Lecturer: Henning Sprekeler

Learning spatial representations with Slow Feature Analysis

The report for this project (no more than 4 pages of text with discussion and interpretation as a PDF file) as well as the source code should be handed in no later than **July 31st** through the Moodle interface.

We encourage you to meet with the project instructor as regularly as possible, but at least three times before deadline, to discuss both your advances and hurdles in the project. In addition, if you have any questions you are encouraged at any time to send an email to your project supervisor.

1 Summary

The goal of this project is to study how Slow Feature Analysis (SFA) establishes spatial representations similar to those found in the rodent hippocampus [Franzius et al., 2007]. Because visual data is very high-dimensional, we concentrate on a computationally simpler sensory system that is vaguely inspired by the echolocation system bats use to locate their prey. You will implement a simple movement pattern of a virtual "bat" in a rectangular room, emulate its sensory input and apply SFA and independent component analysis to learn spatial representations.

2 Detailed description

1. Start by implementing a random walk in a rectangular room. To encode the position in the room use the x and y coordinates. Because a pure random walk (where you add independent position increments at every time step) takes a long time to cover the full room, you should implement some kind of inertia to make the path smoother and more realistic (e.g. by picking random speed and turning values and smoothly interpolating between them). To simplify the problem you can assume that the bat bounces off when it hits the wall (e.g. by mirroring the velocity vector). Plot the trajectory and check that the room is evenly covered for very long trajectories.

2. Now create the sensory input the bat receives during the random walk. To this end, emulate the sonar-like ability of the bat that allows to measure the distance to surrounding objects. Give your bat N sensors that measure its distance to the wall in N different directions that you can specify. You can use the code in *geometry.py* to find points of intersection between the "ultrasound ray" of the sensors and the walls of the room and use this to calculate the distance between the sensors and the walls. Make sure your bat never gets too close to the walls or you could end up with strange sensory data, because geometry.py suffers from some funny floating point rounding errors. To start out, let the sensors always have a fixed orientation with respect to the room, regardless of the bats current direction of travel. The distances returned by the sensors represent the N-dimensional sensory input of the bat.

3. Use SFA to extract slowly varying features from the sensory input. We encourage you to use the MDP library [Zito et al., 2009, http://mdp-toolkit.sourceforge.net], which includes an SFA implementation with a convenient user interface. As a nonlinear expansion you should use a polynomial expansion (have a look at MDP's `PolynomialExpansionNode`).

   As a starting point, use a rectangular room and $N = 2$ sensors with orthogonal orientations aligned to the walls (what do the sensory data represent in this case?). Set the degree of the polynomial expansion to 1. Visualize the outputs of the different SFA components by calculating them on a fine grid of locations in the room and plotting them as a color plot over position (i.e. the color at a certain position encodes the value of one SFA-output). What do you see? What changes when you make the room quadratic? Why?

4. Change the number N of sensors of the bat and set their orientation to random directions. Does this alter the dependence of the first two output signals on the bat's position?
   Hint: With too many sensors, the sensory signals can become linearly dependent. Use MDP's `WhiteningNode` to remove directions with zero variance (check out the `svd` and `reduce` options).

5. Now vary the degree of the polynomial expansion (try e.g. 3, 5 and 7). Plot the position-dependence of the different output signals of SFA. How do the learned features change as the expansion degree increases? What kind of representation does SFA learn for high degrees of the expansion? Change the proportions of the room from quadratic to strongly elongated. What is the effect on the learned spatial representation? Discuss potential similarities with and differences to spatial representations found in rodents (and bats, as well, by the way [Yartsev et al., 2011]).

6. Use a relatively high expansion degree and apply independent component analysis (ICA) to the first J SFA output signals (have a look at MDP's `CuBICANode`). Start by using all of SFA's output signals and plot the output signals of ICA as a function of position. Discuss the results and relate them to the spatial representations found in the rodent brain. Reduce the number of SFA output signals that are used as training data for ICA. Investigate and discuss the effect this has on the spatial representation after ICA.

7. Discuss what you would expect if you would apply SFA to the sensory data received by a different animal with a different sensory system.

8. Bonus: Change the way the sensory data are generated to take into account that the sensors rotate with the head direction of the rat. Plot and discuss the dependence of the SFA outputs on position for different head directions of the bat.

References

[Franzius et al., 2007] Franzius, M., Sprekeler, H., and Wiskott, L. (2007). Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Comput Biol*, 3(8):e166.

[Yartsev et al., 2011] Yartsev, M., Witter, M., and Ulanovsky, N. (2011). Grid cells without theta oscillations in the entorhinal cortex of bats. *Nature*, 479(7371):103–107.

[Zito et al., 2009] Zito, T., Wilbert, N., Wiskott, L., and Berkes, P. (2009). Modular toolkit for data processing (mdp): a python data processing framework. *Frontiers in Neuroinformatics.*