Neuroinformatics — lab exercises manual

authors:
Eduard Bakstein (TA), Daniel Novák (Lecturer)
http://nit.felk.cvut.cz, Prague 2015

May 14, 2015

Abstract

This is supportive text for labs of Neuroinformatics course at Czech Technical University in Prague, Faculty of Electrical Engineering. Exercises for each task are provided in Matlab language.

We further recommend a book [**?**], which is nicely readible. Good summary is provided by [**?**]. Particular topics are covered by [**?**], [**?**] Some advance material cis cover by [**?**], [**?**].

Contents

0 Mathematical apparatus

We will introduce numerical apparatus, which will be applied during activity modelling of neurons.

0.1 Numerical sollution of differential equations

As in other sectors dealing with dynamic systems, we will use differential equations throughout this course. Due to the fact that their exact analytical solutions can often be difficult or even impossible, we introduce in this exercise the approximate numerical solutions.

Euler's method

Let's consider first order

$$\frac{\delta x}{\delta t} = f(x, t), \tag{0.1}$$

Euler's first order metoda consist of discretization $\frac{\delta x}{\delta t} = \frac{\Delta x}{\Delta t}$. Lets' take:

$$\begin{aligned} \Delta t &= t_2 - t_1 \\ \Delta x &= x(t + \Delta t) - x(t), \end{aligned}$$

then we can express eq (0.1) as

$$\frac{\Delta x}{\Delta t} = f(x(t), t)$$

finally

$$x(t + \Delta t) = x(t) + \Delta t f(x(t), t)$$

We can approximate solution by taking into account the slope of the line at that point. However, if the slope is dependent on t, this will be a very rough approximation. In this case, we can refine the solutions by other parameters of the Taylor series according to the formula:

$$x(t + \Delta t) = x(t) + \Delta t \frac{\delta x}{\delta t} + \frac{1}{2}(\Delta t)^2 \frac{\delta^2 x}{\delta t^2} + O, \tag{0.2}$$

where O represents all members of the higher orders. In the second order therefore we model in addition the slope curvature. This approximation should be closer to analytical solutions.

Runge-Kutta metoda

In the methods of higher orders we can additionally refine the solution by estimation not in the point x and $x + \Delta x$, but in the middle of this period - the so-called *„midpoint method"*. The resulting value should be more representative in a given interval and lead to a more accurate estimate. Moreover, if we do not have analytical expression of parameters of the Taylor expansion of higher orders, we can again estimate parameters of high orders using numerical methods.

The actual method of Runge-Kutta numerical integration is the 4th order, which combines estimation in the middle of the interval with a numerical estimate of higher order. It can therefore approximate solution of arbitrary functions. This leads to a higher computational cost, but - as we shall see - leads to very accurate estimates of the solution. Runge-Kutta method is implemented in Matlab function `ode45()`.

Exercise 0.1 Our task is numerical approximation of the following differential equation

$$\frac{dx}{dt} = t - x + 1, \tag{0.3}$$

initial conditions: x (0) = 1 using Euler's method (1 and 2 order) and Runge-Kutta methods.

The analytical solution of this equation has the form $x = t + e^{-t}$ [1]. Solve the equation in the interval $\langle 0, 5 \rangle$, choose $\Delta t = 0.02$.

---

[1] You can check e.g. at http://www.wolframalpha.com/, question: „solve differential equation x '= ... "

Task 0.1 (2 b) Plot the solution $x(t) = f(t)$ for $t = 0, \ldots, 2s$ using Euler's method of first and second order [2] (X_{Euler}) and plot into graph along with the analytical solution.

Task 0.2 (1 b) Solve the rask using Runge-Kutta x_{Runge} and add the solution to the same graph.
    Hints: function `ode45` has as first parameter a callback to the function, which desribes the equation to be solved. In this case the callback can be easily defined using so called *function handle* and *anonymous function*[3] as `ode_func = @(t,x,flag) 1-x+t;`

Task 0.3 (1.5 b) Plot the dependance of relative error of each numerical methods agains analytical solution. For example (for Euler's method) $(x_{Euler} - x_{exact})/x_{Euler} = f(\Delta t)$

Task 0.4 (1.5 b) Plot absolute error of each numerical method taking into consideration size of integration step $\Delta t \in (0.001, 1)s$ in time $t = 1s$.

---

[2] when solving the second order Euler method you can proceed in two ways. 1) Derivating analytic function $f(x,t)$. 2) estimate value of $f'(x,t)$ using the slope $f(x,t)$ between points t and $t + Deltat$

[3] see for example. http://www.mathworks.com/help/matlab/matlab_prog/creating-a-function-handle.html

1 Neuron models

In this part we will describe properties of action potential and behaviour of neurons.

1.1 Model of membrane and synapse: simpler case

Taking into accout that nerve impulses propagate through changes in electrical potential on the cell membrane, it seems natural to create the basic model as an equivalent electrical circuit. The simplest model, that we introduce in this exercise, is working only with one type of channel - leakega chloride channels (always opened) - and takes the form of an RC circuit, the resting battery, replacing the resting potential of the membrane (Nerst potential) . As we shall see, the response to the input excitation is far away from the the real action potential but serve us in understanding the basic concept.

Exercise 1.1 (RC model) Model memrabe behavior using RC model - see figure 1. Input membrane current I_{stim} is rectungalar signal $10pA$ lasting $20ms$. It is neccesary to convert the stimulating current I_{Cl} and I_C to the electrode surface A[4]. The electrode is stimulating and at the same time recording the membrane current (expressed in cm^2). Hence $I'_{stim} = I_{stim}/A \approx 10^{-11} \cdot 10^6 \approx 10^{-5}$. The mebrane parameters are the following:

capacity: $C_m = 1 \ \mu F/cm^2$,
conductance: $g_{Cl} = 0.3 \ ms/cm^2$,
time constant : $\tau = C_m/g_{Cl}$,
mebrane surface : $A \approx 1 \cdot 10^{-6} \ cm^2$
Nerns potential of Cl: $V_{Cl} = -68 \ mV$,
initial conditions: $V(0) = -68 \ mV$, $I_{Cl}(0) = 0 \ \mu A/cm^2$, $I_C(0) = 0 \ \mu A/cm^2$.

Task 1.1 (2 b) Plot dependence $V(t)(= \phi_{in} - \phi_{out})$, $I_{Cl}(t)$, $I_C(t)$ for these time interval $\langle 0, 40 \rangle ms$, $\Delta t = 0.01ms$
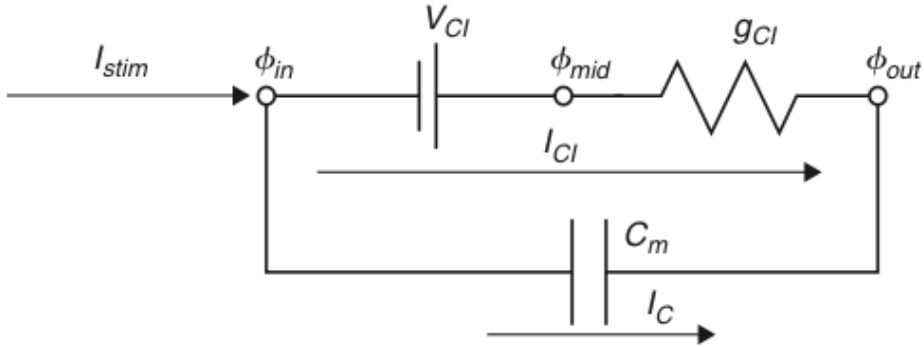
Figure 1: Membrane model with Cl leakage channel

Hence

$$I_C(t) \quad = \quad C_m \frac{dV}{dt}(t) \tag{1.1}$$

$$I_C(t) \quad = \quad \frac{I_{stim}(t)}{A} - I_{Cl}(t) \qquad \rightarrow I_{stim} = A \cdot I_C(t) + A \cdot I_{Cl}(t) \tag{1.2}$$

$$I_{Cl}(t) \quad = \quad g_{Cl}(V(t) - V_{Cl}) \tag{1.3}$$

$$\tau \quad = \quad \frac{C_m}{g_{Cl}} \tag{1.4}$$

---

[4]This is current density when we shoud use the following convention ϱ. However, we will stick to the terminology used during lectures I.

Combining the above equation we get

$$C_m \frac{dV}{dt} = \frac{I_{stim}(t)}{A} - g_{Cl}(V(t) - V_{Cl}) \tag{1.5}$$

$$\tau \frac{dV}{dt} = V_{Cl} - V(t) + \frac{I_{stim}(t)}{Ag_{Cl}} \tag{1.6}$$

$$\tag{1.7}$$

Euler's method (forward)

$$\tau \frac{V(j) - V(j-1)}{dt} = V_{Cl} - V(j-1) + \frac{I_{stim}(j-1)}{Ag_{Cl}} \tag{1.8}$$

$$V(j) = V(j-1) + \frac{dt}{\tau}[V_{Cl} - V(j-1) + \frac{I_{stim}(j-1)}{Ag_{Cl}}] \tag{1.9}$$

Euler's method (backward)

$$\tau \frac{V(j) - V(j-1)}{dt} = V_{Cl} - V(j) + \frac{I_{stim}(j)}{Ag_{Cl}} \tag{1.10}$$

$$V(j)(1 + \frac{dt}{\tau}) = V(j-1) + dt(\frac{V_{Cl}}{\tau} + \frac{I_{stim}(j)g_{Cl}}{Ag_{Cl}C_m}) \tag{1.11}$$

$$V(j) = \frac{V(j-1) + dt(\frac{V_{Cl}}{\tau} + \frac{I_{stim}(j)}{AC_m})}{(1 + \frac{dt}{\tau})} \tag{1.12}$$

Exercise 1.2 (EPSP model) The task is analysis of a model depicted in figure 2. This model incorporates so called „excitatory postsynaptic potential" (EPSP) simulating behavior of membrane dendrite of the postsynaptic's neuron after receiving excitation. Synapse is modelled by chanchagable conductance g_{syn}. In other words: the model contains additionally channels, which react to release to the neurotransmitter (literally they are opening, hence their conductance is decreasing - see result of eq (1.13)) with time constant $\tau_{syn} = 1 \; mS$. The parameters are the same like in the previous case. Stimulation current $I_{stim} = 0$, $V_{syn} = 10 \; mV$. In time $t = 1 \; ms$ the neurotransmitter is released, hence $g_{syn}(1 + \delta) = 1$. Initial conditions $V(1) = 0$, $I_{syn} = 0$, $g_{syn}(1) = 0$, $g_L = 1$.

$$\tau_{syn} \frac{dg_{syn}(t)}{dt} = -g_{syn}(t) + \delta(t - t_{pre} - t_{delay}) \tag{1.13}$$

Task 1.2 (0 b) Plot dependence $V(t)$, $I_{Cl}(t)$, $I_C(t)$, $I_{syn}(t)$. Explain trends and compare them with the previous task 1.1.
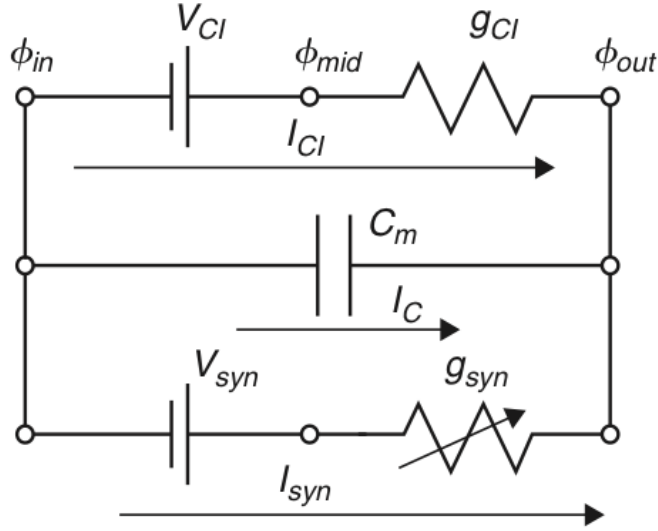
Figure 2: EPSP: Synapse model with leakage Cl channel

1.2 Mebrane and synapse modelling 2: Hodgkin-Huxley

Exercise 1.3 (Hodgkin-Huxley) The taks is analysis of Hodgkin-Huxley(HH) model depicted in figure 3. Compared to the previous models, HH incorporates variables which are dependent both on time model and voltage, hence Na^+ a K^+ channels are controlled by voltage. The model is based on the following set of eqautions
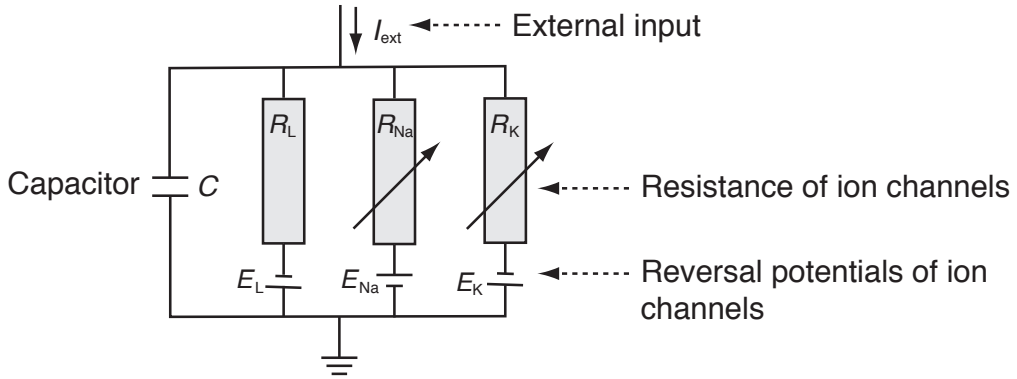
Figure 3: Hodgkin-Huxley model. Each branch represents resting potential a and resistivity of a particular channel: E_L, R_L - channels always opened („Leakage channels"), E_{Na}, R_{Na} - sodium channels controlled by voltage , E_K, R_K - calcium channels controlled by voltage.

Current in ion channels can be described by Ohm's law:

$$I_{ion} = \hat{g}_{ion}(V - E_{ion}) \qquad \left(= (V - E_{ion})/\hat{R}_{ion} \right), \tag{1.14}$$

where $\hat{g}_{ion}$ is the maximum conductance of ion channel

Next, we introduce auxilary variables dependent on voltage and time $n(V,t)$, $m(V,t)$, $h(V,t)$, resulting in the following conductances:

$$\hat{g_K}(V,t) = g_K n^4 \tag{1.15}$$
$$\hat{g_{Na}}(V,t) = g_{Na} m^3 h \tag{1.16}$$

6

Combining eq.1.16-1.14 we get

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = -g_{\mathrm{K}}n^4(V - E_{\mathrm{K}}) - g_{\mathrm{Na}}m^3h(V - E_{Na}) - g_{\mathrm{L}}(V - E_{\mathrm{L}}) + I_{ext}(t) \qquad (1.17)$$

Next, we define time constants

$$\tau_{\mathrm{n}}(V)\frac{\mathrm{d}n}{\mathrm{d}t} = -[n - n_0(V)] \qquad (1.18)$$

$$\tau_{\mathrm{m}}(V)\frac{\mathrm{d}m}{\mathrm{d}t} = -[m - m_0(V)] \qquad (1.19)$$

$$\tau_{\mathrm{h}}(V)\frac{\mathrm{d}h}{\mathrm{d}t} = -[h - h_0(V)] \qquad (1.20)$$

For each variable we get

$$\frac{dx}{dt} = -\frac{1}{\tau_x(V)}[x - x_0(V)], \qquad x \in \{n, m, h\} \qquad (1.21)$$

and after substituing Euler's numerical integration we get

$$x(t + \Delta t) = \left(1 - \frac{\Delta t}{\tau_x}\right)x(t) + \frac{\Delta t}{\tau_x}x_0. \qquad (1.22)$$

Initial conditions hold:

$$x(0) = \frac{\alpha}{\alpha + \beta}, \qquad \tau_x = \alpha\beta, \qquad x \in \{n, m, h\} \qquad (1.23)$$

$$\alpha_n = \frac{10 - V}{100\left(e^{\frac{10-V}{10}} - 1\right)}, \qquad \beta_n = 0.125e^{-\frac{V}{80}} \qquad (1.24)$$

$$\alpha_m = \frac{25 - V}{10\left(e^{\frac{25-V}{10}} - 1\right)}, \qquad \beta_m = 4e^{-\frac{V}{18}} \qquad (1.25)$$

$$\alpha_h = 0.07e^{\frac{V}{20}}, \qquad \beta_h = \frac{1}{e^{\frac{30-V}{10}} + 1} \qquad (1.26)$$

Task 1.3 (0 b) Download model implementation file `cv3_HH.m` from course website. Visualize time output of the model (membrane voltage V), which is excited by the constant current $I_{ext} = 60\mu A/cm^2$ (default settings). Implementation works from time -30s to get steady system. Plot output for time $0 - 100ms$.

Task 1.4 (1 b) Expand visualization by adding time trend of conductance g for each channel type (describe different time series using `legend`)

Task 1.5 (2 b) Visualize dependence of firing frequency on external constant current I_{ext} - so called *activation or transfer function*. Set the input current I_{ext} in the range $0 - 15\ \mu A/cm^2$. You should compute firing frequency after reaching the steady state (hint: use fce `diff`, `find` alt. `findpeaks`). Which shape does the activation function resamble? For which values of the current I_{ext} will the shape of the activation function change significantly (perform a discussion)?

Task 1.6 (2 b) Adding noise to the input current will change behaviour of the model. Visualize membrane voltage V and activation function as a response to input current with added noise. Use $I_{ext} = 30\mu A/cm^2$ and white noise (`randn`) with the standard deviation 60 and zero mean, for plotting activation function use $I_{ext} = \langle 0, 15\rangle\mu A/cm^2$, noise with standard deviation 30. How did the resulting output change? Try out for different noise levels.

Task 1.7 (bonus) (1 b) Plot ISI histogram for the input current $I_{ext} = 30\mu A/cm^2$ and noise with standard deviation 60. Compare to the case with zero noise and discuss the results.

1.3 Spike train modelling

We showed in the previous tasks, how does noise on the input affect temporal and frequency characteristics of a neuron. Noise signal on neuron input is much more realistic than constant current - mainly due to the fact that neuron input is formed by a summary signal from many synapses of pre-synaptic neurons, connected to its dendrites. In this section, we will simulate such signal and use it as input to a neuron model.

You have learned in the lectures, that interspike intervals (ISI) of typical cortical neurons exhibit log-normal distribution. Before we start modelling spiketrains, let us verify this on a model.

Exercise 1.4 (LIF neuron model and Poisson spiketrain) The complex models from previous exercises, such as Hodgkin-Huxley, are able to model the action potential shape based on physiological modelling of ion channels' behavior. If we want to model more complex neural networks, even simpler models (such as Wilson model) are overly complex for such application. When modelling neural networks, the accurate shape of action potential is not as important (we will see it can be simply modelled using α functions) as the spike timing. For these purposes, the *Leaky integrate and fire* neuron is often used. The basics of this model includes simple integrator, which is reset to resting potential when a preset threshold is exceeded.

Task 1.8 (1 b) Download LIF model from the course website - `cv4_lif.m`. Show model output for constant input current (default setting). Plot output voltage, along with spike times (= reset times) over time. Compute firing rate and display an ISI histogram.

Task 1.9 (2 b) Modify previous task by applying input noise with mean 12 and standard deviation 100, instead of constant current (function `randn`). Simulate for times in the range $0 - 10^5$ and integration step $dt = 0.01$. Set bin number to 60 when computing histograms. Convert histograms to probability density function estimates ($\sum pdf = 1$).

Task 1.10 (3 b) Fit lognormal distribution to the data from the previous task. The lognormal distribution is given by:

$$pdf^{lognormal}(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{(log(x)-\mu)^2}{2\sigma^2}} \tag{1.27}$$

One possible solution would be to fit the analytical pdf to ISI histogram from the previous task. This, however, would be completely wrong, as one point in the histogram does not mean one, but many data points (think about this for a while!).

Thus, we will use the function `lognfit`, which estimates lognormal distribution parameters using maximum likelihood estimates. Based on these estimates, calculate the lognormal pdf and display along with the histogram. (You will probably have to adjust the bin number to get a good representation)

2   Real data analysis, modelling of neuronal populations

In this block, we will play with analyzing of real μEEG (*micro-EEG*) data, recorded from brains of Parkinson's disease patients. We will compare the data to our simulations from the previous block and we will analyze correlation with other biosignal - the electrooculogram (EOG).

Further in this section we will simulate more complicated network of artificial neirons and (so called *spiking network*) and analyze its behavior under varying conditions.

2.1   Spiketrain modelling II

Exercise 2.1 (Lognorm spiketrain) We showed in the last exercise how does LIF neuron model respond to a noisy input. In this task, we will show that real signal, produced as a combination of presynaptic potentials, really resembles gaussian noise in many ways.

You know from the lectures, that the *Poisson spiketrain*, - a spiketrain where the number of events per unit time follows poisson distribution - has exponential distribution of interspike intervals (ISIs). Further, the physiological constraints prevent neurons from firing earlier than after the *refractory period* - a fact we have to incorporate in our implementation.

In this example, we will generate a set of spiketrains using exponential distribution, taking refractory period into account, and mix them to create an artificial $muEEG$ signal.

Task 2.1 (1 b) Generate 100 000 ISI intervals from the exponential distribution with mean $E[X] = 50ms$ (hint: use `expnrd` function). Display ISI histogram.

Remove all ISI values lower than the refractory period of 1ms. Display ISI histogram and compare to the one before refractory period filtering. How many spikes were removed? Compute mean firing rate and coefficient of variation before and after filtering and compare.

Convert the resulting ISI ingervals to a spiketrain (logical vector with ones at spike positions hint: `cumsum`). Use time step of $dt = 0.1ms$. Check for errors by plotting the result

Task 2.2 (1 b) To convert binary spiketrain to time course, we will use convolution with $\alpha - function$, wchich will play the role of spike template.

Generate time course of the $\alpha - function$ according to:

$$
\begin{aligned}
I(t) &= c \cdot t \cdot e^{-\frac{t}{t_{peak}}} & (2.1) \\
c &= \frac{g_{peak}}{t_{peak}} e^1 & (2.2)
\end{aligned}
$$

where $g_{peak} = 0.5$, $t_{peak} = 1ms$. Compute the function values for times in the interval $t \in \langle 0, 30 \rangle$, the time step should be again $dt = 0.1ms$. Plot the output. After assessing the result, you may reduce the time range if appropriate.

Convolve the alpha function with Poisson spiketrain from the previous task (`conv`). Check the result again by plotting.

Task 2.3 (1 b) Divide the time course from previous task into 1000 signals of the same length(`reshape`). In this task, we will simulate mixing of presynaptic potentials (represented by the simulated neurons), at dendrites. The resulting summary signal will be given by a linear combination of the individual spiketrains. Coefficients will be given by weights w, which will be modelled as random from exponential distribution with $\lambda = 1$.

In order to have better control of the amplitude of the resulting signal, we will normalize the weights so that their sum equals a constant k. Set the value $k = 100$.

Combine the 1000 spiketrains into one signal, using a linear combination with weights w. You can represent the linear combination as matrix multiplication (elegant + much faster in Matlab).

Compare the resulting signal to gaussian noise (time and distribution). What are the key differences?

Task 2.4 (1 b) Apply the signal from previous task to the input of LIF neuron from Exercise and observe the output. What changes do you observe when you change the normalizing constant $k = 150$. What does the ISI histogram look like, now? Investigate and comment behavior of the LIF neurin.

2.1.1 Artificial and real signal - comparison

We have now generated artificial signal as a combination of Poisson spiketrains, convoluted with alpha function. In these tasks, we will compare the simulated signal to real μEEG, recorded from basal ganglia of a Parkinson's disease patient during deep brain surgery. We will show, that this simple method of data generation is not far from reality.

Task 2.5 (1 b) Use artificial signal generated in the previous task. It is a time course with sampling rate $f_s = 10kHz$. Download μEEG data from the course website. The .mat file contains variable `realmEEG` with 1s recording of activity around Thalamus ($f_s = 24kHz$). Display both signals in parallel plots (with correct time coordinates). Investigate both signals.

Task 2.6 (1 b) From visual comparison of time courses of real and simulated data, we get a rough idea of how similar these two signals are. Your next task is to try out some quantitative comparisons. Choose and implement appropriate methods of signal comparison. You can use e.g.: frequency spectrum, amplitude distribution etc.

2.2 Real data processing: Spike sorting

Exercise 2.2 (Spike sorting) In this exercise we will focus on spike detection and spike sorting (i.e.) in real μEEG data.

Spikes and background noise

The real μ EEG data we will use in this exercise were recorded using microelectrodes placed in the brain of patients suffering from Parkinson's disease, specifically in the area of the thalamus. As we know, the recorded signal consists of activity of a large number of neurons in the vicinity of the electrodes - this is source of a background noise[5] - from which we can get an approximate idea of the activity of the population in the area. However, we are usually interested in neurons in the vicinity of the electrode - here we can recognize individual action potentials of individual neurons and thus accurately analyze their activity. The recorded AP's (*spikes*) from near neurons have higher amplitude and we are usually able to distinguish them from the background - this procedure is called *spike detection*. Then, to assign individual spikes to different neurons we need a method called *spike sorting*, which usually uses unsupervised learning techniques based on the spike shape.

Spike sorting

Suppose we have a μEEG signal with detected spikes. Our goal is to classify spikes to individual neurons. Due to the spatial arrangement of different neurons relative to the electrode (plus possibly their individual characteristics) the shapes of recorded spikes originating in different neurons differ. This can be used to sort spikes on the basis of their shape. Given that the correct solution is unknown and we can not know (the number of neurons in the vicinity of the electrode can not be detected by the available imaging techniques, as well as correct assignment of individual spikes to the neurons), we need to use unsupervised learning methods, namely clustering. Regarding the choice of features describing the shape of individual spikes, we have a great range of options, including amplitude, duration, as well as various transformations - such as wavelet, fourier and many more.
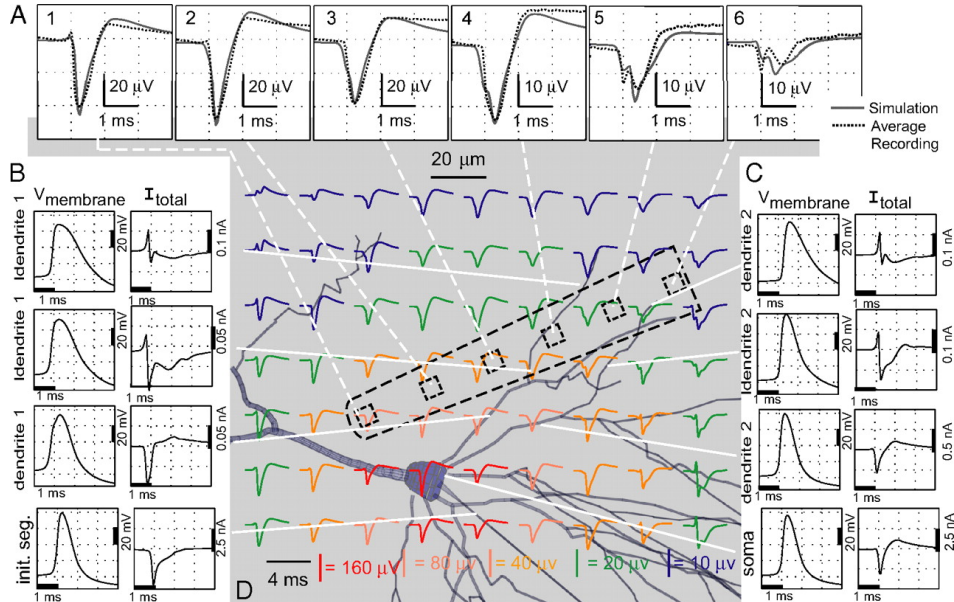
Dependence of AP's shape on location to electrodes. [6]

In this task we will use for spike detection a method designed by R.Q. Quiroga that is available from WaveClus toolbox[7]. The methods detects each AP based on amplitude tresholding. The treshold is determined by the number of standard deviation. Furthremore, signal is filtered by band-pass filter in range $300 - 3000Hz$. You can download a script `cv7_amp_detect.m`.

---

[5]Frequently the noise carries the majority of the signal energy

[7]source code is available here http://www.vis.caltech.edu/~rodri/Wave_clus/Wave_clus_home.htm. Those who are interested can find more information here, specially in „Introduction")

Task 2.7 (1 b) Download `cv7_spikesorting_data` and load it into the MATLAB. Sampling frequency is again $fs = 24kHz$. Plot the data and roughly estimate the spikes position. How long is record (in seconds)?

Download a script for spike detection `cv7_amp_detect.m`. Launch the sript using microelectrode data and plot the time series along with spikes detected with the treshold used for detection (output variable is `index` and treshold is `thr`). .

Task 2.8 (1 b) Variable `spikes` returns spikes detected in rows sorted by max value. Plot a few first spikes and estimate the number of neurons. Design at least one feature that will be used for spikesorting. Set manually threshold and sort the spikes according your feature. Plot the average spike for each group.

Task 2.9 (4 b) Now just imagine the task when we need to sort the signal without manually setting of the treshold. Suggest and calculate at least one more feature. Plot the spikes in feature space and color them based on your sorting from the previous task. Was your threshold calculated well?

Cluster your features by k-means algorithm (function `kmeans` from Statistics Toolbox). Estimate the number of clusters and show results in feature space. Next, show all signals and average signal for each cluster.

Analyse clustering results. Do you consider your features as sufficient? Is there any possible improvement?

2.3 Analysis of real data: Eye movements

Exercise 2.3 (Eye movements) In this exercise, we will show how to analyze reaction of single neuron to a stimulus. The data comes from experiments, where we evaluate connection between eye movements and neuronal activity in human Basal gangial. If the activity of given neuron correlates with eye movements, we should see a change in its firing pattern when eye movements are present. The data come from ten repetitions of the following experiment: The subject is looking at a PC screen, which is black at the beginning. At given time, a white cross appears and the task of the subject to focus at it. Electrooculogram (EOG) is recorded throughout the experiment using electrodes placed near subject's eyes, as well as micro EEG activity in the area of basal ganglia. Once the experiment is over, the data goes through spike detection and spike sorting and neurons, correlating with EOG are identified.

Task 2.10 (2 b) Download exercise data from the course website. Data contain recording of activity from singe neuron in ten repetinions of the same experiments + corresponding EOG. The variable `firingTimes` contains spike times of given neuron, relative to the time of stimulus (cross appears). Variables `eogTimes` and `eogVals` specify corresponding EOG signal for each repetition.

Note: plotting of EOG and corresponding spike times can be done as follows:

```
plot(eogTimes{1},eogVals{1});
hold on;
stem(firingTimes{1},ones(size(firingTimes{1})))
```

Plot firing of the neuron in time, aligned with the stimulus (time 0). What can we say about the activity? How many times did the neuron fire during all repetitions before and after the stimulus?

Task 2.11 (4 b) Use an appropriate way to aggregate activity over all repetitions. (e.g. number of spikes in time windows, instantaneous firing rate, etc.). How does the activity change right after the stimulus? Is there some trend even before the stimulus? How long does it take for the activity to drop and how long to return to original level? Be creative!

3 Information coding and transfer in the brain

In this block, we will model some of the processes taking part in transferring and storing information in the brain. We will go through the basics of cognitive modelling and we will try out the so-called *Self-organizing maps.*

3.1 Hebbian learning

All simulations that we did until now were using fixed synaptic weights - such approach ignores the *synaptic plasticity*, i.e. the adaptation of neuron's input weights. One model for weight adaptation is based on correlation of pre-synaptic and post-synaptic firing and is based on the theory by Donald Hebb.

Exercise 3.1 (Hebbian learning) This exercise will be based on the codes implemented for 2.1 („Lognorm spiketrain"), which will be adapted to include synaptic weight adaptation based on Hebb's learning. Simply put, the theory states that synapses, that fired shortly before the post-synaptic spike was emitted, are strengthened, while those firing after the post-synaptic spike are weakened.

The weight adaptation will be based on the following equation:

$$w_{n+1} = w_n + \delta w - f_{decay}, \tag{3.1}$$

where w_n are actual synaptic weights, f_{decay} is forgetting constant and δw is weight change computed according to:

$$\delta w = \alpha_{learn} \cdot pm \cdot e^{-pm\frac{\Delta t}{\tau}} \tag{3.2}$$
$$\Delta t = t_{post} - t_{pre} \tag{3.3}$$
$$pm = sign(\Delta t) \tag{3.4}$$

Task 3.1 (3 b) Implement function for weight adaptation based on 3.2. Let the parameters be $\tau = 40$, $\alpha = 1$, display the function in the range $\langle -100, 100 \rangle ms$ (do not forget axis labels). Comment the result.

Task 3.2 (7 b) In this task, we will use code for LIF neuron, fed with a large number of pre-synaptic spiketrains from exercise 2.1. Implement code for Hebbian weight adaptation. All synaptic weights will be adapted every time the LIF neuron fires. Use constatnts from the previous task, set forgetting constant to $f_{decay} = 0.01$ a and set initial sum of weigthts to $k = 600$.

Display time course of the summary pre-synaptic potential in time, including post-synaptic spikes, generated by the LIF neuron. Display the changes of weights over time (you can use e.g. `imagesc` function). How do the weights change over time? What is the change during the first 500ms? And during the following time? What effect does the value of forgetting constant have? Experiment with parameter settings and comment the results.

3.2 Simulated neural networks

The previous exercises focused mostly on modelling and examination of single-neuron models and their behavior. We showed factors that affect the variability of firing pattern of a neuron. It is assumed that firing pattern and spike timing is actually the main information carrier in the brain. To get a bit closer to the behavior of real neural networks, we will study artificial spiking neural networks, composed by many neurons.

Exercise 3.2 (Spontaneous activity) In this exercise, we will model spontaneous activity of a random neural network, composed by Izhikewich neurons - we will reproduce outputs of original Izhikewich research paper (available online: http://www.izhikevich.org/publications/spikes.pdf). The result will be a very brief text document, summarizing results of individual tasks.

Task 3.3 (3 b) Study Izikhewich model from the original research paper. Explain, what do the variables u and v mean and what do the constants a, b, c, d do.

Study the bottom two lines of Fig 2. What do these plots represent? What mainly affects the time courses?

Task 3.4 (3 b) Copy and paste source code of simulation of a random network of 1000 neurons from the paper to matlab. Run the code and investigate it thoroughly. Answer the following simple questions:

1. What types of neurons (and how many) have been used in the simulation

2. What unit do the constants a, b, c, d have and what does their size correspond to?

3. What represents the constant S and which way does it implement different neuron types?

4. How is the input current of different neuron types modelled?

5. Which part of the code represents resetting the membrane voltage to resting potential when a threshold voltage is achieved, as defined by Equation (3)?

Task 3.5 (4 b) Extend the implementation of the paper: add storing of post-synaptic potentials of individual neurons (it already stores firing times). Extend the program with the following (put the output figures in the report)

1. Display the time course of output voltage on 2 neurons of different type (select whichever you want). Are there any key differences in the time course?

2. Display power spectral density (e.g. `pwelch`) of averaged output voltage of the whole population. Restrict the spectrum to the range $1 - 100Hz$. What can we say about the character of the spectrum? What is the activity like in terms of brain waves (How about e.g. the alpha range: $8 - 13Hz$)? Place your comments into the report.

3.3 Self-organizing maps (SOM)

In the previous exercises, we have understood and experimented with weight adaptation - the network learning process. In this exercise, we will show how can a neural network be used for data clustering. This exercise will be based on *self-organizing maps* (*SOM*, sometimes also *Kohonen networks/maps*). We will use Matlab's builtin Neural Network Toolbox.

Unsupervised learning

Unsupervised learning is a machine learning technique which is often used for finding hidden structure in data. Typical example is clustering: we let the model to automatically divide the data into groups, based on their similarity. A well known example of clustering algorithm is *k-means*, which we already used in the spike-sorting exercise. The result of k-means clustering is a strict division of input data into classes, which is not necessarily always what we need in data mining.

SOM

Compared to k-means, we can imagine SOM as a projection of source data to a space of lower dimension. This can be useful for data visualization or aggregation. SOM is composed of neurons, distributed in space. Apart from position in space, each neuron is characterized by a weight vector of the same dimension as the input data (which is equal to the feature count).

The aim of SOM learning is to achieve similar response of neighboring neurons to given stimulus. At the beginning, all weights are initialized randomly[8]. For each input sample (=vector of feature values), we identify the neuron with weight vector closest to the input sample. This neuron is declared a winner. Weight vectors of all neurons in the neighborhood of the winner vector are adapted towards current input vector. The result is heavily dependent on the neighborhood function, defining weight adaptation step for given distance from the winner neuron. During learning, input data are presented to the network in multiple cycles, so that higher network consistency is ensured.

When the network is applied (i.e. during classification phase), weights are held fixed. For each input sample, distance between the input vector and each neuron's weight vector is calculated. Typically, this distance is visualized on a color scale. Based on this visualization, it can be defined which region of the network responds to specific input type.

---

[8]Better initialization can be achieved e.g. by principal component analysis (PCA) or heuristic knowledge about the data.

Exercise 3.3 (SOM: Iris dataset) We will make use of the classic Fisher's iris dataset http://en. wikipedia.org/wiki/Iris_flower_data_set, representing blossom properties of three species of iris (flower), which are rather hard to distinguish visually.

Task 3.6 Perform clustering of the iris dataset using SOM. Visualize the results. Follow this procedure:

1. Load iris dataset from matlab repository using `load iris_dataset`. This will load the variables `irisInputs` (feature values) and `irisTargets` (true labels - iris species, annotated by an expert)

2. Initialize the SOM: `net = selforgmap([8 8], "topologyFcn","hextop");`. What do the individual parameters mean? What does `topologyFcn` mean and what possibilities does the toolbox offer? Try out different topologies and visualize them using `plotsomtop`.

3. Learn the network on iris data using `net = train(net,irisInputs);`

4. Visualize position of individual neurons in the dimension of their weights using `plotsompos`

5. Visualize hit counts of individual neurons using `plotsomhits`. Show the same plots for different classes (iris species, specified by `irisTargets`). Compare the plots

6. Try out other possibilities of SOM visualization (you can use the GUI window which opens during learning)

Exercise 3.4 (SOM: wowel clustering) The aim this exercise is clustering of vowels using SOM and visualization of the results.

Task 3.7 Download publicly available dataset „Vowel Recognition" from UCI repository http:// archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Vowel+Recognition+-+Deterding+Data). You need specifically the file `vowel−context.data`, other files are useful for data understanding.

Load the file to matlab using `D = importdata filePath/vowel−context.data`. Data instances (examples) are in rows, attributes (features) in columns. We will need the 4th-13th feature. We will handle the last column separately - these are the data labels. We will also throw away the first three features. We will transpose the data (').

The next step is SOM initialization using reasonable parameters and learning. Observe the results for different classes and compare the results. The procedure is similar to the previous exercise 3.6.

3.4 Cognitive modeling: Bayes

The most typical approaches to computational cognitive modeling are connectionism (neural networks, SOM - implemented previously) and Bayesian models. In this lesson we will focus on the Bayesian approach which is based on the Bayes' rule.

Bayes' rule

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_j P(B|A_j)P(A_j)} \tag{3.5}$$

$P(A_i)$- the prior, is the initial degree of belief in A_i.
$P(A_i|B)$-the posterior, is the degree of belief having accounted for B.

GMM

For finding categories/clusters in the observed data, mixture models can be used. Each models has its own probability distribution. The most widely used distribution is a gaussian distribution, which is restricted to the data with normal distribution: **test**

$$l_k(\mathbf{x}_i|\mathbf{m}_k, \mathbf{S}_k) = (2\pi)^{-d/2}|\mathbf{S}_k|^{-1/2}\exp[-0.5(\mathbf{x}_i - \mathbf{m}_k)^T\mathbf{S}_k^{-1}(\mathbf{x}_i - \mathbf{m}_k)] \tag{3.6}$$

(cluster centers $\mathbf{m}_k$ and covariance matrices $\mathbf{S}_k$)
The learning of such a mixture can be done using MCMC (Markov chain Monte Carlo) method or EM algorithm.

To find the optimal number of clusters, likelihood function itself cannot be used (because it is gradually increasing with number of clusters):

$$LL(\mathbf{\Theta}) = \sum_{i=1}^n \ln(\sum_{k=1}^K r_k l_k(\mathbf{x}_i|\mathbf{m}_k, \mathbf{S}_k)) \tag{3.7}$$

(K- number of components, n - number of datapoints, $\mathbf{\Theta}$ - estimators (approximated model parameters), $l_k(\mathbf{x}_i|\mathbf{m}_k, \mathbf{S}_k)$ - Gaussian densities (similarities of datapoint $\mathbf{x}_i$ with the component k), r_k - components' mixing proportions).

Some penalization function must be added:

$$-2LL(\mathbf{\Theta}) + P(K, n, E, r_i)$$

The most widely used criterion is Akkaike criterion:

$$-2LL(K) + 2\eta(K) \tag{3.8}$$

For d-dimensional GMM with K components is $\eta(K)$:

$$\eta(K) = (K - 1) + dK + dK(K + 1)/2 \tag{3.9}$$

Exercise 3.5 (An application of Bayes' rule to neuroscience) Cognitive neuroscientists investigate which areas of the brain are active during particular mental tasks. In many situations, researchers observe that a certain region of the brain is active and infer that a particular cognitive function is therefore being carried out. Poldrack (2006) cautioned that such inferences are not necessarily firm and need to be made with Bayes' rule in mind. Poldrack (2006) reported the following frequency table of previous studies that involved any language-related task (specifically phonological and semantic processing) and whether or not a particular region of interest (ROI) in the brain was activated: Suppose

	Language Study	Not Language Study
Activated	166	199
Not activated	703	2154

that a new study is conducted and finds that the ROI is activated. If the prior probability that the task involves language processing is 0.5, what is the posterior probability, given that the ROI is activated?

Exercise 3.6 (Coin flipping) Task 3.8 Imagine that you are told that a box contains two coins: one that produces heads 50% of the time, and one that produces heads 90% of the time. You choose a coin, and then flip it ten times, producing the sequence HHHHHHHHHH. Which coin did you pick? What would you think if you had flipped HHTHTHTTHT instead?

Task 3.9 Coin flipping process produces sequences HHTHT and HHHHH. What will be the odds in favor to H1 if prior $P(H_1) = 999/1000$ and $P(H_2) = 1/1000$. What about the sequence HHHHHHHHHH (10x heads).

Task 3.10 In which terms is the model with priors $p(H) = p$ (arbitrary prob.distribution) more complex compared to model with $p(H) = 0.5$?

Task 3.11 Compare different expectations (priors). Assume that priors (probability of heads) are modeled by Beta distribution: $P(H) \sim p^{FH-1}(1-p)^{FT-1}$ (FH - fictional observations of heads, FT- fictionous observations of tails). Plot Beta distribution for different values of FH, FT. What will be the posterior if we see 4 heads and 6 tails for $P(H) \sim Beta(1000, 1000)$ and for $P(H) \sim Beta(3, 3)$?

Exercise 3.7 (Bayesian cue combination) We are interested in understanding how humans localize sound sources, when they can use both hearing and vision. We are also interested in illusions that arise in situations where there is a conflict between the two signals, for example as in ventriloquism [C1]. We first present subjects with visual stimuli v: a low contrast Gaussian blob, very quickly dashed at some position in space x_v. The subjects are asked to localize the stimulus position x. We find that the subjects estimates are unbiased on average but display some variability from trial to trial. Their estimates $P(x|v)$ can be modelled using a gaussian distribution with mean x_v and variability $\sigma_v = 4$ degree of visual angle. We then remove the visual stimuli and present subjects with very briefs auditory stimuli („clicks") originating from different positions in space x_a. We again ask the subjects to localize the source of the sound, and find that their estimates $P(x|a)$ can be modelled using a gaussian distribution with mean x_a and variability $\sigma_a = 3$ degrees of visual angle.
Now we present the visual and auditory stimuli at the same time and at the same location in space.

Task 3.12 If humans are Bayesian optimal in integrating the information from both sources, how precise are they going to be in their localization performance now ? Express mathematically the distribution of the subjects' estimates explaining how this is derived.

Task 3.13 Plot the three distributions (based on vision alone, based on audition alone, and based on both vision and audition) on the same graph.

Task 3.14 Now we trick the subjects. We tell them that the 2 stimuli come from the same point in space and corresponded to a single event (like a ball hitting the screen), but actually we introduce a small displacement between the stimuli: the visual stimulus is displaced 5° rightwards and the auditory stimulus displaced 5° leftwards ($x_V \angle x_A = 10°$, where xv and xa are the spatial positions of the visual and auditory stimuli). How is this affecting the response? Plot the 3 distributions on the same graph.

Task 3.15 Now we keep the auditory stimulus unchanged but vary the blurry-ness of the visual stimulus, we use first a very precise stimulus v_1 and repeat the experiment. We measure in this case $D_{v1} = 1°$. Then we try a very blurry stimulus v_2 for which we measure $D_{v2} = 20°$. Where do subjects localize the source in these 2 cases? Illustrate these examples and comment on your results.
These predictions were precisely tested by [C1] and they found that human behaviour is consistent with the Bayesian predictions. As described in class, this seems to be a general result, with evidence from a number of experimental protocols, in different modalities [C2, C3].

Exercise 3.8 (Demo: Hierarchical probabilistic model of categorization) Objects are represented by features which are processed separately (modelled by GMM) and consequently merged in a hierarchical layer.

Exercise 3.9 (Speech perception) Draw a graphical model for speech perception. The processes involved: speaker chooses a phonetic category, speaker articulates „a target production", noise in the speech signal, listener hears a speech sound. How will be the results affected by selecting different prior probability distribution for phonetic category?

Exercise 3.10 (Language processing) In this Exercise we will propose a model for language processing using Hidden Markov models.

Task 3.16 Assume that you have a acoustic signal for each word (total number of different words is 30, 20 examples for each word). Propose the method for unsupervised/supervised classification of words. What if the signals have different/same length? Discuss the effect of noise in data. Is it necessary to use model which includes temporal information? Why? Draw the graphical model for this task.

Task 3.17 Draw a graphical model (Hidden Markov model) for generating simple sentences describing an object (eg. Big red dotted horizontal rectangle.). Features are: size, color, texture, orientation and shape. Assume the fixed grammar and fixed length of the sentence. How will the transmission matrix looks like? And emission matrix?

Task 3.18 Modify the model from the previous task for variable length of the sentence. What if the grammar is not fixed?

Exercise 3.11 (Demo: SIMPLE model of memory) Matlab and WinBugs/OpenBugs

4 Výpočetní neurověda v systémové úrovni a její robotická implementace

V této části se budeme věnovat kognitivnímu modelování pomocí sofistikovaných SW nástrojů a implementujeme kognitivní síť pro rozpoznávání vizuálních podnětů do humanoidního robota

4.1 Bioloid robot: zpracování a klasifikace obrazových dat

Úloha se věnuje využití samoorganizujících map (SOM) pro učení obrazových dat a následně pro klasifikování geometrických tvarů, které se v obraze nacházejí. Obraz pro učení/testování SOM je získáván z kamery umístěné na hlavě robota Bioloid, který je připojen k počítači pomocí USB kabelu nebo Bluetooth. Úloha je řešena v Matlabu s pomocí SOM Toolboxu a Bioloid Toolboxu, který vznikl jako součást projektu a obsahuje funkce pro snadnou komunikaci s robotem. Úloha má tři části. V první se vygenerují trénovací obrázky, které jsou následně ukazovány robotovi a obrázky získané z robotovy kamery jsou ve druhé části použity pro natrénování SOM. Ve třetí části se vygenerují trénovací obrázky, které jsou ukázány robotovi a získané obrázky z kamery jsou pomocí SOM klasifikovány podle toho, jaký geometrický tvar se na nich nachází. Robot na základě klasifikace zaujme určitou pózu, aby dal vědět jaký tvar na obrázku rozpoznal.

Exercise 4.1 (Automatické nastavení) Nejprve si stáhněte a rozbalte balík NINUloha01.zip. Umístěte robota vhodně před monitor. Nejdříve spusťte soubor uloha01.m s přednastavenými parametry a pozorujte, zda robot reaguje na prezentované obrázky. Najděte nejlepší vzdálenost a umístění robota před obrazovkou.

Exercise 4.2 (Velikost testovací/trénovací množiny) V souboru uloha01.m postupně měňte velikost trénovací/testovací množiny (řádek 23, parametr num a řádek 96, parametr num) a sledujte, jaký má vliv na výslednou úspěšnost klasfiikace. V SOM síti sledujte velikost kvantifikační chyby. Z U-matice SOM sítě a z pozorované reakce robota na testovací obrázky určete, jak velká trénovací množina je potřeba pro úspěšné natrénování robota.

Exercise 4.3 (Variabilita stimulů) V souboru uloha01.m měňte postupně variabilitu obrázků - jak se geometrický tvar může pohybovat po obrázku (řádek 30, parametr shift). Nulová variabilita (tvar se vždy zobrazí ve středu obrazovky), naopak variabilita 1 znamená, že se tvar může pohybovat libovolně po obrazovce. Pozorujte znovu kvantifikační chybu, U-matici a úspěšnost klasfikace na testovací

množině, abyste určili, jak je potřeba změnit velikost trénovací množiny se zvětšující se variabilitou obrázků.

4.2 Bioloid robot: určení pozice vizuálního stimulu

Úloha se věnuje využití samoorganizujících map (SOM) pro učení obrazových dat a následně pro určení polohy objektu, který je pohybován v prostoru pomocí robotické ruky. Obraz pro učení/testování SOM je získáván z kamery umístěné na hlavě robota Bioloid, který je připojen k počítači pomocí USB kabelu nebo Bluetooth. Úloha je řešena v Matlabu s pomocí SOM Toolboxu a Bioloid Toolboxu, který vznikl jako součást projektu a obsahuje funkce pro snadnou komunikaci s robotem.

V první fázi propojte robotickou Úloha má tři části. V první se vygenerují trénovací obrázky, které jsou následně ukazovány robotovi a obrázky získané z robotovy kamery jsou ve druhé části použity pro natrénování SOM. Ve třetí části se vygenerují trénovací obrázky, které jsou ukázány robotovi a získané obrázky z kamery jsou pomocí SOM klasifikovány podle toho, jaký geometrický tvar se na nich nachází. Robot na základě klasifikace zaujme určitou pózu, aby dal vědět jaký tvar na obrázku rozpoznal.

Exercise 4.4 (Propojení robotické ruky s Bioloid robotem) V první fázi propojte robotickou ruku s Bioloid robotem a umístěte na konec robotické ruky objekt, jehož poloha v prostoru bude robotem určována. Stáhněte a rozbalte balík NINUloha02.zip. Spusťte soubor uloha02.m s automatickým nastavením. V první fázi se bude robotická ruka nastavovat do různých poloh (trénovací fáze). Následně bude robotovi prezentována testovací množina, přičemž by robot měl na objekt v dané poloze vždy ukázat svou rukou. Případně upravte pozici robota a robotické ruky.

Exercise 4.5 (Velikost testovací/trénovací množiny) V souboru uloha02.m postupně měňte velikost trénovací/testovací množiny (řádek 23, parametr num a řádek 96, parametr num) a sledujte, jaký má vliv na výslednou úspěšnost klasfiikace. V SOM síti sledujte velikost kvantifikační chyby. Z U-matice SOM sítě a z pozorované reakce robota na testovací obrázky určete, jak velká trénovací množina je potřeba pro úspěšné natrénování robota.

Exercise 4.6 (Počet poloh stimulů) V souboru uloha02.m měňte postupně variabilitu poloh objektu - v jakých různých pozicích se může objekt nacházet. Pozorujte znovu kvantifikační chybu, U-matici a úspěšnost klasfikace na testovací množině, abyste určili, jak je potřeba změnit velikost trénovací množiny se zvětšujícím se počtem možných poloh stimulů.

———————————— EXAMPLE CODE ————————————-

Bibliography