Pedestrian Trajectory Prediction Using Recurrent Neural Network

Team members: Abhishek Angadi, Ana Belen Barcenas, Azucena Morales, Emma Sun

Supervisor: Mohammadreza Soltani, Ph.D.

Abstract

Robots and autonomous vehicles' coexistence with human crowds aim to become ubiquitous in the years to come. Predicting human trajectories to help autonomous robots and vehicles to navigate in environments surrounded by pedestrians is of utmost importance. In this work, we compare multiple models to predict a one-time point and multiple time points ahead based on historical coordinates of pedestrians using publicly available datasets in crowded scenarios. Analyzing this type of scene is a challenging task. The Social LSTM that we used tries to capture and model the social behavior of pedestrians. We compare this model with other approaches, such as Kalman Filter and LSTM, which don't account for the social component.

1 Introduction

1.1 Motivation and importance of the problem

Human trajectory prediction has become important due to the increase in autonomous vehicles or robots navigating through human crowds. Vehicles have to foresee human movements to plan safe trajectories while another type of robot should be able to emulate human trajectories as well as plan efficiently its own paths. Advances in computer vision focused on object detection, object tracking, and multi-step forecasting have made possible to analyze sequences of coordinates to predict future pedestrian trajectories successfully.

Over the last decade, researchers have addressed this challenge by means of traditional dynamic models such as Kalman Filter (KF) where a pedestrian trajectory is predicted based on its own history. Most recent works on this domain have introduced dynamic social behavior, where besides its own past trajectory, the future destination and the environment also matter to adjust their path.

1.2 Summary

In this work, we summarized and compared some of the most common methodologies to predict pedestrian trajectories. We focused our analysis on recursive multi-step forecasting to leverage each data point available and achieve greater performance. We used Kalman Filter, recurrent neural networks (RNN), and some extensions of these methodologies to model and predict human trajectories in crowds.

2 Related works

2.1 Pedestrian Trajectory Prediction Methodologies

The most straightforward methodology for pedestrian trajectory is to consider pedestrians as independent entities and then use the current state to predict the next. These methods include Particle filter, Kalman filter and Gaussian processes. While these approaches have achieved certain successes, they also have significant drawbacks such as ignoring human interaction and not learning enough trajectory cues from historical data [1]. Therefore, other methodologies that would account for influences among pedestrians are explored further.

Modeling pedestrian trajectories trying to capture the social behavior of the individuals in crowded scenarios has been done before. The model of social force uses potential functions to react to other pedestrians and objects [2]. Another work that was influenced by this approach predicts the point of closest approach and use it to make decisions, thus pedestrians move trying to minimize potential collisions [3]. An entirely different take on predicting the trajectories was worked upon wherein the maximum entropy learning method was applied based on features that capture relevant aspects of the trajectories to determine the probability distribution that underlies human navigation behavior[4].

Follow-up work has been done to improve the social LSTM by taking advantage of groups of people that usually tend to move coherently. The idea behind is the same as the social LSTM, except that the social hidden-state tensor only contains information about one pedestrian in each group. Clustering pedestrians were done with a coherent filtering algorithm using the ground truth coordinates [5].

Another approach to predict the human trajectory is by incorporating the scene information within static crowded scenes. The method essentially consists of two-coupled LSTM networks, pedestrian movement LSTMs (one per target) and the corresponding Scene-LSTMs (one per grid-cell), trained simultaneously to predict the next movements. A scene data filter holds important non-linear movement information, allowing to select the relevant information from the grid cell's memory relative to a target's state. The results show that the method reduces the location displacement errors in about 80 % reduction compared to social interaction methods. [6]

2.2 Multi-step Time Series Forecasting strategies

Time series forecasting typically addresses situations where only the very next value of the series has to be predicted. This is called a single-step forecast. However, there are situations when single-step forecasts are not enough, and it is necessary to predict multiple data points in the future. The following strategies are the most widely used to address multi-step time series forecasting situations.

The direct strategy consists of developing a separate model to predict each desired time step in the future. An advantage of this strategy is that it does not accumulate prediction errors since it does not use any predicted value as input to predict the next time step. However, creating a separate model for each prediction is computational inefficient since there are many models to train.

The recursive strategy consists of forecasting one step at a time and then employing the same model and the predicted value as input for the next prediction. This method tends to suffer from low performance especially when the forecasting horizon exceeds the size of the real-time series data. This happens because, at some point, all the inputs will be predicted values instead of actual observations.

The DirRec strategy combines both direct and recursive methodologies by creating a separate model to predict each step ahead and updating the input variables with each prediction obtained. This results in enlarging the input size by one for each additional prediction on the horizon.

Unlike the three previous strategies that involve multiple inputs and a single output, the multiple output methodology predicts the entire forecast sequence all at once. the disadvantage of preserving the dependencies is that we constraint all the future point predictions to be forecasted with the same model structure.

For the purpose of this work, we believe that the dependencies between the previous location and the next one have important predicting power. Thus, we will employ the recursive strategy to take advantage of all the real data points available as well as the predictions made. We possess enough computing power to avoid inefficiencies due to the multiple modeling and training necessary.

3 Models

3.1 Kalman Filter

Kalman filter is an iterative mathematical process that utilizes a series of equations on consecutive data inputs to quickly estimate the true value, position and velocity of the object being measured, when the measured values contain unpredicted or random error, uncertainty or variation. When a new data point is observed, we update the estimate iteratively. By always looking at the error between the estimate and the observation, Kalman filter allows the estimate to quickly settle down to a stable stage and estimates the trend of the data points. It is a traditional mathematical method that is used in trajectory prediction. Therefore, in our case of pedestrian trajectory prediction, we also include it as one of the methods that we would like to explore.

We used the multidimensional model of Kalman filter in our case because our dataset is two-dimensional comprised of x and y coordinates. The algorithm itself is best illustrated in the flow-chart below.

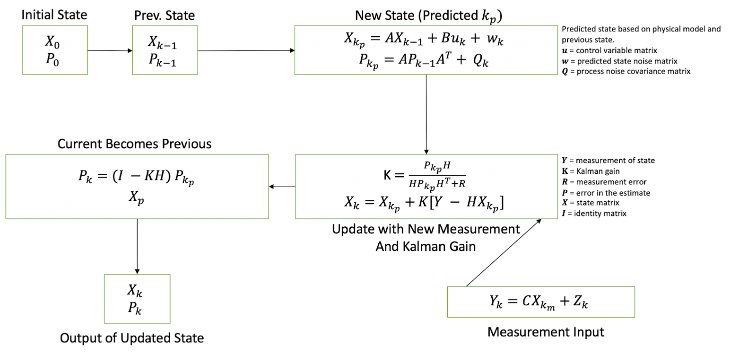
Initial State
X_0 P_0
Prev. State
X_{k-1} P_{k-1}
New State (Predicted k_p)
$X_{k_p} = AX_{k-1} + Bu_k + w_k$
$P_{k_p} = AP_{k-1}A^T + Q_k$
Predicted state based on physical model and previous state.
$\boldsymbol{u}$ = control variable matrix
$\boldsymbol{w}$ = predicted state noise matrix
$\boldsymbol{Q}$ = process noise covariance matrix
Current Becomes Previous
$P_k = (I - KH) P_{k_p}$
X_p
$K = \frac{P_{k_p}H}{HP_{k_p}H^T + R}$
$X_k = X_{k_p} + K[Y - HX_{k_p}]$
$\boldsymbol{Y}$ = measurement of state
$\mathbf{K}$ = Kalman gain
$\boldsymbol{R}$ = measurement error
$\boldsymbol{P}$ = error in the estimate
$\boldsymbol{X}$ = state matrix
$\boldsymbol{I}$ = identity matrix
Update with New Measurement And Kalman Gain
X_k P_k
Output of Updated State
$Y_k = CX_{k_m} + Z_k$
Measurement Input

Figure 1: Kalman Filter Scheme

We implemented the algorithm as below, experimenting on two things: one is to use time window of 12 observations to predict 1 step ahead and the other is to use the same window to predict 12 steps ahead. We calculated MAE and RMSE for both models as the baseline error used to compare with other methodologies that we tried subsequently.

3.2 LSTM

Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. RNN models might suffer from two problems: vanishing gradient and exploding gradient. LSTM models deal with these problems by introducing a memory unit, called the cell into the network.

All recurrent neural networks have the form of a chain of repeating modules of the neural networks. In standard RNNs, this repeating module will have a very simple structure, such as a single layer consisting a tanh activation fuction. LSTMs also have this chain like structure, but the repeating module has more complex structure. Instead of having a single neural network layer, there are four layers, interacting in a very special way.

In a simple way, LSTM networks have some internal contextual state cells that act as long-term or short-term memory cells. The output of the LSTM network is modulated by the state of these cells. This is a very important property when we need the prediction of the neural network to depend on the historical context of inputs, rather than only on the very last input. LSTM networks manage to keep contextual information of inputs by integrating a loop that allows information to flow from one step to the next. LSTM predictions are always conditioned by the past experience of the network's inputs. On the other hand, the more time passes, the less likely it becomes that the next output depends on a very old input. This time dependency distance itself is as well a contextual information to be learned. LSTM networks manage this by learning when to remember and when to forget, through the forget gate weights.

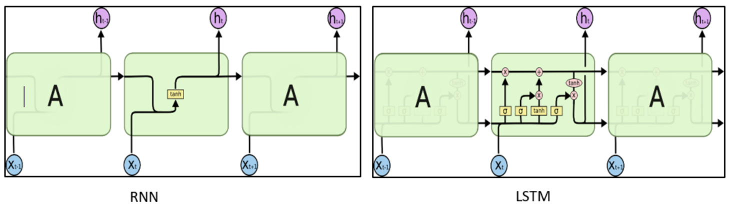

Figure 2: LSTM scheme

3.2.1 Contender Model - LSTM with Parameter Optimization using Genetic Algorithms

To compete with the social LSTM and the Kalman filter model for predicting multiple steps ahead of a pedestrian, we built a LSTM model as a challenger model and focused primarily on parameter optimization. The idea was that a basic LSTM is powerful enough to compete with more sophisticated models if the parameter optimization is performed using a genetic search algorithm technique. Genetic algorithms are inspired by the Darwinian process of Natural Selection, and they are used to generate solutions to optimization and search problems in computer science. Finding the best hyperparameters for a model is an optimization problem for which genetic programming can be used.

The first step was to vary a hyperparameter and chose the few values that best minimize the prediction error. The subsequent step was to vary another hyperparameter for those values and find the values for the 2nd hyperparameter that minimizes the prediction error. The process of selecting the values that minimize the error, fixing them and then varying another hyperparameter is continued across all hyperparameters. Apart from varying the hyperparameters, we also tried different multiple prediction techniques. The loss function used is a MSE loss function. The hyperparameters optimized are:

1. The window size of the input data
2. Batch size
3. Learning Rate
4. Train-Test data split
5. Time-point split – This split pertains to x number of steps considered to predict y number of steps
6. Hidden number
7. Hidden layers
8. Multiple prediction techniques

3.3 Social LSTM

Humans' trajectories depend on many factors, one of them is the interaction with other pedestrians and other objects. The ability of humans to predict others behavior in order to adapt their motion to avoid collisions and follow social conventions is not an easy task to implement in a model. Other works have proposed ways to model human-human interactions to improve predictions, but they fail to capture complex interactions or non-close interactions.

LSTM have been successful in predicting time series data and in NLP problems. In order to account for the human interaction, the paper proposes to include what they call a "social layer". This allows modeling the dynamic crowd interactions without the need for hand-crafted rules. In crowded places, this type of interaction becomes crucial in predicting human trajectories.

The goal is to observe a length sequence t from time 1 to time t, and predict the following sequence for a given length p, time t+1 to time t+p. Each sequence is conformed by x and y coordinates for each pedestrian. All the pedestrians' trajectories information is shared through social tensor. Thus, the model is able to capture both the individual patterns such as velocity and acceleration, as well as the potential interactions with other individuals.

Conceptually, the crux of social LSTM algorithm is to pass the hidden representation of the pedestrians in the same frame to the social pooling layer which produces a new hidden layer H. This new hidden layer H together with individuals' hidden representation at that time point will be passed into LSTM to generate prediction for the next time point. The social pooling is done by first separating pedestrians in the same time frame into neighborhoods based on their spatial distance. The hidden layers of the pedestrians in the same neighborhood are then summed together and then all neighborhoods will be aggregated together by a function.

The following formulae consist a more detailed explanation about how the social tensor is embedded in the LSTM:

A tensor is made with a grid size (N_o x N_o) for each pedestrian in the frame times the dimension of the hidden state D. The tensor is then multiplied by the hidden states of each pedestrian.

$$H_t^i(m,n,:) = \sum_{j \in N_i} 1_{mn}[x_t^j - x_t^i, y_t^j - y_t^j - y_t^i] h_{t-1}^j$$

where h_{t-1}^j is the LSTM hidden state of the jth person at $t-1$,
N_i is the set of neighbors,
$1[x,y]$ is the indicator function.

This social tensor is then passed through a linear layer with an output that equals to the embedding size. Then the tensor is concatenated with the input layer. The final tensor goes inside the cell of the LSTM. The final LSTM output gives a tensor of five numbers for each pedestrian, which is the mean and variance of each coordinate along with the correlation for each pedestrian.

This output is used to get a x and y coordinate sample coming from a normal bi-variate for each pedestrian in the frame.

$$(\hat{x}, \hat{y})_t^i \sim N(\mu_t^i, \sigma_t^i, \rho_t^i)$$

The loss function minimizes the negative log likelihood of bi-variate Gaussian distribution, i.e the log probability of the data given the parameters. Finally, to get the next point estimation the previously predicted estimation is used as the true coordinates.

3.4 Contribution of each member of the team

All of the team members were in charge of doing research about related works in the field, and understanding RNN and LSTM methodologies. Abhishek was in charge of understanding LSTM. He was also in charge of utilizing LSTM to predict one-step and multiple steps. The next part of his task was to optimize hyper parameters using genetic search algorithm technique. He also helped with the literature review.

Azucena was in charge of understanding the code that is used to implement the model. She preprocessed the data and change some of the parameters from the code to run it and compare the vanilla LSTM and social LSTM results with others. She also helped with the social LSTM, worked related and experimental results section.

Ana Belen was in charge of understanding the input and the output dataset employed in the Social LSTM code. She was also in charge of a literature review on (1) multi-step time series forecasting, (2) object detection and object tracking to extract coordinates from videos of pedestrians' trajectories. She also helped with the introduction and abstract sections in the white paper.

Emma was in charge of understanding Kalman Filter and utilizing it to predict pedestrian trajectory as a baseline model to be compared with other methodologies. Kalman filter is applied in both cases of one-time point and multiple time points ahead prediction.

4 Experimental results

The parameters used for social LSTM are the same as in the original paper. The embedding size is 64, the grid area is 4 x 4 m^2, the pooling window size is 32. The embedding layer has a RELU activation. RMS-prop is used as the optimizer with a learning rate of .003.

For all models we utilized 12-time points to predict the next one or the next 8 time windows. Therefore, our error terms are compatible. We thus compared this model with Kalman Filter and an LSTM for each pedestrian, a Vanilla LSTM and Social LSTM. [1]

We compared these four models using common publicly human trajectory datasets: Biwi Walking Pedestrians dataset from ETH, Crowds dataset from UCY, Stanford Aerial Pedestrian Dataset and Multiple Object Tracking Dataset. The datasets were split 70% for the training and 30% for the test. For our metric performance, we decided to use RMSE and MAE.

Table 1 contains the results for one prediction ahead and for multiple predictions for the four models.

	One-step prediction		Multiple prediction	
Model	**MAE**	**RMSE**	**MAE**	**RMSE**
Kalman Filter	1.50	0.39	2.41	0.80
LSTM (with optimized hyper-parameters)	0.34	0.46	1.19	1.90
Vanilla LSTM	.44	.93	.83	1.61
Social LSTM	.42	.89	.81	1.52

Table 1: Model Comparison.

The hyper-parameters chosen for each of the models:

Social and Vanilla LSTM

Embedding size: 64
Grid area: 4*4 squared mts.
Pooling window size: 32
Optimizer: RMS-prop
Learning rate: 0.003
Epoch: 10
Batch size: 5

LSTM (with optimized hyper-parameters)

(12 steps predict the next 8)
Train-Test data split = 70:30
Training window = 8
Learning rate=1e-4
hidden number=900
epoch=5
Batch size=300
Hidden Layers = 1

5 Concluding remarks

The performance of the models vary significantly when predicting one-step as opposed to multi-step. The basic LSTM with optimized hyper-parameters performs the best when predicting a single step ahead. The social LSTM performs the best when predicting multiple steps, while the Vanilla LSTM comes a close second. The LSTM's low performance on multiple prediction can be attributed to the fact that the hyper-parameters were optimized only for a single step and not for multi-step prediction. Hence, the error rate drastically increases when predicting for more than a single step. The Social LSTM's brilliant performance on predicting multiple steps could be attributed to the fact that the

[1]A Vanilla LSTM is the same as the social LSTM without the social layer component

model is able to capture both the individual patterns such as velocity and acceleration, and the potential dynamic close interactions with other individuals.[2]

[2]The basic LSTM and Kalman Filter have a window size of 8 and input data size of 12, in order to predict the next 8 steps. Though the same 8 steps are predicted by Social LSTM and Vanilla LSTM, this results in a different input data format that's fed into the models.

References

[1] Xiaodan Shi, Xiaowei Shao, Zhiling Guo *Pedestrian Trajectory Prediction in Extremely Crowded Scenarios*. Sensors, 2009.

[2] D. Helbing and P. Molnar *Social force model for pedestrian dynamics*. Physical Review E, 51(5):4282–4286, 1995.

[3] S. Pellegrini et al. *You'll Never Walk Alone: Modeling Social Behavior for Multi-target Tracking*. IEEE 12th International Conference on Computer Vision, 2009

[4] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. *Feature-based prediction of trajectories for socially compliant navigation*. In Proceedings of Robotics: Science and Systems, 2012.

[5] Conci, N., Bisagno, N., Cavallaro, A. *Group LSTM: Group Trajectory Prediction in Crowded Scenarios.* In Computer Vision–ECCV, pages Pages 213-225. Springer, 2018.

[6] Huynh Manh and Gita Alaghband. *Scene-LSTM: A Model for Human Trajectory Prediction*. arXiv preprint arXiv:1808.04018 (2018)

[7] Ben Taieb, S., Bontempi, G., Atiya, A., Sorjamaa, A. *A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition.* ArXiv e-prints, 2011

[8] Brownlee, J. (2017, March 8).. *4 Strategies for Multi-Step Time Series Forecasting.* Retrieved from
`https://machinelearningmastery.com/multi-step-time-series-forecasting/`

[9] Brownlee, J. (2017, May 10). *Multi-step time series forecasting with LSTM networks in python.*. Retrieved from
`https://machinelearningmastery.com/multi-step-time-series-forecasting-longshort-term-memory-ne`

[10] Brownlee, J. (2018, October 10). *How to develop LSTM models for multi-step time series forecasting of household power consumption*. Retrieved from
`https://machinelearningmastery.com/how-to-develop-lstm-models-for-multi-steptime-series-foreca`