Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Využití neuronové sítě LVQ v BCI systémech

Plzeň, 2015

Zdeněk Šmucr

Originální zadání

Poděkování

Rád bych poděkoval mému vedoucímu bakalářské práce Ing. Pavlu Mautnerovi, PhD., za odbornou literaturu, jeho rady a poznámky k mé práci. Dále děkuji Ing. Petrovi Brůhovi za vstřícnost při mé žádosti o testovací data a nakonec děkuji své rodině za podporu a psychickou odolnost vůči mým samovolným myšlenkovým pochodům.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 7. května 2015

Zdeněk Šmucr

Abstract

This bachelor thesis deals with the use of an LVQ neural network for the classification of brain activity in BCI systems, which is an important part of neuroinformatics. These systems create an interface between a computer and the human brain. A primary goal of this work is to find the best LVQ network methods and settings for the classification of event-related potentials (ERP) inside signals measured by means of electroencephalography (EEG). This method will be applied in EEGLAB in a Matlab environment, where the best conditions for the proper functionality of this method will be tested. All experiments were performed based on data taken from a random group of people. Data was gathered by a team of researchers from the Department of Computer Science and Engineering (KIV) at the University of West Bohemia in Pilsen.

Abstrakt

Tato bakalářská práce se zabývá využíváním neuronové sítě typu LVQ ke klasifikaci mozkových signálů v systémech BCI, které jsou důležitou částí v oblasti neuroinformatiky. Tyto systémy se zabývají propojením lidského mozku a vnějších počítačových komponent. Prioritním cílem práce je najít metodu a nastavení sítě LVQ, která by klasifikovala kognitivní evokované potenciály (ERP) v signálech naměřených encefalografem (EEG). Dále implementovat tuto metodu v programu EEGLAB z prostředí Matlab a otestovat, za jakých podmínek je její funkčnost nejlepší. To vše za experimentů provedených na datech naměřených z náhodné skupiny lidí. Data byla pořízena skupinou výzkumníků z katedry informatiky a výpočetní techniky (KIV).

Obsah

1 Úvod

V této diplomové práci se budu zabývat zpracováním mozkové aktivity člověka a to konkrétně metodami LVQ, jenž patří do skupiny neuronových sítí. V rámci ZČU zde proběhly podobné výzkumy a to za cílem najít ideální metodu na klasifikaci měřených signálů lidského mozku.

Studium mozkové aktivity je významná složka nejen v klinické praxi. Lze ji použít při diagnostice neurologických onemocnění, poruše pozornosti i při případech ryze experimentálních (tato práce). Mozková činnost se nejčastěji měří elektroencefalografem jako záznam časové změny elektrického potenciálu (EEG). V moji práci se budu zaměřovat na evokované potenciály (ERP), což jsou zaznamenatelné reakce mozku na podněty (stimuly) vizuální či zvukové. Tyto potenciály se budu snažit najít v signálu EEG vybranou metodou.

LVQ je jedna z mnoha umělých neuronových sítí. Neuronové sítě se snaží napodobit chování mozku a tím usnadňují práci při paralelním zpracování dat. Dají se použít k vytvoření tříd a následné klasifikace evokovaných potenciálů.

Signál naměřený pomocí metody EEG je často zašuměný a vyskytuje se v něm plno vad, jako například artefakty (svalová aktivita, pocení, atd.). Takovýto signál je často potřeba filtrovat a průměrovat. Předzpracovaný signál je důležitý k práci v rozhraní mezi počítačem a mozkem (BCI).

Mým úkolem bude vyzkoušet neuronovou síť k účelu klasifikace vzorů podle předem vytvořených tříd. Dále optimalizace metod LVQ na dané signály a zjištění vlivu filtrů a průměrování signálu na přesnost klasifikace.

2 Teoretická část

2.1 Architektura mozku

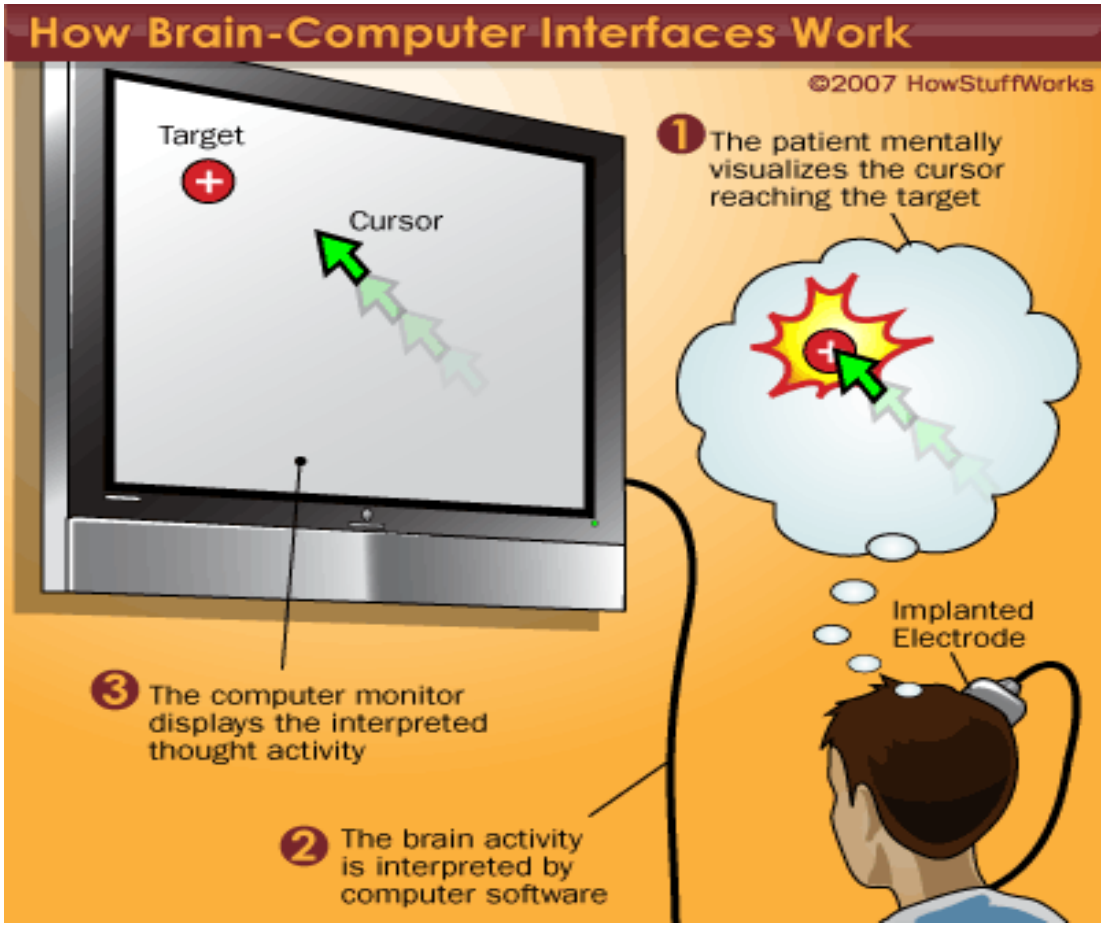
Navzdory obecným představám, mozek není víceúčelový výpočetní stroj s jedním hlavní procesorem. Mozek musíme brát jako soustavu naprosto odlišných subsystémů určených k vykonávání jednotlivých úkonů. Důkladným studováním účinků mozkových úrazů, ale hlavně používání nejmodernějších technologií, sestavili vědci detailní topografické mapy propojující různé části mozku s různými kognitivními funkcemi.

Můžeme provést rozdělení zhruba dvou základních částí. A to na kůru mozkovou a subkortikální oblasti. Subkortikální oblasti jsou fylogeneticky starší a zahrnují části spojené s kontrolou základních životních funkcí jako dýchání, srdeční tep a regulace tělesné teploty. Ovládá ale také základní emoce a instinktivní reakce jako je strach, odměna, reflexy a také učení a paměť. Kůra mozková je naopak vývojově mnohem novější. Jedná se o zatím nejkomplexnější část lidského mozku. Ta je obvykle spojována s myšlením v obrazech. Kůra podporuje smyslové a motorické dění, jako je odůvodňování, plánování, zpracování jazyka a rozpoznávání vzorců. Tato oblast je také ta, na kterou se BCI nejvíce soustředí.

2.2 BCI

BCI (Brain-Computer Interface) systémy měří mozkovou aktivitu, ze které jsou následně odvozeny specifické vlastnosti, jež jsou dále převedeny na jednotlivé signály vhodné pro ovládání koncového zařízení (Obr. 2.1). Tyto systémy nachází nejrůznější využití od rychlých odpovědí jednoduché otázky až po ovládání neuroprotéz a robotických končetin [7].

BCI je rozhraní sloužící k propojení zařízení snímající signály z mozku s vnějším zařízením, které ovládá. Ke snímání mozkové aktivity se nejčastěji používá zařízení založená na principu EEG. V dnešní době se objevuje spousta různých aplikací pracujících s BCI systémy, avšak nejvíce se investuje do zdravotnických zařízení, která se snaží pomoci ochrnutým nebo tělesně postiženým lidem. V armádě si od takovýchto systému slibují rychlejší reakce člověka, který například může ovládat let vrtulníku.



Obr. 2.1: Pohyb kurzorem myši pomocí myšlenek

V praxi je možné se setkat se třemi druhy BCI systémů. Druhy se dělí podle toho, jak je zavedeno snímání mozkové aktivity [2].

Invazní systémy

Prvním druhem jsou invazní systémy, při kterých se senzory zavádí přímo do šedé kůry mozkové. Výstupní signály jsou potom v nejvyšší kvalitě, jaké můžeme snímat. Nevýhodou je možnost rozšiřování zjizvené tkáně vedoucí k zeslabení nebo ztrátě signálu.

Neinvazní systémy

Druhé, neinvazní systémy jsou naopak zcela odlišné. Ke snímání nepoužíváme implantaci senzoru do tkáně. Celý systém je zaveden mimo organismus. Odchyt signálů se provádí pomocí technologie EEG, jež umožňuje měřit napěťové signály skupin neuronů. Tento typ snímá výsledky milionů nervových buněk a navíc je čte silně zkreslené průchodem skrz lebeční kost a kůži.

Částečně invazní systémy

Třetím typem je metoda kombinující obě předchozí metody. Část systému je implementována uvnitř lebky a druhá část je mimo organismus. Tento typ má výhody obou metod. Není potřeba rozsáhlý chirurgický zákrok, čímž se zmenšuje šance na rozšiřování zjizvené tkáně a přitom jsou signály v mnohem lepší kvalitě, než je tomu u neinvazních systémů.

2.2.1 Princip BCI systémů

- **Snímání dat** – Existuje mnoho metod, jak měřit data z mozku. Mezi nejpoužívanější patří EEG (Electroencephalography), MEG (Magnectoencephelograph), SPECT (Single Photon Emission Computed Tomography) nebo fNIRS (Functional Near Infrared Spectroscopy), avšak odborníci se shodují na tom, že pouze EEG a fNIRS jsou metody přenositelné, vcelku levné a bezpečné, což jsou důležité faktory pro práci s HCI(Human-Coputer Interaction).
- **Separace artefaktů** – Artefakty jsou nežádoucí signály vznikající například svalovou aktivitou, mrkáním nebo z vnějších elektrických polí.
- **Prostorová filtrace** – K filtrování dat budeme používat nejlépe dolnopropustní filtr typu IIR nebo FIR k odstranění nežádoucích frekvencí.
- **Parametrizace** – V této části se převádíme signál na parametry, které následně poslouží k práci s koncovým zařízením.
- **Klasifikátor** – Získané parametry jsou klasifikovány na povely a příkazy pro navazující systémy.
- **Akční člen** – Vykonává příkazy klasifikátoru (např. pohyb myší po obrazovce).
- **Zpětná vazba** – Uživateli je umožněno sledovat reakce na jeho mozkovou aktivitu a nadále zdokonalovat spolupráci se systémem.

2.2.2 Princip neinvazního systému

Jak již bylo řečeno, u neinvazních systémů snímají mozkovou aktivitu přístroje na bázi EEG. Převod těchto signálů není jednoduchou záležitostí, protože data získaná z mozku nejsou stacionární. Jejich vlastnosti se mění v závislosti na čase. Dalším problémem je to, že se každý mozek chová jinak. Tudíž algoritmy vytvořené na základě získaných dat z jednoho subjektu, který bude dále používán k ovládání nějakého výstupního zařízení nebo k interakci s počítačem, nemusí být kompatibilní s jiným subjektem. Často se stává, že ani nefunguje nebo funguje jinak.

Proto byly vyvinuty algoritmy, které jsou odolné vůči časovým změnám charakteristik signálu a jsou všeobecně použitelné [7]. Takovéto systémy musí počítat s obrovskou výpočetní složitostí a nejčastěji jsou založené na neuronových sítích.

2.2.3 Kontrolní úlohy

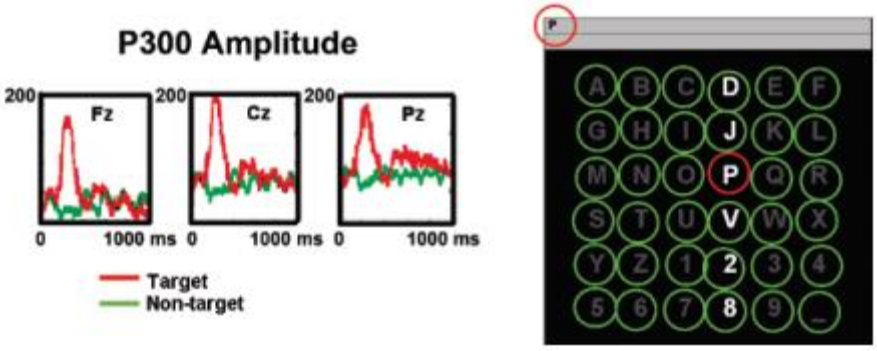
Kontrolní úloha je duševní úsilí BCI uživatele dobrovolně vyvolávat změny v mozkových signálech [2]. Kontrolní úlohy mohou mít mnoho podob. Například fyzický pohyb, vizualizace objektu, tiché popěvování, počítání nebo třeba uklidňující myšlenky. Tyto úlohy lze rozdělit do dvou hlavních kategorií:

1. Exogenní (navozené) paradigmata – Uživatel soustředí svou pozornost na množinu podnětů, které vyvolají samovolnou reakci zaznamenané BCI systémem.
2. Endogenní (vnitřního původu) paradigmata – Uživatel vykoná úlohu v duchu. Například si představí pohyb nebo v duchu počítá, aby vytvořil změnu signálu detekovatelnou BCI.

2.2.3.1 Paradigmata exogenních kontrolních úloh

Při exogenní reakci je potřeba vnějších podnětů, aby systém BCI zaznamenal změnu v mozkovém signálu. V obecném shluku signálů zaznamenáváme vlnu P300. Vlna P300 u zdravého jedince vrcholí 300 milisekund po té, co do mozku dorazí informace o nějaké překvapivé události. Abychom takovouto vlnu zachytili, je potřeba uživatelova soustředěnost.

Pánové Farwell a Donchin, kteří se tomuto tématu věnovali, dali svým studentům tabulku s abecedou (Obr. 2.2)s tím, že si má každý student vybrat jedno písmeno. Ti se pak soustředili na dané písmeno. Písmena začala jedno po druhém blikat a studentův mozek odeslal P300 vlnu, objevilo-li se jeho písmeno [2].



Obr. 2.2: Zachycení vlny P300

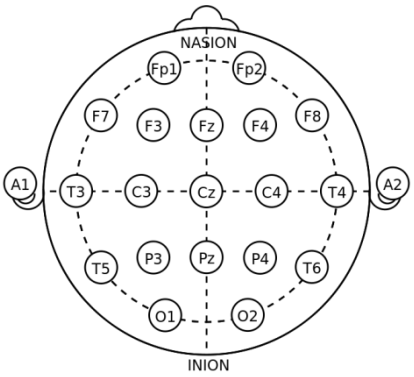
2.2.3.2 Paradigmata endogenních kontrolních úloh

U těchto úloh zaregistrujeme aktivitu určité části mozku uživatele pouhým pomyšlením například na pohybující se ústa. Zde nejsou vyžadovány žádné podněty, ačkoliv s jejich pomocí lze zlepšit odpovídající charakteristiky.

S prvními ovládacími rozhraními přišli Janathan Wolpaw (Murythm response) a Niels Birbaumer [6](Slow Cortical Potencials). Tyto endogenní systémy jsou založeny na nezávislosti podmíněné reakcemi od uživatelů. Systémy na bázi SCP se spoléhají na tom, že se zkušený personál naučí uživatele přepínat polaritu (kladná či záporná) na jejich systému SCP. Wolpawovo systémy pracují na skutečném nebo myšleném pohybu zachyceném v motorické kůře. Tyto systémy zároveň měří pro kontrolu amplitudu signálu. Oba BCI systémy byly demonstrovány na úloze pohybu kurzoru a výběru cíle.

2.3 Elektroencefalografie

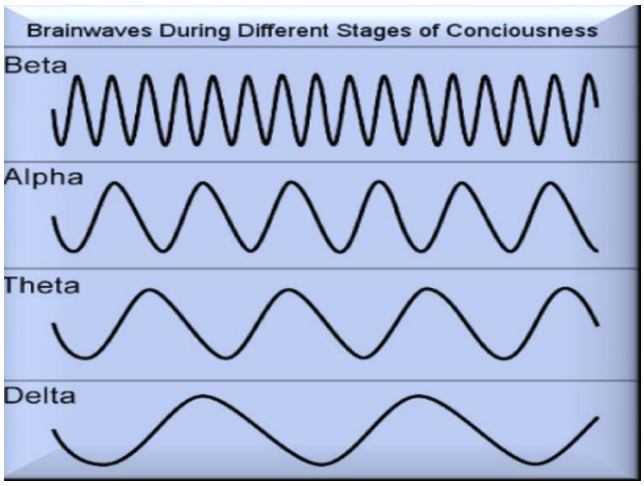
Elektroencefalografie (EEG) je metoda, pomocí které je možno měřit elektrické potenciály různých částí mozku přes lebku za plného vědomí pacienta. Zaznamenává se aktivita měřená v řádech desítek mikrovoltů. Standardní EEG se provádí v klidovém stavu. Měřená osoba leží nebo sedí a na hlavě má upevněnou speciální čepici, na které jsou umístěny elektrody. Tyto elektrody jsou přesně rozmístěné po celém povrchu hlavy. Rozmístění elektrod je pevně stanovené, jedná se o tzv. systém 10/20 (Obr. 2.3). Zkratka znamená vzdálenost elektrod 10 či 20% v podélné i příčné rovině.



Obr. 2.3: Rozmístění elektrod na hlavě, systém 10/20

Aktivita EEG signálu je z pravidla snímána ve čtyřech základních frekvenčních pásmech (Obr. 2.4):

- Alpha (8 -13 Hz)
- Beta (14 - 30 Hz)
- Theta (4 - 7,5 Hz)
- Delta (0,5 - 4 Hz)



Obr. 2.4: Typy mozkových vln

2.3.1 Evokované potenciály (EP)

Evokované potenciály jsou změny elektrického signálu v nervové tkáni v reakci na podněty z vnějšího prostředí. Jedná se o krátko-latentní odpověď mozku na zevní podnět s odezvou kolem 150 milisekund. Evokované potenciály mají amplitudu mezi 0,1-20 μV, což je méně, než EEG aktivita. Proto musíme podněty opakovat vícekrát, abychom byli schopni tyto signály zaregistrovat.

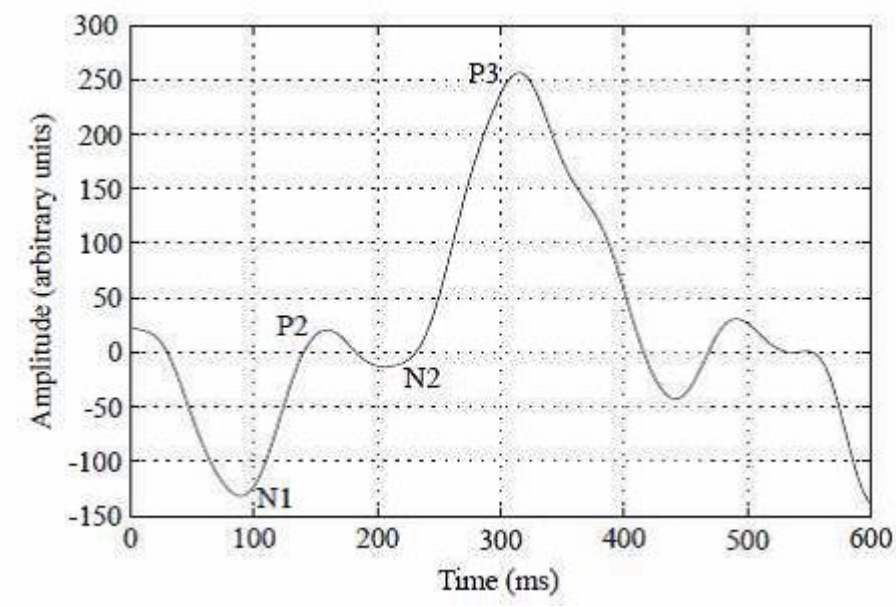
Tyto potenciály se mohou evokovat různou stimulací jako je motorická (MEP), vizuální (VEP), sluchová (AEP) nebo například kognitivní (ERP). Nejčastějším místem registrace evokovaných potenciálů je vertex (elektroda CZ); dále pak C3, C4, Fz, Pz, P3 a P4.

2.3.2 Kognitivní evokované potenciály (ERP)

ERP signály jsou potenciály vázané na nějakou událost. Mají tvar krátkodobých vln a nízké amplitudy, jejichž tvar, latence a velikost závisí na síle stimulace a na mentálním stavu pacienta. Mezi nejčastěji zkoumanou komponentou patří vlna P300, jež se jeví jako pozitivní vlna nacházející se kolem 300 milisekund po stimulu u zdravého jedince při sluchové stimulaci. Při zrakovém podnětu se objevuje mezi 400-550 milisekund [4].

Kromě této nejvýznamnější pozitivní vlny obsahuje signál po stimulaci další řadu komponent (Obr. 2.5). Označují se jako P1, N1, P2 a N2. Písmena znamenají, jestli je komponenta negativní (N) nebo pozitivní (P). Čísla jsou přibližné časy v řádech stovek milisekund, kdy se komponenty objevují.

Pomalé mozkové potenciály jsou ovlivněny vědomou i nevědomou kognitivní aktivitou, která vzniká nejčastěji zapojením pacienta do dané úlohy. Úloha se obvykle skládá s většího množství častých (nevýznamných) a terčových (významných) podnětů. Počet podnětu bývá nastaven 5 : 1.



Obr. 2.5: Komponenty ERP

Pokaždé, kdy dojde ke stimulu, se zaznamená čas a lze tak signál rozdělit na tzv. epochy. Pro hledání komponent v epochách se běžně používá filtračních metod.

2.3.3 Filtry

Naměřený EEG signál obsahuje veliké množství šumu a občas i nějaký artefakt (např. mrknutí) a proto je ho potřeba filtrovat. Epocha s artefakty se odstraňuje.

2.3.3.1 Horní propust

Horní propust je filtr, který utlumí nízké frekvence a propustí ty vyšší. Horní propust je důležitá, protože zmírňuje efekt velikého napěťového posunu zapříčiněného potenciály kůže. Použití tohoto filtru ovšem způsobuje zkreslení ERP komponent [3].

2.3.3.2 Dolní propust

Dolní propust funguje přesně opačně. Tlumí vysoké frekvence a ty malé propouští. Obvykle se používá hraniční frekvence mezi 30 – 100 Hz.

2.3.3.3 Band-pass filtr

Propouští frekvence v daném intervalu a odstraňuje ty mimo něj.

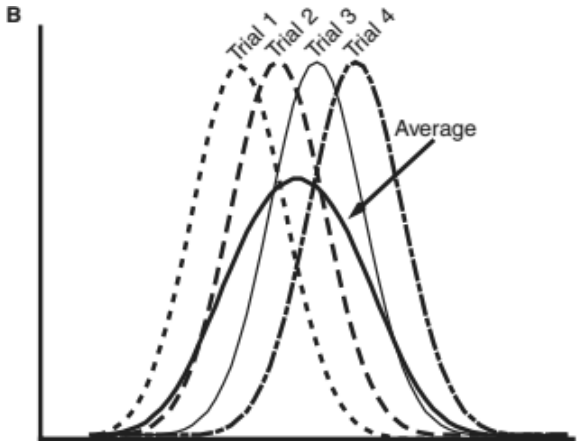
2.3.3.4 Průměrování

Průměrování je důležitá součást zpracování signálu. Ačkoliv můžeme použít různé filtry k oddělení ERP od zbytku signálu, zdaleka nejúspěšnější je právě průměrová-

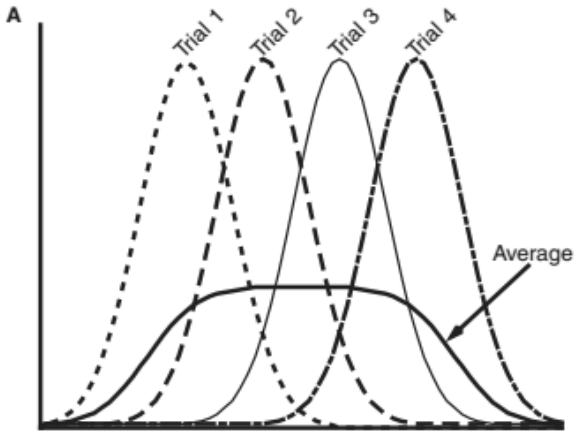
ní. Šum v signálu nebývá konstantní na rozdíl od víceméně podobných ERP komponent. Průměrováním vyniknou komponenty a šum se redukuje. Teoreticky lze říci, že čím více epoch zahrneme do průměrování, tím nám více vynikne ERP a utlumí se šum.

Matematicky lze vyjádřit jako $(1/\sqrt{N}) \cdot R$ za předpokladu, že je R míra šumu jedné epochy a N počet epoch. Jinými slovy, velikost šumu se zmenšuje a tím se i odstup signálu od šumu zvětšuje (S/N), jako funkce druhé odmocniny počtu epoch [3]. Dají se těmito metodami redukovat i artefakty, ačkoliv mrknutí má tak velký napěťový rozsah, že úplně odstranit prakticky nelze.

Asi nejpoužívanější variantou průměrování signálu je obyčejné sečtení epoch a vydělení signálu jejich počtem. Je to jednoduchá metoda, ale přináší menší obtíže. Vzhledem k možným různým latencím hledaných komponent v epochách se zmenšuje napětí vlny (Obr. 2.6, 2.7).



Obr. 2.6: Průměrování ERP s málo odlišnými latencemi



Obr. 2.7: Průměrování ERP s velice odlišnými latencemi

2.4 Klasifikace

Klasifikace v umělé inteligenci spadá do kategorie strojního učení. Je druh problému, máme-li určit, do které z kategorií dat dané pozorování patří. K tomu máme k dispozici trénovací množinu obsahující pozorování (data, třídy), pro která jsou kategorie správně určeny. Učení klasifikátoru může probíhat způsoby učením s učitelem nebo učením bez učitele. Učení s učitelem znamená, že je předem známa množina dat a jejich správné zařazení do tříd. Druhá metoda klasifikuje vzory na základě nějaké podobnosti. Například vzdálenost mezi vektory.

2.4.1 Klasifikátor

Klasifikátor je algoritmus zajišťující klasifikaci. Jedná se o matematický model, který pro daný vstup určí třídu. V případě, že počet výstupních tříd je roven dvěma (odpověď ANO, NE), jedná se o úlohy dichotomické klasifikace.

2.4.2 Bayesův klasifikátor

Model je reprezentován pravděpodobnostním rozložením tříd. Při klasifikaci je vybrána třída s největší pravděpodobností.

Princip:

Mějme množinu dat $D = \{d_1, d_2, \dots \dots, d_r\}$. Každý prvek d_i je reprezentován příznakovým vektorem v_j, kde $V = \{v_1, v_2, \dots \dots, v_v\}$ a zároveň patří alespoň do jedné ze tříd C_k; $C = \{C_1, C_2, \dots \dots, C_c\}$. Zařazení jednotlivých dat do třídy probíhá výpočtem podmíněných pravděpodobností pro každou třídu a výběrem maximální podmíněné pravděpodobnosti použitím Baysova pravidla.

$$P(c_k|d) = \frac{P(c_k)\,P(d|c_k)}{P(d)} \qquad 2.1$$

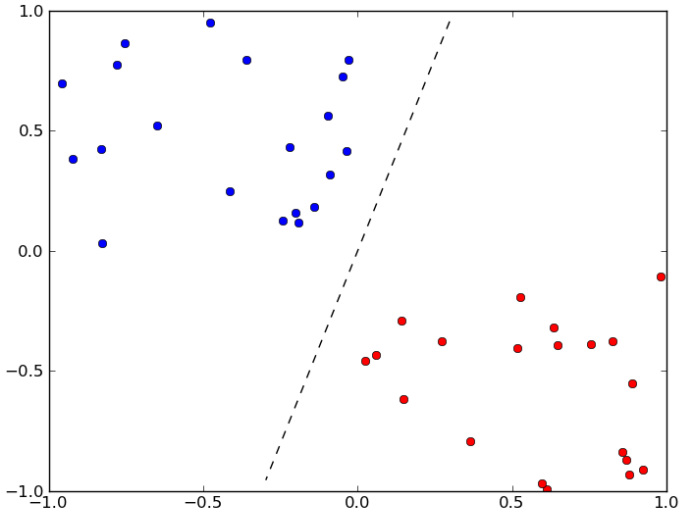
Ve výsledku vyplývá:

$$c_{max} = \arg max_{c_k \in C}\ P(c_k) \prod_{j=1}^{n} P(v_j|c_k) \qquad 2.2$$

Výhoda této metody je ta, že dokáže klasifikovat i neúplně popsané vzory.

2.4.3 Lineární klasifikátor

Lineární klasifikátor je jednoduchá klasifikační metoda, která rozděluje prostor lineárními úseky [9]. Každé ze tříd klasifikátoru patří jeden úsek (Obr. 2.8).



Obr. 2.8: Rozdělení prostoru lineárním klasifikátorem

Předpokládejme n-dimenzionální prostor, váhový vektor $\omega = \{\omega_1, \omega_2, \dots, \omega_n\}$ a práh ω_0. Uvažujme n-dimenzionální vstupní prostor $x \in \mathbb{R}^n$. Potom výsledná je nadrovina následující:

$$g(x) = \omega^T \cdot x + \omega_0 = 0 \qquad 2.3$$

Pro každé x_1, x_2 náležící nadrovině vyplývá, že vektor rozdílu $x_1 - x_2$ je ortogonální k vektoru ω.

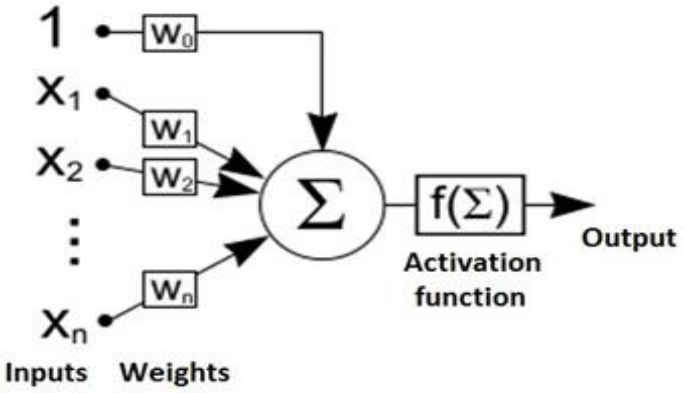
$$0 = \omega^T \cdot x_1 + \omega_0 = \omega^T \cdot x_2 + \omega_0 \rightarrow \omega^T(x_1 - x_2) = 0 \qquad 2.4$$

Lineární klasifikátory se zaměřují především na případ, kdy máme 2 třídy (ano/ne). Mezi nejznámější a nejpoužívanější patří LDA (Linear Discriminant Analysis), SVM (Support Vector Machines) a Perceptron.

2.5 Neuronové sítě

Neuronová síť je algoritmus pro zpracování informací, který má simulovat práci mozku. Základem celé sítě je umělý neuron (Obr. 2.9), jehož předobrazem je biologický neuron. Vstupy neuronů $(x_1, x_2, \dots\dots, x_n)$ jsou ohodnoceny synaptickými vahami $(\omega_1, \omega_2, \dots\dots, \omega_n)$. Na výsledný neuron je aplikována přenosová funkce f a výsledkem je výstupní funkce y. Práh ω_0 označuje prahovou hodnotu aktivace neuronu. Neuron tedy může mít více vstupů, ale jen jeden výstup.

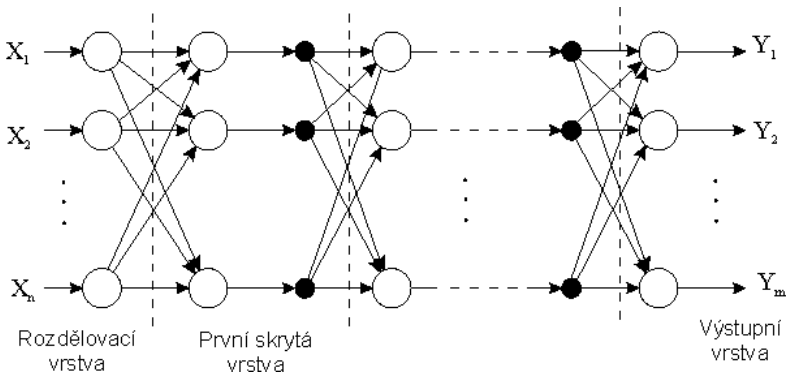
Neuronové sítě se používají v lékařství při prohlubování znalosti o organismu živočichů při simulaci nervových soustav. V softwarovém inženýrství se pak používá na rozpoznávání obrazů, řeči, zpracování signálu a klasifikace vzorů.



Obr. 2.9: Umělý neuron

2.5.1 Architektura sítě

Neuronové sítě se skládají z vrstev. Tomuto typu architektury se říká vrstvená architektura. Základní modely jsou jednovrstvé, kdy jsou propojeny vstupní jednotky s výstupními. Naopak vícevrstvý model (Obr. 2.9) má mezi těmito dvěma minimálně jednu vrstvu navíc, tzv. skryté úrovně. Obě tyto sítě jsou zařazeny jako sítě s dopředným šířením. Tudíž netvoří žádné smyčky.



Obr. 2.10: Vícevrstvá architektura

2.5.2 Učení neuronové sítě

Neuronovou síť učíme za účelem takového nastavení, aby dávala přesné výsledky. Veškerý učební potenciál je kladen na váhy, které v důsledku toho mění svoji velikost. Větší váha znamená větší důležitost vstupu. Učení neuronové sítě rozlišujeme na učení s učitelem a učení bez učitele.

2.5.3 Soutěžení

Některé sítě využívají soutěžní strategie učení. Výstupní neurony spolu soutěží o to, který z nich bude aktivní a bude se učit. V určitém čase je tedy aktivní pouze jediný neuron. Příkladem takové sítě je Samoorganizující Kohonenova sít (SOM) nebo Lineární vektorové kvantování (LVQ).

2.5.4 LVQ

Algoritmus LVQ je založen na Kohonenově síti. Ta je však samoorganizující (učení bez učitele) na rozdíl od LVQ. Obě tyto sítě však používají strategii soutěžení. Jelikož tato síť používá učení s učitelem, budou algoritmu předloženy vzory patřící do určitých tříd. Každý výstupní neuron bude reprezentovat jednu ze tříd. Pod výrazem LVQ se skrývají algoritmy LVQ1, LVQ2, LVQ3 a OLVQ1.

2.5.4.1 LVQ1

Cílem algoritmu je nalezení takového výstupního neuronu, který by (svými vahami ω_i) přibližně odpovídal zadanému vstupnímu vektoru x. Budeme tedy hledat neuron, který je k němu nejblíže. Algoritmus končí tehdy, spadají-li vektor x a ω_i do stejné třídy [1]. Patří-li x a ω_i do rozdílných tříd, poté je neuron přeučen, aby k takovému omylu nedošlo.

Algoritmus:

- Přiřazení tříd T vstupním tréninkovým vektorům $(x_1, x_2, \dots, x_n)$. Inicializace referenčních vektorů. Inicializace parametru učení (α).

1. Pokud není splněna podmínka ukončení, provádět kroky (2 až 5).
2. Pro každý vstupní vektor $x = (x_1, \dots, x_n)$ opakovat kroky 3 až 4.
3. Nalezení takového J, že $c = \arg min \|x - \omega_J\|$.
4. Aktualizace váhových hodnot ω_J:
   pokud $T = C_J$, pak
   $$\omega_J(t+1) = \omega_J(t) + \alpha(t)\left[x(t) - \omega_J(t)\right]$$
   pokud $T \neq C_J$, pak
   $$\omega_J(t+1) = \omega_J(t) - \alpha(t)\left[x(t) - \omega_J(t)\right]$$
5. Aktualizace parametru učení (zmenšení jeho hodnoty).
6. Test podmínky ukončení.

Kde C_J je třída j-tého neuronu, ω_J je váha j-tého neuronu, t reprezentuje diskrétní čas a $\|x - \omega_J\|$ euklidovská vzdálenost mezi vstupním vektorem x váhovým vektorem ω_J.

Parametr učení musí splňovat pravidlo: $0 < \alpha(t) < 1$, přičemž se ve většině případů každou iteraci zmenšuje od původní nastavené hodnoty až k nule. To zaručuje ukončení procesu adaptace. Samotný parametr učení by měl být ze začátku nastaven na hodnotu spíše menší (0.1), než větší. Je to ovšem případ od případu. Větší α na začátku značí rychlejší oddělení neuronů.

2.5.4.2 LVQ3

Klasifikační rozhodování je u LVQ3 stejné, jako u LVQ1, kdežto učení je rozdílné[1]. Máme-li váhové vektory ω_i a ω_j, které jsou nejblíže vektoru x, potom mohou nastat jen 2 situace. Buď ω_j a x spadají do téže třídy a ω_i a x spadají do rozdílných anebo všechny patří do stejné třídy. K tomu $x$musí spadat do tzv. „okénka“, které je definováno kolem středo-roviny ω_i a ω_j.

$$min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s \qquad 2.5$$

Kde $s = \frac{1-w}{1+w}$. Za předpokladu, že d_i a d_j jsou euklidovské vzdálenosti od vstupního vektoru x a w je šířka okénka. Pro první případ se váhové vektory učí podle rovnic:

$$\begin{aligned} \omega_i(t+1) &= \omega_i(t) - \alpha(t)[x(t) - \omega_i(t)] \\ \omega_j(t+1) &= \omega_j(t) + \alpha(t)[x(t) - \omega_j(t)] \end{aligned} \qquad 2.6$$

Kde ω_i a ω_j jsou dva nejbližší váhové vektory ke vstupnímu vektoru x. Přičemž x a ω_j patří do stejné třídy a zároveň x a ω_i patří do různých tříd. Kromě toho x musí spadat do okénka. Pro druhý případ se rovnice mění:

$$\begin{aligned} \omega_i(t+1) &= \omega_i(t) + \varepsilon\alpha(t)[x(t) - \omega_i(t)] \\ \omega_j(t+1) &= \omega_j(t) + \varepsilon\alpha(t)[x(t) - \omega_j(t)] \end{aligned} \qquad 2.7$$

Doporučená hodnota ε se uvádí mezi 0.1 – 0.5 pro okénko velikosti $w = 0.2$ až 0.3. S užšími okénky je vhodné nastavovat menší ε.

Tento algoritmu se zdá být samostabilizující. To znamená, že dostane-li se vektor ω_i do optimální polohy, pokračování v učení tuto polohu nezmění (nezhorší).

3 Realizační část

Cílem práce bylo vyzkoušet algoritmy typu LVQ pro hledání ERP komponent a to hlavně vlny P300. Dále je implementovat v prostředí MATLAB a to buď jako plugin do BCIlabu nebo EEGlabu. Tyto algoritmy se zde budou používat za účelem klasifikace mozkových vln v EEG datech.

Programy EEGLAB a BCILAB jsou k dispozici zdarma. Každý z nich je však nutno nainstalovat jako toolbox do prostředí MATLAB.

3.1 Plugin do programu BCILAB

BCILAB je program založený na EEGLABu a používá funkce ze staré verze EEGLABu. Program obsahuje spoustu algoritmů (LDA, SVM, atd.) pro klasifikaci ERP komponent a to jak v offline, tak v online režimu.

3.1.1 Implementace a výsledky

V BCILABu je nutno implementovat jednu funkci pro vytvoření modelu z dat a druhou pro klasifikaci (predikce). První problém je ten, že v manuálu se nic takového člověk nedozví a musí si projít kód nejlépe v debuging módu. Druhý problém přizpůsobení vstupních a výstupních parametrů těchto funkcí. Musí obsahovat proměnné, které nemusíte u svého algoritmu vůbec využít. Nejen z těchto důvodů považuji BCILAB za vývojářsky nepřívětivý.

Pro splnění všech požadavků pro přidání rozšíření do tohoto programu jsem narazil na další překážky. Jedna z nich byla chybějící průměrovací metoda v procesu předzpracování signálu. Další byla například metoda, která ze vstupních vektorů vyvářela zprůměrované vzorky neregulovatelně zkreslené.

Nepovažuji tento program za špatný, avšak poukazuji na to, že vzhledem k povaze mého úkolu byl BCILAB nevhodný na vytvoření rozšíření.

3.2 Plugin do programu EEGLAB

EEGLAB je nástroj na zpracování dat typu eeg umožňující jeho další zkoumání.

Oproti předchozímu programu je EEGLAB mnohem přívětivější. EEGLAB sám při startu nalezne všechny pluginy v kořenovém adresáři nebo v adresáři „Plugins“. Jediná důležitá věc je, aby tato rozšíření obsahovala soubor nazvaný „eegplugin_*“, přičemž místo hvězdičky stačí jakýkoliv text. Více o rozšířeních do EEGLABu naleznete zde: [5].

3.2.1 ERPLAB

Mluvím-li o rozšířeních EEGLABu, je důležité zmínit ERPLAB. Tento zásuvný modul je zdarma ke stažení a nabízí spoustu funkcí ke zkoumání ERP vln, které v EEGLABu chybí. Já sám jsem jeho funkcí využil k segmentaci, filtraci a následnému zobrazení epoch jednotlivých tříd, které umožňuje libovolně zprůměrovat.

3.3 Načtení signálu

Do EEGLABu se musí načíst soubor formátu Brain Vision. Je nutno mít hlavičkový textový soubor s informacemi o naměřených datech (počet kanálů, informace o filtraci atd.) s koncovkou *.VHDR, dále binární soubor s eeg signály na jednotlivých kanálech - *.eeg a soubor s pozicemi návěští (marker) odpovídající jednotlivým epochám - *.VMRK. Načtením souboru *.VHDR se načtou i ostatní soubory.

3.4 Předzpracování signálu

Mnou vytvořený plugin je jakási nadstavba celého programu. Je tedy nutno vstupní signál alespoň prvotně předzpracovat. EEGLAB umožňuje některá menu a podmenu znepřístupnit, nejsou-li splněny nějaké podmínky předzpracování.

3.4.1 Segmentace

Po načtení signálu ze souboru musíme data rozdělit na jednotlivé epochy. Tomuto procesu se říká segmentace. Segmentaci lze zařídit v toolboxu ERPLAB, který se doporučuje používat společně s mým pluginem. Před extrahováním epoch z celkového signálu se musí nastavit místo, kde epocha začíná. Jedná se tzv. korekci základny (baseline). Pro zkoumání ERP vln se doporučuje rozdělovat epochy po 1 sekundě, přičemž epocha začíná 200ms před stimulem a končí 800ms po podnětu.

3.4.2 Filtrace

Filtrování signálu je již jen doporučený způsob předzpracování. Není nutný k použití mého rozšíření, ale vzhledem k rušivým frekvencím, které občas soubory EEG obsahují je silně doporučen.

Pozn.: Z pokusů, co jsem prováděl na různě frekvenčně zkreslených signálech, se nejlépe jevil filtr IIR Butterworth z ERPLABu s nastavením:

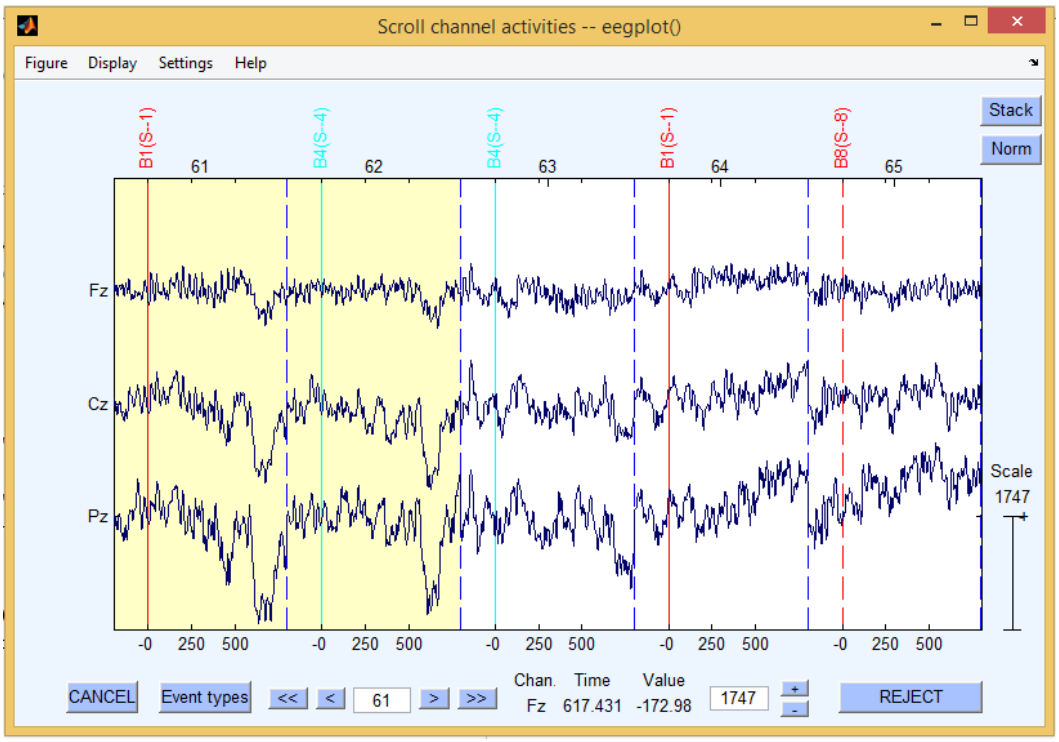
- Dolní propust – 30Hz
- Roll-off – 160 dB/dec

Spolehlivě odstranil všechen výskyt síťové frekvence (50Hz) a zkreslení ERP vln a úbytek napětí byly minimální.

3.4.3 Odstranění artefaktů

ERPLAB nabízí automatickou detekci artefaktů jako je mrknutí nebo posun oka. Ze svých zkušeností vím, že pro úplnou detekci artefaktů je potřeba EEG soubor projít osobně (Obr. 3.1). Například mrknutí může mít různě velikou amplitudu nebo se může vyskytovat vícekrát těsně za sebou a to systém neodhalí. Ruční zamítání domnělých artefaktů je však časově velmi náročné.

Po automatické detekci je v EEGLABu nutné zvolit zamítnutí označených epoch (artefact rejection).



Obr. 3.1: Detekce artefaktů v EEGLABu

3.5 Realizace neuronové sítě

Při aplikaci neuronové sítě LVQ jsem se řídil Kohonenovým návrhem [2].

3.5.1 Prvotní návrh

Sám autor těchto algoritmů navrhl jejich zlepšení pojmenované „Optimized LVQ“ (OLVQ1, OLVQ3). Jedná se zlepšení změny parametru učení.

$$\alpha_c(t) = \frac{\alpha_c(t-1)}{1 + s(t)\alpha_c(t-1)} \qquad 3.1$$

Parametr učení je u těchto modifikací přidělen každému váhovému vektoru ω_i. Rovnice učení se mění jen záměnou obecné α za parametr učení pro vítězný vektor (α_c). Proměnná s se mění v závislosti na tom, zdali je klasifikace správná, či nikoliv.

$$s(t) = +1;\ \text{je-li klasifikace správná}$$
$$s(t) = -1;\ \text{je-li klasifikace nesprávná} \qquad 3.2$$

Čili se učící parametr zmenšuje úměrně s kvalitou učení.

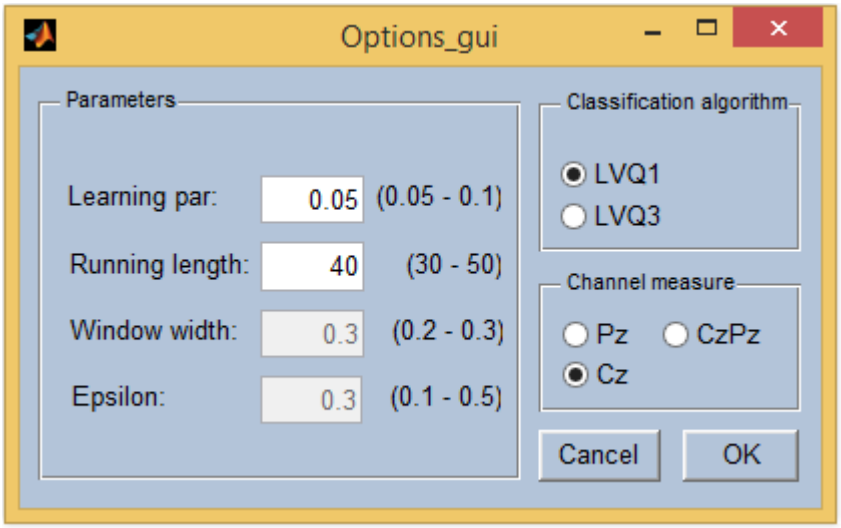
Tento algoritmus jsem zkoušel na naučení modelu. Nemělo však cenu ho nadále používat, protože se samotný algoritmus LVQ1 učí velice dobře beze změny parametru α.

3.5.2 LVQ bez optimalizace

Algoritmus LVQ3 uvedený v 2.5.2 je k dispozici k učení modelu v mém programu spolu s možností nastavení vstupních parametrů (Obr. 3.2).

3.5.3 Modifikace LVQ1

Jak už bylo řečeno, LVQ1 lze navrhnout tak, že není nutná změna parametru učení. Naopak ji nedoporučuji měnit během učení. Podmínkou je učení pouze jednoho zástupce z obou tříd targetů (epochy s ERP vlnami) a non-targetů (vše ostatní). Algoritmus se učí stejně rychle, jako kdyby mu byly předloženy všechny vektory tříd. I výsledkově jsou daleko přesnější. Další výhoda je ta, že na naučené třídy nepůsobí tzv. „unášení tříd“, čili nedochází k přeučení.

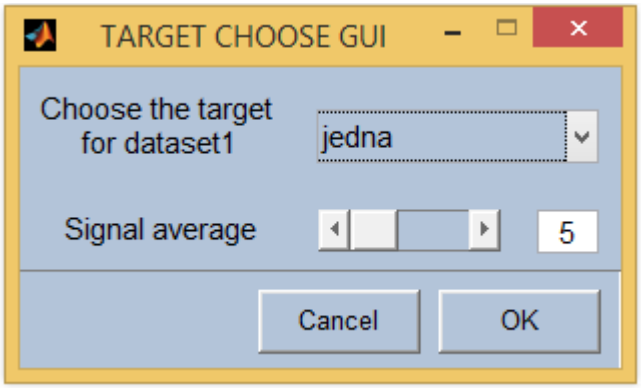


Obr. 3.2: Nastavení klasifikátoru

3.6 Trénování sítí

Jelikož jsou neuronové sítě LVQ typu „učení s učitelem“ (2.5.2), je nutno síť natrénovat a vytvořit tzv. model.

Vstupní předzpracovaná data můj program rozdělí na skupiny dat, kde každá skupina patří jednomu markeru. Učitel dále musí rozhodnout, která skupina odpovídá targetu a nastaví počet průměrovaných dat (Obr. 3.3).



Obr. 3.3: Učení modelu na vstupních datech

Každá skupina dat se omezí na jediného zástupce obsahující právě zprůměrovaná data skupiny. Poté jsou skupiny rozděleny na data obsahující ERP (target) a data neobsahující zkoumané vlny.

Model by neměl obsahovat data jen z jednoho měření. Proto je doporučeno opakovat proces učení i pro další datasety a vytvořit model mnohem komplexnější.

3.7 Klasifikace ERP

Je-li neuronová síť naučena, lze provést klasifikaci vstupního souboru. Soubor musí být (podle 3.4) předzpracován, protože je také nutno rozdělit signál do skupin na základě markerů a následně zprůměrovat minimálně 5 vzorků (epoch). Program k průměrování používá jednoduchou metodu z kapitoly 2.3.3.4. Bereme v potaz, že je epocha obsahující ERP, zprůměrována méně nežli 5x, je nečitelná vůči rozpoznání ERP vln.

K rozpoznání, která ze skupin vstupního signálu patří k targetům, je nutno porovnat podobnost zprůměrovaných vstupních vektorů a modelových vektorů. K tomu slouží následující metody:

3.7.1 Euklidovská vzdálenost

Euklidovská vzdálenost je dána jako vzdálenost dvou bodů x a y o souřadnicích $(x_1, x_2, \dots, x_n)$, $(y_1, y_2, \dots, y_n)$ v n-rozměrném euklidovském prostoru předpisem:

$$D_E = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad 3.3$$

Dosazením vektoru modelu za x a vstupního vektoru za y, dostáváme vzdálenost 2 vektorů. Čím menší, tím je našemu modelu blíže.

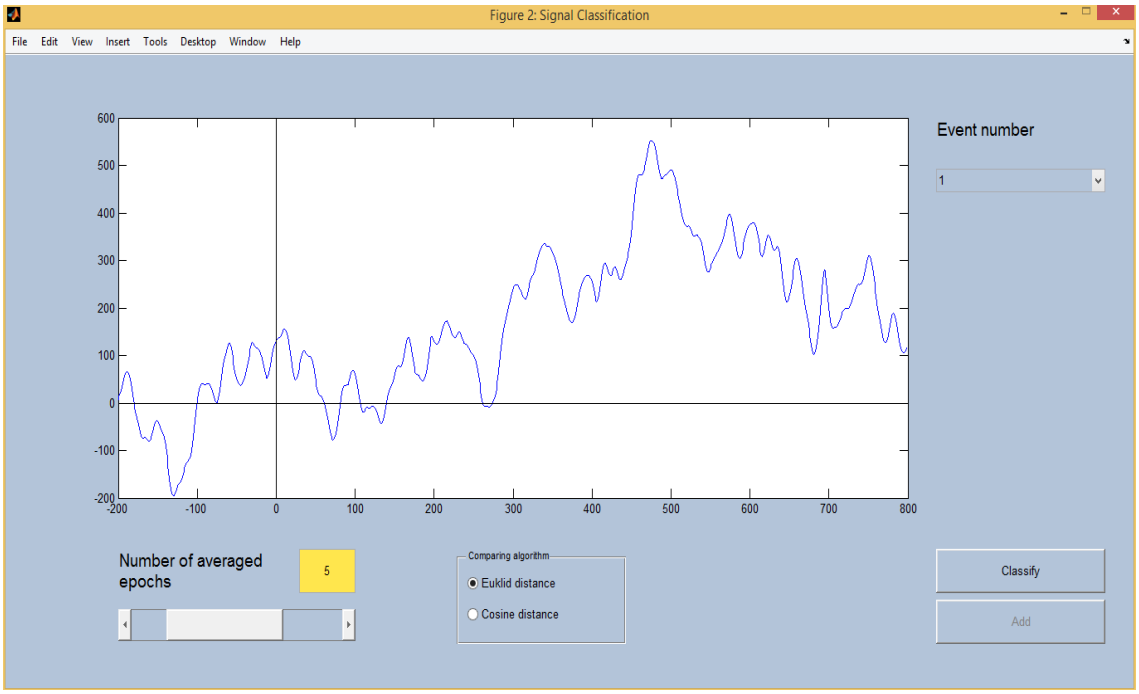
3.7.2 Kosinová vzdálenost

Tato metoda je založena na podobnosti 2 signálů. Vše záleží na úhlech, které signály svírají, čili berou se v potaz jen směry vektorů [8]. Podobnost se počítá podle rovnice:

$$D_c = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_{i=1}^{n} x_i \times y_i}{\sqrt{\sum_{i=1}^{n} {x_i}^2} \times \sqrt{\sum_{i=1}^{n} {y_i}^2}} \qquad 3.4$$

Podobnost se nám může zásadním způsobem hodit. Máme-li například model natrénovaný na signálech s vysokými amplitudami a chceme klasifikovat data, která mají amplitudu malou, ale požadovaný tvar ERP vlny. Podobnost je uvedena celočíselně. Čím větší číslo, tím jsou si signály podobnější.

Obě tyto metody jsou k dispozici při klasifikaci (Obr. 3.4).

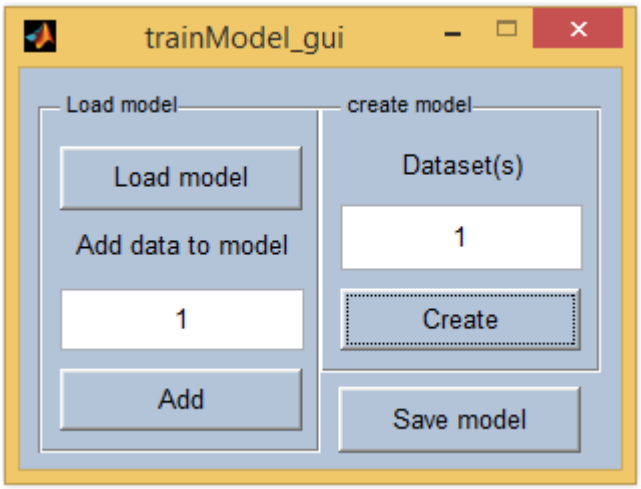


Obr. 3.4: Klasifikační formulář – hlavní okno programu

3.8 GUI

K podrobnému zkoumání evokovaných potenciálů jsem vytvořil grafické rozhraní (Obr. 3.4), kde si lze prohlédnout data jednotlivých markerů s možností průměrování a následné klasifikace (je-li přiložen model). Pro přehlednost je toto GUI v neměnném maximalizovaném stavu.

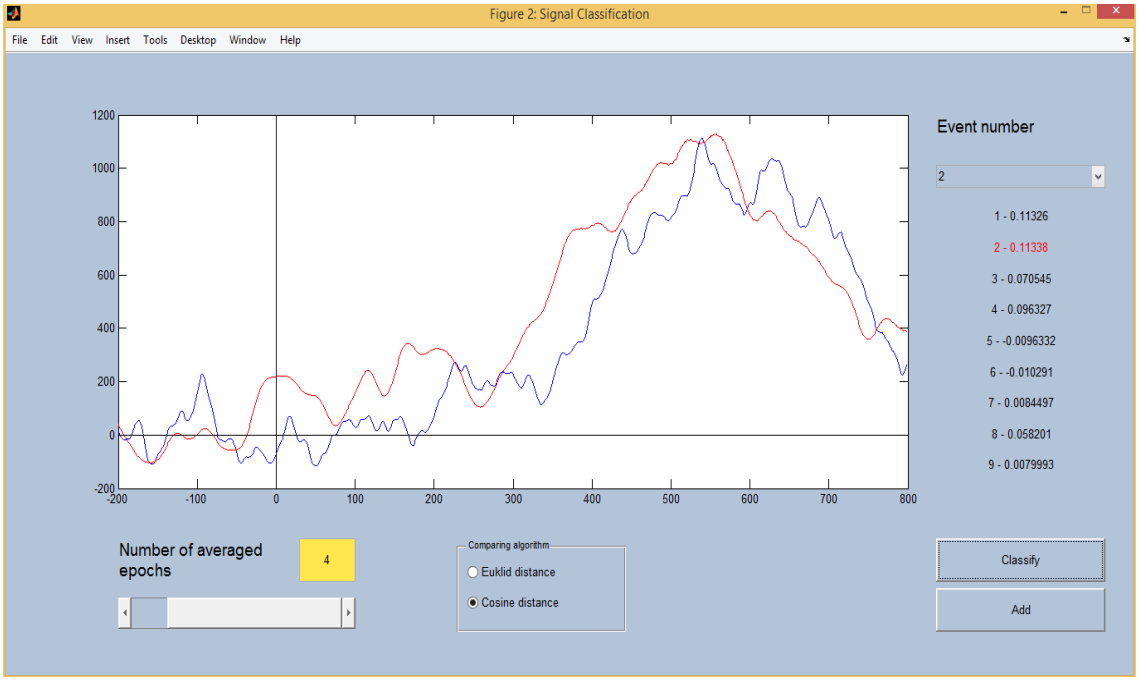
Další a poslední okno (Obr. 3.5) slouží k vytváření, načtení nebo modifikaci modelu. Přidávat signály do stávajícího modelu lze i po klasifikaci v hlavním okně. Více o GUI najdete v uživatelské příručce.



Obr. 3.5: Formulář vytváření modelu neuronové sítě

3.8.1 Zobrazení klasifikace

K zobrazení výsledků klasifikace slouží také hlavní okno. Po výsledném určení targetu je vybrán zprůměrovaný signál vítězné třídy a je zobrazen modře. K němu je zároveň dokreslen (červeně) signál modelu, ke kterému byl nejblíže. Poté se vypíší velikosti vah k daným třídám. Vítězná třída je zobrazena červeným písmem (Obr. 3.6).



Obr. 3.6: Zobrazení výsledku klasifikace

3.9 Implementace

3.9.1 Jazyk a nástroje

Program byl vytvořen v jazyce matlab v prostředí MATLAB R2012a. Měl jsem k dispozici toolboxy zdarma ke stažení EEGLAB 13.3.2, ve kterém je program zasazen jako plugin, ERPLAB, který jsem používal na předzpracování dat a SOMTOOLBOX s algoritmy na základě Kohonenovo [2] myšlenek.

Na spouštění pluginu je nutné mít prostředí MATLAB a program EEGLAB.

3.9.2 Datové úložiště

Plugin používá kořenové úložiště EEGLABu k uchování struktur nastavení (MODEL_SETTINGS), modelu (MODEL) a signálu (EEG). Na toto úložiště lze přistupovat i během vykonávání vnořených funkcí.

3.9.3 Stručný popis funkcí

Program dělí funkce na 3 druhy: „GUIs“, „pop_functions“ a „functions“.

Název funkce	**Popis funkce**
`eegplugin_lvqClassifier.m`	Tato funkce slouží ke spojení pluginu s EEGLABem, k vytvoření struktur v kořenovém úložišti proměnných a k přidání položek menu a jejich podmenu do záložky „Tools“.
`pop_saveModel.m`	Ukládá vytvořený nebo modifikovaný model jako soubor typu **.m* na disk.
`pop_settings.m` `pop_train.m`	Pouští *gui* funkce za účelem změny vstupních parametrů.

`pop_classify.m`
`targetChoose_gui.m`
`Options_gui.m`
`trainModel_gui.m`
	Jednotlivá gui reprezentující položky menu pluginu.

`som_map_struct.m`
`som_randinit.m`
`som_set.m`
`som_topol_struct.m`
`som_train_struct.m`
	Funkce k vytvoření struktury obsahující typ algoritmu, výsledky učení, třídy atd. Slouží také k předdefinování vah k jednotlivým třídám před startem učení.

`lvq1.m` `lvq3.m`	Algoritmy učení typu LVQ.
`trainlvq1.m` `trainlvq3.m`	Funkce, ve kterých se vytváří nebo modifikuje model sítě na základě typu algoritmu.
`separateTarget.m`	Na základě vstupního parametru (třída targetu) rozdělí třídy na targetové a netargetové.
`createDataForModel.m`	Ze vstupních dat vytvoří data srozumitelná pro trénovací funkce.
`predictlvq.m`	Vstupem je model a zkoumaná data a algoritmus, kterým se má klasifikace provádět. Výstupem je struktura s výsledky.
`chooseByEuklid.m`	Provádí výpočet vzdálenosti modelových signálů od

Název funkce	**Popis funkce**
`chooseByCosine.m`	vstupních konkrétním algoritmem.
`AverageAndStore.m`	Metoda, vrací zprůměrovaná data patřící do konkrétních tříd na zadaný počet průměrování.
`binEpochCount.m`	Vrací počet epoch každé třídy.
`easyAverager2.m`	V této funkci probíhá proces průměrování.
`axesOnZero.m`	Vykreslí do grafu kříž na pozici (0, 0).
`getChans.m`	Zjistí z eeg struktury typy kanálů a při existenci kanálů *Cz* a *Pz* vrací jejich pořadí.
`AverageAllBins.m`	Vytváří buněčné pole obsahující všechna možná data zprůměrovaná na minimální až maximální počet epoch.

4 Výsledky testování a měření

V této části budou prezentovány výsledky různých pokusů, které byly prováděny s použitím klasifikátoru na testovacích datech. U každého pokusu bude přiloženo vysvětlení, za jakých podmínek se konal a následná vizualizace v podobě tabulky výsledků nebo grafu.

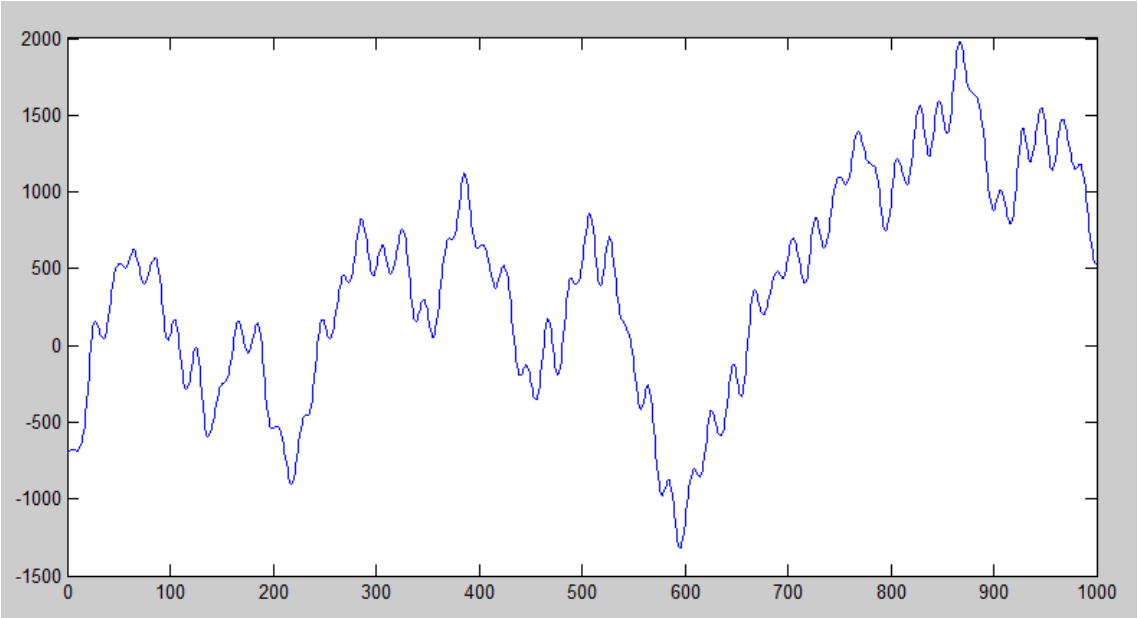
Testy byly prováděny na 20 různých lidech. Jednalo se o děti základních škol, přičemž měření neprobíhalo v nerušeném prostředí. Každé dítě si vybralo číslo od 1 do 9 a ta se poté promítala na monitoru. Měření skončilo, pokud bylo číslo uhádnuto nebo naopak data byla nečitelná a ani po více minutách nebylo možno určit na co testovaný myslí.

Jsou k dispozici data z kanálů Fz, Cz a Pz. Výsledky si je možné ověřit na přiložených datech na CD.

4.1 Vliv předzpracování dat

Jak již bylo řečeno v kapitole 3.4, předzpracování dat je více než doporučeno. Na následujících obrázcích ukážu proč.

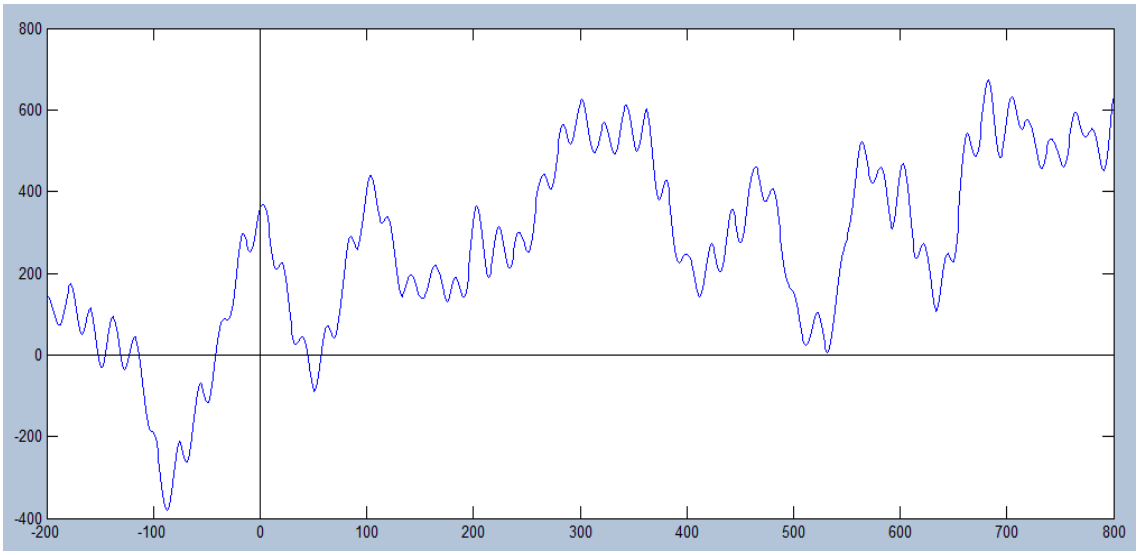
4.1.1 Důsledek mrknutí



Obr. 4.1: Efekt mrknutí

V předchozích kapitolách jsem několikrát zmínil, že se musí odstraňovat mrknutí. Není to ovšem jen z důvodů, že může zkreslit P3 vlnu tak, že ani průměrováním nedocílíme její viditelnosti. Z měření jsem zjistil, že po mrknutí může výrazně nabýt amplituda i do kladných hodnot. To může být problém. V tomto případě (Obr. 4.1) si klasifikátor může mylně vyložit mrknutí jako vlnu typu N, následovanou typem P. Což se po zprůměrování ukázalo jako pravdivé.

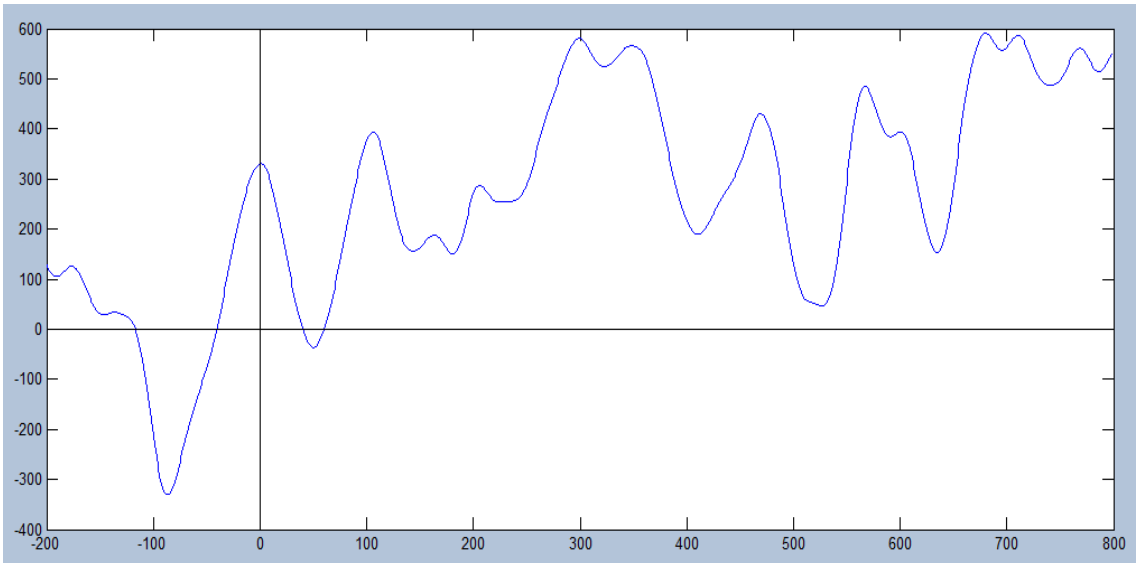
4.1.2 Důsledky filtrace



Obr. 4.2: Epocha dat bez použití filtrace

Na předchozím grafu (Obr. 4.2) je vidět veliké zkreslení signálu frekvencí 50Hz. Tato frekvence vede k tomu, že Euklidovská vzdálenost při porovnávání výrazně naroste díky opakujícím se špičkám. To je efekt, který opravdu nepotřebujeme.

Na rozdíl od takto porušeného signálu, můžeme vidět, že následující obrázek (Obr. 4.3) je vyhlazen od těchto přebytečných frekvencí a je daleko lépe porovnatelný. Ne každý signál je naměřen s těmito frekvencemi, ale přesto je lepší provádět předzpracování u všech testovaných stejným způsobem.

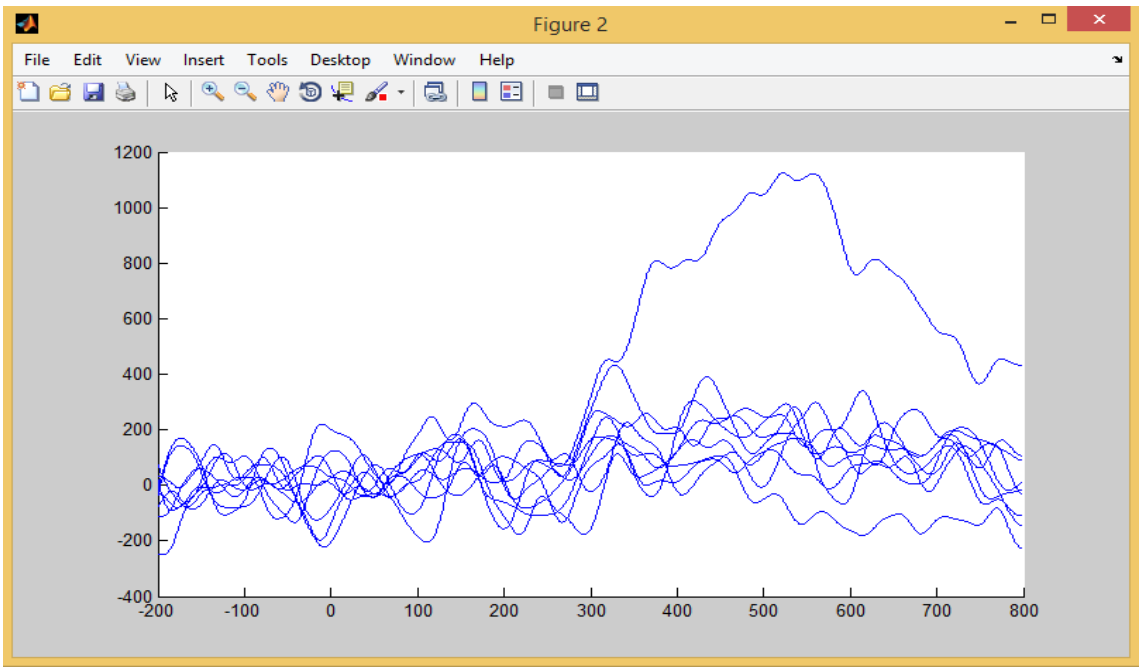


Obr. 4.3: Epocha dat s použitím filtru (dolní propust)

Nevýhoda filtrů je však úbytek napětí, který je viditelný na předchozím obrázku. Není to nic výrazného (0.1 – 0.2 μV), ale s tím třeba počítat. Je také vidět, že se vlny trochu zvýraznily do šířky.

4.2 Učení klasifikátoru

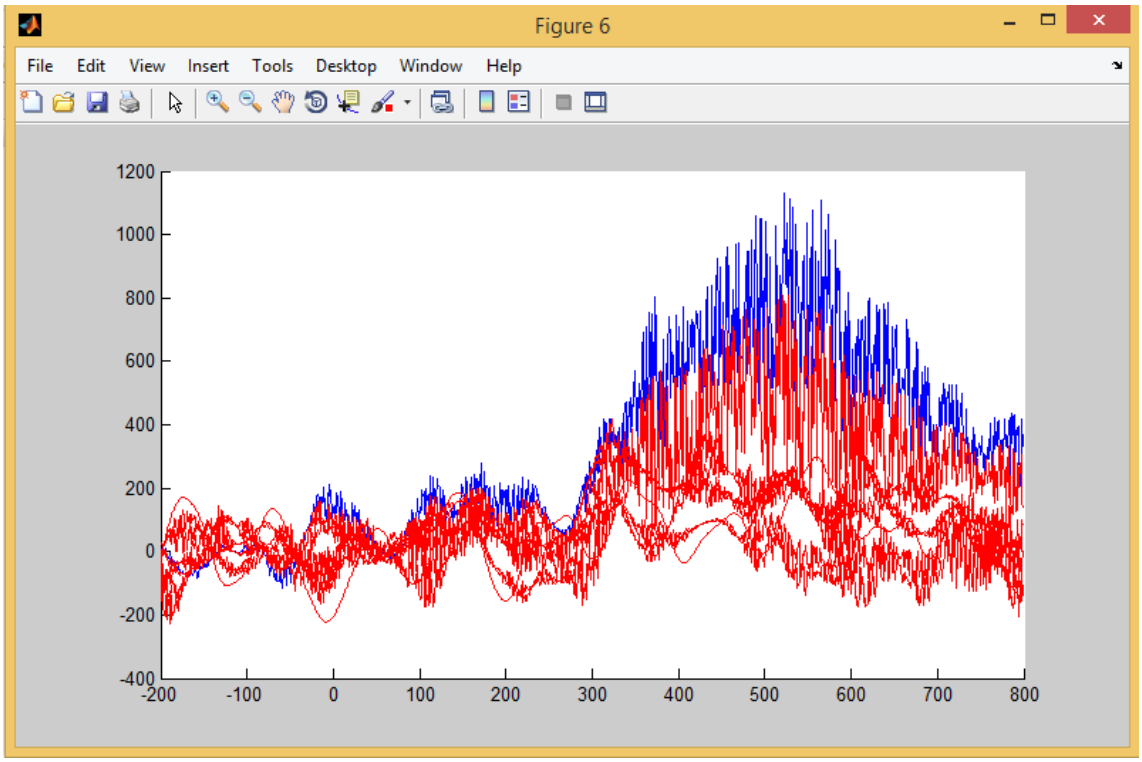
V této kapitole ukáži výsledky učení obou algoritmů LVQ (LVQ1, LVQ3) s různými nastaveními učících parametrů a jak učení vzorů ovlivňují. Poté vyberu vhodnější algoritmus a zdůvodním proč. Nejprve vzorový dataset, ke kterému se budeme chtít přiblížit (Obr. 4.4):



Obr. 4.4: Graf průměrovaných epoch jednoho signálu

U algoritmu LVQ1 je nutno nastavit tzv. učící parametr (α), který zvedá rychlost učení. Udává se mezi 0 (žádné) – 1 (skokové učení). Druhý parametr je počet učících cyklů (rlen), který může a nemusí dělat to samé. Závisí to na tom, jestli se všechny vektory mapy učí.

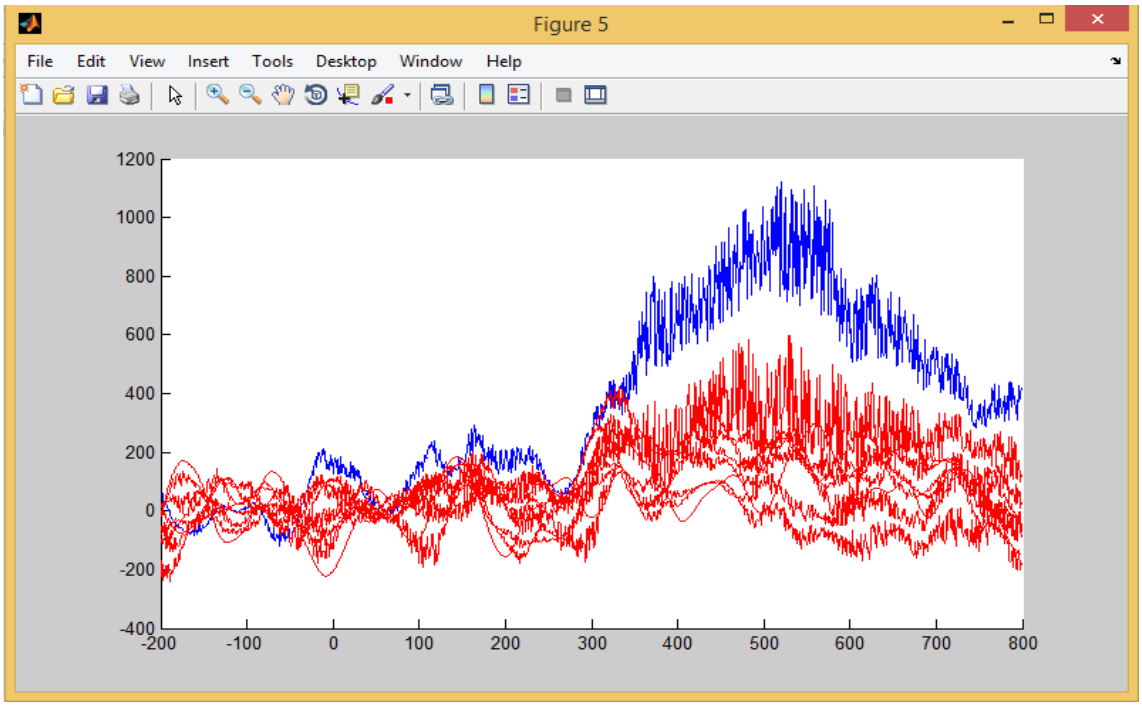
Na prvním obrázku (Obr. 4.5) je vidět naučená síť s nastavením $\alpha = 0.01$ a $rlen = 20$:



Obr. 4.5: Grafické znázornění 1. pokusu naučit síť

První pokus nedopadl dobře (target modře, non-target červeně). Je vidět, že se většina signálů nenaučila. Ve druhém pokusu zvednu počet cyklů.

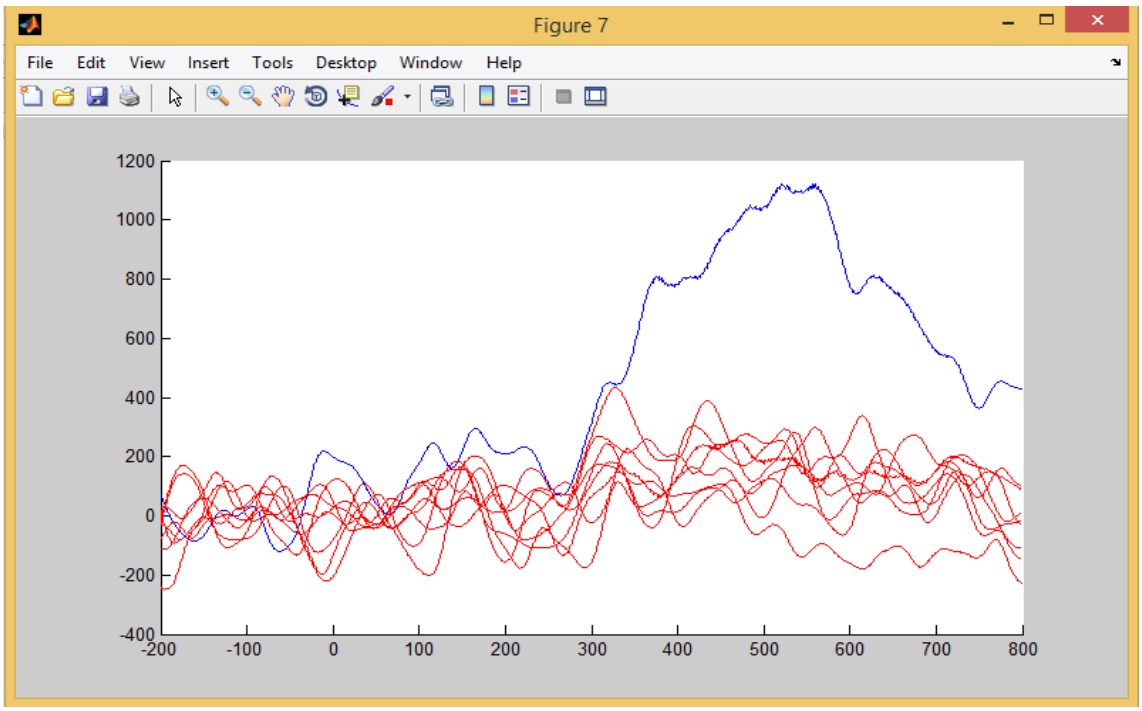
Na druhém obrázku (Obr. 4.6) je vidět naučená síť s nastavením $\alpha = 0.01$ a $rlen = 40$:



Obr. 4.6: Grafické znázornění 2. pokusu naučit síť

Druhý pokus již docela dobře pokrývá výskyt non-targetů a rozdíl od targetu je patrný. Ve třetím pokusu vrátím počet učících cyklů na předchozí hodnotu, ale zvýším učící parametr.

Na třetím obrázku (Obr. 4.7) je zobrazena naučená síť s nastavením $\alpha = 0.1$ a $rlen = 20$:



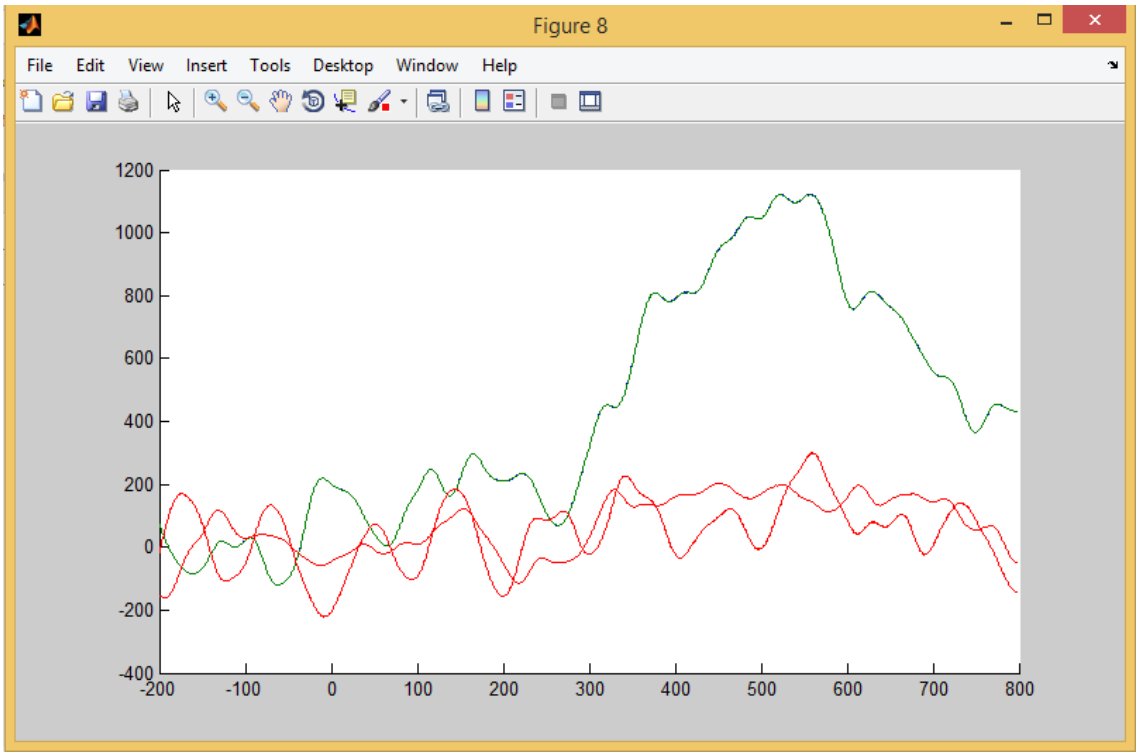
Obr. 4.7: Grafické znázornění 3. pokusu naučit síť

Třetí pokus je již skoro kopií vzorových signálů. Je tedy vidět, že učící parametr trénuje sítě rychleji, nežli počet cyklů. Je také výkonnostně výhodnější. Je tedy vhodné tyto parametry vhodně zkombinovat k libovolným výsledkům.

Na rozdíl od LVQ1, nemáme u algoritmu LVQ3 možnost učit vektory mapu po 2 vektorech. U tohoto učení je totiž základním pravidlem, že hledáme ke vstupnímu vektoru 2 nejbližší. Takže vítězné by byly pořád. Necháváme tedy algoritmus naučit svoji mapu libovolně ze všech vstupních vektorů. Omezíme jen velikost mapy. Přibyly nám 2 učící parametry (window, epsilon).

Rozhodl jsem se pro vstupní mapu o velikosti 10 vektorů, 5 pro každou třídu. Velikost sítě na výstupu z algoritmu může být však jiná. V LVQ3 se totiž soutěží o možnost učení a je v celku pravděpodobné, že se po prvotních pár cyklech budou učit už jen vítězné.

Na čtvrtém obrázku (Obr. 4.8) je zobrazena naučená síť s nastavením $\alpha = 0.1$, $rlen = 20$, $win = 0.3$ a $epsilon = 0.3$:



Obr. 4.8: Grafické znázornění pokusu naučit síť algoritmem LVQ3

Má premisa byla správná. Síť se naučila pouze 3 signály. Z důvodu náhodných vah vstupní mapy předem nevíme, které vektory patřící konkrétním signálů budou vítězné. Proto při měření výsledků používám zásadně trénování algoritmem LVQ1.

4.3 Měření v jednotlivých fázích

Na následujících stránkách otestuji, jak se chová klasifikátor na jednotlivých kanálech Cz, Pz a jak se zachová, spojím-li tyto kanály v jeden. Předem upozorňuji, že nebudu používat kanál Fz a to z důvodů jeho nečitelnosti.

Pro každý z těchto testů bude vytvořena neuronová síť ze stejných dat (4 datasetů), avšak na jiném kanálu. Následně budou s každým testovacím datasetem prováděny stejné akce:

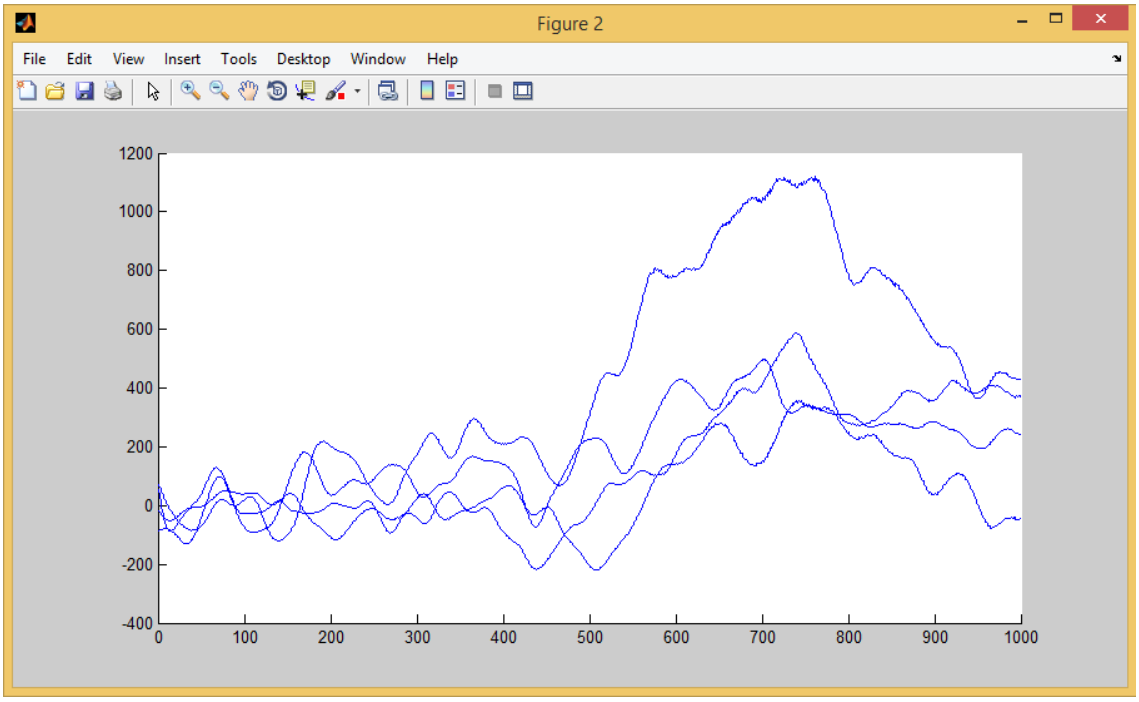
- Měření v jednotlivých fázích předzpracování zprůměrovaných 5 prvních epoch každé třídy.
- Výpočet jak Euklidovské, tak Kosinové vzdálenosti k třídě zvolené klasifikátorem. Byla-li tato klasifikace špatná, zaznamenám vzdálenosti i k třídě označené jako targetové.
- Nebyla-li klasifikace úspěšná u signálu zprůměrovaného po 5 epochách, zvyšujeme počet průměrovaných epoch tak dlouho, dokud není bezpečně určen target nebo již nelze třídy více zprůměrovat (v takovém případě je v tabulce označen křížkem).

Cílem je zjistit vliv předzpracování na úspěšnost klasifikace, nejlepší kanál na měření ERP a vhodný počet zprůměrovaných epoch, při kterém bychom měli s jistotou určit vítěznou třídu (u těchto testů je touto vítěznou třídou hádané číslo).

Kvůli omezující velikosti stránky jsem se rozhodl zavézt některé zkratky popisků tabulky:

- Set – číslo měření (dataset)
- Terč – hádané číslo (target)
- Seg – segmentovaná data
- Filtr – filtrovaná data
- Minimální počet průměrů – kolikrát epoch bylo za potřebí k nalezení správné targetové třídy

Model LVQ1 pro kanál CzPz



Obr. 4.9: Model z epoch obsahující ERP

Vybral jsem si 4 první datasety, jež posloužili jako základ k učení neuronové sítě. Výsledné vlny (Obr. 4.9) by měly dostatečně pokrývat výskyt vlny P300. LVQ1 bylo naučeno s parametrem učení 0.05 a v 50 učících cyklech.

LVQ1		Fáze předzpracování			Euklidovská		Minimální počet průměrů		
Set	Terč	Seg	Bez artefaktů	Filtr	Vzdálenost	Vzdálenost k terčovému	Seg	Bez artefaktů	Filtr
1	9	6	7	7	4318	7173	X	X	X
2	4	4	5	5	4604	5051	5	15	15
3	5	3	3	3	4723	6483	15	10	10
4	7	9	9	9	3286	5319	9	7	7
5	9	6	3	3	5023	5779	X	X	X
6	5	5	5	5	6727		5	5	5
7	8	6	6	6	5617	4999	6	6	6
8	3	4	4	4	4162	6459	12	12	12
9	9	2	6	6	4370	4660	16	12	12
10	7	9	7	7	3582		6	5	5
11	3	3	3	3	3850		5	5	5
12	3	5	3	3	5888		12	5	5
13	5	5	5	4	4219	4780	5	5	X
14	5	7	3	3	5695	5260	X	X	X
15	5	7	7	7	4253	5495	7	7	7

LVQ1		Fáze předzpracování			Euklidovská		Minimální počet průměrů		
Set	Terč	Seg	Bez artefaktů	Filtr	Vzdálenost	Vzdálenost k terčovému	Seg	Bez artefaktů	Filtr
16	7	8	7	7	4200		7	5	5
17	9	3	3	3	7452	10479	7	8	8
18	2	2	2	2	11350		5	5	5
19	6	8	8	8	5535	8442	7	12	12
20	2	2	2	2	6484		5	5	5

Tab. 4.1: Výsledky na kanálu CzPz - Euklidovskou vzdáleností

Úspěšnost klasifikátoru s použitím Euklidovi vzdálenosti (Tab. 4.1) považuji na tomto kanálu za velice špatnou. Data s artefakty měla úspěšnost 30%, bez artefaktů 40% a s filtrem 35% při zprůměrování 5 epoch. Úspěšnost bychom zvednutím průměru o moc nenavýšili.

LVQ1		Fáze předzpracování			Kosinová		Minimální počet průměrů		
Set	Terč	Seg	Bez artefaktů	Filtr	Vzdálenost	Vzdálenost k terčovému	Seg	Bez artefaktů	Filtr
1	9	8	8	8	0.064642	0.079261	18	6	6
2	4	5	5	5	0.069933	0.41566	X	X	X
3	5	1	5	5	0.078776		7	5	5
4	7	9	9	9	0.064847	0.090842	9	8	8
5	9	5	9	9	0.37703		X	5	5
6	5	5	5	5	1.0013		5	5	5
7	8	8	8	8	0.065127		5	5	5
8	3	3	3	3	0.12866		5	5	5
9	9	2	6	6	0.015689	0.37138	6	6	6
10	7	7	7	7	0.098967		5	5	5
11	3	3	3	3	0.063689		5	5	5
12	3	3	3	3	0.11114		5	5	5
13	5	4	4	4	0.078035	0.11674	8	X	X
14	5	3	3	3	0.11985	0.29295	X	18	18
15	5	5	5	5	0.056366		5	5	5
16	7	7	7	7	0.027978		5	5	5
17	9	9	9	9	0.076553		5	5	5
18	2	2	2	2	0.083059		5	5	5
19	6	6	6	6	0.073985		5	5	5
20	2	2	2	2	0.062586		5	5	5

Tab. 4.2: Výsledky na kanálu CzPz - Kosinovou vzdáleností

Naopak pro Kosinovou vzdálenost se zdá toto měření (Tab. 4.2) velice dobré. Úspěšnost klasifikace dat s artefakty 60%, bez artefaktů 70% a pro filtrovaná taktéž 70%. Úspěšnost zde kazí hlavně datasety 2, 13 a 14, o kterých krátce pohovořím na konci této kapitoly.

LVQ1		Fáze předzpracování			Euklidovská		Minimální počet průměrů		
Set	Terč	Seg	Bez artefaktů	Filtr	Vzdálenost	Vzdálenost k terčovému	Seg	Bez artefaktů	Filtr
1	9	9	9	9	5294		5	5	5
2	4	4	6	5	4886	5205	5	15	10
3	5	5	1	5	5763		5	15	5
4	7	9	9	9	3703	4764	9	7	6
5	9	1	1	3	5811	6348	6	6	7
6	5	5	5	5	7798		5	5	5
7	8	8	8	6	4029	4127	5	5	7
8	3	3	3	3	5792		5	5	5
9	9	2	6	9	4399		16	X	5
10	7	9	7	9	4044	4295	6	5	6
11	3	3	3	5	4739	4173	5	5	9
12	3	2	3	3	5622		12	5	5
13	5	4	4	4	4297	6773	8	X	X
14	5	3	3	3	5055	6494	X	X	X
15	5	5	5	5	4644		5	5	5
16	7	7	6	7	4479		5	8	5
17	9	5	5	9	5062		X	X	5
18	2	2	2	2	8435		5	5	5
19	6	6	4	6	6150		5	9	5
20	2	2	2	2	3535		5	5	5

Tab. 4.3: Výsledky na kanálu Pz - Euklidovskou vzdáleností

Klasifikátor měl úspěšnost 60% na datech s artefakty, 55% na datech, kde již byly odstraněny a 60% na filtrovaných. Je nutno říci, že rozdíl v úspěšnosti mezi měřeními s artefakty a bez artefaktů může být právě způsoben tím, že targetové třídy nebyly poškozeny artefakty, ale ostatní ano.

Ze všech měřených (Tab. 4.3) se zdá být na tom s úspěšností nejlépe pokus s filtrováním. Procentuálně dopadl stejně, jako první test, ale podíváme-li se na počet průměrovaných vzorků, který už zaručil uhádnutí targetu, je značně lepší.

LVQ1		Fáze předzpracování			Kosinová		Minimální počet průměrů		
Set	Terč	Seg	Bez artefaktů	Filtr	Vzdálenost	Vzdálenost k terčovému	Seg	Bez artefaktů	Filtr
1	9	3	3	3	0.036732	0.057109	10	9	9
2	4	5	5	5	0.12316	0.44236	X	X	X
3	5	5	5	5	0.067471		5	5	5
4	7	9	9	9	0.10349	0.15185	9	6	6
5	9	5	1	1	0.30529	0.3291	X	X	20
6	5	1	9	9	0.41349	0.74055	6	X	X
7	8	8	8	8	0.042191		5	5	5
8	3	3	3	3	0.10905		5	5	5
9	9	2	6	6	0.10098	0.24899	6	9	9
10	7	7	7	7	0.13671		5	5	5
11	3	6	6	6	0.067997	0.074377	6	7	7
12	3	3	2	2	0.083104	0.088087	5	8	8
13	5	4	4	4	0.11514	0.29722	X	X	X
14	5	3	3	3	0.1095	0.28915	X	X	X
15	5	5	5	5	0.063315		5	5	5
16	7	9	7	7	0.037308		6	5	5
17	9	9	9	9	0.032516		5	5	5
18	2	2	2	2	0.057732		5	5	5
19	6	6	6	6	0.045504		5	5	5
20	2	2	2	2	0.035169		5	5	5

Tab. 4.4: Výsledky na kanálu Pz - Kosinovou vzdáleností

Úspěšnost klasifikátoru u kanálu Pz s Kosinovou vzdáleností bylo 50% u dat s artefakty, 50% u dat bez artefaktů a 50% u filtrovaných (Tab. 4.4). Drobným zvýšením průměrování (např. na 8) lze úspěšnost výrazně zlepšit.

LVQ1		Fáze předzpracování			Euklidovská		Minimální počet průměrů		
Set	Terč	Seg	Bez artefaktů	Filtr	Vzdálenost	Vzdálenost k terčovému	Seg	Bez artefaktů	Filtr
1	9	6	7	6	4255	5245	19	X	X
2	4	5	5	5	5074		X	X	X
3	5	1	3	5	4759	6167	7	15	5
4	7	9	9	9	3407	6449	7	8	7
5	9	5	9	6	6212		6	5	6
6	5	5	5	5	10242	4614	5	5	5
7	8	6	6	6	3486		X	8	X
8	3	3	3	3	8102		5	5	5
9	9	2	6	9	5571	3824	X	14	5

10	7	7	7	9	3376	3625	5	5	6
11	3	9	9	9	3162	4867	6	11	7
12	3	4	3	4	7240		12	5	10
13	5	4	4	5	4210	5304	8	X	5
14	5	3	3	3	5280	4868	X	X	X
15	5	7	7	7	4378		7	6	6
16	7	7	7	7	6993		5	5	5
17	9	9	9	9	8576		5	5	5
18	2	2	2	2	9721	7148	5	5	5
19	6	6	7	5	4836	6964	5	14	12
20	2	1	1	1	2822		6	6	6

Tab. 4.5: Výsledky na kanálu Cz - Euklidovskou vzdáleností

Klasifikátor na kanálu Cz s Euklidovou vzdáleností dosahoval druhých nejhorších výsledků (Tab. 4.5). U dat s artefakty 35%, u dat bez artefaktů 40% a u filtrovaných taktéž 40%. I zde by šla zvýšit úspěšnost klasifikace a to zvednutím průměrování přibližně na 7.

LVQ1		**Fáze předzpracování**			**Kosinová**		**Minimální počet průměr**		
Set	**terč**	**Seg**	**Bez artefaktů**	**Filtr**	**Vzdálenost**	**Vzdálenost k terčovému**	**Seg**	**Bez artefaktů**	**Filtr**
1	9	5	4	4	0.064812	0.13271	14	9	9
2	4	5	5	5	0.05142	0.28011	X	X	X
3	5	1	2	2	0.16853	0.17182	7	6	6
4	7	9	7	7	0.058917		7	5	5
5	9	5	5	5	0.61028	0.62202	X	6	6
6	5	5	5	5	1.2897		5	5	5
7	8	8	8	8	0.10042		5	5	5
8	3	3	3	3	0.1051		5	5	5
9	9	9	2	2	0.4086	0.60626	5	X	X
10	7	7	9	9	0.087288	0.11134	5	6	6
11	3	6	3	3	0.073311		6	5	5
12	3	3	4	4	0.26052	0.28836	5	X	X
13	5	4	4	4	0.062978	0.094144	X	X	X
14	5	3	3	3	0.13228	0.36736	X	18	18
15	5	5	5	5	0.073214		5	5	5
16	7	9	7	7	0.047654		6	5	5
17	9	9	9	9	0.12755		5	5	5
18	2	2	2	2	0.1431		5	5	5
19	6	6	6	6	0.082875		5	5	5

LVQ1		Fáze předzpracování			Kosinová		Minimální počet průměr		
Set	terč	Seg	Bez artefaktů	Filtr	Vzdálenost	Vzdálenost k terčovému	Seg	Bez artefaktů	Filtr
20	2	2	2	2	0.051025		5	5	5

Tab. 4.6: Výsledky na kanálu Cz - Kosinovou vzdáleností

I u tohoto kanálu vychází výsledkově měření s Kosinovou vzdáleností lépe (Tab. 4.6). 55% pro data s artefakty, 55% pro data bez artefaktů a 55% pro filtrovaná data. A jako ve všech proběhlých měřeních na těchto datasetech 2, 13 a 14 jsou nesprávně zařazena.

4.4 Měření se zvětšováním modelu

Jako poslední experiment (Tab. 4.7)jsem provedl měření na výsledkově nejlepším kanálu s tím, že se bude naučená síť zvětšovat o správně přiřazené vzory. Tím by teoreticky bylo možné dosáhnout větší přesnosti. Tentokrát začnu od datasetu 20 až k 1.

LVQ1		CzPz	
Dataset	Target number	Kosinová vzdálenost	Euklidovská vzdálenost
20	2	2	2
19	6	6	6
18	2	2	2
17	9	9	9
16	7	7	7
15	5	5	5
14	5	3	3
13	5	4	4
12	3	3	4
11	3	6	3
10	7	7	7
9	9	6	2
8	3	3	4
7	8	8	6
6	5	5	5
5	9	5	3
4	7	7	9
3	5	5	3
2	4	5	5
1	9	3	6

Tab. 4.7: Výsledky měření s rozšiřující se neuronovou sítí

Je vidět, že nám úspěšnost klesla u Kosinové vzdálenosti o 5%, zato u Euklidovské se zvětšila o 15%. Průměrování bylo nastaveno na 5 epoch.

4.5 Shrnutí měření

Testy nedopadli tak dobře, jak bych se byl domníval. Nebylo to však ovlivněno tím, jak dobře se učí neuronová síť. Problémy se týkaly převážně dat. Ta byla vybrána náhodně, tudíž ač jsem si mohl vybrat nejlépe naměřené, neudělal jsem to, protože by byly testy zidealizované a o ničem by nevypovídaly.

Hlavní zjištění je, že Kosinová vzdálenost je výsledkově lepší, dále spojení kanálů Cz a Pz může posunout amplitudu vlny P300 a tím trochu zkreslit představu o tom, za jak dlouho po stimulaci se vlna vykreslí. Další zjištění je, že zvednutí průměru epoch z 5 na 7 výrazně pomůže přesnosti klasifikace.

Předpoklad, že filtrovaná data budou lepší ke klasifikaci, se nepotvrdil. Statisticky vychází úspěšnější klasifikace dat bez filtrů. Data 2, 13 a 14 nebývala moc často klasifikována správně a to zapříčiněním posunuté amplitudy P300 nebo třídy, která byla po zprůměrování výraznější v oblasti po 300ms.

Poslední měření s přidáváním vzorů do neuronové sítě byl na jednu stranu pozitivní a na druhou ne. Důvodem může být zvětšující se počet naučených non-targetových dat, který je mnohonásobně vyšší, než targetových. Klasifikátor pak začíná přiřazovat třídy jako netargetové.

5 Závěr

V této bakalářské práci bylo hlavním úkolem zjistit, zda lze v oblasti BCI vhodně využít neuronové sítě LVQ. Vycházelo se z předpokladu, že když fungují Kohonenovy sítě typu SOM, mělo by být možné použít LVQ podobným způsobem. Dále bylo nutné se seznámit se signály EEG, pochopit problematiku kolem měření a prostudovat teorii kolem komponent, nacházejících se uvnitř těchto signálů. Právě pro ERP komponenty bylo třeba navrhnout klasifikátor s použitím neuronové sítě LVQ a vyzkoušet robustnost tohoto klasifikátoru na určitém počtu lidí.

Nejprve bylo nutno navrhnout a otestovat metody LVQ. Návrh probíhal podle teoretických předloh od Kohonena [2]. Posléze jsem vybral metodu, která byla úspěšnější učení vzorů (LVQ1). Tuto metodu jsem následně upravil, jak bylo řečeno v kapitole 3.5.3. Dále jsem tuto metodu aplikoval v klasifikátoru, který jsem přidal do zvoleného programu EEGLAB jako zásuvný modul a klasifikaci otestoval na 20 různých lidech v různých fázích předzpracování signálu za stejných podmínek.

Mnou navržený klasifikátor se nezdál výsledkově moc úspěšný, ale potvrdil se předpoklad, že alespoň částečné předzpracování signálu vede ke zlepšení úspěšnosti klasifikace. Veškeré testy také ukázaly, že by měl být model robustnější. A to hlavně kvůli posunuté latenci vlny P300. Ukázalo se, že pro vybraná data, kde jsem zkorigoval základnu na $-200ms$; $+800ms$, dosahovala tato komponenta vrcholu kolem $500ms$ a existovaly případy, které ji měly ještě dál. Navrhnutý model by tedy měl pokrývat teoreticky i pozdější latence. Proto pro další vývoj tohoto zásuvného modulu by bylo dobré otestovat průměrovací metodu s posouváním latencí.

Bylo také zjištěno, že každý testovaný subjekt má rozdílnou sílu signálu. U většiny testovaných byla amplituda komponenty na začátku větší, než na konci, avšak u některých tomu bylo naopak. Dále lidé, u kterých byla změřena vlna P300 průměrně větší (cca 12 μV), měli celkový signál měřený ve vyšších napětích a těch, co měli amplitudu naopak. Nabízí se otázka, zda by nebylo vhodné implementovat algoritmus na škálování signálu. Nehodlám se zde dále zabývat teoretickými otázkami, ale jsem přesvědčen, že by to stálo za otestování.

Celkově se dají sítě LVQ označit, jako vhodné pro klasifikaci ERP komponent. Je však nutno celkový BCI systém dobře navrhnout.

Seznam použitých zkratek

EEG	Electroencephalography (Elektroencefalografie)
EP	Evoked Potentials (Evokovaný potenciál)
MEP	Motoric Evoked Potentials (Motoricky evokované signály)
VEP	Visual Evoked Potentials (Zrakově evokované potenciály)
AEP	Acoustic Evoked Potentials (Zvukově evokované potenciály)
ERP	Event-Related Potentials (Kognitivně evokované potenciály)
MATLAB	Matrix laboratory (Vývojové prostředí matematického softwaru)
LVQ algoritmus	Linear Vector Quantization (Lineární vektorové kvantování)
SOM algoritmus	Self-Organizing Maps (Samo-organizující mapy)
LDA algoritmus	Linear Discriminant Analysis (Lineární diskriminační analýza)
SVM algoritmus	Support Vector Machines (Podpůrné vektorové stroje)
BCI	Brain-Computer Interfaces (Mozkově-počítačové rozhraní)
SCP	Slow Cortical Potentials (Pomalé kortikální potenciály)
GUI	Graphical User Interfaces (Uživatelské rozhraní)
IIR	Infinite Impulse Response (Nekonečná impulzní odezva – filtr)

Citovaná literatura

1. **Tan, Desney S. a Nijholt, Anton.** *Brain-Computer Interfaces: Applying our Minds to Human-Computer Interaction.* London : Springer-Verlag, 2010. ISBN 978-1-84996-272-8.

2. **Kohonen, Teuvo.** *Self-Organizing Maps.* 3. edice. Berlin, Heidelberg, New York : Springer-Verlag, 2001. ISBN 3-540-67921-9.

3. **Luck, Stephen J.** *An Introduction to the Event-Related Potential Technique.* 1. edice. London : MIT Press, 2005. ISBN 978-0-262-62196-0.

4. **prof. MUDr. Martin Bareš, Ph.D.** *Kognitivní evokované potenciály.* [Článek] místo neznámé : www.prolekare.cz, 2011.

5. **Arnaud Delorme, Scott Makeig.** EEGLAB wiki. [Online] http://sccn.ucsd.edu/wiki/EEGLAB.

6. **Birbaumer, Niels.** *Breaking the silence: Brain–computer interfaces (BCI).* [Publikace] Maryland : Blackwell Publishing Inc. Printed in the USA, 2006.

7. **Žák, Roman.** trilobit.cz. [Online] 1. Červen 2012. http://trilobit.fai.utb.cz/zpracovani-mozkove-aktivity-v-bci-systemech_e498d494-fd80-4948-a8d6-f5f3720bba2d.

8. **Darvishi, Ali.** *Translation Invariant Approach for Measuring Similarity of Signals.* [Publikace] Babol : Journal of Computer Engineering, 2009.

9. Lineární klasifikátory. *www.kiv.zcu.cz.* [Online] http://www.kiv.zcu.cz/studies/predmety/tks/Klasifikatory/Linearni_klasifikator.pdf.

Přílohy

Seznam obrázků

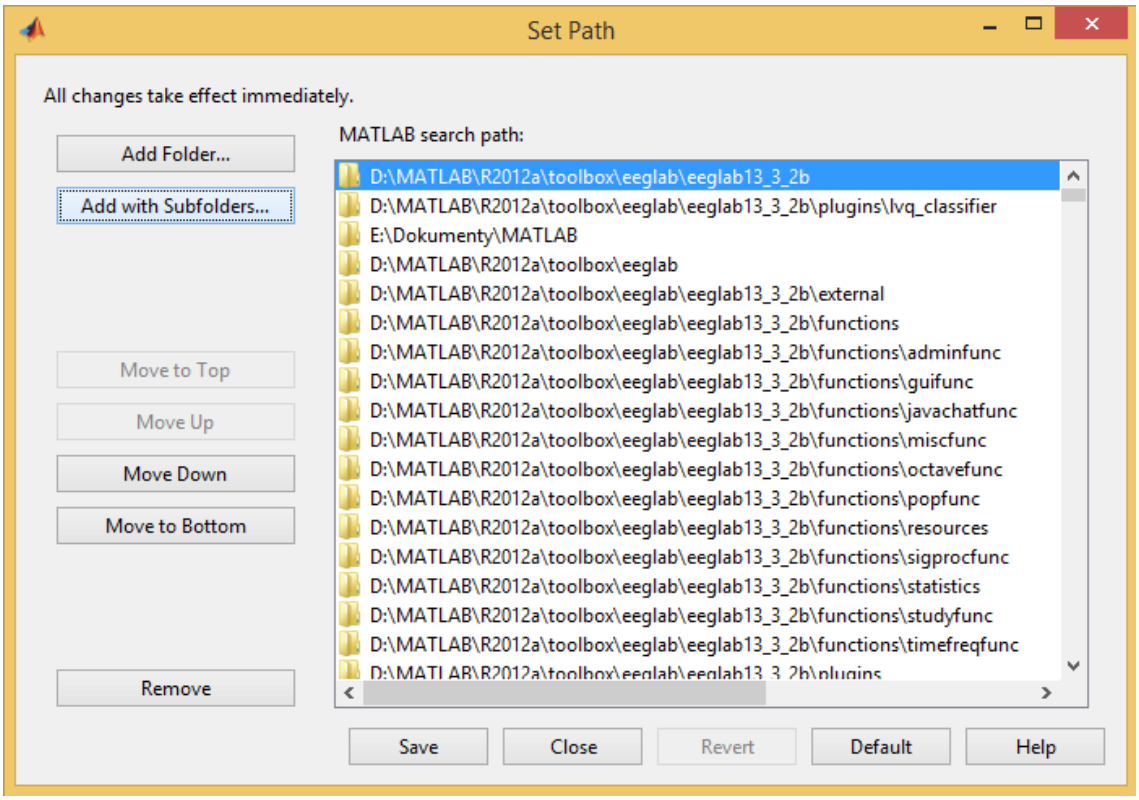
Uživatelská příručka

V této příloze bude popsáno, jak program nainstalovat a spustit.

Instalace

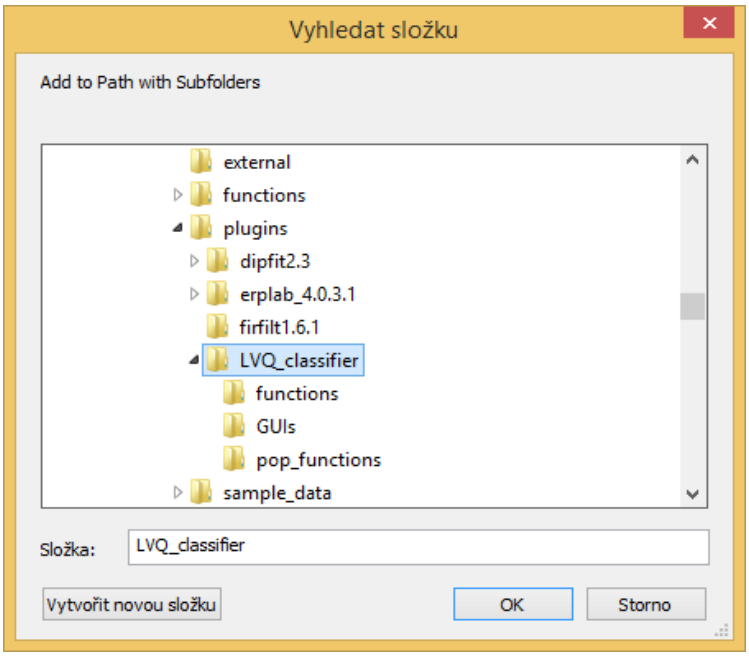
Uživatel musí vlastnit prostředí Matlab a program EEGLAB. Na místě, kde je nainstalovaný Matlab existuje adresář „toolbox“, kde by se měl nacházet EEGLAB. Otevřete adresář eeglab\eeglab_(verze)\plugins a nakopírujte rozšíření „LVQ_classifier“ přiložený na CD.

Poté otevřete prostředí Matlab, nastavte cestu k zásuvnému modulu přes panel File -> Set Path. V okně, co se otevře, zmáčkněte tlačítko „Add with Subfolders“ (Obr. 6.1)



Obr. 6.1: Přidání pluginu

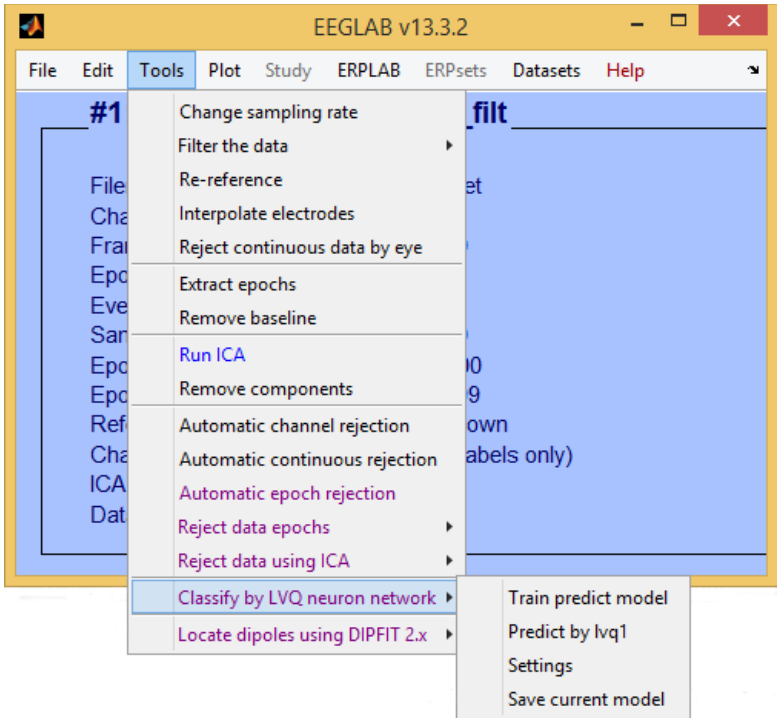
Najděte cestu k pluginu (viz. první odstavec) a přidejte celou složku (Obr. 6.2). V předešlém okně uložte změny a okno zavřete.



Obr. 6.2 Nalezení cesty k zásuvnému modulu

Spouštění

Rozšíření by se mělo automaticky přidat po spuštění EEGLABu. Nově přidané funkce reagují až poté, je-li vybrán nějaký načtený a segmentovaný soubor se signály. Funkce se objeví v menu „Tools“ (Obr. 6.3).



Obr. 6.3: Přidané funkce v menu „Tools“