7D-05

Enriching Graph Information for Pedestrian Behavior Learning

Nahum Alvarez Chenyi Zhuang

National Institute of Advanced Industrial Science and Technology (AIST)

Introduction

Pedestrian behavior simulation is a difficult task to perform due to the performance requirements and the necessary information to learn meaningful behavior patterns. Inverse Reinforcement Learning techniques help in solving those issues, as they learn from a set of observed behaviors provided by an expert and can be processed before the simulation. We developed a variant that includes contextual actions and multiple reward functions and adapted it to work with a multi-agent based pedestrian simulator. The agents in the simulator are able to navigate the map with no information other than the learned behavior patterns and obtain better results than other methods in terms of goal clear times and trajectory optimization. We called them "Contextual Action Multiple Policy Inverse Reinforcement Learning" (CAMP-IRL) agents.

However, we found several instances when only having the data from expert trajectories was not enough to obtain the desired knowledge. For example, under certain conditions, traversing certain areas of unknown layout to reach concrete goals is difficult for trained agents, whilst for humans such information should be trivial to deduct. In order to avoid such situations, we devised a method to improve the available graph information contained in the trajectory database.

Pedestrian Simulation

Our simulator simplifies the city map into a 1-dimensional network consisting of nodes and links. The model of the map consists in a custom xml that describes the map in the form of network where nodes represent intersections and links represent streets or paths. A link also has length and width attributes influencing how long the agents need to walk from a end to another and how many agents can walk in parallel, and can be two-way or one-way. Nodes also can have features, and information describing what facilities are on that location.

The inputs of our method are the city map in this model and a file containing the trajectories we want to train the agents with. This method is performed before the simulation as a pre-processing task, so even if it can take a long time depending of the complexity of the map it does not represent a big impact in the simulation speed as the decision process of the agents once we have these files is enough fast to use it in real time. In our experiments, the CAMP-IRL agents outperform by far other types of agents that do not work with multiple policy functions or contextual actions, but we identified one issue that hindered their behavior. We observed that some useful information that should be extracted from the map and the routes was not being reflected in the learning process; in some of our experiments, one of the goals was a scarce feature that only was present in four nodes of the map; the agents were able to find it, but the wandered excessively before to do it. The main reason was that the system switching between different policies without finding any goal.

After analyzing why this was happening, we found that this situation was due to the coincidence of two factors: scarcity of the goal feature in the map and having only a few and indirect ways to reach those features. In one example, in order to reach one of its goals, agents had to cross from one area of the map to another which could only be reached by crossing three links between them, but those links were not very remarkable in terms of learned value for the selected policies to reach that feature. Thus, agents were conducted by their policies to go towards the feature, but when reaching the nearby areas of the map they could not find the crossing point which was far away. We plan to solve this issue by improving the learning process by adding enriched information to the map, trying to establish semantic relations between nodes of the map like those crossing points and the featured nodes using a method we explain in the next section.

Map Enrichment

Before training the pedestrian behaviors, we first propose a method to improve the available graph information contained in the trajectory database. The intuitive idea we have is that by enriching each graph node information by its neighbors, an agent could make better decisions when choosing the next node to move.

In our experiments, the graph we collected from the trajectory database contains the following information: (1) there are 13 categories (i.e., *hostel, books, convenience, restaurant, café, dry_cleaning, hospital,*

supermarket, *fast_food*, *kindergarten*, *telephone*, *cinema* and *post_office*) describing each node; and (2) the graph structure is stored as an adjacency matrix. Therefore, the input of this method are a feature matrix $X \in \mathbb{R}^{n\times 13}$ recording the category information for all the n nodes and an adjacency matrix $A \in \mathbb{R}^{n\times n}$ recording the graph structure. The output of this method is a new feature matrix $\hat{X} \in \mathbb{R}^{n\times k}$, where the value of k depends on how many different filters we will used in this method.

Having defined the input and output of this method, in the remainder of this section, we will introduce the method in detail. Similar to signal processing, our method consists on three steps: (1) by using discrete Fourier transform, we first transform the feature matrix X from the graph vertex domain to the graph spectrum domain that is donated as matrix X'; (2) then, we do filtering on X' in the graph spectrum domain; (3) at last, by using inverse discrete Fourier transform, we transform the filtered feature matrix from the spectrum back to the vertex domain.

1. Transform $\mathbf{X}$ in graph vertex domain to $\mathbf{X}'$ in graph spectrum domain:
To do the graph Fourier transform, we first need to calculate the eigenvectors and eigenvalues of the graph Laplacian matrix $L = D - A$, where A is the adjacency matrix and $D = diag(\sum_{j\neq i} A_{i,j})$ is the degree diagonal matrix. After obtaining the Laplacian matrix L, we obtained its eigen- vectors and values using the following factorization:

$$L = U \Lambda U^{*},$$

where all the eigenvectors are stored as columns in matrix U, the diagonal matrix Λ records all the eigenvalues and $*$ is the conjugate transpose operator. Then, the graph Fourier transform is defined as:

$$\mathrm{X}' = \mathrm{U}^{*}\,\mathrm{X}.$$

Since each node has 13 categories in our case, by regrading X as a signal having 13 channels, the equation above maps the signal from vertex domain into the spectrum domain, i.e., X'.

2. Do frequency-based filtering on $\mathbf{X}'$:
Then, we apply different filters to the obtained X'. Without loss of generality, if we define any filter as a function g, the frequency-based filtering process is: $\mathrm{g}(\Lambda)\mathrm{X}'$.

Similar to signal process, the input of function g are the n eigenvalues stored in Λ, which can be regarded as *graph frequencies*. By defining different filters, we can adjust the final results. Since we do not know which eigenvalues are important for our final pedestrian simulation in advance, we constructed a filter bank to record as many filters as possible. In our experiments, we utilized heat-kernel based filters [1] and Meyer filters [2]. Heat-kernel based filters are low-pass filters (only allowing small eigenvalues to pass the filter) and Meyer filters cover all the frequency ranges, which can allow low-pass, band-pass and high-pass. By comparing these different filtering strategies, we want to verify whether map filtering would improve the final pedestrian simulation performance.

3. Transform X' in graph spectrum domain back to graph vertex domain, i.e. the output $\hat{X}$:
After filtering, we finally transform the information in spectrum domain back to the vertex domain. Assuming we used two filtering functions $\mathrm{g}_1, \mathrm{g}_2$ in the second step, the final output would be calculated as:

$$\hat{X} = [\mathrm{U}(\mathrm{g}_1(\Lambda)\mathrm{X}')] \oplus [\mathrm{U}(\mathrm{g}_2(\Lambda)\mathrm{X}')],$$

where $\oplus$ is the matrix concatenating operator along the second dimension.

Using different filtering functions g would lead to different graph filtering results. To automatically identify which filters are better than others, is advisable to perform a cross validation process where the simulation output feeds the map enrichment method and a final simulation would automatically do the selection.

Once the map enrichment process finishes, it generates a modified map file where the nodes' features are modified and can be passed to our inverse reinforcement learning method to generate finally the behavior patterns and pass them to the CAMP-IRL agents.

Performance comparison

We compared the performance of the simulator using a map without processing with the ones obtained using enriched maps. Our initial experiments showed positive results with the agents obtaining better times in reaching goals when using enriched maps. However, still it is necessary to perform the mentioned cross validation for the different filtering functions, as the resulting performance varies greatly depending on it. And also, different filters have different sets of hyperparameters so their tuning has to be selected as well.

References

[1] Nicole Berline, Ezra Getzler, and Michele Vergne. Heat kernels and Dirac operators. Springer Science & Business Media, 2003.
[2] Nora Leonardi and Dimitri Van De Ville. Wavelet frames on graphs defined by fmri functional connectivity. In Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on, 2136–2139.