HUMAN-ROBOT INTERACTION

ROBOT LEARNING

J. Verhaert
S1047220
Master Artificial Intelligence | Intelligent Technology
joost.verhaert@student.ru.nl

T. Gelton
S4480783
Master Artificial Intelligence | Intelligent Technology
thijs.gelton@student.ru.nl

1 Introduction

2 Methods and algorithms

The two algorithms that were used for the task of robot learning were a Feedforward Neural Network (FFN) and a Mixture Density Network (MDN). Regarding the popularity of deep learning, the FFN is widely used and does not need an indepth explanation. In short, it allows the modelling of non-linear distributions and performs exceptionally well on unstructured data. A MDN is a network that allows to make predictions under uncertainty. Meaning that a learned MDN outputs Gaussians and these are then used to sample from. This allows it to model 1tomany relations, where the input has multiple possible outputs. Bishop[1], shows an inverse sinusoid and how a MDN is able to model this where a FFN is not.

However, for any algorithm the first step to take is to gather data. For this we re-used some of Nao's parts from the first assignment, i.e. the keyboard functionality and the segmentation of the ball in the image. In a step-by-step manual fashion, the arm is first moved, then the ball is manually aligned and then 'E' is pressed to create a new data point. This CSV file is imported into a PyTorch Dataset object (see notebook) and deduplicated to make sure that every X and Y coordinate pair of the ball only has one corresponding roll (θ_1) and pitch (θ_2) angle.

After importing the data, which consisted of 72 samples after deduplication, the MDN was trained first. For this we used the Adam optimizer and a Negative Log-Likelihood (NNL) loss function. The Adam optimizer needed an epsilon of $1e-05$ to compensate for the often occurring vanishing gradient when using the NNL loss function. For this very reason, using a train and validation set was infeasible. The size of the dataset strongly impacted this vanishing gradient and thus all samples were needed in order to train the MDN model. Only one sample was reserved for testing. The network itself has two inputs that are connected, via 1 hidden layer of 6 nodes, to the pi-(π), sigma-(σ) and mu(μ) layer. Both the MDN and FFN use Tanh activation function. The number of Gaussians, and thus the number of π, σ and μ outputs, depends on the domain and it is up to the designer to specify this number. In the results (page 3) section we will compare the differences in the number of Gaussians.

To be able to compare the FFN and MDN, only one hidden layer with 6 nodes was used for the FFN as well. The network was then trained using the Mean Squared Error (MSE) and the same Adam optimizer as with the MDN.

One possibility with deep neural networks is to have the network extract the ball from an image itself instead of using the distilled coordinates by segmentation. In order to do so, the first layer(s) should be substituted by convolutional layers. For this controlled toy experiment this would not be necessary.

3 Results

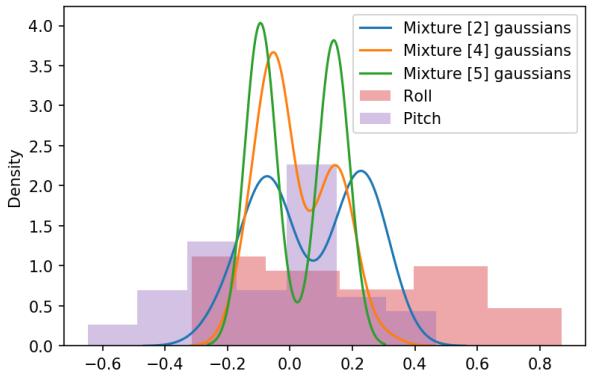

Figure 1: Mixture Models at different number of Gaussians and the target distribution. Input to each model was the mean of the x and y coordinates of the ball.

Eventually we chose to use 2, 4 and 5 gaussians and compare the results. The learned distributions can been seen in figure 1. In this specific case, different combinations of θ_1 and θ_2 could not lead to the same position and thus the MDN would quickly converge to a point where the outputted π would become a one-hot encoded vector. Formulated differently, the model would not be doubting about a multitude of distributions describing the actual distribution best, it would only have to converge to one to best model it. This meant that the Mixture Model would converge to only one Gaussian. When inspecting figure 1, you can clearly see that there are two means for each mixture and that number of gaussians correlate to the size of the covariance. It seems that the amount of Gaussians gave the model more freedom to become more specific with the distribution as the amount went up.

Aside from inspecting Nao's behavior the FFN and the different MDN models were tested against the validation sample (N = 1). In figure 2 you can see that the best performing model is the MDN with 5 gaussians. Do note that this is not representative as it is only 1 sample, but in order to be able to train the MDNs without the vanishing gradient nearly all samples were needed for training. However, this one sample is unseen for all networks.

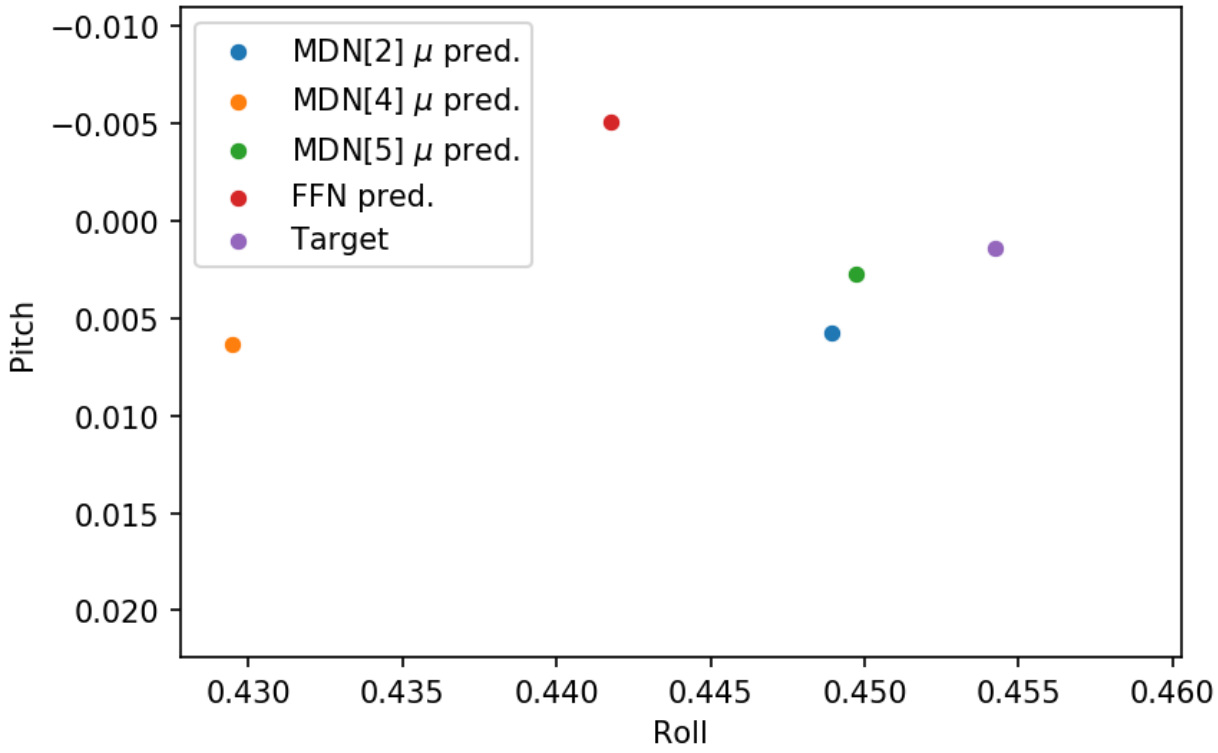

Figure 2: Predictions of the MDN models and the FFN model on the validation sample (N = 1).

A factor that was impossible to capture by still images, was the actual performance of all models in the simulator. Eventually, the FFN seem to outperform the MDNs as the MDN caused some oscillation whereas the FFN tracked the ball in a seemingly perfect and smooth manner. The MDN with 4 gaussians showed the best tracking capability by a small margin compared to 5 gaussians.

4 Conclusion

There are multiple conclusions to draw from this experiment. First and foremost it is important to state that learning in intelligent systems is indispensable. In order to create an autonomous intelligent system, learning is key ingredient. However, for autonomous learning this present study is inadequate, as this was done in a supervised manner. A robot in an unknown and hostile environment has to be able to orient itself, explore its environment, recover from failures and must generalize well in order to solve a multitude of different tasks[2].

Still, the difficulty of a task can be seen as a spectrum and each task will have its own solution. For the ball tracking behavior by (Nao's) hand both the models showed promising results. As described in section 3, the problem at hand only required Nao to use two degrees of freedom to learn the *Inverse Kinematics*. Contradictory to the example by Bishop[1] (which also uses two degrees of freedom), different θ_1 and θ_2 values could not result in the same outcome of the arm. Therefore, comparing these two models in this study would be senseless as both have their own usefulness and referring back to the beginning of this paragraph: are a solution to a specific task. A better use of the MDN is displayed by [3], where they use two robotic arms that perform under 7 degrees of freedom (DOF) that need to avoid each other while touching an air bound thrown object. Here, the MDN is used to compute different *feasible intercept points* for the thrown object in respect to the arm.

Eventually, even when taking into account that both of us already understood the FFN, the MDN still seemed to be a more difficult concept to grasp and to implement. Thus concluding, for this study the FFN would be our model of choice for the problem. If the problem would have more degrees of freedom, then the MDN would be the better choice.

References

[1] C. M. Bishop, "Mixture density networks," 1994.

[2] S. Thrun and T. M. Mitchell, "Lifelong robot learning," *Robotics and autonomous systems*, vol. 15, no. 1-2, pp. 25–46, 1995.

[3] S. S. Mirrazavi Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1205–1232, 2018.