A6M33NIN — instrukce ke cvičení

autoři:
Eduard Bakstein°, Karla Štěpánová, Daniel Novák
http://nit.felk.cvut.cz, 2013

°cvičící pro LS 2014

3. ledna 2014

Abstrakt

Toto je podpůrný text pro cvičení v rámci magisterského předmětu A6M33NIN. Podklady pro jednotlivé úlohy jsou ve většině případů dále rozšířeny o kódy v jazyce MATLAB, na jejichž rozšíření se v rámci úloh pracuje.

Pro zájemce o hlubší znalost lze doporučit knihu [**?**], která je dobře čitelná a je v dostatečném počtu k dispozici v knihovně katedry kybernetiky. Česká kniha [**?**] je přehledová a zpracovává látku z lékařského hlediska, jako anglickou knihu shrnující neurovědy jako celek lze doporučit [**?**]. Z literatury k jednotlivým tématům pak nabízíme m.j. [**?**], [**?**] či z pokročilejších [**?**],další literaturu z naší knihovny lze doporučit na dotaz.

Obsah

0 Matematický aparát

V této úvodní části cvičení se prakticky seznámíme s numerickými metodami, které dále bohatě využijeme při modelování aktivity neuronů.

0.1 Numerické metody řešení dif. rovnic

Stejně jako v dalších odvětvích, zabývajících se dynamickými systémy, setkáváme se v Neuroinformatice s popisem systémů pomocí diferenciálních rovnic. Vzhledem k tomu, že jejich přesné analytické řešení může být často obtížné až nemožné, přichází na řadu přibližná numerická řešení, s nimiž se seznámíme v rámci tohoto cvičení.

Eulerova metoda

Uvažujme lineární diferenciální rovnici prvního řádu ve tvaru

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

Eulerova metoda prvního řádu spočívá v diskretizaci $\frac{\delta x}{\delta t} = \frac{\Delta x}{\Delta t}$ a následující úvaze. Vezmeme-li:

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

pak lze rovnici 0.1 psát ve tvaru

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

a konečně

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

Tímto způsobem tedy můžeme modelovat řešení v prvním přiblížení - bereme v potaz sklon přímky v daném bodě. Pokud je však sklon závislý na t, bude toto přiblížení velmi hrubé. V takovém případě můžeme řešení zpřesnit dalšími členy Taylorova rozvoje dle vzorce:

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

kde O reprezentuje všechny členy vyšších řádů. Ve druhém přiblížení tedy modelujeme kromě sklonu křivky i její zakřivení a výsledky by měly tudíž být bližší analytickému řešení.

Runge-Kutta metoda

V metodách vyšších řádů můžeme navíc zpřesnit řešení tím, že hodnotu členů vyšších řádů odhadujeme nikoliv v bodě x či $x+\Delta x$, ale v polovině tohoto intervalu - tzv. *"midpoint method"*. Výsledná hodnota by tak měla lépe reprezentovat sledovanou veličinu v daném intervalu a vést k přesnějšímu odhadu. Navíc, pokud nemáme k dispozici analytické vyjádření členů Taylorova rozvoje vyšších řádů, můžeme je opět odhadnout pomocí numerických metod, což sice vede k dalším (pravděpodobně nepřesným) odhadům, ale v konečném důsledku může výsledný odhad průběhu řešení rovnice výrazně zpřesnit.

Samotná metoda Runge-Kutta je tedy numerickou metodou 4. řádu, která kombinuje odhad ve středu intervalu s numerickým odhadem členů vyšších řádů. Lze tedy aproximovat řešení libovolné funkce a analytické vyjádření řešené rovnice není třeba. To vede k vyšší výpočetní náročnosti, ale - jak uvidíme - vede k velmi přesným odhadům řešení. Metoda Runge-Kutta je implementována v Matlabu ve funkci `ode45()`.

Úloha 0.1 Úkolem je numericky aproximovat řešení diferenciální rovnice

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

za poč. podmínky poč. podmínky: x(0)=1 pomocí Eulerovy metody (1 a 2 řádu) a metody Runge-Kutta.

Analytické řešení této rovnice má tvar $x = t + e^{-t}$ [1]. Řešení rovnice vypočtěte na intervalu $\langle 0, 5)$, dále zvolte $\Delta t = 0.02$.

Úkol 0.1 (1.5 b) Vykreslete řešení $x(t) = f(t)$ pro $t = 0, \ldots, 2s$ pomocí Eulerovy metody prvního řádu (x_{Euler}) a vykreslete do grafu spolu s analytickým řešením.

Úkol 0.2 (1.5 b) Úlohu vyřešte dále pomocí metody Runge-Kutta x_{Runge} a doplňte do grafu.

Nápověda: funkce `ode45` má jako první parametr callback na funkci, implementující řešenou rovnici. V tomto případě můžeme callback funkci definovat snadno pomocí tzv. *function handle* a *anonymous function*[2] jako `ode_func = @(t,x,flag) 1-x+t;`

Úkol 0.3 (1.5 b) Vykreslete závislost relativní chyby jednotlivých numerických metod oproti analytickému řešení. Př. (pro Eulerovu metodu) $(x_{Euler} - x_{exact})/x_{Euler} = f(\Delta t)$

Úkol 0.4 (1.5 b) Vykreslete absolutní chybu jednotlivých numerických metod v závislosti na velikosti integračního kroku $\Delta t \in (0.001, 1)s$ v čase $t = 1s$.

Úkol 0.5 (bonus) Úlohu vyřešte také pomocí Eulerovy metody 2. řádu. Řešení zahrňte do výstupů.

[1] můžete ověřit např. na http://www.wolframalpha.com/, dotaz: "solve differential equation x'=..."

[2] viz např. http://www.mathworks.com/help/matlab/matlab_prog/creating-a-function-handle.html

1 Modely neuronů

V této části cvičení se budeme detailně věnovat generování akčního potenciálu a chování jednotlivých neuronů. Jednotlivé aspekty generování akčních potenciálů a jejich sledů si vyzkoušíme na několika různých matematických modelech o různé úrovni přiblížení.

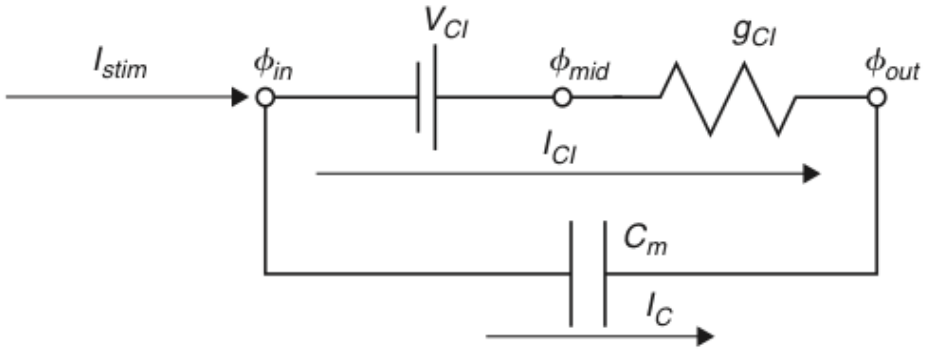
1.1 Modelování membrány a synapse 1: jednodušší modely

Vzhledem k tomu, že se nervové vzruchy šíří formou změn elektrického potenciálu na buněčné membráně, jeví se jako přirozené vytvořit základní model jejího fungování jako ekvivalentní elektrický obvod. Nejjednodušší model, který si představíme v této úloze, pracuje pouze s jedním typem kanálů - chloridovými stále otevřenými kanály - a má formu RC obvodu, doplněného článkem, nahrazujícím klidový membránový potenciál. Jak uvidíme, jeho odezva na vstupní vzruch je sice skutečnému akčnímu potenciálu poměrně vzdálena, poslouží nám však pro porozumění základním parametrům buněčné membrány a usnadní studování složitějších modelů.

Úloha 1.1 (RC model) Namodelujte chování membrány pomocí RC členu - viz obrázek 1. Vstupní proud membrány I_{stim} je obdélníkový signál $10pA$ po dobu $20ms$. Je nutné stimulační proud převést na stejné jednotky jako proudy I_{Cl} a I_C, které jsou vztaženy k ploše elektrody[3], která současně stimuluje a snímá proudy buňky v cm^2. Tedy $I'_{stim} = I_{stim}/A \approx 10^{-11} \cdot 10^6 \approx 10^{-5}$. Parametry membrány jsou následující:

kapacita membrány: $C_m = 1\ \mu F/cm^2$,
vodivost membrány: $g_{Cl} = 0.3\ ms/cm^2$,
časová konstanta: $\tau = C_m/g_{Cl}$,
povrch membrány: $A \approx 1 \cdot 10^{-6}\ cm^2$
Nerstův potenciál Cl: $V_{Cl} = -68\ mV$,
počáteční podmínky: $V(0) = 0\ mV$, $I_{Cl}(0) = 0\ \mu A/cm^2$, $I_C(0) = 0\ \mu A/cm^2$.

Úkol 1.1 (2 b) Vykreslete závislost $V(t)$, $I_{Cl}(t)$, $I_C(t)$



Obrázek 1: Model membrány s volným iontovým Cl kanálem

Platí

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

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

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

$$\tau \frac{dV}{dt} = V_{Cl} - V(t) + \frac{I_{stim}(t)}{A g_{Cl}} \tag{1.4}$$

$$\tau = \frac{C_m}{g_{Cl}} \tag{1.5}$$

[3]Ve skutečnosti se tedy jedná o proudové hustoty, které by bylo správnější značit dle konvencí ϱ. Pro konzistenci s přednáškami se však přidržíme značení I.

Eulerova metoda (dopředná)

$$\tau\frac{V(j)-V(j-1)}{dt} = V_{Cl}-V(j-1)+\frac{I_{stim}(j-1)}{Ag_{Cl}} \tag{1.6}$$

$$V(j) = V(j-1)+\frac{dt}{\tau}[V_{Cl}-V(j-1)+\frac{I_{stim}(j-1)}{Ag_{Cl}}] \tag{1.7}$$

Eulerova metoda (zpětná)

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

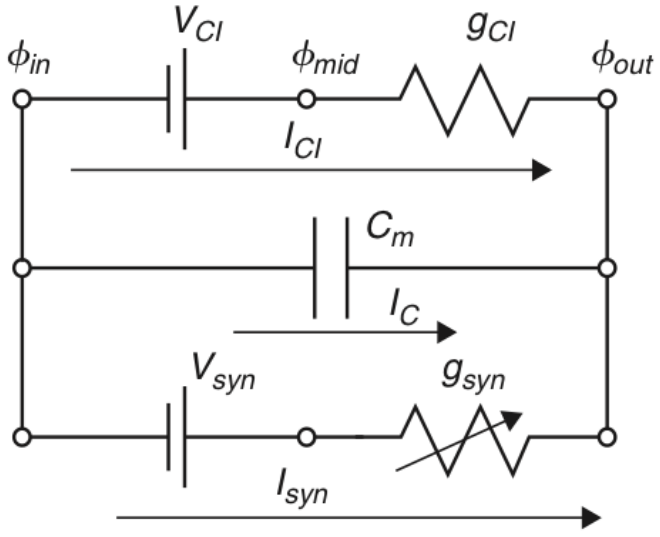
$$V(j)(1+\frac{dt}{\tau}) = V(j-1)+dt(\frac{V_{Cl}}{\tau}+\frac{I_{stim(j)}g_{Cl}}{Ag_{Cl}C_m}) \tag{1.9}$$

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

Úloha 1.2 (EPSP model) Úkolem je analýza modelu synapse vyobrazené na obrázku 2. Jedná se model se zahrnutím tzv. "excitatory postsynaptic potential"(EPSP), tedy chování membrány dendritu postsynaptického neuronu po přijetí vzruchu. Synapse je namodelována pomocí měnící se kondutivity g_{syn}. Časová konstanta $\tau_{syn} = 1\ mS$. Konstanty jsou stejné jako v předchozím příkladě. Stimulační proud $I_{stim} = 0.$, $V_{syn} = 10\ mV$. V čase $t = 1\ ms$ dojde k uvolnění neurotransmitteru, tedy $g_{syn}(1+\delta) = 1$. Počáteční podmínky $V(1) = 0$, $I_{syn} = 0$, $g_{syn}(1) = 0$, $g_L = 1$.

$$\tau_{syn}\frac{dg_{syn}(t)}{dt} = -g_{syn}(t)+\delta(t-t_{pre}-t_{delay}) \tag{1.11}$$

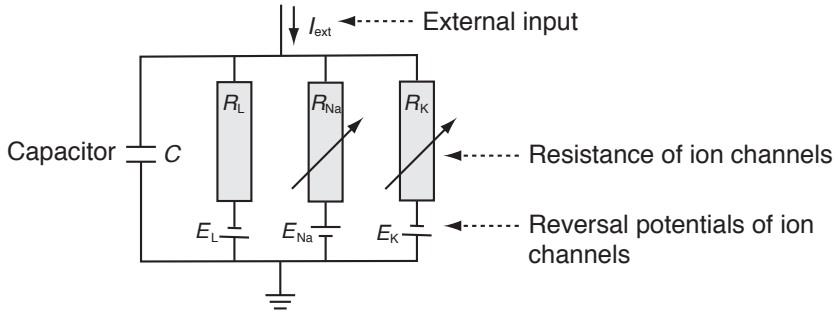
Úkol 1.2 (0 b) Vykreslete závislost $V(t)$, $I_{Cl}(t)$, $I_C(t)$, $I_{syn}(t)$. Vysvětlete souvislosti průběhů ve výsledném grafu a porovnejte je s výsledky Úkolu 1.1.



Obrázek 2: Model synapse s volným iontovým Cl kanálem

1.2 Modelování membrány a synapse 2: Hodgkin-Huxley

Úloha 1.3 (Hodgkin-Huxley) Úkolem je analýza Hodgkin-Huxley(HH) modelu dle schematu na obrázku 3. Na rozdíl od předchozích modelů zahrnuje HH model proměnné závislé nejen na čase, ale také na napětí, tedy Na^+ a K^+ napětím řízené kanály. Model je založen na následující sadě rovnic.



Obrázek 3: Schéma Hodgkin-Huxley modelu. Jednotlivé prvky vyjadřují vždy reverzní potenciál a rezistivitu příslušného typu kanálu: E_L, R_L - stále otevřené kanály ("Leakage channels"), E_{Na}, R_{Na} - sodíkové, napětím řízené kanály, E_K, R_K - draslíkové, napětím řízené kanály. Rezistivita napětím řízených kanálů je funkcí napětí a času.

Proud v iontovém kanálu lze vyjádřit dle Ohmova zákona:

$$I_{ion} = \hat{g}_{ion}(V - E_{ion}) \qquad \left(= (V - E_{ion})/\hat{R}_{ion}\right), \tag{1.12}$$

kde $\hat{g}_{ion}$ je maximální hodnota vodivosti daného kanálu, ostatní veličiny pak dle uvedeného schematu.

Dále zavádíme pomocné veličiny, závislé na čase a napětí $n(V,t)$, $m(V,t)$, $h(V,t)$, na jejich základě jsou vyjádřeny vodivosti:

$$\hat{g_K}(V,t) = g_K n^4 \tag{1.13}$$
$$g_{\hat{N}a}(V,t) = g_{Na} m^3 h \tag{1.14}$$

Sloučením rovnic 1.14-1.12 dostáváme

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = -g_{\mathrm{K}} n^4 (V - E_{\mathrm{K}}) - g_{\mathrm{Na}} m^3 h (V - E_{Na}) - g_{\mathrm{L}}(V - E_{\mathrm{L}}) + I_{ext}(t) \tag{1.15}$$

Dále definujeme časové konstanty

$$\tau_{\mathrm{n}}(V)\frac{\mathrm{d}n}{\mathrm{d}t} = -[n - n_0(V)] \tag{1.16}$$
$$\tau_{\mathrm{m}}(V)\frac{\mathrm{d}m}{\mathrm{d}t} = -[m - m_0(V)] \tag{1.17}$$
$$\tau_{\mathrm{h}}(V)\frac{\mathrm{d}h}{\mathrm{d}t} = -[h - h_0(V)] \tag{1.18}$$

Pro libovolnou z proměnných dostaneme

$$\frac{dx}{dt} = -\frac{1}{\tau_x(V)}[x - x_0(V)], \qquad x \in \{n, m, h\} \tag{1.19}$$

a po řešení pomocí Eulerovy metody konečně dostáváme

$$x\left(t + \Delta t\right) = \left(1 - \frac{\Delta t}{\tau_x}\right) x(t) + \frac{\Delta t}{\tau_x} x_0. \tag{1.20}$$

Pro počáteční podmínky pak platí:

$$x(0) = \frac{\alpha}{\alpha+\beta}, \qquad \tau_x = \alpha\beta, \qquad x \in \{n, m, h\} \tag{1.21}$$

$$\alpha_n = \frac{10-V}{100\left(e^{\frac{10-V}{10}}-1\right)}, \qquad \beta_n = 0.125e^{-\frac{V}{80}} \tag{1.22}$$

$$\alpha_m = \frac{25-V}{10\left(e^{\frac{25-V}{10}}-1\right)}, \qquad \beta_m = 4e^{-\frac{V}{18}} \tag{1.23}$$

$$\alpha_h = 0.07e^{\frac{V}{20}}, \qquad \beta_h = \frac{1}{e^{\frac{30-V}{10}}+1} \tag{1.24}$$

Úkol 1.3 (1 b) Implementaci modelu si stáhněte ze stránek cvičení - soubor `cv3_HH.m`. Zobrazte časový výstup modelu (napětí na membráně V), buzeného konstantním napětím $I_extt = 60\mu A/cm^2$ (výchozí nastavení). Implementace pracuje s časem od -30s pro počáteční ustálení systému. Výstup zobrazte pro časy $0 - 100ms$.

Úkol 1.4 (1 b) Rozšiřte zobrazení o časové průběhy vodivostí g jednotlivých typů kanálů, vodivosti popište (příkaz `legend`)

Úkol 1.5 (3 b) Nasimulujte a zobrazte závislost frekvence pálení na konstantním externím proudu I_{ext} - tzv. *Aktivační funkci* neuronu modelovaného HH modelem. Vstupní proud I_{ext} uvažujte v rozsahu $0 - 15\ \mu A/cm^2$. Frekvenci pálení počítejte z dostatečně ustáleného úseku časového signálu (nápověda: pomohou fce `diff`, `find` popř. `findpeaks`). Jakou obecnou funkci vám tento průběh připomíná? Při jaké hodnotě I_{ext} se významně mění průběh aktivační funkce (diskutujte).

Úkol 1.6 (3 b) Přidání šumu do budicího proudu způsobí změnu časových vlastností neuronu. Vykreslete časový průběh membránového napětí V a aktivační funkci z předchozích úkolů s přidáním šumu. Pro časový průběh použijte budicí proud $I_{ext} = 30\mu A/cm^2$ a normální šum (`randn`) o směrodatné odchylce 60, pro aktivační funkci opět rozsah $I_{ext} = \langle 0, 15\rangle \mu A/cm^2$ a šum o směrodatné odchylce 30. Jak se změnil tvar obou průběhů? Vyzkoušejte pro různé hodnoty šumu.

Úkol 1.7 (bonus) (1 b) Vykreslete ISI histogram (inter-spike-intervalů) pro budicí proud $I_{ext} = 30\mu A/cm^2$ a šum o směrodatné odchylce 60. Porovnejte s časovým průběhem. Průběhy vysvětlete.

1.3 Modelování spiketrains

Na předchozích úlohách jsme si ukázali, jaký vliv má šum na vstupu neuronu na jeho časové a frekvenční charakteristiky. Šumový signál na vstupu neuronu podstatně lépe odpovídá praktické situaci než konstantní proud. To je dáno především tím, že vstupem neuronu je součtový signál, tvořený synapsemi mnoha dalších neuronů, zapojenými na jeho dendritech. V tomto oddíle cvičení si takový signál vygenerujeme a prozkoumáme výstup neuronu, na nějž je přiveden.

Na přednáškách jste se dozvěděli, že interspike intervaly (ISI) typických neuronů mozkové kůry vykazují logaritmicko-normální rozdělení. Než se tedy pustíme do modelování spiketrains, zkusme tuto skutečnost ověřit na modelu.

Úloha 1.4 (LIF neuron a Poissonovský spiketrain) Modely, kterým jsme se věnovali v předchozích úlohách, jsou schopny velice přesně modelovat tvar akčního potenciálu na základě fyziologického modelu chování iontových kanálů. Chceme-li ale modelovat složitější sítě neuronů, ukazuje se, že i zjednodušené modely (např. Wilsonův) jsou pro takové použití zbytečně komplexní. Při modelování sítí neuronů není ani tak podstatný přesný tvar akčního potenciálu (ten, jak si ukážeme později, může být snadno modelován pomocí α funkcí), jako spíš správné modelování časování pálení neuronu. K těmto účelům se využívá jednoduchý *Leaky integrate and fire* neuron, popsaný v přednášce č. 3. Základním principem modelu je jednoduchý integrátor, který se po překročení nastaveného prahu resetuje na klidový potenciál (více v již zmiňované přednášce)

Úkol 1.8 (1 b) Stáhněte si model LIF neuronu ze stránek cvičení - `cv4_lif.m`. Zobrazte výstup modelu pro konstantní vstupní proud (výchozí nastavení). Vykreslete průběh výstupního napětí a okamžiky spikes v čase. Vypočtěte frekvenci pálení a zobrazte ISI histogram.

Úkol 1.9 (2 b) Upravte zadání z předchozího bodu tak, že na vstup přivedete namísto konstantního proudu šum se střední hodnotou 12 a směrodatnou odchylkou 100 (opět využijte funkci `randn`). Simulujte pro časy $0 - 10^5$ s krokem $dt = 0.01$. U ISI histogramu nastavte počet binů na 60. ISI histogram převeďte na odhad pravděpodobnostní funkce ($\sum pdf = 1$).

Úkol 1.10 (3 b) Na data z předchozího bodu nafitujte logaritmicko-normální rozdělení. Předpis logaritmicko-normálního rozdělení je dán rovnicí

$$pdf^{lognormal}(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(log(x)-\mu))^2}{2\sigma^2}} \tag{1.25}$$

Jedním z možných řešení by tedy bylo použít jako vstup odhad pravděpodobnostní funkce ISI z předchozího bodu a analytickou funkci na něj nafitovat pomocí funkce `fminsearch` nebo jí podobných. Vzhledem k tomu, že však jedna hodnota v pravděp. funkci neodpovídá jednomu, ale celé řadě pozorování, byl by takový postup nesprávný. Problematiku nastudujte zde: http://www.mathworks.com/products/statistics/demos.html?file=/products/demos/shipping/stats/cfitdfitdemo.html#6.

Z uvedených důvodů využijeme pro fitování funkci `lognfit`, která odhadne parametry rozdělení pomocí metody maximální věrohodnosti. Na základě odhadnutých parametrů vypočtěte a zobrazte analytický průběh logaritmicko-normálního rozdělení na ISI pravděpodobnostní funkci. Přesnějšího zobrazení můžete dosáhnout mapř. úpravou počtu binů v histogramu.

1.4 Modelování spiketrains II

Úloha 1.5 (Lognorm spiketrain) V minulém příkladu jsme si ukázali, jak vypadá výstup LIF modelu, buzeného signálem, silně zašuměným gaussovským šumem. Na této úloze si ukážeme, že skutečný signál, vzniklý součtem výstupů (z hlediska přijímajícího neuronu presynaptických potenciálů), Gaussovský šum v mnohém připomíná.

Na přednášce jste se dozvěděli, že vlastností *Poissonovského spiketrainu*, tedy takového, u nějž počet pálení za jednotku času má poissonovské rozdělení je, že interspike intervaly (ISI) mají rozdělení exponenciální. Z fyziologických vlastností neuronu dále vyplývá, že je velice nepravděpodobné, aby byly mezi dvěma AP kratší úsek než tzv. refraktorní perioda. I tuto vlastnost tedy ve své implementaci zohledníme.

V tomto komplexním příkladě si tedy vytvoříme sadu spiketrains pomocí exponenciálního rozdělení, respektující refraktorní periodu, se kterou provedeme některé experimenty. Více již v jednotlivých úkolech.

Úkol 1.11 (3 b) Vygenerujte soubor 10000 ISI intervalů pomocí exponenciálního rozdělení se střední hodnotou $E[X] = 50ms$ (lze použít funkci `expnrd`). Zobrazte histogram hodnot.

Z dat odfiltrujte ISI, které budou menší než refraktorní perioda. Ta bude opět modelována jako náhodná veličina s rozdělením

$$pdf = \begin{cases} N(1,6) & \text{if } t > 1 \\ 0 & jinde \end{cases} \tag{1.26}$$

Opět zobrazte histogram a porovnejte s histogramem před filtrací. Kolik spikes jsme filtrací ztratili? Pro obě rozdělení (před a po filtraci) vypočtěte koeficient variace a mean firing rate a porovnejte.

Výsledné ISI intervaly převeďte na spiketrain (logický vektor s jedničkami na pozicích spikes - nápověda: `cumsum`). Použijte časový krok $dt = 0.1ms$. Zkontrolujte správnost výsledku.

Úkol 1.12 (2 b) Abychom z binárního spiketrain získali časový průběh, pomůžeme si konvolucí s $\alpha - funkcí$, která obstojně nahradí tvar akčího potenciálu.

Vygenerujte tedy průběh $\alpha - funkce$ podle vzorce:

$$I(t) = c \cdot t \cdot e^{-\frac{t}{t_{peak}}} \tag{1.27}$$

$$c = \frac{g_{peak}}{t_{peak}} e^1 \tag{1.28}$$

kde $g_{peak} = 0.5$, $t_{peak} = 1ms$. Průběh funkce vypočtěte pro rozsah časů $t \in \langle 0, 30 \rangle$, opět použijte časový krok $dt = 0.1ms$. Výsledný průběh zobrazte. Po zhodnocení průběhu funkce můžete časový rozsah vhodně zkrátit.

Nyní proveďte konvoluci vypočteného průběhu se simulovaným Poissonovským spiketrain z předchozího úkolu (`conv`). Výsledek opět zkontrolujte zobrazením.

Úkol 1.13 (2 b) Vypočtený časový průběh z minulého úkolu vhodným způsobem rozdělte na 100 stejně dlouhých spiketrains (`reshape`). V tomto úkolu budeme simulovat míchání presynaptických potenciálů, reprezentovaných právě našimi spiketrains, na dendritech neuronu. Výsledný celkový vstupní proud do neuronu bude dán lineární kombinací jednotlivých spiketrains. Koeficienty budou tvořeny vahami w, které si opět zvolíme jako náhodné s rovnoměrným rozdělením (`rand`). Váhy upravte tak, aby jejich celkový počet bylo možno nastavit jednou konstantou k. Hodnotu konstanty zpočátku nastavte na 100.

Za pomoci těchto vah smíchejte rozdělené spiketrains do jediného, který bude představovat součtový signál na vstupu námi zkoumaného neuronu. Signál si zobrazte a prohlédněte. Jak vypadá v porovnání s gaussovským šumem?

Úkol 1.14 (2 b) Signál z minulého úkolu přiveďte na vstup LIF neuronu z příkladu 1.4 a pozorujte výstup. Jsou na výstupu LIF neuronu nějaké spikes? Co se změní když změním konstantu k, udávající součet vah, na 150? Jak nyní vypadá histogram výstupních ISI intervalů? Chování LIF neuronu prozkoumejte a okomentujte.

2 Analýza reálných data, modelování neuronových populací

V této části si vyzkoušíme analýzu reálných μEEG (*micro-EEG*) dat, získaných z mozků pacientů s Parkinsonovou chorobou. Data porovnáme s našimi simulacemi z minulého bloku a vyzkoušíme analyzovat korelaci odezvy konkrétního neuronu s dalším biologickým signálem - elektrookologramem. Data jsme získali díky naší spolupráci s 1. Neurologickou klinikou v Praze.

Dále si v této části vyzkoušíme nasimulovat složitější síť neuronů (tzv. *spiking network*) a budeme pozorovat její chování za různých podmínek.

2.1 Umělý a reálný signál - porovnání

Úloha 2.1 (Porovnání s reálnými daty) V rámci minulé úlohy jsme si vygenerovali signál na vstupu neuronu jako směs mnoha Poissonovských spiketrainů, konvolvovalných s alfa funkcí. V této krátké úloze si tato data srovnáme s reálnými μEEG daty, nahranými pomocí mikroelektrod v mozku pacenta s Parkinsonovou chorobou během implantace hluboké mozkové stimulace. Ukážeme, že tento jednoduchý způsob generování umělých dat není realitě příliš vzdálen.

Úkol 2.1 (2 b) Vygenerujte směs časových signálů s náhodnými vahami podle zadání z předchozí úlohy. Získáte tak simulovaný časový průběh se vzorkovací frekvencí $f_s = 10kHz$. Ze stránek cvičení si stáhněte μEEG data. Mat soubor obsahuje proměnnou `realmEEG`, obsahující 1s nahrávku nervové aktivity v oblasti Thalamu ($f_s = 24kHz$). Oba signály si zobrazte v paralelních grafech a průběhy prozkoumejte.

Úkol 2.2 (2 b) Při zkoumání signálů je patrné, že reálný signál je oproti simulovanému mírně vyhlazený. To může být dáno impledancí snímací elektrody, stejně jako vlastnostmi prostředí (signál je nahráván extracelulárně). Simulovaný signál filtrujte klouzavým průměrem o vhodné délce a výsledek opět vykreslete. Jsou si signály nyní podobnější?

Úkol 2.3 (3 b) Z porovnání pouhým okem máme hrubou představu, zda se signály podobají. Nyní je na čase vyzkoušet některé kvantitativní nástroje. Vyberte a implementujte vhodný způsob porovnání obou signálů (filtrovaného a reálného). Pro inspiraci můžete použít např. frekvenční spektrum, rozložení amplitud apod.

2.2 Zpracování reállných dat: Okohybné pohyby

Úloha 2.2 (Okohybné pohyby) V této úloze si na jednoduchém příkladu ukážeme, jak lze analyzovat reakci konkrétního neuronu na daný podnět. Data vznikla během experimentů, vnichž vyhodnocujeme provázanost aktivity neuronů v oblasti bazálních ganglií a okohybných pohybů. U neuronu, jehož aktivita s EOG koreluje, jednoduše předpokládáme změnu chování v souvislosti s pohyby oka. Data, která máte k dispozici, jsou záznamem deseti opakování experimentu, v němž pacient hledí na obrazovku. Ta je zpočátku černá, v určitou chvíli se na ní náhle oběví křížek. Úkolem pacienta je na křížek zaměřit svůj zrak. Během experimentu je zaznamenáván μEEG signál pomocí mikroelektrod v mozku pacienta, stejně jako okohybné pohyby (EOG, neboli elektrookulogram). Po skončení experimentu je na datech provedena detekce spikes a spike sorting a jsou nalezeny neurony, korelující s EOG.

Úkol 2.4 (2 b) Ze stránek cvičení si stáhněte EOG data. Data obsahují záznam aktivity jednoho EOG neuronu v deseti opakováních stejného experimentu. Proměnná `firingTimes` určuje relativní časy pálení vzhledem k stimulu, proměnná `expParts` určuje, ke kterému opakování experimentu daný čas náleží.

Vykreslete pálení neuronu v čase pro jednotlivá opakování experimentu vzhledem k stimulu (čas 0). Co lze o aktivitě na první pohled říci? Kolikrát v jednotlivých opakováních neuron pálil?

Úkol 2.5 (4 b) Vhodným způsobem zprůměrujte aktivitu přes všechna opakování experimentu (např. počet spiků v oknech, průměrná doba pálení - *mean firing rate* apod.). Jak se změní aktivita daného neuronu bezprostředně po stimulu? Lze vysledovat nějaký trend již před stimulem? Po jakém čase se aktivita vrátí opět na původní hodnotu?

2.3 Zpracování reállných dat: Spike sorting

Úloha 2.3 (Spike sorting) Doposud jsme v úlohách pracovali převážně s umělými daty, jejichž generování i výstup byly plně pod naší kontrolou. Použili jsme různé přístupy (modely neuronu, generování pomocí známého rozdělení apod.) a vyzkoušeli jsme, jak signál v nervové tkáni vzniká a jaké jsou jeho hlavní charakteristiky. Nyní si naopak vyzkoušíme, jak je možné analyzovat reálná data, o jejichž vzniku máme pouze přibližné informace.

Spikes a šum pozadí

V této úloze opět využijeme reálná μEEG data, získaná pomocí mikroelektrod z mozku pacientů s parkinsonovou chorobou, konkrétně z oblasti thalamu. Jak víme, je podobný nahraný signál složením aktivity velkého počtu neuronů v okolí elektrody – ten v praxi tvoří jakýsi šum pozadí[4] – ze kterého můžeme získat přibližnou představu o aktivitě populace v dané oblasti. Co nás však obvykle zajímá více, jsou neurony v blízkosti elektrody, u kterých jsme schopni rozeznat jednotlivé akční potenciály, a tudíž velmi přesně analyzovat jejich aktivitu. Jednotlivé AP (*spikes*) mají obvykle oproti zbytku signálu výrazně vyšší amplitudu a jsme tedy obvykle schopni je poměrně jednoduše od pozadí rozeznat – této proceduře říkáme *spike detection*. Problémem, na který však brzy narazíme, je, že v blízkém okolí elektrody se nachází obecně neznámý počet neuronů – může to být jeden, tři, nebo také žádný. Analýza směsice AP od více neuronů nám sice o aktivitě v oblasti poskytne výrazně detailnější informaci, než analýza pozadí, přesto je např. pro výzkum fyziologie fungování neuronů v dané oblasti nevhodná. Zde přichází na řadu tzv. *spike sorting*, jehož cílem je roztřídit jednotlivé detekované AP k jednotlivým neuronům. A právě spike sorting je těžištěm této úlohy.

Spike sorting

Předpokládejme, že máme k dispozici jednotlivé detekované spikes. Naším cílem je roztřídit spikes k jednotlivým neuronům. Výhodou je, že vzhledem k charakteristice jednotlivých neuronů a jejich prostorovému uspořádání vzhledem k elektrodě, se zaznamenané tvary spikes jednotlivých neuronů liší. Toho můžeme s výhodou využít a třídit spikes na základě jejich tvaru. Vzhledem k tomu, že správné řešení neznáme a znát nemůžeme (počet neuronů v okolí elektrody nelze dostupnými zobrazovacími technikami zjistit, stejně jako příslušnost jednotlivých spikes k nim), budeme muset využít metody učení bez učitele, konkrétně clustering do neznámého počtu tříd. Co se týče volby příznaků, popisujících tvar jednotlivých spikes, máme k dispozici velkou škálu možností, včetně amplitudy, délky, stejně jako různých transformací - např. waveletové.

V úloze využijeme pro spikedetekci metodu R.Q. Quirogy z veřejně dostupného toolboxu WaveClus[5]. Metoda detekuje jednotlivé akční potenciály na základě prahování amplitudy. Detekční práh je určen počtem směrodatných odchylek signálu. Pro odstranění nežádoucího šumu je navíc signál předem filtrován pásmovou propustí v rozsahu $300 - 3000Hz$. Upravená verze spikedetekce s nastavenými parametry `cv7_amp_detect.m` je dostupná na stránkách cvičení.

Úkol 2.6 (1 b) Ze stránek cvičení si stáhněte `cv7_spikesorting_data` a načtěte je do MATLABu. Vzorkovací frekvence je opět $fs = 24kHz$. Data si zobrazte a zhruba odhadněte, kde by se mohly nacházet spikes. Jaká je délka záznamu (ve vteřinách)?

Úkol 2.7 (1 b) Ze stránek cvičení si stáhněte funkci pro detekci spikes `cv7_amp_detect.m` s přednastavenými parametry. Spusťte ji na zadaných datech a zobrazte časový průběh signálu, spolu s detekovanými spikes (výstupní proměnná `index`) a detekčním prahem (proměnná `thr`).

Úkol 2.8 (4 b) V proměnné `spikes` vrací funkce v řádcích detekované spikes, seřazené podle maxima. Zobrazte několik prvních spikes a odhadněte, jaký počet blízkých neuronů do signálu přispívá. Na základě Vašeho pozorování průběhů navrhněte a vypočítejte alespoň jeden příznak pro spikesorting. Pro příznak ručně stanovte práh a detekované spikes podle něj rozdělte do skupin a spikes v jednotlivých skupinách zobrazte. Dále zobrazte zřetelně průměr pro každou ze skupin.

[4] Nezřídka tento šum nese převážnou většinu energie signálu

[5] původní zdrojové kódy http://www.vis.caltech.edu/~rodri/Wave_clus/Wave_clus_home.htm, kde zájemci naleznou i kvalitně připravené informace různé úrovně detailu (pro začátek doporučuji "Introduction")

Úkol 2.9 (4 b) Nyní si představte situaci, kdy potřebujete signály třídit automaticky bez ručního stanovování detekčního prahu. Navrhněte a ze signálu spočítejte alespoň jeden další příznak (celkově alespoň dva příznaky). Zobrazte všechny spikes v prostoru příznaků a obarvěte je na základě třídění z předchozího bodu. Byl Vámi nastavený práh zvolen správně? Kde se v grafu nachází?

Data, resp. vámi zvolené příznaky, roztřiďte pomocí algoritmu k-means (funkce `kmeans` ze Statistics Toolboxu) do vhodného počtu shluků a výsledek zobrazte v příznakovém prostoru. Dále stejně jako v předchozím bodě zobrazte průběhy jednotlivých spikes v nalezených clusterech a průměr pro každý cluster.

Zhodnoťte fungování automatického shlukování. Jak by podle Vás fungovalo na jiných datech? Považujete navržené příznaky za dostatečně robustní? Jak by bylo případně možné je vylepšit?

2.4 Simulace neuronových sítí

V předchozích příkladech jsme se zabývali modelováním a zkoumáním vlastností jednotlivých neuronů na neuronových modelech. Ukázali jsme si, jak se mění variabilita frekvence pálení neuronu při různých průbězích a směsích presynaptických potenciálů. Jak se předpokládá, je to právě časování akčních potenciálů, jež je nositelem informace v mozku. Abychom se v úlohách opět o stupeň přiblížili komplexitě reálného chování mozku, budeme se v této sekci věnovat studiu chování sítí, složených z mnoha neuronů.

Úloha 2.4 (Spontánní aktivita) V této úloze budeme modelovat spontánní aktivitu náhodné sítě neuronových modelů Izikhewichova typu - reprodukujeme výstupy původního Izhikewichova článku (dostupný online zde: http://www.izhikevich.org/publications/spikes.pdf). Výstupem úlohy bude velice stručný textový dokument, shrnující v bodech odpovědi na jednotlivé úkoly.

Úkol 2.10 (3 b) Nastudujte Izhikewichův model neuronu z původního článku. Vysvětlete, co znamenají veličiny u a v a k čemu zhruba slouží konstanty a, b, c, d.

Prostudujte spodní dva řádky Fig 2. Co tyto grafy reprezentují? Čím jsou vykreslené průběhy dány?

Úkol 2.11 (3 b) Ze článku si do matlabu zkopírujte zdrojový kód simulace náhodné sítě o 1000 neuronech. Kód si spusťte důkladně prostudujte. Odpovězte na následující jednoduché otázky:

1. Jaké typy neuronů jsou v simulaci použity a v jakých počtech?
2. Jakého rozměru jsou konstanty a, b, c, d a čemu tyto rozměry odpovídají?
3. Co vyjadřuje proměnná S a jakým způsobem implementuje rozdílné typy neuronů?
4. Jak je modelován vstupní proud do jednotlivých typů neuronů?
5. Která část kódu reprezentuje resetování výstupního napětí neuronů po překročení prahu, jak jej definuje rovnice (3)?
6. Kód spusťte a stručně popište výstup, který generuje.

Úkol 2.12 (4 b) Implementaci z článku rozšiřte tak, aby se kromě okamžiků pálení ukládaly také postsynaptické potenciály jednotlivých neuronů. Následně rozšiřte program o následující body, výsledné grafy zkopírujte do reportu.

1. Zobrazte průběh výstupního napětí na libovolných 2 neuronech rozdílných typů. Liší se nějak výrazně tyto průběhy?
2. Zobrazte výkonové frekvenční spektrum zprůměrovaných výstupních napětí celé populace. Spektrum zobrazte v rozsahu $1 - 100Hz$. Co lze říci o charakteru spektra? Jaká je aktivita sítě z hlediska mozkových vln (co například pásmo Alfa: $8 - 13Hz$)?

3 Přenos a kódování informace v mozku

V tomto bloku si namodelujeme některé procesy, související s přenosem a uchovávání informace v mozku. Zabrousíme také do základů kognitivního modelování a vyzkoušíme si tzv. *samoorganizující mapy.*

3.1 Hebbovské učení

V předchozích simulacích jsme pracovali se neuronovou sítí s konstantními náhodnými vahami. Takový přístup ignoruje synaptickou plasticitu, tedy průběžné změny vah jednotlivých synapsí, související s korelací pálení jednotlivých presynaptických neuronů. Toto asociativní Hebbovské učení si ukážeme na následující úloze.

Úloha 3.1 (Hebbovské učení) V tomto příkladě vyjdeme z kódů, naimplementovaných pro příklad 1.5 ("Lognorm spiketrain - obrácená úloha") a příklad obohatíme o úpravy vah pomocí Hebbovského učení. To je založeno na adaptaci vah presynaptických neuronů, jejichž aktivita koreluje s postsynaptickým výstupem (tj. pálením neuronu, o jehož synaptické váhy se jedná). budeme upravovat podle následující rovnice:

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

kde w_n jsou aktuální hodnoty synaptických vah, f_{decay} je konstanta zapomínání a δw je změna vypočtená z následující rovnice:

$$\delta w = \alpha_{learn} \cdot pm \cdot e^{pm\frac{\Delta t}{\tau}} \tag{3.2}$$

$$\Delta t = t_{post} - t_{pre} \tag{3.3}$$

$$pm = -sign(\Delta t) \tag{3.4}$$

Úkol 3.1 (3 b) Implementujte funkci adaptace vah dle rovnice 3.2. Parametry zvolte $\tau = 40$, $\alpha = 1$, rozdíly časů post a presynaptických AP volte v rozsahu $\langle -100, 100 \rangle ms$. Funkci pro daný rozsah časových rozdílů zobrazte v grafu (nezapomeňte popsat osy!). Výsledek komentujte.

Úkol 3.2 (7 b) V tomto úkolu použijeme kód pro LIF neuron buzený šumovým signálem z příkladu 1.5. Kód doplňte o adaptaci vah pomocí Hebbovského učení. Váhy budete upravovat pokaždé, když zaznamenáte na výstupu LIF neuronu AP. Konstanty volte stejné jako v předchozím úkolu, navíc nastavte konstantu zapomínání na $f_{decay} = 0.01$ a v kódu upravte počáteční sumu vah $k = 600$.

Zobrazte časový průběh součtového pre-synaptického potenciálu v čase včetně AP, generovaných LIF neuronem. Dále zobrazte vývoj vah v čase (lze použít např `imagesc`). Jak se mění hodnoty vah během prvních 500ms? A jak v následujícím čase? Experimentujte s nastavením parametrů učení a výsledky komentujte.

3.2 Samoorganizující se mapy (SOM)

V minulých cvičeních jsme modelovali neurony a jejich sítě a pochopili princip adaptace vah - neboli učení sítě. V tomto cvičení si na příkladu samoorganizujících sítí (*self-organizing maps - SOM*, někdy také *Kohonenovy sítě/mapy*) vyzkoušíme, jak jde učení neuronových sítí využít ke shlukování na dvou různých datasetech. Využijeme implementaci v Matlab Neural Network Toolbox.

Učení bez učitele

Principem učení bez učitele (*unsupervised learning*) je využití modelů k hledání skrytých souvislostí v datech. Typickým příkladem takové úlohy je shlukování, kdy necháváme model automaticky rozdělit data do skupin (*tříd*) na základě podobnosti jejich vlastností. Příkladem shlukovacího algoritmu je také *k-means*, který jsme využili v úloze na třídění akčních potenciálů. Jeho výstupem je "ostré"rozdělení do skupin, cože nemusí být pro vytěžování dat vždy výhodné.

SOM

Naproti tomu samoorganizující se neuronové sítě si lze představit jako projekci zdrojových dat do prostoru nižší dimenze a lze je tak využít např. pro vizualizaci nebo agregaci dat. SOM sestává z neuronů (uzlů), rozmístěných v prostoru. Kromě své pozice přísluší ke každému neuronu také vektor vah o stejné dimenzi, jako je dimenze vstupních dat (= počet příznaků).

Cílem učení SOM je zajistit, aby jednotlivé oblasti sítě vykazovaly podobnou odezvu na daný vstup. Na počátku učení jsou hodnoty vah inicializovány zpravidla náhodně[6]. Pro každý vstup se pak nalezne neuron, jehož váhy jsou hodnotám vstupního příkladu nejbližší a ten je prohlášen za vítěze. Hodnoty vah neuronů v blízkosti vítězného se pak upraví směrem k danému příkladu. Na výsledek má velký vliv volba funkce, určující míru úpravy vah okolí - tzv. *neighborhood function*. Vstupní data jsou prezentována síti zpravidla v několika cyklech, aby byla zaručana lepší konzistence sítě.

Při aplikaci sítě na data již k úpravám vah nedochází a pouze se vyhodnocuje vzdálenost vah jednotlivých neuronů od daného příkladu. Typicky se tato zobrazuje na barevné škále v prostoru a posuzuje se, pro jaké vstupy jsou charakteristické oblasti v síti.

Úloha 3.2 (SOM: Iris dataset) V úloze využijeme klasický dataset Iris http://en.wikipedia.org/wiki/Iris_flower_data_set, představující naměřené parametry okvětních lístků několika na pohled obtížně rozlišitelných druhů kosatců.

Úkol 3.3 Proveďte shlukování dat iris za pomoci samoorganizující mapy. Výsledky vhodně vizualizujte. Postupujte podle následujícího návodu:

1. Načtěte iris dataset z repozitáře matlabu pomocí `load iris_dataset`. Načtou se pole `irisInputs` (samotná data) a `irisTargets` (skutečné třídy - druhy kosatců, označené expertem)
2. Inicializujte samoorganizující mapu: `net = selforgmap([8 8], "topologyFcn","hextop");`. Co znamenají jednotlivé parametry? Co znamená `topologyFcn` a jaké možnosty toolbox nabízí. Vyzkoušejte různé topologie a zobrazte je pomocí `plotsomtop`.
3. Naučte síť na Iris datech pomocé `net = train(net,irisInputs);`
4. Zobrazte jednotlivé neurony jako pozici v prostoru vah - `plotsompos`
5. Zobrazte počty "zásahů"(hits) jednotlivých neuronů pomocí `plotsomhits`. Zobrazte stejné grafy pro jednotlivé třídy (jako parametr vložíte data, která vytřídíte na základě `irisTargets`). Grafy porovnejte.
6. Vyzkoušejte další možnosti vizualizace SOM (lze použít grafické okno, které se otevře během učení.)

Úloha 3.3 (SOM: Shlukování samohlásek) Předmětem úlohy je shlukování vyslovovaných samohlásek pomocí SOM a vizualizace jejích výstupů. Jedná se o bonusovou úlohu, pokud už jste předchozí úkoly splnili, pusťte se do ní!

[6] K lepší inicializací lze použít např. analýzu hlavních komponent (PCA)

Úkol 3.4 Z UCI machine learning repository si stáhněte veřejně dostupný dataset "Vowel Recognition"http://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Vowel+Recognition+-+Deterding+Data). Potřebujete především soubor `vowel-context.data`, ostatní slouží pro porozumnění datům.

Soubor načtěte do matlabu pomocí `D = importdata cesta-k-souboru/vowel-context.data`. Instance dat (příklady) jsou v souboru v řádcích, atributy ve sloupcích. Budeme potřebovat čtvrtý až třináctý atribut, poslední sloupec oddělíme zvlášť - jedná se o anotaci dat (třídy). První tři atributy zahodíme. Data transponujeme (').

Inicializujeme kohonenovu mapu rozumně zvolených parametrů a naučíme na data. Sledujeme výstupy sítě pro jednotlivé třídy, výstupy porovnáváme. Postup je obdobný úloze 3.3.

3.3 Kognitivní modelování: Bayes

Nejtypičtějšími přístupy k modelování kognitivních procesů lze charakterizovat buď jako konekcionistické (Neuronové sítě, SOM - viz předchozí úlohy) a Bayesovské modely. V této úloze se zaměříme na Bayesovský přístup, založený na Bayesově pravidle.

Bayesovo pravidlo

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

$P(A_i)$ - apriorní pravděpodobnost - počáteční pravděpodobnost jevu A pokud bychom neměli k dispozici žádnou další informaciA_i.

$P(A_i|B)$ - aposteriorní pravděpodobnost, kdy bereme v potaz pozorování B.

GMM - Gaussovské směsi

Při reprezentaci pravděpodobností jednotlivých jevů můžeme narazit na problém velmi vysokého počtu parametrů: když např. potřebujeme zjistit, jaká je pravděpodobnost, že náhodně vybraný člověk bude mít výšku 175.5 cm, potřebujeme pravděpodobnosti všech možných hodnot výšky (např. od 100 do 230cm), a tedy i velmi mnoho trénovacích dat. K reprezentaci pravděpodobnostního rozdělení pozorované veličiny se proto často využívají dodatečné předpoklady o tvaru rozdělení, které umožní výrazně snížit počet učených parametrů. Pokud např. předpokládáme, že má daná veličina normální rozdělení $N(\mu, \sigma)$, sníží se počet parametrů na 2. Pro modelování složitějších ("negaussovských") rozdělení můžeme použít gaussovské směsi - tzv. *Gaussian Mixture Models*, které sestávají z více gaussovských komponent. Jsou definovány následovně:

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

(kde $\mathbf{m}_k$ jsou výběrové střední hodnoty and $\mathbf{S}_k$ kovarianční matice)

K učení takovýchto modelů můžeme použít EM algoritmus (*Expectation Maximization alg.*). Ten sestává z opakování následujících dvou kroků:

1 - E(*expectation*): výpočet odhadované hodnoty Log-likelihood funkce $f_k(\mathbf{x}_i)$:

$$f_k(\mathbf{x}_i) = \frac{r_k l_k(\mathbf{x}_i|\mathbf{m}_k, \mathbf{S}_k)}{\sum_{k'=1}^{K} r_{k'} l(\mathbf{x}_i|\mathbf{\Theta}_{k'})} \tag{3.7}$$

2 - M(*maximization*): nalezení parametrů modelu, které maximalizují log-likelihood. Předpokládá se, že pravděpodobnosti $f_k(\mathbf{x}_i)$ jsou známé

$$r_k = \frac{1}{N}\sum_{i=1}^{N} f_k(\mathbf{x}_i) \tag{3.8}$$

$$\mathbf{m}_k = \frac{\sum_{i=1}^{N} f_k(\mathbf{x}_i)\mathbf{x}_i}{\sum_{j=1}^{N} f_k(\mathbf{x}_j)} \tag{3.9}$$

$$\mathbf{S}_k = \frac{\sum_{i=1}^{N} f_k(\mathbf{x}_i)(\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^T}{\sum_{j=1}^{N} f_k(\mathbf{x}_j)} \tag{3.10}$$

K nalezení optimálního počtu komponent nelze použít samotnou likelihood funkci, neboť ta narůstá s narůstajícím počtem komponent (nejvyšší hodnotu likelihood funkce dosáhneme pro počet komponent rovný počtu pozorování $K = n$):

$$LL(\mathbf{\Theta}) = \sum_{i=1}^{n} \ln(\sum_{k=1}^{K} r_k l_k(\mathbf{x}_i|\mathbf{m}_k, \mathbf{S}_k)) \tag{3.11}$$

(K- počet komponent, n - počet pozorování, $\mathbf{\Theta}$ - odhadnuté parametry modelu, $l_k(\mathbf{x}_i|\mathbf{m}_k, \mathbf{S}_k)$ - Gaussovské parametru $\mathbf{x}_i$ pro komponentu k), r_k - koeficient pro poměry komponent).

Dále je nutno přidat penalizaci:

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

Nejpoužívanější kritérium je Akkaikeho:

$$-2LL(K) + 2\eta(K) \tag{3.12}$$

Pro d-rozměrný GMM s K komponentami je $\eta(K)$:

$$\eta(K) = (K-1) + dK + dK(K+1)/2 \tag{3.13}$$

Úloha 3.4 (Lotto problem) Společnost Bizzaro inc. provozuje loterii

- Každý den je vyhlášeno výherní číslo, x
- Výherní číslo je vždy dvoumístná celočíselná hodnota.
- V průběhu každého týdne je výherní číslo taženo náhodně z neznámého rozsahu mezi l a u: $10 <= l <= x <= u <= 99$
- Na konci každého týdne jsou čísla l a u zveřejněna a jsou zvoleny nové hodnoty.

Váš přítel - booker - chce nabídnout další sázkový systém:

- Kdokoliv může zvolit číslo y kterýkoliv den v týdnu a pokud je y mezi l a u, vyhrává.
- Pokud chce, aby byly všechny sázky spravedlivé, jaké kurzy by měl nabídnout pro y?

Příklad:

- V neděli, společnost zvolí $l = 15, u = 39$ (tohle nikomu neříkejte!)
- Poté jsou v průběhu týdne tažena následující čísla Pondělí:31, Úterý:15, Středa:37, Čtvrtek:20, Pátek:20

Co potřebuje booker vědět?

- Let $X = (x_1, \ldots, x_k)$ bet he lotto data for k days, x_i is the winning number on day i, Nechť $C = (l, u)$ je skutečný rozsah
- Náš booker potřebuje znát pravděpodobnost $y \in C$, pokud známe dosavadní pozorování X, tedy: $P(y \in C|X)$

Úkol 3.5 Čísla v loterii jsou mezi 10 a 99, každá hypotéza h určuje možnou volbu l a u. Kolik možných hypotéz máme?

Jaké budou apriorní pravděpodobnosti?

Úkol 3.6 Stáhněte si skript lotto_missing_values.m a přidejte pravé strany rovnic na řádky: 25,26,54,58

Úkol 3.7 Spusťte skipt pro různé vektory výherních hodnot X.

Úloha 3.5 (Gaussian distribution) Úkol 3.8 Stáhněte a rozbalte si archiv gm_distribution.zip. Spusťte skript gm_distribution.m pro datasety 3,4,5 a 6 s 15 komponentami (1 opakování) a porovnejte výsledky GMM s učitelem a bez učitele.

Úkol 3.9 Implementujte GMM bez učitele pro neznámý počet shluků. Nejjednodušší způsob: spusťte skript pro různé počty shluků dokud není splněno zvolené kritérium. Implementujte Akkaikeho informační kritérium podle rovnic 3.12 a 3.13.

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.

Úloha 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.

Úloha 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.

Úloha 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.

Úloha 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.

Úloha 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.

Úloha 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.

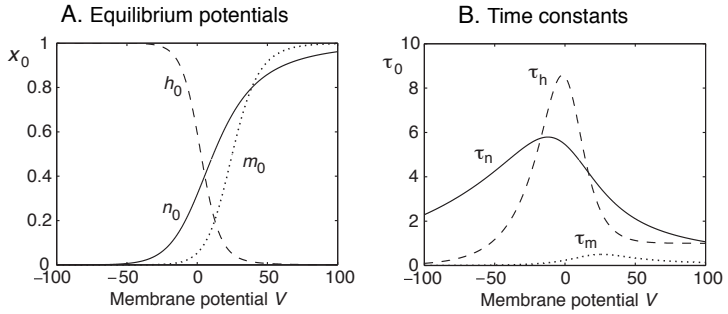
neuron type	$\tau_R[ms]$	g_T	g_H
Fast spiking (FS) neocortex	1.5	0.25	0
Regular spiking (RS)	4.2	0.2	5
Bursting	4.2	2.25	9.5

Tabulka 1: Požadovaná nastavení wilsonova modelu pro simulace z úkolu .1

Appendices

Wilsonův model

Jak jsme viděli v předchozích příkladech Hodgkin-Huxleyho model, přesně modelující vlastnosti neuronu krakatice, je velice složitý a neumožňuje rozsáhlejší analytické zkoumání (jak jsme viděli m.j. na příkladu výpočtu aktivační funkce). Pozdeji v historii se ukázalo, že úpravami rovnic HH modelu lze dosáhnout obdobných výsledků s výrazně nižší výpočetní složitostí. Za příklady mohou sloužit Wilsonův nebo LIF model. Wilsonovu modelu je věnována tato krátká úloha, o LIF modelu bude řeč později.



(a) (b)

Obrázek 4: Hodgkin-Huxley: časové konstanty modelu

Zjednodušené modely musejí z pochopitelných důvodů zahrnovat základní mechanizmy Na^+ a K^+ kanálů. Podstatou Wilsonova zjednodušení byla úvaha, že časová konstanta HH modelu τ_m (viz obrázek 4(a)) je velmi malá a tudíž zanedbatelná pro všechny možné hodnoty napětí na membráně a na ní závislá hodnota m se tak rychle blíží maximální hodnotě m, že může být touto hodnotou přímo nahrazena. Wilson dále usoudil, že otevírání K^+ kanálů je v podstatě doplňkem uzavírání Na^+ (viz obrázek 4(b) - n_0 vs h_0) kanálů a můžeme tedy použít další zjednodušení, totiž že $h = 1 - n$. Přestože výsledný model obsahuje pouze 2 následující diferenciální rovnice, je jeho výstup v zásadě srovnatelný s výstupem HH modelu.

$$\begin{aligned} C\frac{dV}{dt} &= -g_K R(V - E_K) - g_{Na}(V)(V - E_{Na}) + I_{ext}(t) \\ \tau_R\frac{dR}{dt} &= -[R - R_0(V)] \end{aligned}$$

Úkolem tohoto cvičení je prozkoumání chování Wilsonova modelu pro různá nastavení časových konstant.

Úkol .1 (A1) Stáhněte si ze stránek cvičení implementaci Wilsonova modelu pomocí Eulerovy metody - *wilson.m*. Vyzkoušejte modelování chování neuronů pomocí různých nastavení Wilsonova modelu z tabulky 1. Ostatní parametry ponechte ve výchozím nastavení. Výstupy zobrazte a porovnejte s grafy v přednášce 2.