Neural Networks
Group 04
Andrea Jemmett (2573223)
Bas van der Drift (1707507)

Final Assignment

The main Matlab script for this assignment can be found attached as 'final.m'.

Dataset Enhancement

As requested we augmented the size of the provided dataset with 35 more agents based on the paths of the 35 already present. For each of these paths (a vector of x-y coordinates) we generated a new path by rotating the path vector around the source of panic by 0.05 radians. By doing this we maintain the same path behaviour of the original agent but through a different "ray" irradiating from the shouting individual.

Before training we removed the mean from each agent's path and from the source of panic.

Network Architecture

We have chosen to tackle this regression problem with a Radial Basis Function Neural Network because those kind of networks are often used for regression tasks with good results. The first design decision we made was about the architecture of the network; we decided to feed the net with the coordinates of the agent at a given timestep and its (euclidean) distance from the source of panic (the shouting individual) and expect as output the coordinates of the agent at the next timestep. In addition to this configuration we decided to try to include the angle in radians between the X axis and the line that crosses the coordinates of the agent with the source of panic. We compare the results of both input configurations to see which one performs better than the other. Another important choice that needs to be made in the context of RBF networks is the number of hidden nodes; because there is no way of knowing a priori the best number of hidden units cross-validation is used to find the best number of them.

To sum up we tested a number of RBF networks with gaussian hidden units, 3 or 4 input nodes, a number of hidden nodes (precisely from 4 to 100 with a step of 4, for a total of 25 cases) and 2 output nodes. In total we tested 25+25=50 different networks.

Error Measure and Validation

We have decided to test the learned network for three tasks: predict the position of the agent at the next timestep, at the next three timesteps and at the next 46 timesteps. We do this to be able to say which settings work best for different depths of forecast. Because this task can be seen as a time-series prediction problem and there are not so many training data, we decided not to use a 10-fold cross-validation but a different one. For each agent we have 47 timesteps, so we implemented the leave-47-out cross-validation as a "leave-1-agent-out" cross-validation in order to be able to test the prediction capabilities of the network for the full path prediction task; that is, we leave an entire agent's path as test set. So to predict for example the next 46 positions of the test agent we feed the starting coordinates, the distance from the source of panic and optionally the angle to the neural network and we use the output as a starting point for the prediction of the next position. After the prediction is completed, we can compute the error of it. We do this by computing the euclidean distance between each point in the predicted path and the respective one in the original path (this is the raw error), then we calculate the Root Mean Squared Error of those distances. Clearly the lower the RMSE, the higher is the similarity between the predicted path and the original one.

We repeated the training and testing process using each single agent as test set and the averaged the results (shown in the next section). To cope with the randomness of the standard K-mean random initialization we tested each agent five times and that averaged the errors.

Results

As can be seen from Figure 1 and 2 the number of hidden nodes does not have much effect on the resulting error.

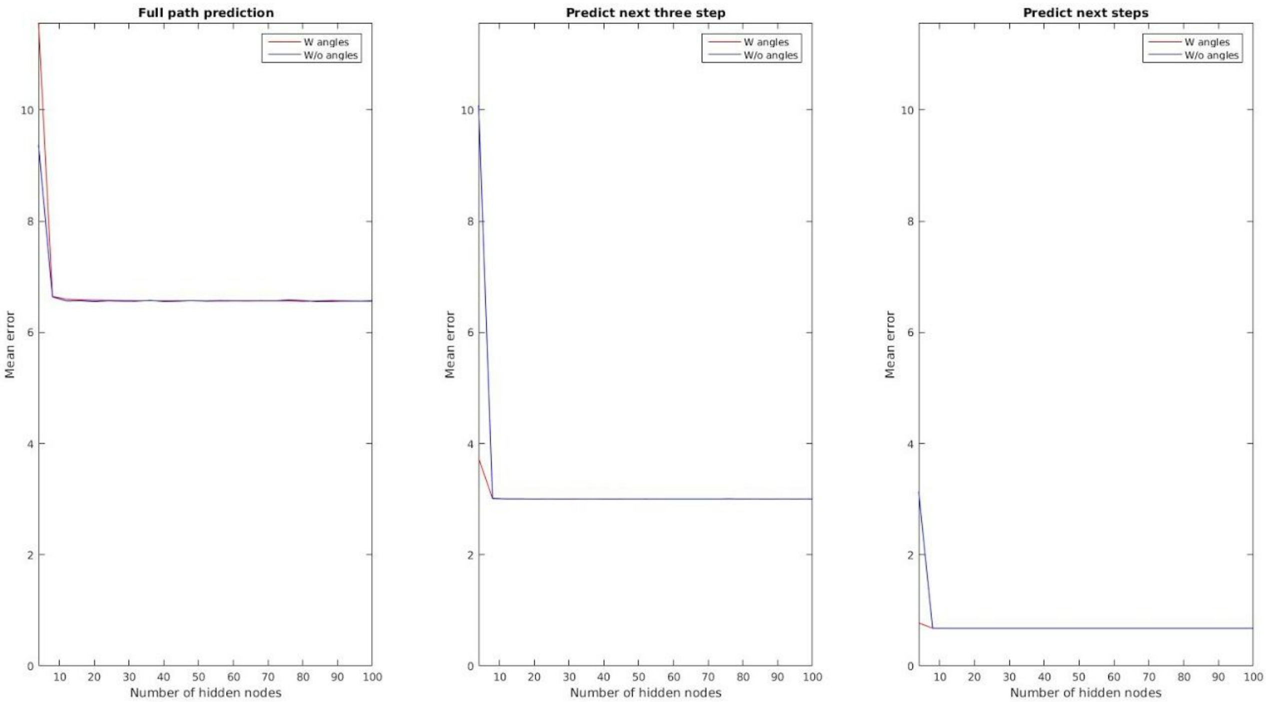

Figure 1. The resulting mean RMSE for the different kinds of prediction on varying network hyperparameters

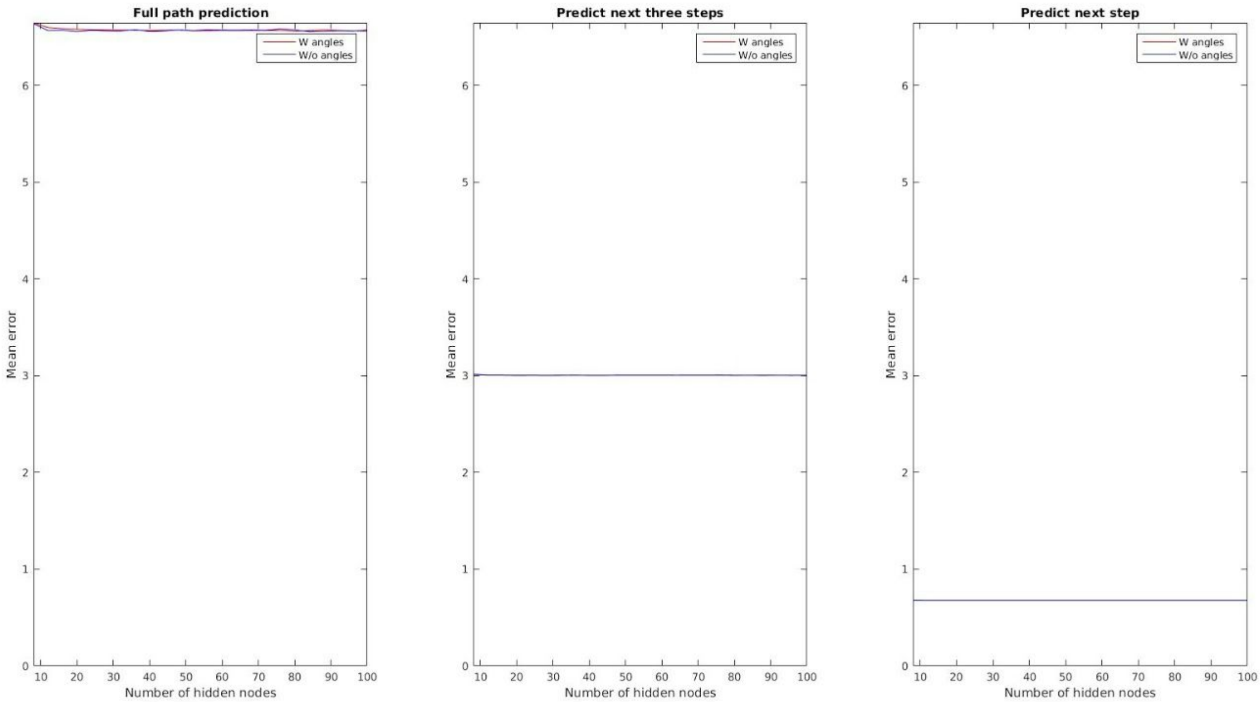

Figure 2. Same plot as before but without the results for 4 hidden units to highlight the flat trend

From Table 1 in the Appendix section, which contains the same data used to create the plots above, it is possible to see how the mean error over the number of hidden nodes used presents slightly different results for the three different type of prediction. We see that using also angles as input data performs on average better for the next three and next step tasks (respectively 3.0333 and 0.6805 with angles and 3.2872 and 0.7748 without). For the full path prediction task, the network trained without angles reports a mean error of 6.6809 which is better than the mean error with angles (that is 6.7721). Moreover from the same table it is possible to see how for the networks trained with angles the error for the full path prediction task decreases almost monotonically for higher number of hidden units; this behaviour is not present for the networks trained without angles and the error has a more oscillating behaviour.

In Table 2 we present the networks ranked using the mean RMSE for the full path prediction task. It is noticeable how networks trained with angles show a better performance for higher numbers of nodes, instead the networks trained without angles rank high also with for example 20 (2^{nd} position vs. 21^{st}), 40 (3^{rd} position vs. 12^{th}) and 32 (5^{th} position vs. 16^{th}).

Future Work

Two main improvements can be carried on to try to boost the performance of the network. One is to use knowledge about previous positions as input to the network; that is for timestep *i* we feed the network with, (x_i, y_i), (x_{i-1}, y_{i-1}), (x_{i-2}, y_{i-2}) and the distance and/or the angle of the last position, where (x_i, y_i) are the coordinates of the agent at timestep *i* (in this example we incorporated knowledge about the positions two steps behind the actual). Another possible improvements concerns the K-means clustering algorithm used to train the hidden layer of the RBF network. Our implementation uses random initialization, but there are a lot of improved algorithms just for that purpose. One is K-means++ that tries to initialize the centers so that the *k* initial clusters' centers are spread out as much as possible. The first one is chosen randomly and the remaining *k-1* are chosen from the remaining data points with probability proportional to its squared distance from the point's closest existing cluster center.

Appendix

Table 1						
	With angles			Without angles		
Hidden nodes	**Full path**	**Next three**	**Next step**	**Full path**	**Next three**	**Next step**
4	11.568	3.7389	0.77486	9.3694	10.085	3.13
8	6.6455	3.0099	0.6771	6.6368	3.0146	0.67808
12	6.5961	3.0058	0.6767	6.5651	3.0041	0.6767
16	6.5847	3.0049	0.67665	6.5687	3.0043	0.67668
20	6.5779	3.0043	0.67659	6.5547	3.002	0.67637
24	6.5747	3.004	0.67657	6.5673	3.0033	0.67651
28	6.5721	3.0037	0.67655	6.5621	3.0023	0.67641
32	6.5701	3.0039	0.6766	6.5606	3.0028	0.6765
36	6.5689	3.0037	0.67658	6.5744	3.0045	0.67671
40	6.567	3.0034	0.67654	6.5562	3.002	0.67636
44	6.5686	3.0037	0.67658	6.5609	3.0024	0.67641
48	6.5705	3.0037	0.67657	6.5714	3.0042	0.67671
52	6.562	3.0029	0.67648	6.5652	3.0044	0.67674
56	6.5632	3.0034	0.67655	6.5723	3.0039	0.67663
60	6.5654	3.0029	0.67647	6.5694	3.0045	0.67674
64	6.5641	3.0032	0.67654	6.5684	3.0037	0.67664
68	6.5656	3.0036	0.6766	6.5703	3.0046	0.67677
72	6.5647	3.0034	0.67657	6.5681	3.0041	0.67667
76	6.5668	3.0035	0.67658	6.5822	3.0052	0.67679
80	6.5587	3.0024	0.67641	6.5721	3.0037	0.67659
84	6.5647	3.0036	0.67659	6.5546	3.003	0.67655

88	6.5737	3.0045	0.67672	6.5582	3.0027	0.67648
92	6.5672	3.0037	0.67662	6.5627	3.0031	0.67652
96	6.5627	3.003	0.6765	6.5608	3.0027	0.67644
100	6.5594	3.0029	0.6765	6.5701	3.0037	0.6766
mean	**6.7721**	**3.0333**	**0.6805**	**6.6809**	**3.2872**	**0.7748**
std	**0.9992**	**0.1470**	**0.0197**	**0.5603**	**1.4163**	**0.4907**

Table 2

With angles		**Without angles**	
Hidden nodes	Mean RMSE	Hidden nodes	Mean RMSE
80	6.5587	84	6.5546
100	6.5594	20	6.5547
52	6.562	40	6.5562
96	6.5627	88	6.5582
56	6.5632	32	6.5606
64	6.5641	96	6.5608
72	6.5647	44	6.5609
84	6.5647	28	6.5621
60	6.5654	92	6.5627
68	6.5656	12	6.5651
76	6.5668	52	6.5652
40	6.567	24	6.5673
92	6.5672	72	6.5681
44	6.5686	64	6.5684
36	6.5689	16	6.5687
32	6.5701	60	6.5694
48	6.5705	100	6.5701
28	6.5721	68	6.5703
88	6.5737	48	6.5714
24	6.5747	80	6.5721
20	6.5779	56	6.5723
16	6.5847	36	6.5744
12	6.5961	76	6.5822
8	6.6455	8	6.6368
4	11.568	4	9.3694