EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Computer Science Department
Neuro-Cognitive Modeling

Summer Term 2022

Recurrent and Generative Neural Networks

Exercise Sheet 04

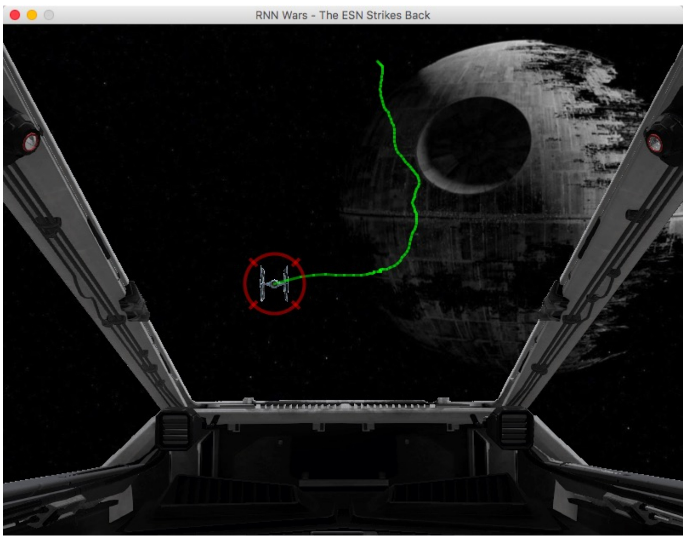
Release: June 3, 2022 Deadline: June 16, 2022 (23:59 pm)

General remarks:

- Download the file `exercisesheet04.zip` from the lecture site (ILIAS). This archive contains files (Java classes) and libraries, which are required for the exercises.
- **Add a brief documentation (pdf) to your submission**. The documentation should contain protocols of your experiments, parameter choices, and a discussion of the results.
- When questions arise start a forum discussion in ILIAS or contact us via email:
  Sebastian Otte (`sebastian.otte@uni-tuebingen.de`)

Introduction

This exercise sheet addresses *Echo State Networks* (ESNs) [1] as part of a neural network based missile control system within a simple 3D space combat simulation. The goal is predict the flight trajectory of the target in order to estimate, where it will be (most likely) in the future.



Excercise 1 – Movement Prediction with Echo State Networks [80 points]

(a) ESN Implementation [40 points]

Implement an ESN learning algorithm. The already implemented RNN from the previous exercise sheet can be reused here. Note that the output feedback can be simulated using the input layer.

The required pseudo inverse calculations can be accomplished using the *Efficient Java Matrix Library* (EJML)[1]. It is recommended to implement the method from [2], which is based on *Differential Evolution* (DE) [3].

A DE implementation with an example can be found in the archive `exercisesheet04.zip`. Nonetheless, you can also implement a more traditional ESN learning method, e.g., randomly generating reservoirs with spectral radius normalization and output feedback scale [4]. In the latter case, learning will perhaps require thousands of trials until success.

(b) Data Acquisition and Training [20 points]

Use the raw simulation to generate a flight trajectory of a certain length covering washout phase, training phase, and test phase. It is advisable to learn the position relative to the origin of the enemy's trajectory using `getRelativePosition()`. The best point to record the data is within the method `simulationStep` of the class `AIMComputer`. This method is called after every simulation step.

Now use your ESN implementation to find an ESN that works well on your data trajectory and report the achieved test results using the *Root Mean Square Error* (RMSE). Note that a good working ESN should be able to predict the trajectory correctly for at least 500 time steps into the future (without teacher forcing). Hint: You will most likely need an output feedback scale of $\approx 10^{-8}$ as reported in the literature [4, 2].

To start the simulation run `main` within `SpaceSimulationMain`. The GUI can be controlled via keyboard: locking/unlocking target (l), launching missile (enter), manual missile control (left, right, up, down), pause (p), reset (r).

(c) Motion Prediction [20 points]

Integrate a trained ESN module in the simulation setup within the class `AIMComputer` by replacing the dummy method. When done right, a green trajectory will appear representing the predicted future trajectory of the RNN, which, optimally, the enemy ship will follow precisely.

References

[1] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks," Fraunhofer Institute for Analysis and Information Systems AIS, Sankt Augustin, Germany, Tech. Rep. GMD Report, 148, 2001.

[2] S. Otte, M. V. Butz, D. Koryakin, F. Becker, M. Liwicki, and A. Zell, "Optimizing recurrent reservoirs with neuro-evolution," *Neurocomputing*, vol. 192, pp. 128–138, Jun. 2016.

[3] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[4] D. Koryakin, J. Lohmann, and M. V. Butz, "Balanced echo state networks," *Neural Networks*, vol. 36, pp. 35–45, 2012.

[1] `http://ejml.org`