AI & Robotics: Research
Exercise 2

Marc Toussaint
Learning & Intelligent Systems Lab, TU Berlin
Marchstr. 23, 10587 Berlin, Germany

Summer 2020

Please prepare some written solutions to the following exercises that you can share by screen in our session. The notes can be brief, but esp. equations and derivations (as needed for the 2nd and 3rd exercise) should be written precisely. Try to use LaTeX.

1 Parameters & Fully Conv Residual Networks

The TossingBot paper uses "fully convolutional residual networks" at several places. Let's learn about them in detail! A core reference is [13].

a) For the perception module, the TossingBot paper uses "a 7-layer fully convolutional residual network [4, 13, 19] (interleaved with 2 layers of spatial 22 max-pooling), which outputs a spatial feature representation μ of size $45 \times 35 \times 512$ that is then fed into the grasping and throwing modules."

   In a yet unpublished journal version, they give more details: "Our network architecture consists of the following layers for each module:

   - Perception: C(3,64)-MP-RB(128)-MP-RB(256)-RB(512)
   - Grasping: RB(256)-RB(128)-UP-RB(64)-UP-C(1,2)
   - Throwing: RB(256)-RB(128)-UP-RB(64)-UP-C(1,1)

   where C(k,c) denotes a convolutional layer with k×k filters and c channels, RB(c) denotes a residual block [14] with two convolutional layers using 3×3 filters and c channels, MP denotes a 3×3 max pooling layer with stride = 2, and UP denotes a bilinear 2× up-sampling layer."

   Additionally, assume that all layers include bias parameters. Also, assume zero padding to preserve the input size in a convolutional layer.

   Describe in all detail the architecture of the perception network. How many layers are there? For each layer, what's the size (width, height, channels)? Which layers are connected how? What is the exact total number of parameters that this network has?

   (A comment by a teaching assistant: "It's quite interesting because the students could either calculate it by hand or just implement it and count. Both is an interesting exercise I think.")

b) The ResNet paper [13] talks about a degradation phenomenon. Explain this phenomenon. What does this tell us about the difficulty of learning identity maps? Why do the authors use this to motivate ResNet? How do the authors of [13] exactly initialize the weights of their networks?

2 Reinforcement Learning

We discussed in the course that the TossingBot paper does not formally introduce the problem as an RL problem – but it could be formulated as RL problem. Please do so yourself:

- Rigorously define the state and action space
- Rigorously define the Markov Decision Process: What are the transition probabilities (you might not be able to specify this analytically, but what are they in principle?) Define a reward function!

– Assuming the robot collected a large batch data set of tuples $(p, I, \phi_g, \phi_t, s_g, \bar{p})$, where $s_g \in \{0, 1\}$ is the grasp success, and, if $s_g = 1$, $\bar{g}$ is the true landing box. Additionally you might store all details of the motion and robot control signals. (Did I miss something?) How could you train a Q-function?

3 End-to-end Training

The TossingBot mentions end-to-end training. However, at each end there are parts of the system not jointly optimized. Try the following:

a) Assume the analytic physics model for predicting the 'default' throw velocity has an uncertain parameter – say the gravitational constant g would be uncertain and you want to learn it. Derive all necessary equations to learn g end-to-end and jointly with the rest of the system (perception, grasp, and throw network). That is, embed the analytical prediction model in the computation graph and propagate gradients through!

b) Equally, assume there are free parameters on the motion execution side (parameters of the motion generation system). You also want to learn those end-to-end jointly with the rest of the system. Sketch how you could formulate this and how to propagate gradients also to these parameters.