Project report

Exercise 3
Visual Planning

Deep Learning Lab Course 2017
University of Freiburg

Autors: Fabien Jenne 3725777 Johannes Engler

Robotics & Neurorobotics track

17.12.17

Contents

1 Outline

The task of this exercise is to implement a convolutional neural network (CNN) which facilitates visual planning. The well established A^* planning algorithm is used for reference.
The environment to plan on is given by a 25×25 block-sized 2D grid-world. The only input the CNN receives is a partial local view of the world whose size is adjustable as hyperparameter.
The training data provided is generated by performing A^* search on the given world. The labels $y \in \{0, ..., 4\}$ encode the possible actions the agent can take:

$$0 = \text{no action} \mid 1 = \uparrow \mid 2 = \downarrow \mid 3 = \leftarrow \mid 4 = \rightarrow .$$

The aim of this project is to train the CNN such that based only the local view of the environment and a history of some earlier states, the CNN predicts actions which eventually lead the agent to the goal location. Furthermore, it should do it in an 'optimal' fashion, reaching a performance level as close to A^* as possible.

2 Implementation

We implement the CNN in Python using the Keras framework which is based on the Tensorflow API.

Our network has the following final architecture:

- Layer 1: 32 filters, 3x3 kernel, ReLu activation
- Layer 2: 64 filters, 3x3 kernel, ReLu activation
- Max-pooling layer: 2x2 kernel
- Dropout: 25%
- Fully connected layer: 128 filters, ReLu activation
- Dropout: 50 %
- Loss layer: Cross-entropy loss, Softmax activation

The `Adadelta` optimizer is used. Training is performed with 500 mini-batches, each consisting of 32 data samples. The default local window size is 5×5. In this exercise we stick to a fixed epoch size of 30.

Implementation notes We find the implementation of the CNN itself straight-forward after the last exercises. They pervious exercises also helped a lot to get an idea of what is going on 'behind the scenes'. We faced most difficulties getting the data into the right format and to maintain consistency of labels and dataset. After resolving these issues, the network performes nicely.

3 Evaluation

3.1 History length

history length	test accuracy	$\frac{\text{episodes solved}}{\text{episodes}}$
1	0.93	0.756
2	0.94	1
3	0.953	1
5	0.954	1
10	0.978	1
20	0.968	0.943

Table 3.1: Test set accuracy and planning performance for several history lengths.

In the first task, we examine the effects of different history lengths as seen in table 3.1. The CNN does not model temporal relations. It just 'sees' several states stacked together. Therefore, it is not surprising that for a history length of $h = 1$ (only the current state) the accuracy is not that high. The state-space is just to large to infer a precise action based only on one local view. The ambiguity of which action is best is too high.
Increasing the history size resolves this issue. Also, the planning performance of finding the goal within an limited interval of applied actions is ways more sophisticated now. In fact, our CNN finds the goal in every case leading to a planning accuracy of 100 %. Around 40-400 episodes are tested for each history length.
Interestingly, the performance decreases again for a large history length of $h = 20$. We assume this is because for that many states stacked together without temporal relation, the stacked augmented states the network works with might be similar, whereas the most current state might differ a lot. This then leads to misclassifications.
For the following tasks, we stick to a history length of $h = 4$.

history length	test accuracy	$\frac{\text{episodes solved}}{\text{episodes}}$
3	0.88	0.898
5	0.974	1
7	-	-

Table 3.2: Test set accuracy and planning performance for different window sizes.

3.2 Local window size

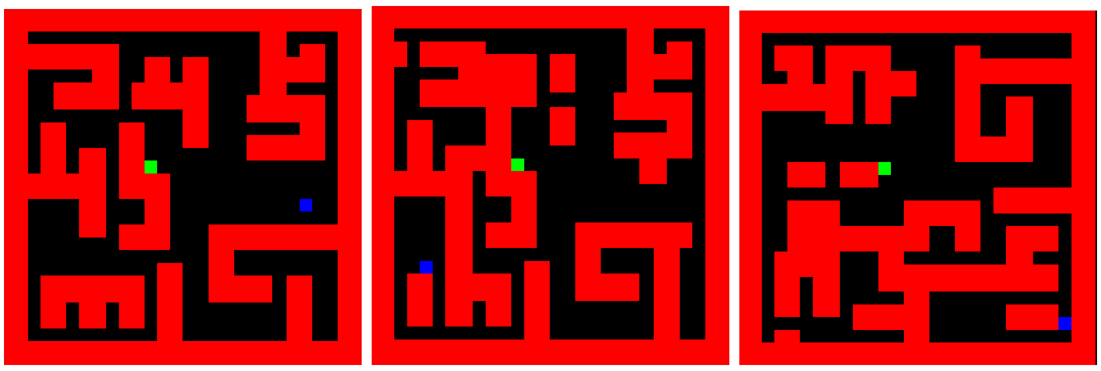
This task is about changing the window size the CNN sees as state observation. We first decreased the local view to a 3×3 window. As seen in table 3.2, the accuracy and performance both decrease as well. Increasing the window to a size of 7×7 did not work, since it would require to enlarge the border of the map from 2 to 3. We can clearly see that a larger window size makes the state the agent sees more identifiable. The configuration space and therefore the uniqueness of a state increases significantly with each enlargement. This yields higher overall accuracies. It also corresponds to our human intuition. The more we see of the environment that surrounds us, the more certain we usually get where we are. Interestingly, the CNN agent show exactly the same behaviour. If it does not yet know where in the world it is, the agent starts some sort of exploration until it localizes. Once localization is acquired, the agent chooses the nearest path to the goal in A^* like fashion.

3.3 Target location

When the target location is changed after training, the agent has severe difficulties finding its goal. Only if the target is 'by chance' on the way the agent travels along, it reaches the goal state. It is interesting to observe that the agent tries to reach the goal location it was trained on before and looks it up several times. In fact, this is exactly the behavior we expected. The agent (the CNN) was trained to mimic the A^* algorithm with the old goal location. Compared to A^* which contains a heuristic function, there is nothing in the CNN that accounts for a new target location.

3.4 The map

This task is about the effects of changing the map of the environment the agent was trained on. Therefore, we first alter the base map a little by blocking some paths and



(a) base map (b) slight changes (c) major changes

Figure 3.1: Illustration of different maps used for evaluation of the CNN.

adding some new ones. The resulting map differs, although the main structure is preserved. Evaluation shows that this way we at least preserve the chance that the agent reaches the target. With a maximum number of steps $\varepsilon = 75$, the agent finds the goal location in $\approx 65\%$ of the cases by averaging over mutiple runs of the `test_agent` procedure.

Going further, we apply major adjustments to the map, e.g. rotating the map but keeping the target location at the old place. Here, the agent is incapable of reaching the target location. For $\varepsilon = 75$ steps at maximum, the agent arrives at the goal only at $\approx 10\%$ of the cases.
This task shows that the CNN agent generalizes badly. With just a few minor changes, it still reaches the goal in most cases. However, as soon as there are major changes to the map, the agent has no chance of getting to the target. This is reasonable, as the environment changed from the state the CNN was trained on. With the lack of a heuristic function, the CNN agent has no option to account for the changed conditions.

3.5 Further ideas

As pointed out in the previous sections, in the current configuration our CNN does not generalize at all for different maps. With a CNN the agent learns on the input horizon we provide. While the map stays constant, the agent can simply learn the correct actions. Given a new map, the old (learned) sequences just don't work anymore. We first propose to use an agent which learns on different maps. As a result, the feature selection would be completely different here. For example, the agent cannot rely on the input sequence anymore but has to come up with different techniques such as using the walls for navigation.

Further adjustments to improve generalization could be to train differently on different maps and use some sort of confidentially measure for the predictions. Then an arbiter could choose the most probable action.
In the course of this exercise we implement the first idea mentioned above (training on different maps), but the results unfortunately did not generalize any better either. For further improvements, we hope to try a more suitable NN architecture than CNNs for this kind of task the next time.