Misbehavior Detection in VANETs

Manish Dash, 140101087
Rodney Stephen Rodrigues, 174101036
M. Krishnananda Singh, 174101005
Longkiri Bey, 140101035

Motivation

- As AI advances, technologies like “smart cars” are a reality.
- Importance of VANETs has increased to make road travel safer and comfortable.
- Security of such networks are critical to avoid loss of life and property.
- VANETs are inherently data centric. It is critical to make sure the nodes are communicating correct data.

Misbehavior in VANETs

- VANETs depend on nodes (vehicular or infrastructural) for its data.
- Nodes must be *trustworthy* and must *coordinate*.
- Any node that is not generating data or producing false data is said to **misbehave**.
- Misbehaving nodes can cripple the network and can cause performance degradation.
- In the worst case, misbehaving nodes may lead to traffic accidents and possible loss of life.
- It is, therefore, essential to detect and correct misbehaving nodes.

Approach

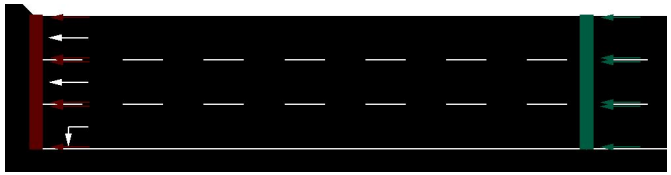
Our approach can be divided into two parts:

- **Data generation**
  - Absence of a good dataset for misbehavior in VANETs.
  - Simulation of Urban Mobility(SUMO) was used to simulate VANET and create artificial misbehaviors.
- **Prediction Model**
  - Generated data in form of a Time-Series data sequence.
  - Prediction involved learning non-linear relationship between the nodes.

Data Generation

Misbehaving Nodes

- Chose infrastructural nodes as they measure the flow statistics of the vehicles in the road network.
- Correct working is critical for efficient performance of the VANET.
- Used the entry-exit detectors(E3) in sumo.
- E3 detectors produce aggregate statistics of the road segment.



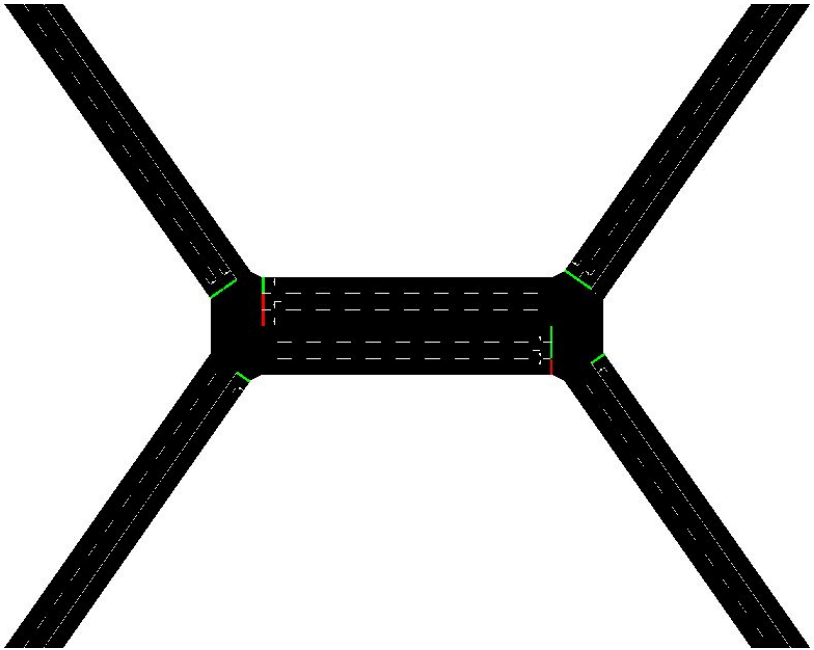
E3 detector:e3_0_1 Parameter

Name	Value	Dynamic
vehicles within [#]	18	✓
mean speed [m/s]	4.73	✓
haltings [#]	11	✓

Data Generation

Road network

- A main connecting road with four “arms”.
- Placed detectors on the five road segments.
- Aim is to predict the data generated in the middle road on the basis of data generated by the neighboring nodes.



Modelling Misbehavior

- SUMO doesn’t allow to simulate misbehaviors
- Misbehavior modelled as a prediction problem
- Deviation of reported value (H_R) from estimated value (H_E):

$$D = |H_R - H_E|$$

- If the deviation is beyond a certain threshold then misbehavior.
- If model can predict H_E accurately then misbehaviors can be detected too.

Analysis of Models

- Simulated data is a time-series sequence. Prediction of value X_t depends on the past values X_{t-1}, X_{t-2}, …
- Training data formatted as predicting current *meanHaltingPeriod* on the basis of observations of past 5 time-stamps.
- The loss function for the prediction model is mean square error (MSE): $MSE = avg\ \{\ |\ label - prediction\ |^2\ \}$
- Models divided into three classes: Linear, Non-linear and DNNs

Comparison of Linear models

Four linear models compared:

- Linear regression: models linear relationship between the features *X* and label *y*
- Lasso regression : L1 regularization on coeff. with Linear regression.
- Ridge regression : L2 regularization on coeff. with Linear regression.
- ElasticNet regression : combines both L1 and L2 regularization on coeff. of linear regression

Comparison of Linear models

TABLE I. COMPARISON OF LINEAR MODELS

Linear model	Metrics		
	R^2 score	*MSE*	*Time (s)*
Linear regressor	0.288	4846.98	0.17
Lasso	0.425	4006.93	3.5
Ridge	0.466	3721.82	0.05
Elastic Net	0.418	4058.76	4.6

Comparison of Non-linear models

- Non-linear models expected to learn the relationship between the nodes' readings.
- Analyzed many popular statistical and machine learning non-linear models.
- Observed a significant gain in accuracy over the linear models

Decision Trees

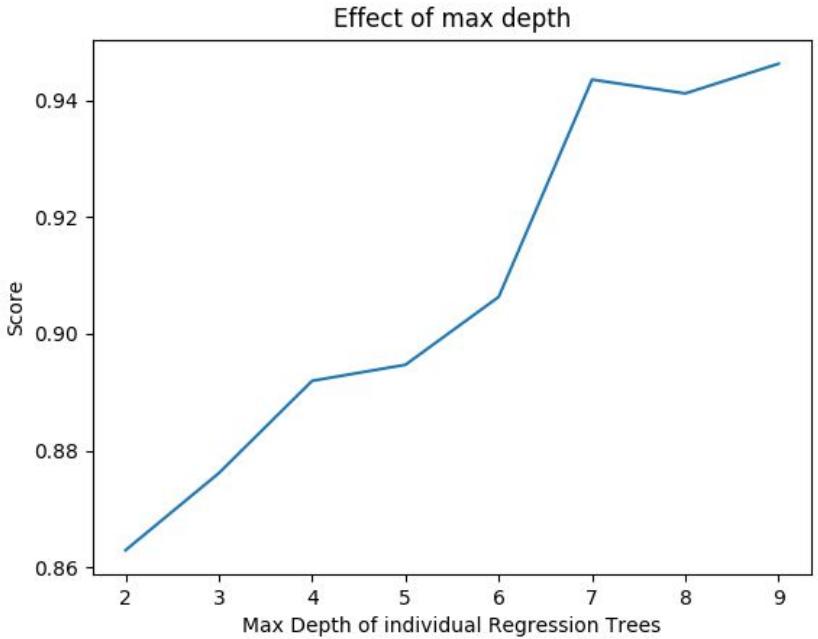
- Popularly known as Classification and Regression Trees (CART).
- Rule-based model which partitions the training data on basis of value of a single feature at every node.
- Able to learn non-linear relationships by the combination of rules from root to leaf.
- Extend using ensemble techniques: *bagging* and *boosting.*

Boosting

- Model learns many *weak learners*, the main *strong learner* learning from the weak learners' mistakes.
- Adaboost DTs: modifies the data distribution by increasing the weights of errored data, and reducing those of correctly predicted data.
- Gradient Boosting: weak learners trained on the residual errors of the past learners.
- Due to many learners being trained, significantly slower than simple DTs. But much more effective.

Boosting

- Performance of Gradient Boosting regressor as the max allowed depth of individual learners in increased.
- Shows that as learners become more complex, they individually become fully-grown DTs, and the model captures even more trends.

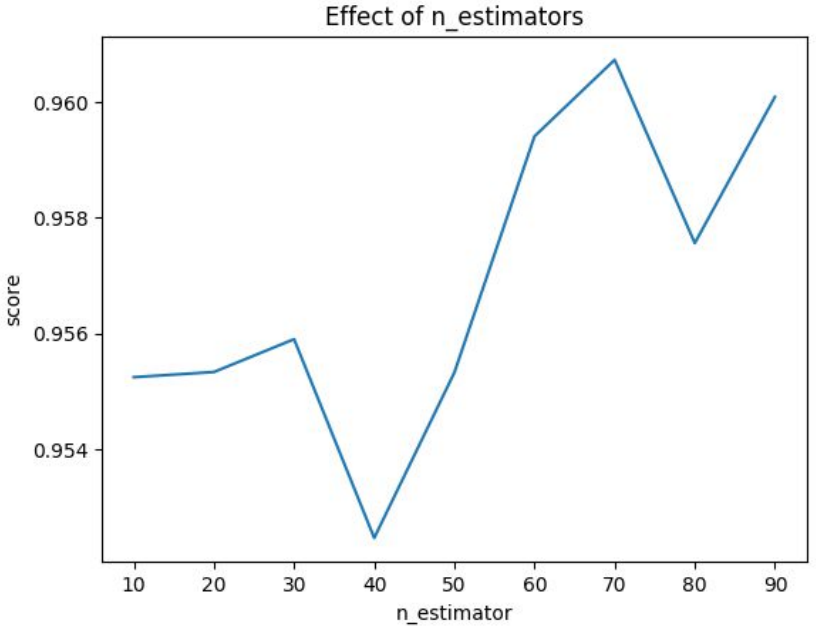


Bagging

- Instead of learning many learners, models are trained on different subsets of the training data in parallel.
- Prediction is the average value of all the models.
- Idea is to be able to capture the small trends in the data, evident in the smaller subsets of data.
- Performs much better than simple DTs, and even better than the boosting algorithms.

Bagging

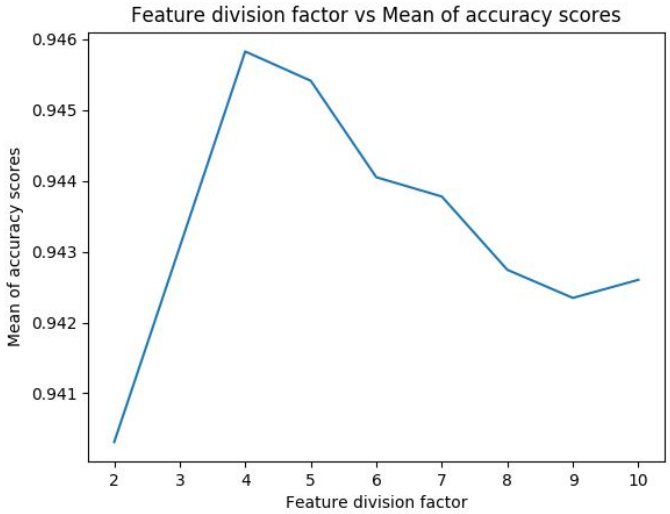
- A plot of the number of estimators versus the score for the Bagging DTs
- Shows that a large number of estimators do not necessarily mean better results.

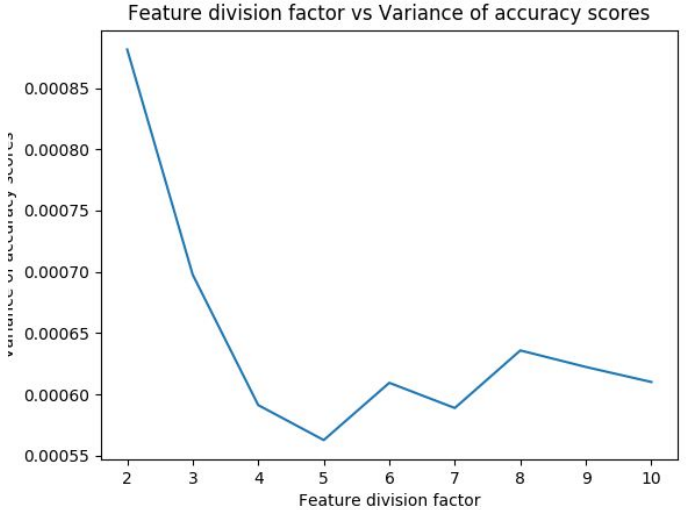


Random Forests

- Random forests are a popular solution for large classification and regression datasets.
- Similar to Gradient Boosting, but Random forests train fully grown DTs.
- We trained a RF with 20 DTs. The training time is large compared to a single DT.
- Performed the best among all the non-linear models.

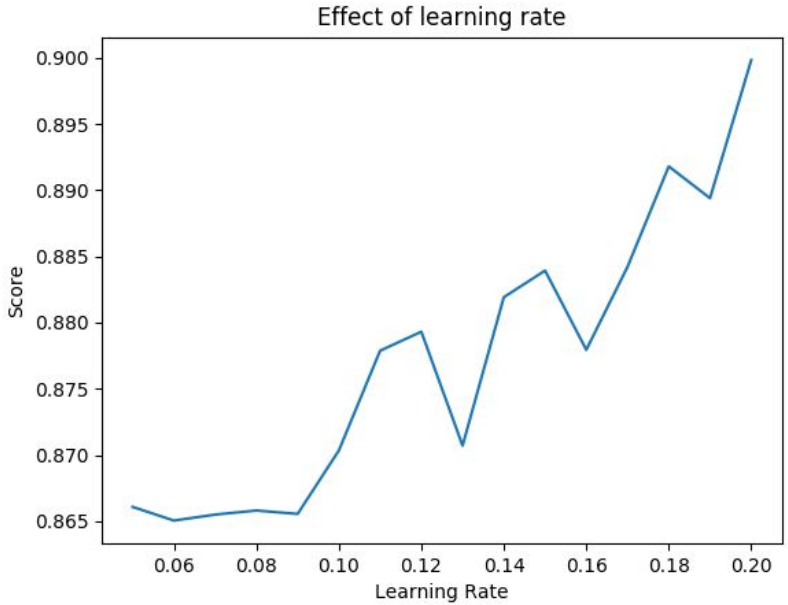
Random Forests





XGBoost

- Extreme Gradient Descent (XGBoost) is a well tuned version of Gradient Boosting algorithm.
- Performs similar to the Gradient Boosting regressor.



Comparison of Non-linear models

TABLE II. COMPARISON OF NON-LINEAR MODELS

Non-linear model	Metrics		
	R^2 *score*	*MSE*	*Time (s)*
Decision Tree	0.755	1709.67	0.32
Adaboost Decision Tree	0.808	1358.88	3.67
Bagged Decision Tree	0.882	821.91	5.05
Gradient Boosting Regressor	0.829	1186.36	11.294
KNN	0.801	1384.90	0.518
Random Forests	0.890	766.0	6.515

Deep Neural Networks

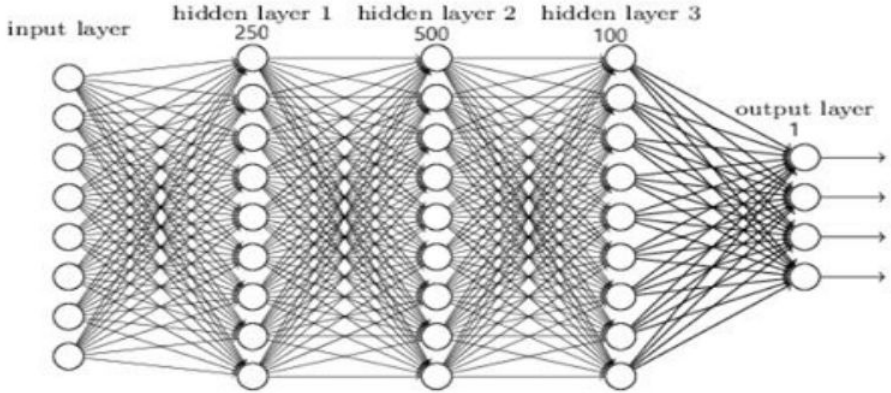
- Recent achievements of DNNs in various fields merit experiments in our problem.
- Two expectations:
  - Online learning
  - Remember past sequences
- We trained two different models with the two goals in mind.

Online MLP

- Multi-layer perceptron models shown to learn non-linear relationships quite effectively.
- Our MLP model capable of online learning.
- First the model is pre-trained on a batch of older data.
- Then the model is deployed on the node. Following “learning never stops” the model can have a feedback for every correct prediction.

Online MLP

- Trained for 500 epochs, with Adam optimizer at learning rate 0.001
- Performed at par with average non-linear model
- The network light enough for online learning.

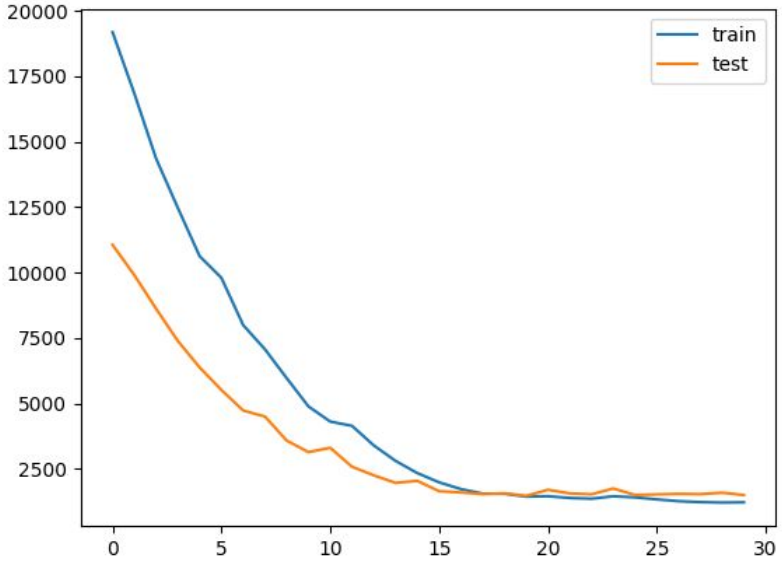


Long Short Term Memory Networks

- RNNs very capable of learning and remembering trends in time-series data.
- LSTMs have been shown to be very effective in time-series forecast.
- Our LSTM model had two LSTM layers with 50 and 100 cells each. The last layer was a fully-connected layer.
- Dropout of 50% introduced for regularization. Loss function MSE was optimised using Adam at learning rate 0.0001

Long Short Term Memory Networks

- LSTMs perform almost same as the Random forest.
- This shows that LSTMs were capable of learning the trends in periodic data.



Comparison of Deep Learning Models

TABLE III. COMPARISON OF DNNS

DNN	Metrics		
	R^2 score	*MSE*	*Time (s)*
online-MLP	0.815	1400.51	119.2
LSTM	0.885	801.78	213.6

Future Work

- The problem can be extended to include the dynamic vehicular nodes. The changing neighbors will be a challenge.
- DNN models can be modified to learn that the net traffic flow in a junction is constant.
- There can be hybrid models: two non-linear models or a non-linear and a DNN can be stacked to increase performance.