Activity forecasting

Bo Fan

ECE Department, Tufts University
Bo.Fan@tufts.edu

Abstract

This project aims to deal with the problem of predicting the behavior and decisions an agent will make via learning the past trajectories. It uses the Hidden Markov Decision Process model (HMDP) to simulate the decision making process of an agent. In the training step, it learns the weights from different features. In the planning step, optimal policies are generated through a backward algorithm, and a final visitation distribution will be propagated via a forward approach. Experimental results demonstrate that the HMDP model and the backward forward algorithm can accurately predict human behavior.

Our Work

This project [1] is quite comprehensive. To make it simple, we directly use the bird view images, ground truth/observed trajectories, and different features from the author's website without doing image projection, super pixel tracking [2] and semantic scene labeling [3]. The weights of those features will be computed according to the author's demo called IOC (inverse optimal control). After the simplification, we have done the following work:

We build up the Visual C++2012 and OPENCV 2.9.0 IDE, and solve two bugs from the author's online code to run his IOC and OC demos smoothly.

Then, we design an XML to MAT conversion function, which is able to extract 40 feature matrices from the original XML file and put them into an MAT matrix. The MAT file is only 1/3 of the original XML size and easier for processing through Matlab.

We study the Hidden Markov model, the maximum entropy framework and the exponentiated gradient descent method. Using Matlab, we also rewrite the backward-forward pass algorithm (OC), and successfully predict the paths for one and multiple goals cases. Finally, we develop a conversion function from one channel matrix to HSV/RGB format image, which can plot the rewards, value functions or the paths directly on the bird view image. It is shown that our simplified code can perfectly predict the trajectories of a person given one starting position and N destinations.

HMDP model and algorithms

The idea behind this project

The MDP can be used to express the dynamics of a decision making process, and the reward at state "s" is simply the sum of all the weighted features at that state.

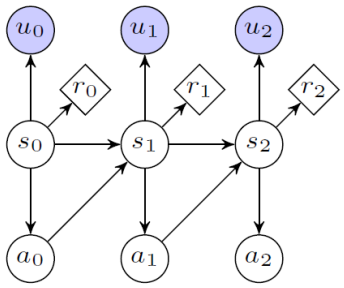
$$r(s;\boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{f}(s)$$

,where θ are unknown weights of those features.

In the maximum entropy framework, the distribution over a state sequence is:

$$p(s;\boldsymbol{\theta}) = \frac{\prod_t e^{r(s_t)}}{Z(\boldsymbol{\theta})} = \frac{\prod_t e^{\boldsymbol{\theta}^T \mathbf{f}(s_t)}}{Z(\boldsymbol{\theta})}$$

In a vision based system, we can't get access to the true states. Therefore, Hidden Markov Decision process can deal with the observation uncertainty. If the tracker output has low precision, the corresponding weights will be decreased during the training step to minimize the reliance on the tracker output. A typical HMDP model is depicted as follows:



For a HMDP model, state observations "u" are used to represent the uncertainty. The joint distribution over state and observations can be as follows:

$$p(s,u;\boldsymbol{\theta}) = \frac{\prod_t p(u_t \mid s_t) e^{\boldsymbol{\theta}^T \mathbf{f}(s_t)}}{Z(\boldsymbol{\theta})} = \frac{e^{\sum_t \{\boldsymbol{\theta}^T \mathbf{f}(s_t) + \log p(u_t|s_t)\}}}{Z(\boldsymbol{\theta})}$$

Training and inference phase

In the training phase, we want to infer the weights θ and an optimal policy $\pi_\theta(a \mid s')$ by maximizing the entropy of the conditional distribution as bellow:

$$p(s \mid u;\boldsymbol{\theta}) = \frac{e^{\{\sum_t \boldsymbol{\theta}^T \mathbf{f}'(s_t)\}}}{Z(\boldsymbol{\theta})}$$

,where the feature vector $f'(s_t)$ can include the features from the tracker.

To maximize the entropy, we use exponentiated gradient decent to minimize the gradient of its log likelihood. The gradient is the difference between the empirical mean feature count and the expected mean feature count (they are shown in algorithm 3).

Empirical mean feature count is the average features accumulated over M demonstrated trajectories. The expected mean feature count is the average features accumulated by trajectories generated by the learned weights and optimal policy. For the initial training step, weights are all set to 0.5.

The following algorithms 1,2,3 show how to learn the weights. V is a soft estimation of the expect cost from s to the goal. Q is also a soft estimation of the expect cost of reaching the goal after taking action "a" from s. D is the expected visitation count at state s.

Algorithm 1 computes V and Q iteratively from any state "s" to the goal. When it stops, an optimal policy will be generated under the maxim entropy assumption.

Algorithm 2 will use the policy from backward pass and accumulate the probability at any state.

The third algorithm just updates the weights by computing the difference between the empirical and expected mean features.

Algorithm 1 Backward pass

$V(s) = -\infty$

For n=N,…,2,1 do

$V^{(n)}(s_{goal}) = 0$

$Q^{(n)}(s,a) = r(s;\theta) + V^{(n)}(s')$

$V^{(n-1)}(s) = soft\max_a Q^{(n)}(s,a)$

end for

$\pi_\theta(a \mid s) = e^{Q(s,a)-V(s)}$

Algorithm 2 Backward pass

$D(s_0) = 1$

For n=1,2…,N do

$D^{(n)}(s_{goal}) = 0$

$D^{(n+1)}(s) = \sum_{s',a} P^s_{s',a} \pi_\theta(a \mid s') D^{(n)}(s')$

End for

$D(s) = \sum_n D^{(n)}(s)$

Algorithm 3 Exponentiated gradient descent

$\bar{\mathbf{f}} = \frac{1}{M}\sum_m^M \mathbf{f}(s_m)$ (empirical mean feature count)

$\hat{\mathbf{f}}_\theta = \sum_s \mathbf{f}(s) D(s)$ (expected mean feature count)

$\nabla \mathbf{L}_\theta = \bar{\mathbf{f}} - \hat{\mathbf{f}}_\theta$

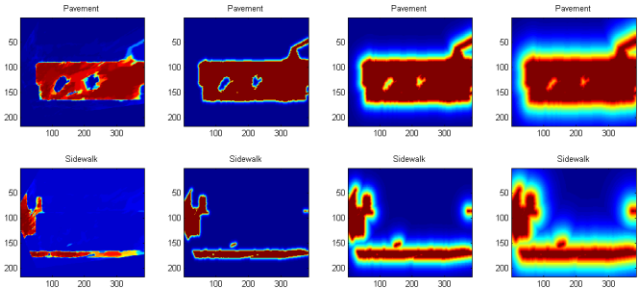
$\theta = \theta e^{\lambda \nabla \mathbf{L}_\theta}$

Planning phase

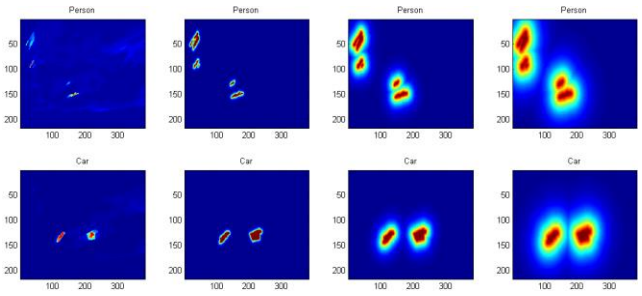
For the planning phase, we consider those weights as known parameters, and run algorithm 1 and 2 without running algorithm 3 to predict the trajectory distribution. When algorithm2 stops, the expected visitation count D is the trajectory distribution. This algorithm can also predict multiple goals from one starting point. We just need to change the role of starts and goals from algorithm 1 and 2.

Experimental results

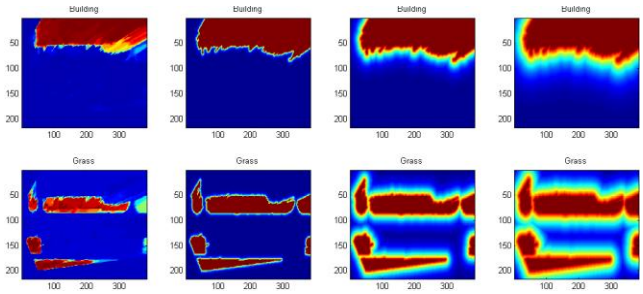
In our experiments, we use 40 features from the author's web page. There are 9 semantics labels including grass, pavement, sidewalk, curb, person, building, fence, gravel and car. For each semantic label, there are four features, including the raw probability and three types of distance-to-object features. There are also three features from the pedestrian tracker by blurring its raw trajectory with 3 different Gaussian filters. The following features are derived according to the author's XML files:



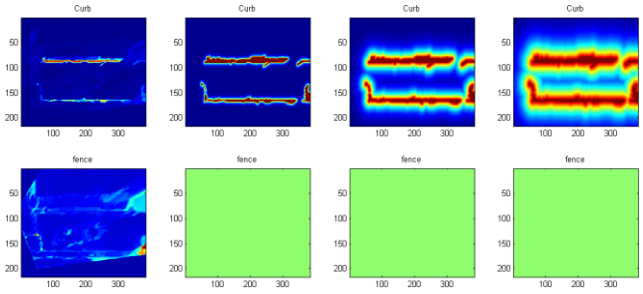
Pavement and side walk



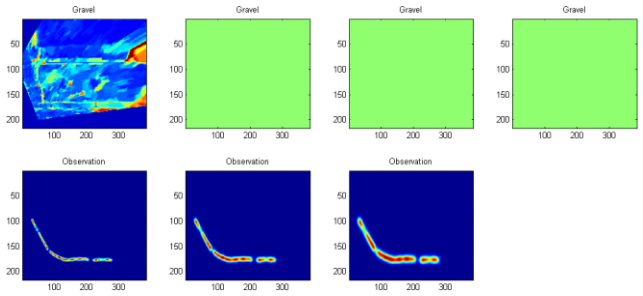
Person and car



Building and grass



Curb and fence



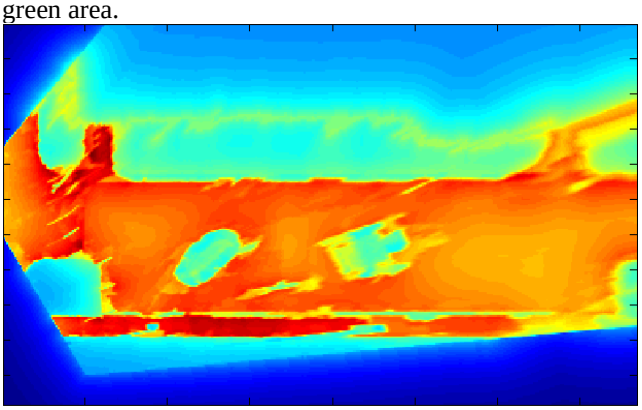
Gravel and observation

Tab. Weights learned from the training datasets.

	raw	Dist 1	Dist 2	Dist 3
Pave ment	0.718705	0.0275457	0.0903541	0.209453
Side- walk	0.91435	0.0510705	0.015433	0.025499
Per- son	0.0862693	0.0644116	0.0475729	0.0625473
Car	0.0896602	0.104552	0.125872	0.12193
Buildi ng	0.087527	0.0827183	0.0716442	0.144106
Grass	0.0496668	0.0679752	0.109652	0.119538
Curb	0.0145358	0.000286201	0.0016849	0.0085608
Fence	0.0859901	0.0879989	0.0879989	0.0879989
gravel	0.0878917	0.0879989	0.0879989	0.0879989

For the following experiment, there is one starting point (94,28) at the top left corner of the bird view image. The location of the destination is set to (160, 350).

The reward function is computed from 37 features except for the observation features. The region with red colors means it will have higher rewards compared to blue or green area.



Reward function map

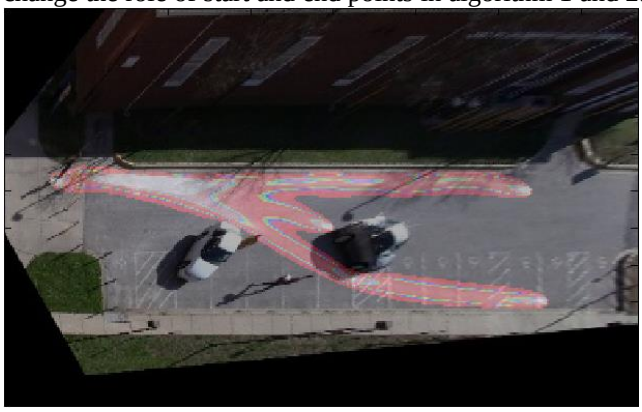
In the backward pass phase, system will compute the value function from the end point, and the value at each state will be propagated until the stopping criteria is satisfied.

Value function map

The forecasting distribution D was printed onto the bird-view image. We need to do image processing task to convert a 2d matrix into an RGB color space.

Forecasting distribution for one goal

For the previous case, we only try one start and one goal. This time, we will do the simulations for one person with multiple goals. In the following experiment, there are three goals with coordinates (160,330), (120,200) and (100,320). Compared to the previous experiment, we only need to change the role of start and end points in algorithm 1 and 2.



Forecasting distribution for multiple goals

Destination forecasting and future work

Previous experiments only show the future path when we know the goals and the starting points. However, the destination of a person may not be known. If we have a sequence of observations, what is the probability that an agent will reach goal A instead of the other goals? That problem is called Destination forecasting. It can infer the destination of the person from noisy observations, but all possible destinations have to be given. Following [1][4], the posterior over goals can be approximated by the following formula:

$$p(s_g \mid s_0, u_{1:t}) \propto p(u_{1:t} \mid s_0, s_g) \cdot p(s_g)$$
$$\propto e^{V_{u1t}(s_g) - V(s_g)} \cdot p(s_g)$$

However, the definition of those value functions are not clear in [1], and in some sense the ratio of those state log functions are computed differently from [4].

Apart from the computation of state distribution and destination forecasting, the author's maximum entropy framework can also deal with sequence smoothing and robust generalization problems. Sequence smoothing means the distribution will be updated with noisy observations. Robust generalization (knowledge transfer) can use learned parameters for forecasting in a new scene.

Conclusions

This project includes many techniques from 3D-2D image transform, semantic scene labeling, super pixel tracking, maximum entropy framework, parameters inference, backward-forward pass algorithm and etc. The hardest part is sequence smoothing and destination forecasting, which are shown in the author's paper without specific formulations. I have tried to solve those two problems myself, but did not have time to figure them out correctly. One clue I have found is to use the idea from HMM or CRF.

The author does not use the features from noisy trajectories for training, but applies ground truth trajectories to compute the empirical features. Although he claims that his IOC algorithm can deal with noisy observations, the learning effect might be sensitive to noisy features, which makes me doubt about whether his model can deal with noisy observations properly.

References

[1] K. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In ECCV, (2012)

[2] Wang, S., Lu, H., Yang, F., Yang, M.H.: Superpixel tracking. In: ICCV. (2011)

[3] Munoz, D., Bagnell, J.A., Hebert, M.: Stacked hierarchical labeling. In: ECCV.(2010)

[4] Ziebart, B., Maas, A., Bagnell, J., Dey, A.: Maximum entropy inverse reinforcement learning. In: AAAI. (2008)