Universität Hamburg
Department Informatik, ABe NATS, WTM & TAMS
Wolfgang Menzel, Stefan Wermter, Jianwei Zhang

Contact: Cornelius Weber
Office F-233
Tel 42883-2537

(for groups focusing on the SOM)

Exercises to the module: Algorithmisches Lernen

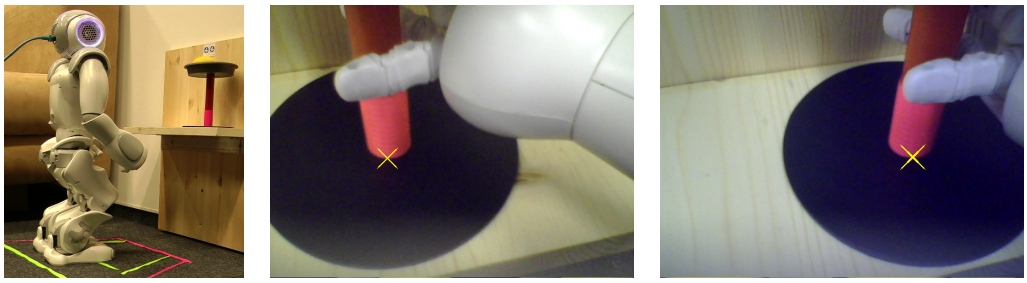
SS 2012 Sheet 8 (for groups focusing on the SOM)

Due: 06.06.2012

Task 8.1 Visuomotor Coordination

The topic of this exercise is a Nao robot that sees an object and generates the required arm posture for grasping that object. The object will be represented by coordinates (x,y) that are extracted from the camera image, and the robot must generate the matching arm joint angles (5 angles). We need only two visual coordinates to locate the object in 3D, because its position is constrained to a table surface.

The pictures show the scenario and two examples of the Nao camera image. The yellow cross in each camera image denotes the extracted point that defines the object's position.



The task is to learn the arm postures (5 coordinates) given the object coordinates from the camera image (2 values). A data set with 51 data pairs is in the MinCommSy: the file `visual.dat` contains the inputs, the file `angles.dat` contains the outputs.

To this end, you extend the SOM by another set of weights, which can generate outputs. The output weights are easily trained with the remaining algorithm without changing the existing structure. The update of the weight between map neuron k and output neuron j is as follows:

$$\Delta w_{kj}^{out} = \epsilon \, \eta_k \, (y_j - w_{kj}^{out})$$

Here, $\vec{\eta}$ is the map activation (i.e. the Gaussian around the winner unit), which is determined solely by the *input* data, as in the canonical SOM. $\vec{y}$ is the target output, which, apart from this learning rule, does not influence the learning algorithm elsewhere. After training, the network output is determined by the output weight vector $\vec{w}_{k*}^{out}$ of the winning unit $k*$ (the neighborhood interaction function $\vec{\eta}$ can be neglected, because, due to a small interaction width after training, it resembles a delta function).

(a) Visualize the input data and how the trained map lies in this input space. Does it seem to interpolate – and to extrapolate – well?

(b) Train the supervised SOM. Devide the data into training and test data. How do you quantify the error? Does the SOM perform well?

Wolfgang Menzel
(menzel@informatik...)
Stefan Wermter
(wermter@informatik...)
Jianwei Zhang
(zhang@informatik...)

Vorlesung:
Mi 8:15-11:45, F-009
Übungen:
Mi 12:15-13:45, F-009

Dienstgebäude
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg

Öffentliche Verkehrsmittel
Buslinien 181, 281
Haltestelle: Informatikum

(c) Finally, we visualize the visuomotor coordination on the Webots robot simulator. Please log in to any Linux computer in room F-234, which is mostly open. Go into the directory:
/informatik/isr/wtm/public/installations/webots

There, start:
./start_webots.sh

In the following graphical menu, choose "Your Project" and then via — Datei — Open World ... — open the following file:
/informatik/isr/wtm/public/installations/webots/projects/Algorithmisches Lernen/worlds/NaoGrasp.wbt

Now you have a scenario running in Webots. The Nao robot therein is controlled from any directory by a Python script, obtainable from the MinCommSy, that is started by:
`python 01_start_Nao.py`

In this Python script, you may first have to read the comment in the beginning and set the PYTHONPATH accordingly.

Then you insert into this script, line 33, the arm joint angles that you have obtained from the SOM output.

Hint: the "Revert" button in Webots restarts the scenario without having to restart Webots.

Compare the hand posture as seen from the robot in Webots with the pictures from the data set, which can be found in the following diectory:
/informatik/isr/wtm/public/installations/webots/projects/Algorithmisches Lernen/Pictures2

(d) You could also train the SOM in the opposite direction: inputs would be the arm angles, outputs the corresponding object coordinates. What could that be used for?

You could also solve this task with a multi-layer perceptron. Which differences would you expect?