Solution of Task Sheet 4

Nico Schmidt (nico.schmidt@uzh.ch)

Notes

The questions in this task sheet are very similar to the ones that you will find in the final examination. Please remember to write both your name and student ID number at the top of each sheet before it is handed in. You are encouraged to work in groups of two. If you do so, please hand in only one solution and write both names and student IDs on it. Please make an effort to write legibly.

When you hand in the solutions, please staple the sheets together. Alternatively, you can send your solution as one single PDF file by email to nico.schmidt@uzh.ch.

Questions

1. **Hebbian learning.** Consider a synaptic weight of a neuron with input and output signals denoted by $x_j(n)$ and $y_k(n)$. The adjustment applied to the synaptic weight at time step n is expressed in the general form 4P

$$\Delta w_{kj}(n) = F\left(y_k(n), x_j(n)\right)$$

where $F(*,*)$ is a function of both output and input signals.
The simplest form of Hebbian learning is described by

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$$

where η is a positive constant that determines the learning rate.

a) Explain how the weights are modified.

Hebbian learning strengthens the output in turn for each input presented, so frequent input patterns will have most influence in the long run, and will come to produce the largest output.

b) Explain the limitations of this implementation of Hebbian learning.

The weights keep on growing without bound and learning never stops.

c) How does Oja's rule solve the limitations of this implementation?

Oja's rule prevents the divergence of plain Hebbian learning by constraining the growth of the vector $\mathbf{w}$.

d) Write the implementation of Hebbian Learning including Oja's rule.

$\Delta w_{kj}(n) = \eta y_k(n)(x_j(n) - y_k(n)w_{kj})$

2. **Sanger's rule** The generalized Hebbian algorithm, or Sanger's rule, states that the output of a neuron is: 2P

$$y_i = \sum_{j=1} w_{ij} x_j \tag{1}$$

The weights are modified by using the following equation:

$$\Delta w_{ij} = \eta y_i \left(x_j - \sum_{k=1}^{i} w_{kj} y_k \right)$$

where x_j is the j^{th} component of the input vector x.
What is the relationship between Sanger's rule, Oja's rule and PCA?

Oja's rule attempts to implement a 1-output network that obtains the first principal component. In addition, Sanger's rule extracts the first m principal components individually in order of decreasing variance (in a network with m output nodes).

3. **DAC Architecture** A Khepera robot is equipped with six collision sensors, six proximity sensors, and six light sensors. Fig. 1 shows the arena in which the robot will be trained. Note: For each question (a,b,c) the neural network has to be trained from scratch (random weights at the beginning).

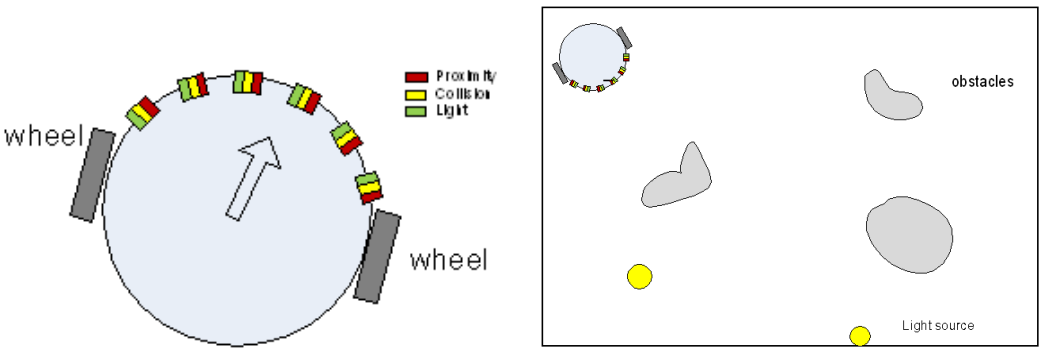
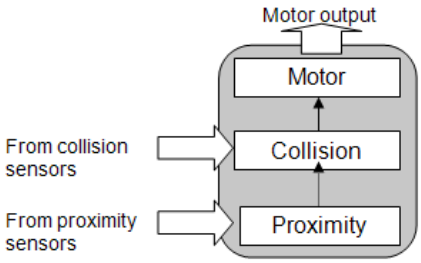


Figure 1: Robot and sensors

a) Design a DAC architecture whose emergent behavior (after the learning) will be to avoid the obstacles. Built-in reflex: if the robot hits with a collision sensor on the right, it will turn to the left and vice-versa. Draw a diagram or schematic of the network. Which kind of activation functions do you use? Explain how the architecture works. 2P

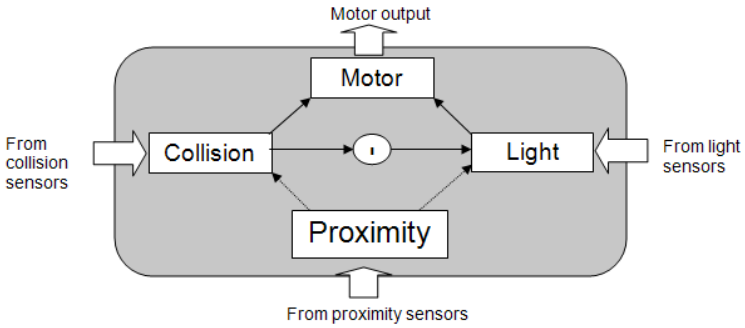
Hint: ignore the light sensors for now.

Step functions for the neurons connected to the collision sensors and linear functions for the ones connected to the proximity sensors. See script for further explanation...



b) Let's suppose there are a number of light sources near each wall of the corridor (learning phase). Built-in reflexes: the same as in question "a" plus a "light following" reflex. Enhance the previous design in order to allow the architecture to learn the "wall following" behavior. Which kind of activation functions do you use? Explain. 2P

The light sensors from the sides are connected to a new layer. See script for further explanation...



c) In order to achieve the "wall following" behavior, which kind of learning rule should be used between the proximity and collision layers. The following figures show the evolution of the connection weights between the proximity and the collision layers as the robot interacts with the environment. Please give an explanation of the diagrams shown below. Dark means high activation values. 2P

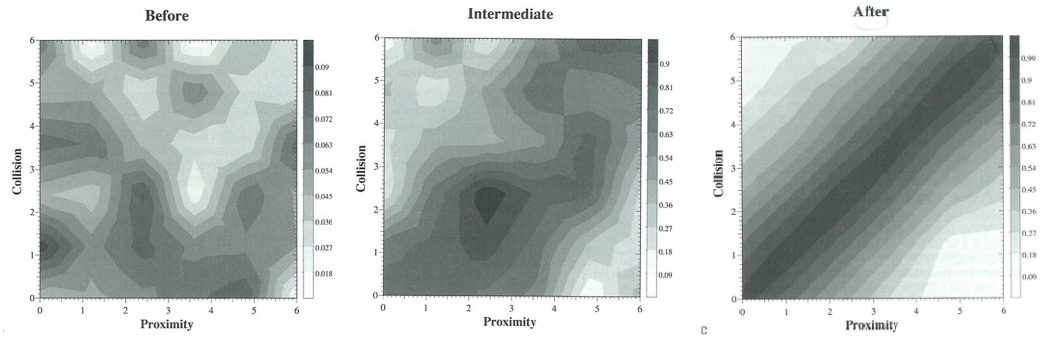


Figure 2: **left:** Weight pattern at the beginning of the trial. **middle:** Weight pattern after the robot has collided 10 times. **right:** Weight pattern after the robot has encountered a wall another 10 times

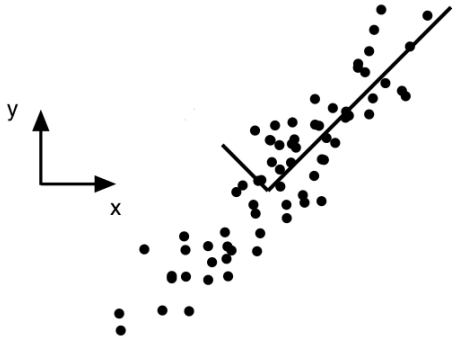
i) There is no apparent structure; the weights are randomly distributed.

ii) A diagonal pattern begins to emerge.

iii) The connectivity pattern has a clear diagonal structure reflecting the fact that neighboring proximity and collision nodes are simultaneously active. This correlation has been picked up by the Hebbian learning mechanism.

4. **PCA (Principle Component Analysis)**

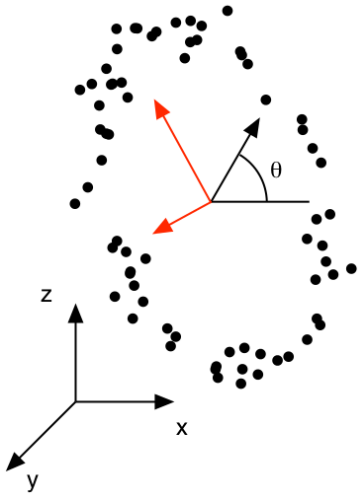
a) Sketch a 2D example of a data set and indicate the 1^{st} and 2^{nd} principle components that would be identified by PCA. 1P



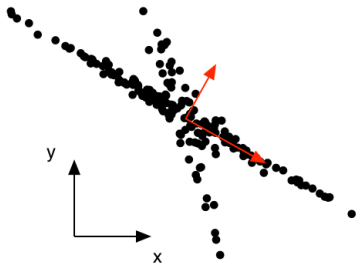
The longer line is the 1^{st} PC, the shorter one is the 2^{nd}.

b) In principle component analysis we made three assumptions about the structure of our data set. Explain each of them. For each of the three assumptions, sketch a counterexample where PCA would fail to extract the interesting features if the assumption does not hold for the data. 6P

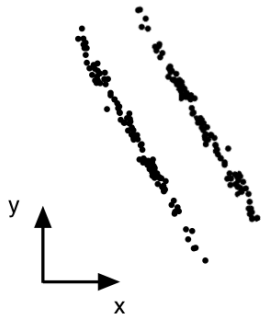
i) **Linearity.** The interesting structure lies on a linear subspace of the data.
Counterexample:



ii) **Orthogonality.** The different features are orthogonal to each other.
Counterexample:



iii) **Importance of Variance.** The interesting features lie along the direction of maximum variance.
Counterexample:



5. **Kohonen networks** The goal of this exercise is to get an intuition about how the learning in a Kohonen network depends on the various learning parameters: the learning rate, the size of the neighborhood and their respective decay rates. Use the Java simulator to build a 10^2 Kohonen network with two inputs (of type "line"). Use the training pattern "kohonnen.testpattern" that is provided with the exercise. Experiment with different values for these four parameters and explain your observations:

   a) Explain the effect of high/low learning rate, neighborhood radius and their respective decay rates on the resulting map. 4P

   The learning rate controls how strong the winning node (and its neighbors) are moved towards the input in each iteration. A too low learning rate doesn't let the map unfold and cover the space, whereas a too high learning rate makes the winner move a lot towards the input, which prevents convergence and is sensitive to outliers. Often it makes sense to start with a high learning rate to let the map unfold and use a decay < 1 to 'freeze' the map over time. The neighborhood radius defines how many neighbors are moved together with the winner. If it is too low, only the winner moves and the topology of the map will be lost. If it is too high, the map jumps back and forth (the winner is pulling its neighbors too much towards its new position) and prevents convergence. Again it is often good to start with high values and let it decay over time

   b) Which combination of values works best? 2P

Many parameters may work. These settings work quite well:
Learning rate: 0.9 with decay 0.9
Neighborhood radius: 10 with decay 0.9

Notes:

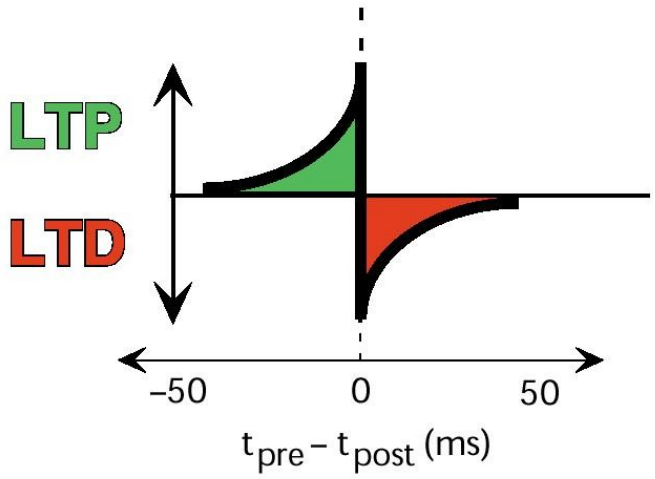
- To start the simulator from a command line, change into the "Neural-Networks" folder and type "java controller.Laboratory"
- Click "create" on the left of the main window to create a network.
- Click "open" on the right of the main window to load the training pattern.
- Decay rates for the learning rate and the size of the neighborhood work as follows: For a decay rate of 0.9 the learning rate for example will decay like this: $\eta_{t+1} = 0.9\eta_t$. The same applies for the size of the neighborhood.
- Remember to "reset the weights" after each trial (first press "change" and then "reset the weights")
- To start training, click the "unfold" button in the "learn control"-window.

6. **Spiking networks**

a) How does a high activation manifest itself in a typical artificial neuron and in a typical biological neuron? 1P

An artificial neuron with high activation has simply an output with a large magnitude that is sent along the weights to its followers. The magnitude (voltage) of the output of a biological neuron with high activation would typically not differ very much from its output with a lower activation. What makes the difference is the firing frequency which is higher when the activation is higher.

b) In the graph below, sketch the paradigm of spike-time-dependent plasticity (STDP) and explain it. What is meant by long-term potentiation (LTP) and long-term depression (LTD)? (Don't forget to label the axes!) 4P



In the spike time dependent plasticity paradigm, the change of the synaptic strength (weight) is dependent on the timing of the firing of the pre- and post-synaptic cells. If the time-difference between their spikes is slightly negative (pre fires shortly before post), the connection is strengthened (long-term potentiation, LTP). If it is slightly positive, the connection is weakened (long-term depression, LTD). If both fire with large time-difference, nothing changes.