Andrey_Vagin

Report

I decided to use Reinforce Policy Gradient Algorithm. The main part of the task solution is to choose appropriate state representation as I decide how my agent will perceive the world.

Environment description

We have indefinite world size, desirable area placement and size, number of cargoes and their shape. My agent is represented as ANN which is fixed in size, but it works for any world configuration, so I came up to following state representation:

```
cargo_state = {
                'upper_distance': (self.p_upper - lower) / parking_height,
                'lower_distance': (self.p_lower - upper) / parking_height,
                'left_distance ': (self.p_left - right) / parking_width,
                'right_distance': (self.p_right - left) / parking_width,

                'relative_area': cargo_relative_parking_area,

                'upper_intersec': intersections[0],
                'lower_intersec': intersections[1],
                'left_intersec': intersections[2],
                'right_intersec': intersections[3],

                'is_intersected': False if cargo_parking_reward == cargo_intersected_reward else True
            }
```

State is represented as dictionary with information about each cargo. Through the training and inference process I iterate in cycle each cargo and give it’s state to the ANN. The model can decide to move cargo to one of the 4 directions or leave it at the same place. This is how problem of uncertain number of cargoes is solved.

First 4 parameters are differences of cargo’s bounding box coordinates from the desirable area. They are measured vice versa. For example distance between lower part of cargo and upper part of desirable area. By the sign of such difference the model can understand where to move cargo and whether it already partly in the desirable area. As we do not know environment size these distances are scaled regarding the environment size.

relative_area is *area of* cargo placed in the desirable area divided by the cargo size.

Also I wanted somehow represent relative position of intersection with other cargoes to give the model understanding where to move cargo in case of intersection. Next 4 parameters are bool

values which indicate with which part cargo is intersected by the other cargo.

If cargo has odd shape then parts in the center are not considered as left or right and upper or above. To represent fact of intersection of small cargoes with one of the shape parameters equal to 1 there is the last parameter *is_intersected.* It is true if cargo is intersected by other cargoes.

Neural Network Architecture

I have chosen 2-layer ANN as policy estimator. The number of cargoes is different for different maps, so ANN chooses actions for all cargoes sequentially. The output of the model is 5 values which stand for different actions that can be performed on the given cargo.

Action to perform is chosen with probability of corresponding softmax value. The first action is not to move given cargo (if it is in optimal position now), other actions stand for one step moves in 4 directions.

With such architecture the model can operate on any number of cargoes.

Training Method

Training consists of several epochs. Each epoch the model is trained on all the maps sequentially. I have created different maps to help the model to learn general rules for task solving. These maps have different sizes, placements of desirable area, number of cargoes and their shapes. There are exist as simple maps (with small number of cargoes with simple shapes) which big number of solutions and more complex ones where harder to place all the cargoes.

Model gets difference between current obtained reward and previous one. Such a trick can help the model understand whether performed action good or bad. If such reward is greater than zero then action was performed in right direction, if negative then action was wrong.

Difficulties

The main part of the thinking process was dedicated to choosing what the state will look like. We have unknown parameters for all the sizes and ANN as estimator. ANN is fixed in it’s configuration and it should work for any map.

There were three main problems:

1. Unknown sizes of map, desired area and corgoes.

   It was solved via taking relative distances and sizes.

2. Give the model an understanding of the relative parking position.

   Was solved by taking difference of coordinates of sides of the desirable area and cargo's bounding box sides. The sides were taken opposite (upper of desirable area with lower of cargo's bounding box) to give an understanding of whether the cargo is at least partially in the desirable area. If it's lower side position is lower than upper side of the desirable area and the same holds for left or right then cargo is at least partially inside the desirable area.

3. Unknown number of cargoes.

   Model considering all the cargoes sequentially and decides take or not to take action on the given cargo. (special 5-th action was added for that)

Environment Architectures

There was one main architecture that was modified through experiments.

I started with cargoes representation. **Cargo class** can estimate cargo's position relatively defined by upper left corner. This class can define upper, lower, left, right parts to determine intersections with other cargoes. Cargo can be moved only to the valid positions, appropriate methods are included into the implementation.

The second class which is responsible for environment representation is **World class.** At first it scans given map and initializes Cargo objects for cargoes found on the map, also coordinates of desirable area are found. I tried to follow interface of *gym* environments, so my class has *reset()* and *step()* methods. World class is responsible for moving cargoes, counting rewards and intersections of cargoes in the desirable area.

All of the above was implemented to simplify the use of the environment in training as much as possible. You should provide only read .txt map to the **World class** and then it will initialize all the things and will do all the needed work by itself.

New Knowledge

Now I am experienced in writing an environment and the agent for it from scratch. In this task I had to analyze the problem, choose appropriate environment representation, provided reward, the way of training, choose valuable training maps and model architecture. Finally obtained solution was evaluated.

These are steps of whole process of solving an RL problem.

Conclusion

My algorithm do not converge well (It solves 3 of 7 maps in a reasonable number of steps). I think better representation can be chosen and adding additional rewards during the training process can help to solve the issue.