Week 8 Tutorial: Traffic Prediction using Machine Learning

Xuehao Zhai and Fangce Guo

8th March 2024

Module: CIVE70110 - Intelligent and Autonomous Transport
Room: SKEM 208 - Munro Computer Room

Introduction

Problem Description Short-term Prediction is the process of estimating the anticipated traffic conditions in the short-term future given historical and current traffic information. Making short-term traffic prediction accurate and robust is one of the key components for building Intelligent Transport System (ITS) applications, which can assist transportation network managers in devising advanced strategies for forecasting and mitigating network issues, thereby reducing network congestion.

Mathematically, the goal of short-term prediction is to build a mapping function to regress the traffic variables in the next several time steps based on the historical traffic sequence. A time step is a basic time interval of recording, and input features (e.g., vehicle volume, average vehicle speed) contain the traffic variables that happened in each time interval. Assuming the length of the historical traffic sequence is n and the number of future steps to predict is k, then we can formulate a basic expression for the short-term prediction problem as follows:

$$\hat{X}_{t+1}, \cdots, \hat{X}_{t+k} = \mathcal{F}_\theta \left(X_{t-m+1}, \cdots, X_{t-1}, X_t \right) \tag{1}$$

where the parameters set of the prediction model are θ.

Sliding Window Mechanism The sliding window mechanism is a critical process for extracting a fixed length of sequence from historical traffic data to make short-term predictions. This mechanism involves moving a fixed-size window across the traffic data sequence to capture and use specific segments of data for predicting future traffic conditions.

Formally, given a historical traffic sequence $X_{t-m+1}, \ldots, X_{t-1}, X_t$, we define a sliding window of size n ($n \leq m$) that iteratively moves over the sequence one time step at a time. At each position, the window captures a sub-sequence $X_{t-n+1}, \ldots, X_t$, which is used as the input for the prediction model to estimate traffic conditions for the next k steps. This process can be mathematically represented as:

$$\text{For each window position } p, \text{ where } t - m + 1 \leq p \leq t - n + 1 : \tag{2}$$

$$\mathcal{W}_p = (X_p, X_{p+1}, \cdots, X_{p+n-1}) \tag{3}$$

$$\text{Predict } \hat{X}_{p+n}, \cdots, \hat{X}_{p+n+k-1} \text{ using } \mathcal{F}_\theta(\mathcal{W}_p) \tag{4}$$

Through this mechanism, the model captures and leverages the temporal dependencies and patterns within the traffic data, which are essential for making accurate and robust short-term predictions. By training the model to minimize the prediction loss across various window positions and corresponding future time steps, we can enhance the model's ability to generalize and perform well on unseen data.

Temporal Modelling For a basic version of temporal deep learning, this mapping function $\mathcal{F}$ contains (at least) a **temporal layer** and an **output layer**, as shown in Figure 1.

- The **temporal layer** processes the input sequence $X_t, X_{t-1}, \ldots, X_{t-n+1}$ into a hidden layer H with dimension of hidden size s ($H \in \mathbb{R}^s$).
- The **output layer** (typically a full-connected layer) is used to transform H into a series of predictions of the traffic variables in k of future time steps $\hat{X}_{t+1}, \ldots, \hat{X}_{t+k}$.

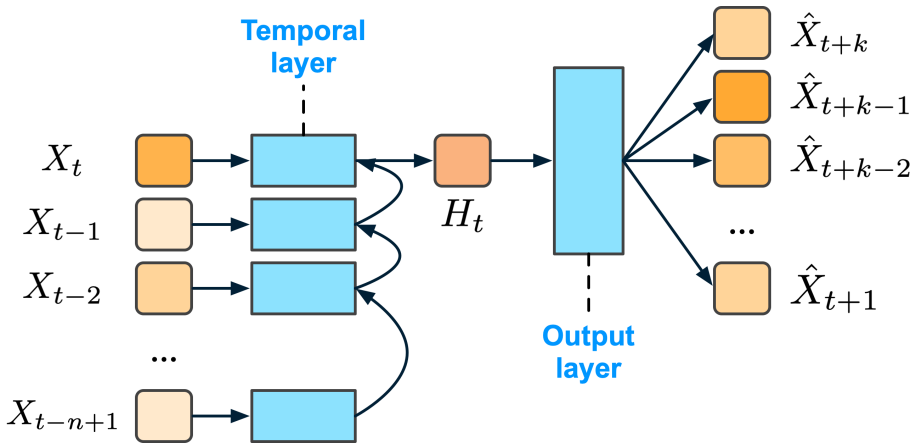

Figure 1: A simple framework of temporal models for multi-step prediction

We train this framework to minimise the loss by optimising the parameter set θ (including weights and bias in each layer):

$$\theta = \underset{\hat{\theta}}{\text{argmin}}\ \text{loss}(X, \hat{X}) \tag{5}$$

Tutorial Tasks

The data and provided python script is available here.

Tasks

Task 1: Exploring Sliding Windows

1. Begin by examining the structure and dimensions of the raw data contained within `data.csv`.

2. Run the `preprocessing.py` script. Adjust the control parameters to save preprocessed files in the `.npz` format for different prediction steps (`output_seq_len`): 1, 3, and 6.

3. Print and compare the dimensions of the subsets contained within these `.npz` files to observe how the dimensions change with varying `output_seq_len`.

Task 2: Evaluating the LSTM Model

1. Employ the LSTM model for the task of prediction.

2. Compare the performance outcomes for different prediction steps (`output_seq_len`) by examining accuracy tables and plotting line graphs.

3. Discuss the reasons behind the observed performance variations as the `output_seq_len` is altered.

Task 3: Comparing RNN, GRU, and LSTM Models

1. Compare the RNN, GRU, and LSTM models using the same prediction tasks.

2. Analyze and compare the results, including accuracy tables and line graph visualisations, for each model.