 = 1 - \frac{SSR}{SST}$$

where SSR is the sum of squared residuals ($\sum_{i=1}^{n}(actual - predicted)^2$) and SST is the total sum of squares ($\sum_{i=1}^{n}(actual - mean)^2$).

For each approach, we averaged the results from the 10 samples to obtain the mean result. We then compared these mean results among the different approaches. Additionally, by plotting the actual values against the predicted values, we can gain further insights and make more informed conclusions about the different scenarios.

Page:15

Author: Date: 0000-00-00 00:00:00

use table

delet extra info

these are trivial