Robot Learning
Weekly Exercise 11

Marc Toussaint & Wolfgang Hönig

Learning & Intelligent Systems Lab, Intelligent Multi-Robot Coordination Lab, TU Berlin

Marchstr. 23, 10587 Berlin, Germany

Summer 2024

1 Literature: Neural-Swarm2

Here is the paper we discussed in the lecture that uses (and extends) deep sets for a control problem that arises in multi-robot aerial swarms [1]:

G. Shi, W. Honig, X. Shi, Y. Yue, and S.-J. Chung. Neural-swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions. *IEEE Transactions on Robotics*, 38(2):1063–1079, Apr. 2022. doi:10.1109/TRO.2021.3098436

The paper is an extension of the NeuralLander paper to the multi-robot case we discussed in exercise 8.

Questions:

a) How does the dataset exactly look like? How was the data obtained? What sensing/measurement capabilities were needed to obtain such data?

Raw dataset: synchronized flights of multiple UAVs with information about motor commands, acceleration, and orientation.
Preprocessed dataset: relative state and kind of neighbors; labelled f_A
Data collection: Random flights with a simple collision avoidance module; curriculum data collection to also get data with closer proximity.
Capabilities: IMU (accelerometer, gyro)

b) Write down pseudo code on how one can use SGD or Adam and train a 2-group permutation-invariant function using the heterogeneous deep sets proposed in (9).

We want to approximate functions $z = f_{x/y}(\mathcal{X}, \mathcal{Y})$. In total, we have to train 4 NN: $\rho_x, \rho_y, \phi_x, \phi_y$. The dataset is initially $D = \{type, \{x\}_{i=0}^{N}, \{y\}_{i=0}^{M}\}\}$, i.e., the type ($x$ or y) and then a number of neighbors for each type.

```
# prepare data to have batches with same type, N and M
group_data = group_by_key(D, "type_N_M")
data = batch_data(group_data)
# this is the regular training loop!
for z, type, X, Y in data:
    optimizer.zero_grad()
    outputs = model(type, X, Y) # (9)
    loss = loss_fn(outputs, z)
    loss.backward()
    optimizer.step()
```

c) Consider the use-case of motion planning (Fig. 6). Explain how the neural network is applied inside the motion planner.

[1]A shorter and perhaps easier to follow earlier work is G. Shi, W. Honig, Y. Yue, and S.-J. Chung. Neural-swarm: Decentralized close-proximity multirotor control using learned interactions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3241–3247, Paris, France, May 2020. IEEE. doi:10.1109/ICRA40945.2020.9196800

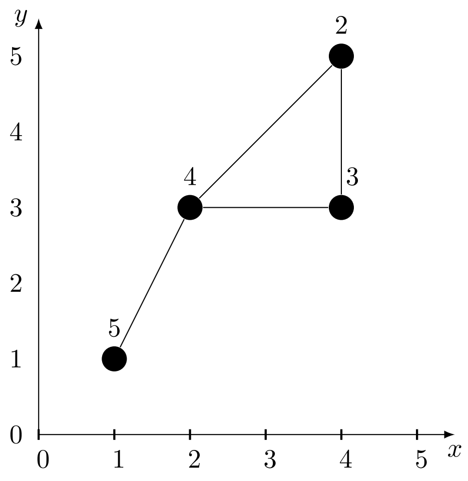
The motion planner has two parts: kinodynamic RRT and SCP. In both parts, the neural network is used as part of the step function/dynamics model.

d) In the considered examples for K-group permutation invariant functions, K is relatively small (4 in the paper). Consider the case where K is large or unknown, for example if we are able to measure the size of the neighboring robot (a real value). How could learning be applied in this case?

Heterogeneous deep sets do not work in this case, since K needs to be known apriori and defines the number of neural networks to be learned. One option would be to use a "regular" (non-heterogeneous) deep set, where the networks (ϕ and potentially also ρ) receive the other measured quantity as input/conditioning as well.

2 Encodings for Environmental Monitoring

Consider a team of robots that is spatially distributed as shown below. In the figure, circles are robots, the numbers are their associated measurements (such as temperatures), and lines indicate the existence of a communication link. The goal is to find the minimum of their sensor measurements. In this question we will explore various concrete encodings for such problem.



a) First consider the abstract, centralized setting with function $f(\mathcal{X}) = \min_{x\in\mathcal{X}} x$, where $\mathcal{X}$ is a set of real numbers. In other words, the function takes one or more numbers as input and returns the smallest element of these numbers. Recall that the deep set

$$f(\mathcal{X}) \approx \rho\left(\sum_{x\in\mathcal{X}} \phi(x)\right) \tag{1}$$

should be able to approximate this function. Provide concrete (differentiable) functions for ρ and ϕ for this case. Hint: You can find some inspiration in the original Deep Set paper or the paper from question 1.

See `https://en.wikipedia.org/wiki/Smooth_maximum` for some smooth functions. In the Crazyswarm2 paper, the Boltzmann maximum was used. In this context we have:

$$\phi(x) = [xe^{\alpha x}, e^{\alpha x}]^\top$$
$$\rho(z = [a, b]) = \frac{a}{b}$$
$$\lim_{\alpha\to-\infty} \rho\left(\sum_{x\in\mathcal{X}} \phi(x)\right) = \lim_{\alpha\to-\infty} \frac{\sum_x xe^{\alpha x}}{\sum_x e^{\alpha x}} = \min_x x$$

b) Now assume the case where robots have a limited communication radius. One example is shown in the figure, where the lines show communication links. Define the `Aggregate` and `Update` functions of a simple message-passing neural network.

Demonstrate in the example above, how the node at $(1, 1)$ computes the minimum value.

Aggregate$(\{h_m^{(l)} : m \in \mathcal{N}(n)\}) = \min_{h_m^{(l)}}$ (or the differentiable version defined in a)
Update$(h_n^{(l)}, z_n^{(l)}) = \min(h_n^{(l)}, z_n^{(l)})$ (or the differentiable version defined in a)

For this network, we would need at least 2 such layers to converge.
Concrete example:
Layer 0:
$(1,1) : \min(5, \min(4)) \to 4$
$(2,3) : \min(4, \min(5,2,3)) \to 2$
$(4,3) : \min(3, \min(2,4)) \to 2$
$(4,5) : \min(2, \min(4,3)) \to 2$
Layer 1:
$(1,1) : \min(4, \min(2)) \to 2$
$(2,3) : \min(2, \min(4,2,2)) \to 2$
$(4,3) : \min(2, \min(2,2)) \to 2$
$(4,5) : \min(2, \min(2,2)) \to 2$

c) How could a CNN be used for the case with limited communication radius? Be specific about the layers the CNN should have.

Discretize the visible nearby state space and use a single channel. Set the value to a very large number if no neighbor is present and the minimum value of visible neighbors in each grid cell. Use multiple ”min-pooling” layers (down to a single pixel) to aggregate to the desired value.

d) For the use-case outlined above, what are advantages and disadvantages of the three encodings (Deep Sets, GNN, CNN)? Consider both small (=few neighbors) and large (=many neighbors) cases.

	Method	Pros	Cons
Few robots	Deep Set	easy to train, fast inference	only a single hop
	GNN	easy to train, fast inference	requires some sort of synchronization
	CNN	fixed inference time	only a single hop

In the many neighbor case, there is a scalability issue with deep sets and GNNs (for both training and inference).

References

[1] G. Shi, W. Honig, X. Shi, Y. Yue, and S.-J. Chung. Neural-swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions. *IEEE Transactions on Robotics*, 38(2):1063–1079, Apr. 2022. doi:10.1109/TRO.2021.3098436.

[2] G. Shi, W. Honig, Y. Yue, and S.-J. Chung. Neural-swarm: Decentralized close-proximity multirotor control using learned interactions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3241–3247, Paris, France, May 2020. IEEE. doi:10.1109/ICRA40945.2020.9196800.