Lab Assignment 10 Report

Soham Panigrahi

Introduction:

In this assignment, I use the probability density for measurements to be accurate to predict correct flow values from what I believe to be reliable data. The lane measurement data that I am provided with consists of three fields flow, speed and occupancy for multiple lanes in different zones along with a probability density measure rating the reliability of the data. I aim to find the predicted flow measure by following three methods that predict a flow measure along with their confidence scores. Giving due weightage to the respective confidence scores I compute the predicted flow measure.

Approach: Analytics and statistical approaches

Linear Regression: Is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called simple linear regression. In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

Method 1: Nearby lanes+ LR

Here I apply Linear Regression to predict the correct flow measurement by taking all the other lanes excluding the lane to be predicted as the nearby lanes. Therefore, those lanes together comprise the features matrix and the linear regression sklearn fit commands are run to predict the flow values. The intuition behind this is that if flow of one lane is very high, then I expect the flow of another lane will also be high because vehicles tend to automatically balance the load of different lanes. The confidence value of each measurement is the mean of the probabilities of all the other lanes (nearby lanes) in the corresponding row of the same zone.

Predicted(flow) = a*Nearby_Measured(flow) + b

where (a,b) are LR model parameters

This part of the code takes less than 2 minutes to run.

Method 2: Nearby Timestamps + Weighted Sum

The predicted flow is taken to be the weighted average of the flow values in the preceding and succeeding nearby time intervals. They are weighted by the corresponding probability densities. In our implementation, I add 0.00001 to the denominator to avoid 0 divided 0 situation in the case where probability densities are 0. The confidence value of this measurement is taken as the minimum of the probability densities of the preceding and succeeding timestamp's flow values. A time interval is a nearby time interval if the magnitude of difference between its timestamp (say preceding) and the current timestamp is less than a threshold (in our code that is 300seconds). In case a flow value does not have say the preceding timestamp within the threshold then its corresponding weight is taken as 0 and the predicted flow comes entirely from the succeeding timestamp and vice-versa. If a flow doesn't have either preceding or succeeding timestamps within threshold, then its predicted flow value is the same as its current value and so is its confidence value. The intuition behind this is that measurements between consecutive time intervals are usually similar. Thus, I can predict the flow measurement by average of preceding and following time intervals, as given by: Predicted(flow) = w1*Preceding_Measured(flow) + w2*Following_Measured(flow) where (w1,w2) are weights between (0,1) such that w1+w2=1. I can calculate the (w1,w2) as: w1 = c1/(c1+c2), w2 = 1-w1 where c1 is probability density of preceding measurement, similar for c2: c1 = Prob_Density(Preceding(flow, speed, occupancy))

Finally, the confidence of this prediction can be estimated as: Confidence(flow) = min(c1,c2). This part of the code takes under 3 minutes to run. Method 3: Keeping the measurement unchanged The intuition behind keeping the measurements unchanged is that most of the measurements are correct, so keeping all flow measurements unchanged might not lead to a result that is too bad. In this case, I simply predict a correct flow value with the measured flow value, as:

Predicted(flow) = Measured(flow)

The confidence of this prediction can be estimated by probability density of this point:

Confidence(flow) = Prob_Density((flow, speed, occupancy))

The part of the code takes under 1 minute to run.

<u>Merging Multiple Predictions:</u>

The predicted flow values along with their confidence values generated from the above three methods are merged to generate the merged predicted flow value. The formula is:

Merged(flow) = w1*Predicted_1(flow) + … + w3*Predicted_3(flow)

where (w1, w2, w3) are weights between (0,1) such that w1+w2+w3=1. The w1 can be defined as

(like w2 and w2):

w1 = c1/(c1+c2+c3)

where c1 is confidence of the flow prediction given by method 1, c2 is the confidence of the flow prediction given by method 2 and so on. This part of the code takes under 3 minutes to run. Therefore, overall the code takes under 10 minutes to run.

Salient implementation features:

1. The running time is improved by generating shifted column (shifted by one row) of timestamps for method 2. This way expensive for loop can be avoided and calculations can be performed using column operations.

2. To handle empty strings and/or other nan values, for loop is again avoided and data frame is converted to numpy matrix to achieve the results using matrix operations.

3. Missing measurements in the input file are outputted as empty strings.

Analysis & Key Evaluation (Self Evaluation):

The running time of the code is under 9 minutes. I get results for predicted flow values which have considerably good confidence values associated with them. I observe that the predicted flow values are pretty close to the original flow values. The negative values in the flow measurements mostly get replaced with positive predicted values.

Summary (Improvement of the final results):

This is a prediction task where I need to predict the correct flow measurements. For this I are using 3 methods and trying to come up with the merged flow value. Method 1 uses linear regression and using a combination of all the other lanes as nearby lanes as the features. Method 2 uses values flow values from the preceding and succeeding timestamps. This is only when those time intervals are within 300 seconds. The third method keeps the flow values intact. I create the merged flow values using the confidence values generated by the three methods and output it as text files. I see that the final predicted values are close to the original values and have a good confidence values associated with them.

Challenges:

- Improving efficiency of the loops that predict the flow values.
- Merging of the data frames that were separated to regenerate the entire dataset and verifying that the data is behaving roughly as expected. This is especially difficult because the size of the data and the increased possibility of overlooked anomalies from cropping up.
- Handling the empty strings and NaN values that are present for few of the zones.

Intuitions:

By looking at the data:

- Nearby lanes provide a good measure for flow prediction on any given time because vehicles often tend to automatically balance off the loads into different lanes as per congestion.
- The flow measurements in time intervals that are sufficiently close to one another are nearly the same. Hence I can compute a predicted flow measurement from averaging the preceding and succeeding flow metrics provided they are within a threshold time gap.
- I consider that the measurements made for most of the time could be the accurate measurements and to bring this into account of flow prediction I add a third predicted flow metric that is simply the flow measurement that is made for the corresponding lane.

Implementation steps:

- The predicted flow as per method one is computed by the following equation: [Predicted(flow) = a*Nearby_Measured(flow) + b] where a and b are linear regression model parameters and the confidence score of this equation is the mean of the probability density metrics of all the nearby lane.
- The predicted flow metric computed from the second method is given by the following equation: [Predicted(flow) = w1*Preceding_Measured(flow) + w2*Following_Measured(flow)] Here w1 is c1/(c1+c2) and w2 is 1-w1, c1 is the probability density metric of the previous timestamp and c2 is the probability density metric of the succeeding timestamp. The confidence score for the computed predicted flow metric is taken to the minimum of c1 and c2 taking the worst case into consideration.
- To improve the efficiency of computing the predicted flow I break down the dataframe in terms of the continuity of the timestamps in the form of continuous timestamps with respect to previous

timestamp, next timestamp and discontinuous timestamps. I consider the preceding and succeeding timestamps to be continuous only when the time difference value computed is within 300 seconds.

- Next I create two columns that shift the flow values one up and the next one down to have the flow values of the previous and next timestamps and hence the predicted flow is computed row wise.
- The predicted flow metric from the third method is simply the flow metric measured for the given lane irrespective of the fact that it is erroneous or not supported by a confidence score that is computed for it from the corresponding prob.tsv
- I merge the three predicted flow metrics by giving due weightage to the confidence scores that are assumed for them. I use the flowing formula: [Merged(flow) = w1*Predicted_1(flow) + … + w3*Predicted_3(flow)] Here w1= c1/(c1+c2+c3), w2= c2/(c1+c2+c3) and w3= c3/(c1+c2+c3).
- The merged flow metric is added on to the data frame
- The table is now printed into a text file name by its zone_id.flow (zone_id.flow.txt)