University Tübingen
Self Driving Cars WiSe 2018/19

Andreas Rist
Joshua Marben

Excercise 1

21. November 2018

Task 1: Network Design

b) The module *training.py* contains the training loop for the network. Read and understand its function *train*. Why is it necessary to divide the data into batches? What is an epoch? What do lines 43 to 48 do? Please answer shortly and precisely.
Solution: It is helpful to divide your data into batches for two reasons. First there is less memory used for a training epoch of the network, since we do not have to hold all data in memory for a training update (only the data of one batch). Second the network learns faster with smaller batch sizes, since there are many incremental updates to the weights. By not using all the data in every iteration, we introduce a certain noise, which helps the gradient descent.
An epoch is is one forward and one backward pass of all training data.
We take the batch of our training data, calculate the prediction of the network, compare them to the target values using the crossentropy-loss-function. Then the loss is used to propagate the error back through the network to adapt the weights in the network.

c) Define the set of action-classes you want to use and complete the class-methods actions to classes and scores to action in *network.py*
Solution: The provided data of the expert imitations comes as a set of three:

$$(steer, gas, brake)$$

with

$$\begin{aligned} steer &\in \{-1, 0, 1\} \\ gas &\in \{0, 0.5\} \\ break &\in \{0, 0.8\} \end{aligned}$$

We want to use following classes:
{steer_left}
{steer_right}
{steer_left and brake}
{steer_right and brake}
{brake}
{gas}
{chill}
Accelerating and steering at the same time makes no sense to us, because it would conclude in a driving a donut.

e) Motivate your choice regarding the number of image-channels. Can you achieve better results when changing the hyper-parameters? Can you explain this?

Solution: We use all three color-channels, since we expect the network to better interpret the important difference between the green grass and the grey road! Regarding the hyperparameters: Yes, it can be archieved. As you change the hyperparameters, you change the behaviour or the calculation of the network. Therefore it is possible, that you can optimize by changing them. E.g. more epochs leads to a closer adaption to the training data, the number of epochs can thus influence the ability to abstract a certain situation. Many epochs lead to a low training error, however the network can only produce correct actions for situations very similar to the training data. More layers increase the learning capacity, however it is difficult to find a good number of epochs.

f) (I) What is 'good' training data?
Solution: Good training data is data which contains every possible situation or situations with actions which are very close the reality. Therefore training-data should be versatile, contain many different situations to achieve good generalization.

(II) Is there any problem with only perfect imitations?
Solution: Perfect imitations are bad because the network can't adapt to situations which are not in the training data. The network needs to learn how to behave on unexpected situations and how to correct mistakes.

Task 2: Network Improvements

1. **Observations:** What does it do?
Solution: These enables you to encoperate informations of speed, abs, steer and gyroscope into your network. Therefore not only positions of the vehicle can be considered by the network, but also any of these sensors. Therefore it is possible to make different conclusions for seemingly same situations (sometimes the car would just not drive because with the street-layout ahead, no gas would be optimal in most cases - however, with sensor data, this should not happen because the network can see that with no acceleration it is never good to not drive at all).

How does the performance change?
Solution: Curves after high speed passes now enable the car to brake first, as it would usually conclude into leaving the racetrack.

2. **MultiClass prediction:**
Solution: While we did not evaluate this, we expect this to require higher training times, because we have more complex legal target structures. We also have to change the interpretation (obviously) because we need not the maximum of the

predicted probabilities, but all above a certain threshhold (maybe left/right and up/down SHOULD be exclusive...)

3. **Classification vs Regression:**
   Solution: Here, we want to directly predict the action-tuple (steer, gas, brake). Mean squared error could be used as loss-function. This approach allows a much finer control of the car, since we can accelerate a little bit and steer left and right much smoother. However, our training data might not be suited, since it contains only information for a fixed amount of different tuple-combinations. Maybe the network could interpolate the values inbetween - however this would be much better with more versatile (and exploratory) training data.

4. **Data Augmentation:**
   Solution: We expect the performance to increase, since the network sees more possible actions with possible observation and the thus introduced noise can help avoid overfitting and increase performance by providing more examples on good actions as well.

5. **Fine-Tuning:**
   Solution: All the things from the text are good ideas. Especially class imbalance might be important. An automated search for hyperparameters might be rewarding as well. Next time, we'll start sooner!!