Prediction of Pedestrian Trajectories Final Report

Mingchen Li (limc), Yiyang Li (yiyang7), Gendong Zhang (zgdsh29)

December 15, 2017

1 Introduction

As the industry of automotive vehicles growing rapidly, the ability of those vehicles to predict trajectories of pedestrians becomes more crucial than ever. Any autonomous vehicle navigating such a scene should be able to foresee the future positions of pedestrians and accordingly adjust its path to avoid collisions [1]. As stated in [1], this trajectory prediction problem can be viewed as a sequence generation task, where we are interested in predicting the future trajectory of people based on their past positions. In this project, we present comparisons among the performances of different machine learning models.

The input to our algorithm is arbitrary number of people's previous positions in x-y coordinates and the output is the people's next position. The methods been applied are K-Nearest Neighbors (KNN) combined with linear regression, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). The latter two are different cell types of Recurrent Neural Network (RNN).

2 Related Works

The problem of predicting pedestrian trajectories has been studied by many research groups around the world for decades and many models have been proposed. Some groups used traditional machine learning algorithms by fitting data into different models, such as Gaussian mixture model [2], Mixed Markov-chain model [3], and Gaussian process regression [4]. Other groups considered the social forces between pedestrians. Helbing *et. al.* [5] modeled pedestrian motions with attractive and repulsive forces. However, Helbing *et. al.* [5] used hand-crafted functions that might not be applicable in complex settings.

The most advanced trajectory prediction method that considered social force is the "Social-LSTM" proposed by Alahi [1]. We also applied LSTM in the project, and we focused on predicting pedestrian's single next position based on his/her previous positions. Besides trajectory prediction tasks, RNN has been successfully applied to other sequence prediction tasks: language modeling [6], human dynamics [7], etc. However, it may not be advantageous to apply RNN to different problems such as image classification where Convolutional Neural Network (CNN) would achieve better results [8].

3 Methods

The methods we applied are linear regression based on KNN, LSTM, and GRU. Linear regression is used as a baseline for this project. Since the distribution of the pedestrians could be very much random, it might be hard for the entire dataset to be separated by a single hyperplane. KNN provides a solution to this problem, which is a non-parametric method used for classification and regression. The algorithm stores the entire dataset and predicts the numerical targets based on a similarity measurement [9]. In this project, we picked K nearest neighbors for each person and calculated mean distance between that person and its K neighbors using Euclidean equation (shown in equation (1)) to perform regression.

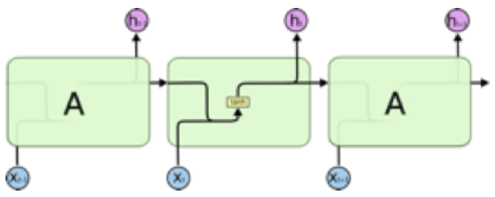
$$d(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{1}$$

RNN is a type of advanced artificial neural network with directed cycles in memory (shown in Figure 1(a)

). RNNs are the best choice for this project given that they can make use of sequential information. This advantage comes from their special “cell state” on the side of regular “output state” of regular neural networks. They perform the same task for every element of a sequence, with the output being depended on the previous computations [10].

However, a regular RNN cell may perform poorly on predictions that requires long term memory dependencies, because input data is changing from cell to cell. The LSTM cell is a special kind of RNN cell that can handle long term memory, because input data runs straight down the entire chain, with only minor linear interactions, as indicated in Figure 1(b). More specifically, each LSTM cell contains three “gates” where sigmoid functions are applied to determine how much of the previous information will be passed onto the next states. “Forget gate” decides what information will be thrown away from the previous cell state. “Input gate” decides what new information will be stored in the new cell state. Finally, the “output gate” decides what information will be output based on filtered cell state data. [11]

The GRU cell is a variation of LSTM, which combines “input gate” and “forget gate” inside an LSTM cell into a single gate to make it run faster.


(a) Tradition RNN

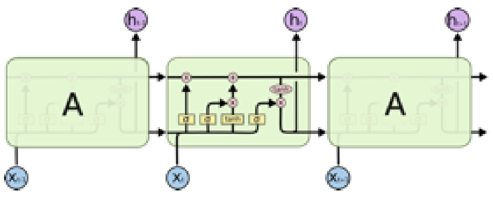

(b) LSTM RNN

Figure 1: Two Types of RNN [11]

Since this is not a classification task, we cannot use accuracy as the evaluation matric for the model performances. As the result, we chose to use cost function to evaluate the model. The cost function is mean square error (MSE), defined in equation (2):

$$L(\hat{p}, p) = (\hat{p} - p)^2 = \frac{1}{m}\sum_{i=1}^{m} \|\Delta p^i\|_2^2, where \Delta p^i = \begin{pmatrix} x^i \\ y^i \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix}, \tag{2}$$

where $\begin{pmatrix} x^i \\ y^i \end{pmatrix}$ is the predicted next position of the person, and $\begin{pmatrix} x \\ y \end{pmatrix}$ is actual value of that position. We calculated the norm of the difference between two vectors to evaluate how close prediction points are to the corresponding points.

4 Data Processing

Our dataset comes from the publicly available resources of UCY [12]. In order to use our dataset fully and efficiently, we did some preprocessing of the data. We used Numpy to tailor our data into a more suitable format so that our training methods can manipulate our data smoothly. Table 1 shows a glance of our dataset. As can be seen that Frame represents the time unit at which the pedestrian’s coordinates was captured, the Pedestrian label is used to distinguish between pedestrians, and Y and X are 2-D coordinates. We found that the number of coordinate pairs (X, Y) captured for every pedestrian is not the same. So as to train our models properly, we set a minimum number of sequential steps, which is the number of feature of the input data, and the pedestrians whose number of data was less than this number plus one (plus one for the target value) were deleted. In addition, before we trained our models, we split the data set into training set, development set and testing set in the ratio of 6:2:2, which ended up to be 1449 pedestrians in training data, 483 in development and 483 in testing.

Table 1: Partial dataset

Frame	0	10	20	30	40	50	60	70
Pedestrian Label	1	1	1	1	1	1	1	1
Y	0.5607	0.5551	0.5495	0.5438	0.5373	0.5286	0.5199	0.51128
X	0.59722	0.622	0.6467	0.6715	0.6971	0.7244	0.7518	0.7792

5 Feature Selection and Experiments

In order to choose appropriate features for the input data, we plotted the feature selection curves for number of steps versus development cost on LSTM model, as shown in Figure 2. According to the curve, there was an underfitting problem when the number was 4 and an overfitting problem when the number was 6, so we chose 5 as the final number. For number of cells of RNN models and the value of K of KNN,

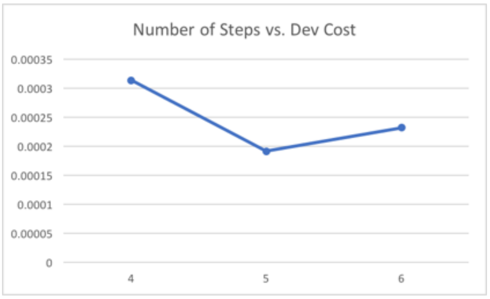
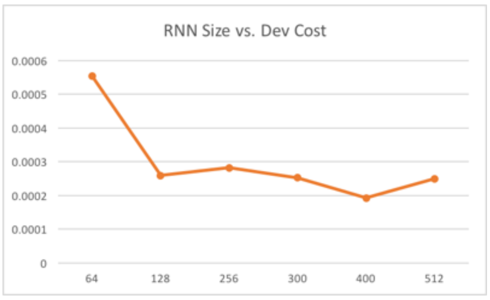
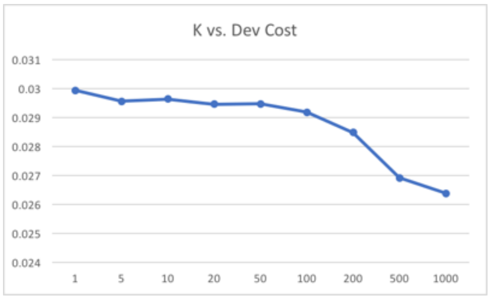


Figure 2: Feature selection curve for number of steps

we ran similar tests on these two models and plotted the corresponding feature versus development cost plots, as shown in Figure 3(a) and 3(b). Due to the same underfitting and overfitting reasons, we chose 400 for the number of cell of RNN models. We also found 1000 for K value of KNN performed well on development set, and therefore, 1000 was used. Note that this 1000 is not the number of pedestrians but the number of coordinates pairs.



(a) RNN size



(b) K

Figure 3: Feature selection curves

Before gathering results from actual experiments, we performed hyperparameter selections by running different combinations of parameters. Table 2 shows the average cross-validation cost on combinations between learning rates and batch sizes, where each of the value is averaged from 10 tests. The first row contains different learning rates and the first column contains different batch sizes. From the table, we decided to use 0.0005 as the learning rate and 30 as the batch size.

Table 2: Cross-validation results on learning rates and batches sizes

	Learnig Rate									
Batch Size	0.02		0.005		0.002		0.0005		0.0002	
	Train Loss 1*e-4	Test Loss 1*e-4	Train Loss 1*e-4	Test Loss 1*e-4	Train Loss 1*e-4	Test Loss 1*e-4	Train Loss 1*e-4	Test Loss 1*e-4	Train Loss 1*e-4	Test Loss 1*e-4
15	4.47	3.97	2.30	1.96	1.85	1.46	1.86	2.11	2.76	2.88
30	9.02	12.3	3.48	3.43	5.36	5.09	1.65	1.82	1.95	2.29
110	7.47	9.15	4.50	4.49	3.59	3.79	2.61	3.44	3.66	4.00

6 Results

After determining the hyper parameters, combined with number of sequential steps as 5, RNN size as 400 and the value of K as 1000, we performed the predictions on the testing data. Table 3 shows the testing results for the three models. The first two rows contain the ground truth x and y coordinates, and the remaining rows contain the predicted coordinates from each model.

Table 3: Testing results

Real							Average Cost
X	0.84	0.77	0.75	0.92	0.84	0.77	
Y	0.11	0.17	0.19	0.19	0.22	0.24	
LSTM							1.1917e-4
X	0.85	0.77	0.74	0.93	0.84	0.77	
Y	0.10	0.16	0.19	0.19	0.21	0.24	
GRU							5.226e-4
X	0.86	0.77	0.73	0.93	0.84	0.76	
Y	0.09	0.15	0.19	0.18	0.20	0.23	
KNN							3.031e-2
X	0.58	0.54	0.51	0.64	0.59	0.54	
Y	0.38	0.36	0.35	0.44	0.41	0.39	

According to the results, the LSTM and GRU outperformed KNN by approximately two orders of magnitude. Both of the RNN models have the cost in the order of 1e-4, but the KNN model has the cost in the order of 1e-2. Since the cost function is the square of the error, the actual error between RNN and KNN models should be nearly ten times different. In addition to what shows in Table 3, the runtimes of LSTM and GRU are a bit different. The LSTM needed 50 epochs in average to converge, while GRU only needed 15 epochs in average to converge. In order to visualize the model prediction, we plotted the results of LSTM model, as shown in Figure 4. In the figure, the circles are the predicted coordinates, the crosses are the real coordinates, and the tails are the sequential steps of the corresponding pedestrians.

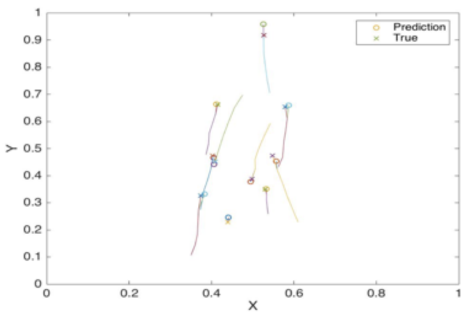
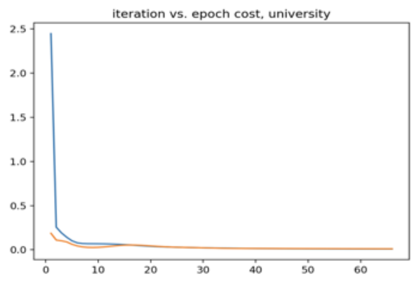


Figure 4: Prediction Results of LSTM model

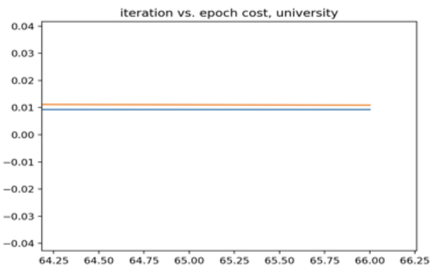
7 Discussion

According to the testing results, the performance of the models in descending sequence is LSTM, GRU, KNN. GRU is faster than LSTM in terms of convergence speed. These result are within expectation: KNN does not take into account of time sequentiality of the data, and GRU combines two gates of LSTM into one gate, which sacrifices the performance for speed.

Besides the comparisons among performances, we also plotted the learning curve for training set and development set. An interesting thing to notice is that sometimes the cost of development set is lower than that of training set. This happens due to the fact that unlike the accuracy, the cost is accumulative through batch iterations for training set. Since the cost of the first few batches can be extremely higher than latter ones, the first epoch of the training set may have a very large cost. However, in development set, the parameters (w and b) of the first epoch are already trained from the batches of the training set, they will be likely to produce less cost on the development set. This procedure happens in every epoch, which means the development set always uses better trained parameters than the training set. Figure 5(a) shows the learning curve of the LSTM model where cost of the development set is less than that of training set. Figure 5(b) is a zoom-in window.



(a) Full window



(b) Zoom in window

Figure 5: Learning curve of LSTM

8 Conclusion

LSTM, GRU and KNN combined with linear regression have been tested. As the results, LSTM and GRU out performed KNN significantly. The first two had cost around 1e-4 and the last one was around 1e-2. For the comparison of the LSTM and GRU, the LSTM had a better performance but longer runtime, and the GRU had a shorter runtime but worse performance.

9 Future Work

In this project, we only used the first 5 steps as to predict the 6th step. In the future, we could use arbitrary number of consecutive points to predict the next few points, implementing sequential-to-sequential prediction. We could also use embedding data processing method to increase the accuracy further. In addition, We could use other version of LSTM, such as social LSTM, besides the vanilla version of LSTM to train our data set.

10 Contributions

Mingchen Li: Wrote programs on preprocessing of datasets, read data.py, and wrote programs on building of LSTM model and training of the data, train test.py; wrote Results, Discussion and partial Features and Experiments sections of the report.

Gendong Zhang: Wrote programs on experiment examination, visualization of the predictions and targets, result.m; Helped debug and improve the read data.py and train test.py; Wrote Data Processing, Conclusion, Future Work and partial Features and Experiments sections of the report.

Yiyang Li: Researched KNN algorithm; Implemented and tested sample KNN method; Helped write programs on preprocessing of datasets; Helped debug and improve the read data.py and train test.py; Wrote Introduction, Related Works and Methods sections of the report.

11 Reference

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces." In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on,* IEEE.

[2] J. Wiest, M. Hoffken, U. Kresel and K. Dietmayer, "Probabilistic trajectory prediction with Gaussian mixture models," in *Proc. IEEE Intelligent Vehicles Symposium (IV),* pp. 141-146, 3-7 June 2012.

[3] Akinori Asahara, Kishiko Maruyama , Akiko Sato , Kouichi Seto, "Pedestrian-movement prediction based on mixed Markov-chain model," *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* November 01-04, 2011, Chicago, Illinois

[4] D. Ellis, E. Sommerlade, and I. Ried, "Modelling pedestrian trajectory patterns with gaussian processes," in *IEEE 12th International Conference on Computer Vision (ICCV) Workshops,* 2009.

[5] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E,* 51(5):4282, 1995.

[6] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," In *Proceedings of the 31st International Conference on Machine Learning (ICML-14),* pages 1764–1772, 2014.

[7] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics."

[8] C. Cao, X. Liu, Y. Yang, Y. Yu, J.Wang, Z.Wang, Y. Huang, L.Wang, C. Huang, W. Xu, et al, "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks," *ICCV,* 2015.

[9] Machine Learning Mastery. (2016). K-Nearest Neighbors for Machine Learning. [online] Available at: machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/ [Accessed 15 Dec. 2017].

[10] WILDML. (2015). Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs. [online] Available at: www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/ [Accessed 15 Dec. 2017].

[11] Colah's blog. (2015). Understanding LSTM Networks. [online] Available at: colah.github.io/posts/2015-08-Understanding-LSTMs/ [Accessed 15 Dec. 2017].

[12] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," in *Computer Graphics Forum,* volume 26, pages 655–664. Wiley Online Library, 2007.