Summer Reading 5: Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data

Social Robot Navigation Project @ Bot Intelligence Group

Paper:

Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, Marco Pavone. Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data: https://arxiv.org/abs/2001.03093

Summary:

Abstract

Most methods for trajectory forecasting ignore agents' dynamic constraints and do not account for environmental information (e.g., camera images, lidar maps, etc.) that modern robots have access to.

The authors propose ***Trajectron++***, a modular, graph-structured recurrent model that predicts the trajectories of a general number of diverse and time-varying agents while **incorporating agent dynamics and heterogeneous data (e.g., semantic maps) and accounting for multimodality in human behavior (which has potential for many different futures).**

Introduction

Multi-agent behavior prediction, reasoning about other agents' actions, could improve decision making, path planning, and proactive actions in human-robot interaction.

The authors are interested in developing <u>a multi-agent behavior prediction model</u> that **accounts for the dynamics (or dynamic constraints) of the agents (ground vehicles in this case)** and providing ways to **incorporate environmental information from camera images, lidar maps, etc. use heterogeneous data about the surrounding environment to**

Related Work

Deterministic Regressor

- Early works in human trajectory prediction were deterministic regression models. The earliest model used the Social Forces (attraction to goals and repulsion from other agents). Later models used a time-series regression.

Generative Probabilistic Approaches

- **Recent methods use generative approaches. They focus on multimodality (multiple plausible trajectories; a distribution of potential future trajectories) rather than focusing on the single best trajectory prediction.** Most of the recent models use CVAE (Conditional Variational Autoencoder) or GAN (Generative Adversarial Network), but **they ignore dynamic constraints (e.g. ignore the fact that their vehicle/robot cannot move side to side). <u>Thus, the authors propose to use Trajectron (CVAE-based and GAN-based) that accounts for dynamic constraints.</u>**

Problem Formulation

Trajectron++

1. Scene Representation
    a. The current scene is abstracted as a spatio temporal graph. Nodes in the graph represent agents and edges in the graph represent agents' interactions.
    b. The authors model the scene as a directed graph (can represent a more general set of scenes and interaction types) while previous approaches used undirected graphs.
2. Modeling Agent History
    a. The model (Trajectron ++) encodes each node's (agent's) current state, history, and how it is influenced by its neighboring nodes (neighboring agents).
    b. The model encodes the history of nodes (agents) [feeds the current and previous states (typically, positions & velocities) of the nodes (agents)] into a LSTM network with 32 hidden dimensions.
    c. Ideally, agent models should be chosen to best match their semantic class (pedestrians, cars, etc.). The authors modeled pedestrians as single integrators and wheeled vehicles as dynamically-extended unicycles to account for no side-slip constraints.
3. Encoding Agent Interactions
    a. Trajectron ++ encodes edges (interactions) in the graph to model neighboring nodes' (neighboring agents') influence on the modeled node (agent).
        i. Edge (interaction) information is collected from neighboring nodes (neighboring agents) of the same semantic class (pedestrians, cars, etc.) through an element wise sum instead of concatenation.
    b. The aggregated states are fed into an LSTM with 8 hidden dimensions. Then, the encodings from all edge types that connect to the modeled node are aggregated to obtain an influence representation vector.
        i. The influence representation vector represents the effect that all neighboring nodes have.
    c. Finally, the node (agent) history and edge (interaction) influence encodings are concatenated to produce a single node representation vector.
4. Incorporating Heterogeneous Data
    a. Using heterogeneous data (Lidar data, camera images, etc.) from modern sensors is more useful than just tracking trajectories of other agents. Trajectron++ encodes a local map, rotated to match the agent's heading, with a CNN.

Experiments

The authors used 3 publicly available trajectory prediction benchmark datasets: ETH, UCY, and nuScenes.

The authors used 4 different error metrics:

1. Average Displacement Error (ADE): Mean L2 (Euclidean) distance between the ground truth and predicted trajectories
2. Final Displacement Error (FDE): L2 (Euclidean) distance between the predicted final position and the ground truth final position
3. Kernel Density Estimate-based Negative Log Likelihood (KDE NLL): Mean NLL of the ground truth trajectory under a distribution created by fitting a kernel density estimate on trajectory samples
4. Best-of-N (BoN): The minimum (lower the better) ADE and FDE from N randomly-sampled trajectories

The authors have 5 datasets from ETH and UCY. They used a leave-one-out strategy to evaluate their model where the model is trained on 4 datasets and is evaluated on the fifth dataset.

Trajectron++ with dynamics integration and map encoding shows more accurate predictions than Trajectron++ base model (no dynamics integration nor map encoding).

Conclusion

Trajectron++ with incorporated heterogeneous data (data from sensors such as camera images, Lidar data, etc.) and dynamics integration (respecting the modeled agent's dynamic constraints) would be useful in downstream robotic tasks such as motion planning, decision making, and control.

Glossary:

Regressor

- In statistics, a regressor (x variable) is the name given to any variable (y variable) in a regression model that is used to predict a response variable.
  https://www.statology.org/regressor/#:~:text=In%20statistics%2C%20a%20regressor%20is,A%20manipulated%20variable

Online parameter estimation

- Online parameter estimation is typically performed using a recursive algorithm. To estimate the parameter values at a time step, recursive algorithms use the current measurements and previous parameter estimates. Therefore, recursive algorithms are efficient in terms of memory usage. Also, recursive algorithms have smaller computational demands. This efficiency makes them suited to online and embedded applications.
  https://www.mathworks.com/help/ident/ug/what-is-online-estimation.html

Strides

- Stride is a component of convolutional neural networks, or neural networks tuned for the compression of images and video data. Stride is a parameter of the neural network's filter that modifies the amount of movement over the image or video. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time. The size of the filter affects the encoded output volume, so stride is often set to a whole integer, rather than a fraction or decimal.
  https://deepai.org/machine-learning-glossary-and-terms/stride

CNN (Convolutional Neural Networks)

- A convolutional neural network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images. Convolutional neural networks are widely used in computer vision and have become the state of the art for many visual applications such as image classification, and have also found success in natural language processing for text classification.

  Convolutional neural networks are very good at picking up on patterns in the input image, such as lines, gradients, circles, or even eyes and faces. It is this property that makes convolutional neural networks so powerful for computer vision. Unlike earlier computer vision algorithms, convolutional neural networks can operate directly on a raw image and do not need any preprocessing.

  A convolutional neural network is a feed-forward neural network, often with up to 20 or 30 layers. **The power of a convolutional neural network comes from a special kind of layer called the convolutional layer. Convolutional neural networks contain many convolutional layers stacked on top of each other, each one capable of recognizing more sophisticated shapes.** With three or four convolutional layers it is possible to recognize handwritten digits and with 25 layers it is possible to distinguish human faces.

The usage of convolutional layers in a convolutional neural network mirrors the structure of the human visual cortex, where a series of layers process an incoming image and identify progressively more complex features.
https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network

Latent variable

- In statistics, latent variables are variables that are not directly observed but are rather inferred through a mathematical model from other variables that are observed.
https://en.wikipedia.org/wiki/Latent_variable

Feedforward Neural Network

- A feedforward neural network is a type of artificial neural network in which nodes' connections do not form a loop. All information flows in a forward manner only.
https://www.analyticsvidhya.com/blog/2022/01/feedforward-neural-network-its-layers-functions-and-importance/