Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Akvizícia proprioceptívno-dotykových reprezentácií tela u humanoidného robota

Diplomová práca

2019
Martin Pecen

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Akvizícia proprioceptívno-dotykových reprezentácií tela u humanoidného robota

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	prof. Ing. Igor Farkaš, Dr.

Bratislava, 2019
Martin Pecen

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Martin Pecen
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Akvizícia proprioceptívno-dotykových reprezentácií tela u humanoidného robota
Acquisition of proprioceptive-tactile body representations in a humanoid robot

Anotácia: Ľudia sú schopní dosahovať tzv. somatosenzorické ciele, špecifikované proprioceptívnymi (uhly kĺbov) a hmatovými (dotykovými) informáciami, bez spoliehania sa na zrak. Dotyky vlastného tela predstavujú typický vývinový proces, ktorý umožňuje autonómne vytváranie komplexného vzťahu medzi týmito dvoma modalitami, ako súčasti schémy tela.

Cieľ: Navrhnite, vyskúšajte a otestujte vhodný model neurónovej siete na extrakciu vzťahov oboch modalít obsiahnutých v tréningovej množine, ako aj jeho generalizáciu. Za týmto účelom nazbierajte dáta z robotického simulátora iCub, v ktorom robot vykonáva dotyky svojho tela a aktivuje proprioceptívne a hmatové receptory.

Literatúra:
Hoffmann M. & Bednárová N. (2016). The encoding of proprioceptive inputs in the brain: knowns and unknowns from a robotic perspective. In Kognice a umělý život XVI, pp. 55-66.
Metohajrová L. (2016). Biologically inspired predictive model of proprioceptive body representations, Bc. thesis, FMPI, Comenius University.
Seelke A.H. et al. (2011). Topographic maps within Brodmann's area 5 of macaque monkeys. Cerebral Cortex, 22(8):1834-50.

Vedúci: prof. Ing. Igor Farkaš, Dr.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 24.10.2017

Dátum schválenia: 15.11.2017 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Ďakujem školiteľovi diplomovej práce, prof. Ing. Igori Farkašovi, Dr. za cenné rady a pripomienky pri písaní tejto práce.

Abstrakt

Je známe, že nemluvňatá, t.j. malé deti v prvom roku života, si v raných štádiách vývinu vytvárajú reprezentáciu (schému) vlastného tela pomocou viacerých mechanizmov, jedným z ktorých sú dotyky vlastného tela. Nemluvňatá si dávajú do súvislostí dve modality, konkrétne pózy tela (propriocepcia) s polohou dotykov, ktoré nastanú ak vznikne kontakt medzi jednotlivými končatinami, prstami alebo inými časťami tela. Tento asociačný proces si nevyžaduje prítomnosť ďalšej modality - zrakových vnemov, aj keď zrak sa do procesu zapája v neskorších štádiách. V diplomovej práce sme navrhli, implementovali a otestovali biologicky inšpirovaný model založený na umelých neurónových sieťach, ktorý simuluje proces učenia (akvizície) proprioceptívnych a dotykových reprezentácií vnemov. Pre tento účel sme navrhli poloautomatický algoritmus zberu dát s použitím simulátora humanoidného robota iCub, ktorý sa sám dotýkal oboma rukami na rôznych miestach predlaktia a dlane. Natrénovali sme model, pozostávajúci z viacerých samoorganizujúcich sa máp a obojsmerného učenia na vytváranie asociácií medzi oboma modalitami. Realizovali sme viacero simulačných experimentov a podarilo sa nám výsledný model úspešne natrénovať, s dobrou generalizáciou a schopnosťou predpovedať, kedy nastane dotyk pri odpovedajúcich pózach, a na ktorých miestach na tele. Takýto model môže byť využitý ako ako stavebný blok komplexnej schémy tela.

Kľúčové slová: iCub, dotyk, propriocepcia, neurónová sieť, obojsmerné asociačné učenie, samorganizujúca sa mapa

Abstract

It is known that infants in their first year of life are creating a representation (schema) of their body by means of several mechanisms, one of which is the self-touch. Infants associate two modalities, namely the body poses (proprioception) with the positions of touch that occur when there is contact between individual limbs, fingers or other body parts. This association process does not require the presence of another modality – visual perception, although vision becomes involved in the process in later stages. In this diploma thesis we designed, implemented and tested a biologically inspired model based on artificial neural networks, which simulates a learning (acquisition) process of proprioceptive and tactile representations. For this purpose we have designed semi-automatic data collecting algorithm using a humanoid robot simulator iCub. Robot touches itself with both hands at different points of the forearm and palm. We trained a model consisting of several self-organizing maps and a bidirectional learning model to create associations between the two modalities. We performed several simulation experiments and we have succeeded in creating successfully trained model with good generalization and the ability to predict occurrence of touch in matching poses. Such a model can be used as a building block of a complex body scheme.

Keywords: iCub, touch, proprioception, neural network, bidirectional association learning, self-organizing map, natural motivation

Obsah

Zoznam obrázkov

Úvod

V posledných rokoch sa vedci v oblasti kognitívnej robotiky pokúšajú simulovať procesy ľudskej mysle pre vytvorenie autonómneho systému, ktorý by vedel vykonávať rôzne úlohy. Výskum ukazuje, že človek sa učí spracovávať svoje proprioceptívne vnemy v období nemluvňaťa. Táto časť ľudského života sa vyznačuje tým, že jedinec ešte nemá dobre vyvinutý zrak a teda spoznáva svoje telo bez zrakových vnemov. Výskum hovori, že dieťa spoznáva svoje telo na základe pohybov končatín, ktoré sa nekontrolovateľne dotýkajú a tým dieťa produkuje stimulácie potrebné pri učení.

Táto diplomová práca sa skladá z dvoch častí. Prvou časťou je získavanie proprioceptívnych informácií a k nim odpovedajúce dotykové vnemy z humanoidného robota iCub. Robot iCub je technológia vytvorená v Taliansku a následne adoptovaná vo vyše dvadsiatich laboratóriách vo svete. ICub bol ako prvý robot na svete vybavený umelou kožou a dodnes slúži ako výskumný nástroj na. Druhou časťou diplomovej práce je navrhnutie komplexného modelu s využitím neurónových sietí, ktorý týmito nazbieranými informáciami budeme učiť asociovať pre účely predikcie. Model pozostáva z troch častí. Prvá časť pozostáva z viacvrstvového perceptrónu predstavujúceho filter, slúži na odhalenie všetkých nedotykových konfigurácií rúk. Druhá časť neurónového modelu pozostáva z aplikácie samoorganizujúcich sa máp na vstupný a výstupný vzor s cieľom zjednodušiť ho, aby model v treťom kroku vedel jednoduchšie asociovať tieto informácie. Tretí krok predstavuje samotná asociácia. Asociujeme už zjednodušené vstupné dáta reprezentujúce proprioceptívne informácie robota so zjednodušenými výstupnými dátami reprezentujúcimi dotyk. Na tento účel sme použili nový model obojsmernej neuronóvej siete BAL. V kapitole 1 budeme hovoriť o motivácii diplomovej práce a načrtneme poznatky z oblasti kognitívnej vedy pre vysvetlenie základných pojmov. V kapitole 2 predstavíme technológiu humanoidného robota iCub. V kapitole 3 povieme o problematike pri zbere dát z humanoidného robota iCub, navrhneme algoritmus na automatický zber dát a opíšeme jeho implementáciu na robotovi. V kapitole 4 opíšeme všetky modely neurónových sietí použitých v tejto práci. V kapitole 5 opíšeme postup pri vytváraní komplexného modelu na predikciu dotykov na základe polohy rúk robota. V kapitole 6 opíšeme trénovanie jednotlivých častí komplexného modelu a zhodnotíme dosiahnuté výsledky.

Kapitola 1

Biologická motivácia

V tejto kapitole predstavíme reprezentáciu ľudského tela vytvorenú ľudskou mysľou. Taktiež si predstavíme dôležité biologické pojmy pre lepšie pochopenie problematiky. Výskum ukazuje, že nemluvňatá (t.j. malé deti v prvom roku života) si vytvárajú reprezentáciu svojho tela na základe dotykov (Rochat, 1998). Nemluvňa si svojimi nekoordinovanými pohybmi naráža do končatín, vďaka čomu si mozog dieťaťa vie dať do súvislostí ako hýbať končatinami, tak aby sa vzájomne nezrazili. Súčasne sa naučí ako sa pohnúť, aby nastal dotyk na správnom mieste. Tento proces sa uskutočňuje v ranných štádiách ľudského života, keď ešte nemluvňa nemá plne vyvinutý zrak a teda si túto schopnosť buduje bez zrakových vnemov (Kandel et al., 2000). Výsledkom tohto procesu je napríklad schopnosť vedieť sa dotknúť so zavretými očami špičky nosa.

1.1 Reprezentácia ľudského tela

Človek cíti svoje telo pomocou vnemov. Tie sú okamžite posielané do centrálneho neurónového systému, kde sú ďalej spracované a kde sa rozhoduje o reakcii na daný vnem. Tieto vnemy ako pocítiť dotyk, tlak, bolesť, teplotu nazývame somatické vnímanie. Somatické vnímanie môžeme rozdeliť na dve úrovne (Longo et al., 2010): Fyziologickú somatickú reprezentáciu a kognitívnu somatickú reprezentáciu.

- **Fyziologická somatická reprezentácia** zahŕňa procesy neurónovej sústavy na fyzickej úrovni. Sú to procesy ako napríklad lokalizácia stimulov na povrchu ľudského tela, vnímanie polohy jednotlivých častí (napríklad, človek vie, že má ruku nad hlavou bez toho, aby sa na ňu pozrel alebo by sa niečoho dotýkala) alebo vnímanie si rozmerov svojho tela.

- **Kognitívna somatická reprezentácia** vnímania zahŕňa kognitívne procesy ľudskej mysle ako sú encyklopedické, lexikálne a topologické znalosti o orgánoch vo všeobecnosti. Predstavuje prepojenie medzi fyzickým a psychologickým vnímaním.

1.2 Propriocepcia

Mozog človeka sa pochopiteľne zaujíma o svoje telo a teda vníma stavy rôznych modalít tela a okolitý svet, v ktorom sa nachádza (napr. bolesť, dotyk, teplota okolia). Sherrington (1907) vo svojom článku rozdelil takéto vnímanie do troch kategórií:

- **Exterocepcia** – receptory zachytávajúce vnemy tela vnímajúce podnety z okolitého sveta, teda mechanický kontakt, teplota, zvuk, svetlo. Taktiež zahŕňa aj reakcie tela na tieto podnety.
- **Interocepcia** – receptory zachytávajúce vnemy tela vnímajúce podnety z vnútra tela (napr. bolesť alebo vo všeobecnosti informácie z orgánov).
- **Propriocepcia** – receptory tela vnímajúce podnety. Tvorba reakcií na vlastné podnety. Propriocepcia zahŕňa vnímanie pozície tela, pohyb, smer pohybu a taktiež veľkosť vyvinutého stresu svalstva pri pohybe.

Vo všeobecnosti by sa dalo povedať, že propriocepcia je zodpovedná za vnímanie polohy a koordinovanie pohybu trupu a končatín. Táto vlastnosť nevyžaduje zrakové vnemy (Kandel et al., 2000) na svoje fungovanie. Z biofyzikálneho hľadiska sa dá propriocepcia chápať ako schopnosť tela zbierať informácie z kožnej tkaniny, svalových vlákien, kĺbov a zabezpečí ich následné doručenie do somatosenzorického kortexu šedej mozgovej kôry. Neuróny zodpovedné za prenos týchto vnemov sa nazývajú aferentné neuróny. V šedej kôre sa tieto informácie ďalej spracujú a na základe nich sa vygeneruje stav, v ktorom sa telo nachádza. Ak prijaté informácie hovoria, že nastal podnet, tak sa vygeneruje aj reakcia, ktorá sa posúva ďalej (Proske and Gandevia, 2012).

1.3 Reprezentácie propriocepcie a dotykov v mozgu

Oblasť mozgu zvaná somatosenzorická kôra S1 je miesto v mozgu, ktoré vyhodnocuje fyzické zmyslové vnemy. Do tejto oblasti sa zbierajú informácie zo somatosenzorických receptorov. To sú receptory zodpovedné za taktilné vnemy (dotyky), nocicepciu (bolesť) a termocepciu (vnímanie teploty). Podľa Brodmannovho členenia sa táto oblasť delí na oblasti 1, 2 a 3 (Kandel et al., 2000). Brodmannová oblasť 3 sa ďalej

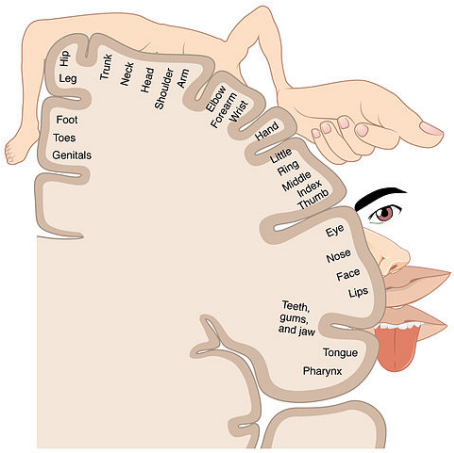
delí na oblasti 3a a 3b. Oblasť 3a sa sústreďuje na propriocepciu ako takú. Oblasť 3b reprezentuje najmä taktilné informácie. Výskum somatosenzorických dát ukazuje (Kim et al., 2015), že tieto informácie sú spracúvané ako prvé práve v oblastiach 3a a 3b, a až potom sú posúvané ďalej na spracovanie a vyhodnocovanie do oblastí 1 a 2. Z tohto dôvodu dostali spoločný názov primárna somatosenzorická kôra.

Obe oblasti 3a aj 3b obsahujú reprezentácie jednotlivých častí ľudského tela. Tieto reprezentácie znázornené na obrázku 1.1 sú zvyčajne graficky usporiadané. Na rozdiel od 3a oblasť 3b má túto reprezentáciu pomerne presne utriedenú. Umiestnenie reprezentácií v oblasť 3b sa mierne odlišuje od fyzického usporiadania. Reprezentácie rúk sú príliš blízko tváre a nohy pri genitáliách. Taktiež je pre oblasť 3b typické pomerné zastúpenie jednotlivých častí ľudského tela zodpovedajúce hustote ich inervácie. Napríklad prsty rúk majú väčšie plošné zastúpenie než lýtko. Typické pre oblasť 3a je, že nemá striktné rozloženie jednotlivých častí, ale u rôznych jedincov sa rozloženie môže jemne líšiť. Taktiež nastávajú rozdiely vo veľkostiach zastúpenia jednotlivých častí tela. Na rozdiel od 3b, oblasť 3a nemá zastúpenie podľa inervácie ale podľa frekvencie používania jednotlivých častí (Krubitzer et al., 2004).

Za hlavné sídlo propriocepcie u primátov sa považuje Brodmannova oblasť 3a, ktorá je situovaná v temennom laloku v časti somatosenzorická kôra (Krubitzer et al., 2004). Základnou cestou, ktorou prichádzajú proprioceptívne informácie do tejto oblasti, sú miechové uzliny – spinálne ganglie. Somatosenzorické informácie prejdú cez časť miechy zvanej nucleus cuneatus do kontralaterálnej časti nucleus cuneatus. To znamená, že už v mieche sa tieto informácie zrkadlovo prekrížia a vstupujú do opačnej strany centrálneho nervového systému než sú orgány, ktoré ich vysielajú. Oblasť 3a prijíma tieto informácie aferentnými signálmi priamo z medzimozgového lôžka (thalamus).

1.4 Schéma tela u humanoidného robota

Cieľom tejto diplomovej práce je vytvoriť model tela robota, ktorý bude vedieť predikovať dotyky končatín. Ak sa pozrieme ako sa vytvárajú reprezentácie propriocepcie a reprezentácie dotykov v ľudskom mozgu, tak zistíme, že vznikajú veľmi komplikované reprezentácie, ktorým ešte stále nie úplne dobre rozumieme. Procesu tvorby takýchto reporezentácií tela sa v dnešnej dobe kladie veľký význam, lebo by otvárali dvere k lepšiemu porozumeniu procesov v mozgu.



Obr. 1.1: Somatosenzorický homonkulus. V somatosenzorickej kôre sú jednotlivé zastúpenia častí tela topologicky usporiadané. Časti tela, ktoré sú blízko seba sú blízko aj v kôre. Polohy jednotlivých častí nie sú fixné a môžu sa u jedincov líšiť. Taktiež si môžme všimnúť, že veľkosť oblastí nezodpovedá proporciám jednotlivých častí ale ich veľkosť súvisí s používateľnosťou. To znamená, že napríklad časti predstavujúce prsty na rukách budú väčšie než časti zodpovedajúce prstom na nohách.[2]

Jednou zo súčastí schémy tela je posturálna schéma, ktorá mapuje priestorové vlastnosti tela. Za týmto účelom budeme kombinovať dva aspekty somatopercepcie, propriocepciu a hmatové vnímanie. Napriek tomu, že ľudské telo zahŕňa aj iné modality, ako je napríklad zrak, v našej práci sa budme zaoberať výlučne somatosenzorickými informáciami. Tento prístup bol inšpirovaný správaním sa nemluvniat a niektorých zvierat, u ktorých sa predpokladá, že si vytvárajú a zdokonaľujú svoje telesné reprezentácie sebastimuláciou (Roncone et al., 2014).

[2] `https://en.wikipedia.org/wiki/Cortical_homunculus`

Kapitola 2

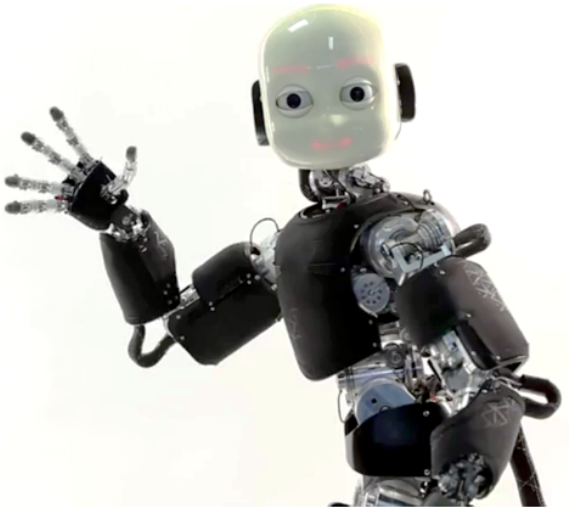
Humanoidný robot iCub

V tejto kapitole predstavíme európsku open-source technológiu humanoidného robota iCub vybaveného umelou kožou, simulátorom `iCub_SIM` a `YARP` technológiu, v ktorej je iCub simulovaný implementovaný.

Ľudstvo sa už od nepamäti pokúša vytvárať nástroje na zjednodušenie svojej práce. Vytvára rôzne stroje, ktoré nám dovoľujú vykonávať úkony, ktoré by sa ťažko vykonali bez nich. V dnešnej modernej dobe, keď sa veda posúva míľovými krokmi vpred je veľmi pochopiteľné, že sa ľudstvo snaží vytvoriť technológiu, ktorá by vedela úplne nahradiť človeka. Je prirodzené, že sa pokúšame vytvoriť robotického človeka (humanoidného robota). Takéto autonómne roboti budú veľmi užitočné napríklad pri domácich prácach. A keďže sú nahraditeľné, tak sú aj výbornou voľbou pri vykonávaní prác v priestoroch, ktoré sú pre človeka nebezpečné (skúmanie vesmíru, zaisťovanie bomby).

2.1 Základná výbava robota

ICub patrí k humanoidným robotom. Bol vytvorený pre výskum kognitívnych vied, neurovedy a umelej inteligencie. ICub bol vyvinutý v Talianskom technickom inštitúte ako súčasť európskeho projektu RobotCub a následne adoptovaný vo vyše 20 laboratóriách po svete (Metta et al., 2010). Na obrázku 2.1 znázornený iCub meria 105 *cm*, váži 24 kilogramov a má 53 motorov, ktoré hýbu hlavou, jednotlivými časťami rúk, trupom a nohami. Dokáže počúvať a vidí (Metta et al., 2010). Taktiež bol vyzbrojený senzormi zaznamenávajúcimi dotyky. Simulátor `iCub_SIM` dostatočne dobre simuluje skutočného robota (Tikhanoff et al., 2008), takže v našej práci budeme používať tento simulátor namiesto skutočného robota. Simulátor je šírený pod otvorenou licenciou.



Obr. 2.1: Známa póza kývajúceho humanoidného robota iCub.[1]

2.1.1 Propriocepcia robota

Propriocepcia robota je reprezentovaná systémom 53 motorov, ktoré ovládajú kĺby robota. Každý kĺb sa môže otáčať v jednom až troch smeroch otáčania. Každý z motorov otáčajúci kĺby je obmedzený na rozsah pohybu, tak aby čo najvierohodnejšie napodobnil človeka. Jeden takýto kĺb/motor a jeho rozpätie sa nazýva stupeň voľnosti (degree of freedom) a udáva sa v stupňoch a uhloch. ICub má 6 stupňov voľnosti na hlave, 3 v trupe, 16 v každej ruke a 6 v každej nohe. Každý stupeň voľnosti má svoju vlastnú základnú polohu, ktorá veľmi často nie je nulová. Rozpätie všetkých stupňov voľnosti je medzi 44° až 270°, pričom jednotlivé stupne nemusia byť symetrické okolo nuly (Metta et al., 2010).

Robot iCub v každom okamžiku pozná všetky stupne voľnosti. Tie v našom prípade predstavujú zjednodušenú analógiu proprioceptívnych informácií v mozgu robota. U živých tvorov sú to informácie z kože, šliach a kĺbov. U robota tieto informácie sú zjednodušené len na veľkosti uhlov zvierajúce jednotlivé kĺby robota. V práci budeme často nazývať jednotlivé polohy rúk vyjadrené stupňami voľnosti (proprioceptívne informácie robota) ako konfigurácie alebo ako proprioceptívne informácie robota.

2.1.2 Taktilný systém robota

Ruky a trup iCuba sú pokryté umelou kožou. Táto koža je poskladaná z trojuholníkov, pričom každý obsahuje desať dotykových senzorov (taxelov), ako vidieť na obrázku 2.2. Umelá koža je poskladaná spolu z 18 skupín trojuholníkov. Jedna skupina obsahuje do 160 taxelov, teda obsahuje šestnásť trojuholníkov. Jednotlivé

[1] `http://wiki.icub.org`

skupiny trojuholníkov sú rozmiestnené nasledovne: po jednom na každej dlani, dva na každom predlaktí, štyri na každom nadlaktí a štyri na na trupe iCuba.

Obr. 2.2: Ukážka skupiny taxelov, ktoré sa pripevnia robotovi na predlaktie.[2]

2.2 Simulátor iCub

Napriek tomu, že robot iCub je sériovo vyrábaný, tak je veľmi drahý. Aj z tohto dôvodu bol vyvinutý simulátor `iCub_SIM` a to ako otvorený softvér (Tikhanoff et al., 2008). ICub simulátor bol naprogramovaný tak aby čo najdôveryhodnejšie napodobnil správanie skutočného robota. Keďže simulátor iCub má otvorenú licenciu, tak je často používaný práve ľuďmi, ktorý nemajú prístup k robotovi. Taktiež sa používa z bezpečnostných dôvodov na prvotné testovania zdrojových kódov robota.

2.2.1 YARP

YARP (Yet Another Robot Platform) je technológia na multiprocesorovú komunikáciu (Metta et al., 2006). YARP bol primárne vytvorený pre jazyk C++, ale pomocou niektorých iných technológií ako je SWIG sa dajú jeho API funkcie zabaliť do iných jazykov (Java, C#, Python, Ruby, Perl, Matlab).

Komunikácia medzi jednotlivými procesmi v YARP je realizovaná pomocou portov. YARP tieto porty spravuje a poskytuje možnosť vytvárania a spájania portov, ktoré sa používajú na tok dát ľubovoľného typu. YARP porty sú programované cez sieťové rozhranie počítača, jednotlivé správy sa posielajú cez sieťovú kartu. To znamená, že jednotlivé procesy môžu bežať na rôznych počítačoch a stále vedia komunikovať medzi sebou. Za zmienku stojí fakt, že aj keď spájame porty sú na

[2] `http://wiki.icub.org/wiki/Tactile_sensors_(aka_Skin)`

jednom počítači, tak YARP pošle tieto správy do sieťovej karty. Na komunikáciu prostredníctvom YARP sa používajú tieto príkazy

- Príkaz `yarpserver` spustí prostredie YARP.
- Príkaz `yarp read /portName` vytvorí port na čítanie.
- Príkaz `yarp write /portName` vytvorí port, kam sa zapíše informácia (ktorá bude dostupná pre iné porty).
- Príkaz `yarp connect /portName1 /portName2` vytvorí spojenie medzi portami `/portName1` a `/portName2`. V tomto momente si porty môžu začať posielať správy.
- Príkaz `yarp disconnect /write /read` zruší spojenie medzi portami.

Okrem príkazov v príkazovom riadku sa dajú používať aj knižničné funkcie v príslušnom jazyku. Tieto funkcie fungujú na rovnakom princípe ako programy v príkazovom riadku.

2.2.2 Moduly iCub simulátora

Po spustení programu `yarpserver` môžeme spustiť simuláciu `iCub_SIM`, ktorá nastaví všetky potrebné porty a otvorí 3D grafické rozhranie so simulovaným robotom. Je dôležité zapnúť kolízie robota a simuláciu kože v konfiguračnom súbore simulátora, inak by tento simulovaný robot mohol nastaviť ruky do pozície, v ktorej sa pretínajú. Robot bez týchto nastavení by prekrížil ruky (ruky by nekolidovali) a nezaznamenal žiaden dotyk. Okrem samotnej simulácie má iCub simulátor aj iné pomocné moduly, slúžiace na vizualizáciu a pomoc pri pohybe robota.

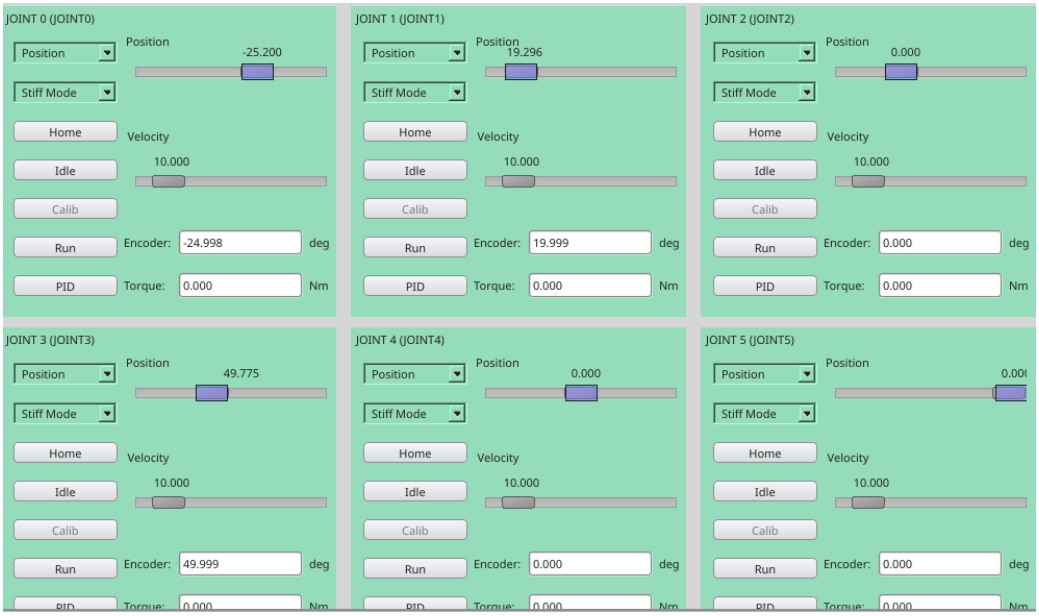
MotorGui

`MotorGui` je modul s grafickým rozhraním na jednoduché ovládanie jednotlivých kĺbov robota (obrázok 2.3). `MotorGui` sa po spustení sám pripne na zapisujúce porty simulátora `iCub_SIM`. Pre každý stupeň voľnosti ponúka posuvník na manuálne ovládanie robota. Tento modul sme používali predovšetkým na rýchle zisťovanie uhlov stupňov voľnosti a ladenie nášho kódu.

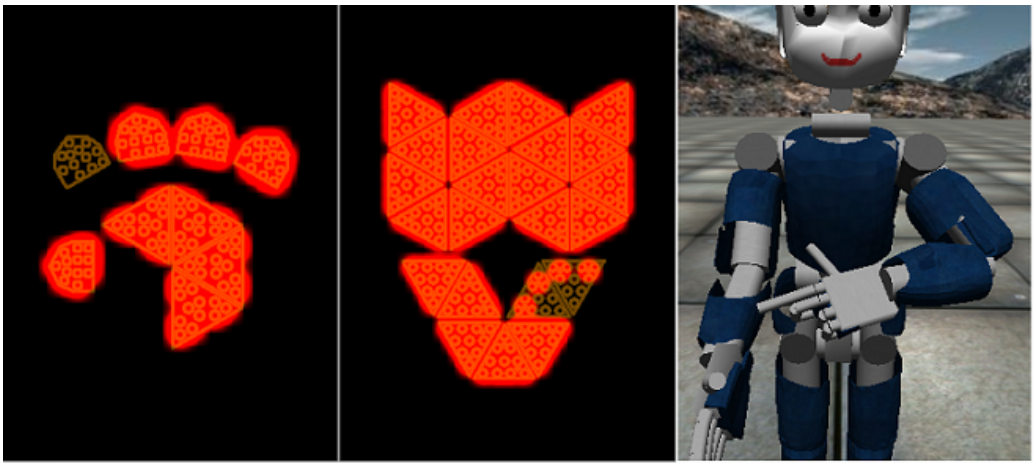
iCubSkinGui

`ICubSkinGui` je vizualizačný modul kože simulátora iCub. Tento modul slúži na zobrazovanie dotykov, ktoré iCub zaregistruje. Na rozdiel od `MotorGui` je potrebné porty tohto modulu ručne pospájať. `ICubSkinGui` zobrazuje iba jednu časť robota. To znamená, že ak chceme spustiť zobrazovanie viacerých častí naraz, musíme spustiť pre každú časť jednu inštanciu modulu. Na vybratie konkrétnej časť

tela použijeme prepínač `-from [meno časti kože]`. Na obrázku 2.4 môžeme vidieť jednotlivé časti vizualizovanej kože počas dotyku.



Obr. 2.3: Znázornenie grafického rozhranie modulu `motorGui` pre jednu ruku. Pre sprehľadnenie je zobrazených iba 6 zo 16 ovládačov.



Obr. 2.4: Použitie vizualizátora kože na zobrazenie dotyku prstu na predlaktí. Vľavo je znázornený dotykový vnem ľavej dlane. V strede je znázornený dotykový vnem predlaktia pravej ruky. Vpravo je znázornená poloha rúk robota iCub.

Kapitola 3

Zber dát z humanoidného robota

V tejto kapitole opíšeme spôsob zberu dát prostredníctvom iCuba pre trénovanie nami navrhnutého modelu. Predstavíme myšlienku algoritmu, pomocou ktorého by sme mohli automaticky získavať dáta a ako sme sa vysporiadali s ťažkosťami pri jeho použití.

Cely proces zberu dát sme uskutočnili v operačnom systéme Linux Mint, v ktorom sme spustili `yarpserver` aj simulátor iCuba. Keďže YARP je vyvíjaný v programovacom jazyku C++, poskytuje aj knižnicu pre tento jazyk na komunikáciu so serverovým démonom. Preto sme na účel zberu dát cez YARP používali programovací jazyk C++.

Pre účely diplomovej práce sme sa zamerali na vzájomné dotyky rúk. Kvôli obmedzenosti pohybu simulátora (obr. 3.1) sme sa museli obmedziť na dotyky rúk a predlaktí. Simulátor robota sa nevedel dotknúť ramien viac ako jedinou konfiguráciou. Dotykové dáta sme preto zbierali len z dlaní a predlaktí oboch rúk. Pre prvé pokusy o vhľad do problematiky sme použili modul `motorGui` (kap. 2). Na zachytávanie informácií z robota sme implementovali funkcie `getTouch` a `getArmPos`, ktoré majú za úlohu pripojiť sa k `yarpserver` a pozbierať všetky proprioceptívne informácie a informácie z dotykových senzorov robota. Spolu s funkciami `moveArm` a `isTouch` sme mali pripravené vhodné API, cez ktoré je možné získavať dáta automatizovane.

3.1 Získavanie informácií o dotykoch s využitím modulov kinematiky

Hlavným problémom zbierania dotykových konfigurácií je ovládať ruky rukami robota tak, aby sme vedeli docieliť dotyk. Problém je v tom, že robotovi vieme zadať len vektor uhlov voľností, ktoré určia simulátoru ako pohnúť jednotlivými motormi a teda pohnúť rukou. Pre tento účel existuje modul inverznej kinematiky. ICub

Obr. 3.1: Ukážka robota iCub, ktorý sa pokúša dotknúť ramena.

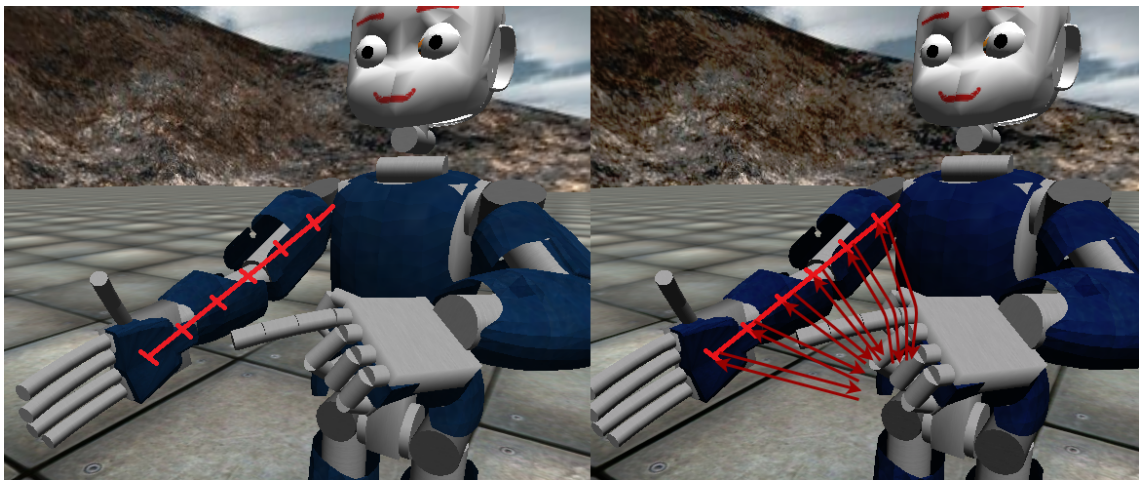
je štandardne vybavený softvérom riešiacim výpočet hodnôt stupňov voľnosti pre konkrétne umiestnenie v priestore. To znamená, že môžeme programu zadať bod v trojrozmernom priestore a program vypočíta hodnoty stupňov voľnosti, pri ktorých motory robota dosiahnu požadovaný bod napríklad dlaňou. Tento modul beží v samostatnom procese, ktorý sa spojí so simulátorom robota aby získal informácie o aktuálnej konfigurácii motorov. Následne pomocou `yarpserver` sa môžeme pýtať na konkrétne polohy končatín robota (Nori et al., 2015).

Okrem inverznej kinematiky iCub obsahuje aj softvér na doprednú kinematiku, ktorá vypočíta pozíciu (napr. dlane) v priestore podľa stupňov voľnosti danej končatiny robota. Dostupnosť oboch kinematík umožňuje navrhnúť postup pre automatický zber dáta. Je výrazne jednoduchšie, ak sa musíme zaoberať iba jednou rukou. Preto si jednu ruku zafixujeme a druhou sa budeme dotýkať prvej. Pri každej zafixovanej konfigurácií jednej ruke by sme sa chceli dotýkať tou druhou postupne na viacerých miestach. Tento proces dosiahneme postupným posúvaním dotykov od dlane k ramenu. Pseudoalgoritmus vyzerá nasledovne (pozri aj obrázok 3.2):

1. Vyber náhodnú polohu ľavej ruky - rozumná poloha by bola taká poloha, kde existuje vysoká šanca, že na ňu robot dosiahne pravou rukou.
2. Doprednou kinematikou zisti pozíciu dlane ľavej ruky robota - pozíciu ramena nemusíme zisťovať, tá je vždy rovnaká.

3. Vypočítaj body v priestore medzi dlaňou a ramenom ľavej ruky - napríklad si určíme konštantnú vzdialenosť a rozdelíme priestor na úseky tejto dĺžky.
4. V cykle prejdi cez všetky body medzi dlaňou a ramenom
   (a) pohni prstami pravej ruky do bodu
   (b) zaznamenaj dotyk/nedotyk a pozície rúk
   (c) prejdi naspať na počiatočnú pozíciu

       V skutočnosti robot nemusí prejsť naspäť do pôvodnej pozície ale stačí oddialiť pravú ruku od miesta dotyku. Dá sa to dosiahnúť natvrdo zakódovaním kroku späť, napríklad otočením ramena o malý uhol. V takomto prípade sa oplatí zaznamenávať aj nedotykové konfigurácie po kroku späť, keďže sú vždy rôzne a pridávajú nám tesnejšie hranice medzi dotykovými konfiguráciami a tými nedotykovými.



Obr. 3.2: Pohyb ruky robota podľa algoritmu používajúceho kinematiku.

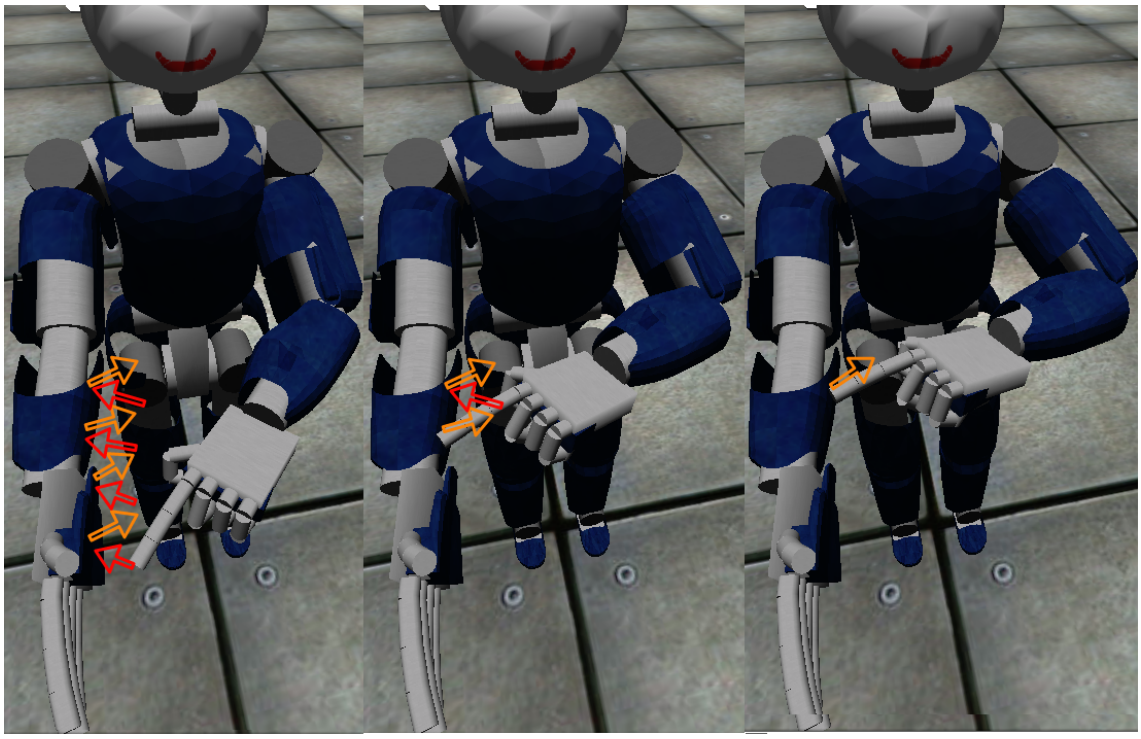
Tento postup môžeme veľa krát opakovať, aby sme získali dostatočný dotykových konfigurácií. V našom prípade takýto ideálny postup ale nebol možný. Modul dobre počíta pozície, iba ak sú na rovnakej strane tela ako ruka. Pre naše účely počíta chybne stupne voľnosti pre pozície ruky, ak sme mu zadali pozíciu za polovicou tela. V takýchto prípadoch vypočíta výsledok vždy pre to isté miesto v oblasti slabín. Túto polohu môžeme vidieť na obrázku 3.3, čo je zlý výsledok pre naše účely. Z tohto dôvodu sme museli hľadať iné riešenie, ktoré by nevyužívalo inverznú kinematiku robota.

Obr. 3.3: Póza robota, ktorá sa vždy vypočíta, ak si vyžiadame polohu za polovicou tela.

3.2 Získavanie informácií o dotykoch bez využitia modulov kinematiky

Vzhľadom na ťažkopádnosť postupu spomínaného v predchádzajúcej časti sme hľadali iný automatizovaný spôsob, ako získať dáta z robota na trénovanie našich modelov. V krajnom prípade by sme vedeli dotykové konfigurácie nazbierať aj ručne s pomocou `motorGui` ale keďže na tréning treba veľa dát, tak by to bolo pracné a nepraktické. Skúmanie viedlo k možnosti ponechať myšlienku z algoritmu, ktorý používa kinematiku, pri čom sme nahradili volania do kinematického modulu definovanými posunmi. Výsledkom skúmania bol poloautomatický algoritmus, ktorý potrebuje asistenciu užívateľa. Pri každej iterácií cyklu zberu dát algoritmus potrebuje:

- pozície - začiatočné pozície oboch rúk
- krok vpred - vektor zmien stupňov voľnosti, ktoré sa aplikujú pri pokuse dotknúť sa
- krok späť - vektor zmien stupňov voľnosti, ktoré sa aplikujú pri oddialení sa od druhej ruky



Obr. 3.4: Pohyb ruky pri algoritme bez modulu inverznej kinematiky.

Algoritmus má nasledovne kroky:

1. posun(pozícia) // nastav ruky robota na začiatočné pozície
2. uložData()
3. opakuj n-krát: // vhodné n = 5
    1. pozícia = pozícia + krok vpred // po zložkách
    2. posun(pozícia)
    3. uložData()
    4. pozícia = pozícia + krok späť // po zložkách
    5. uložData()

Tento algoritmus na rozdiel od algoritmu používajúceho kinematiku si vyžaduje interakciu s človekom. Preto je len výskumným riešením, keďže nám nefungovala inverzná kinematika. Je však významne lepší než zbierať všetky dáta ručne, lebo vie pozbierať 11 (v našom prípade) konfigurácií za každú interakciu na rozdiel od ručného zberu, kde by sme zbierali dáta v pomere 1:1. V tomto algoritme sa dá parametrizovať počet cyklov, ak chceme vykonávať jemnejšie alebo hrubšie zmeny.

Po viacerých pokusoch zberu dát poloautomatickým algoritmom sme si všimli, že vymyslieť vektor posunov nie je až také jednoduché a často sa stáva, že po viacerých posunoch pravej ruky sa ruky buď veľmi vzdialia od seba a už nenastáva dotyk po kroku vpred, alebo sa veľmi priblížia a krok späť už nevie ruky "oddeliť od seba"

(robotovi prikážeme dať ruky tak, aby prechádzala jedna cez druhú a krok dozadu nie je dostatočne veľký na to, aby dostal ruky do nedotykového stavu). Preto tento algoritmus vylepšíme:

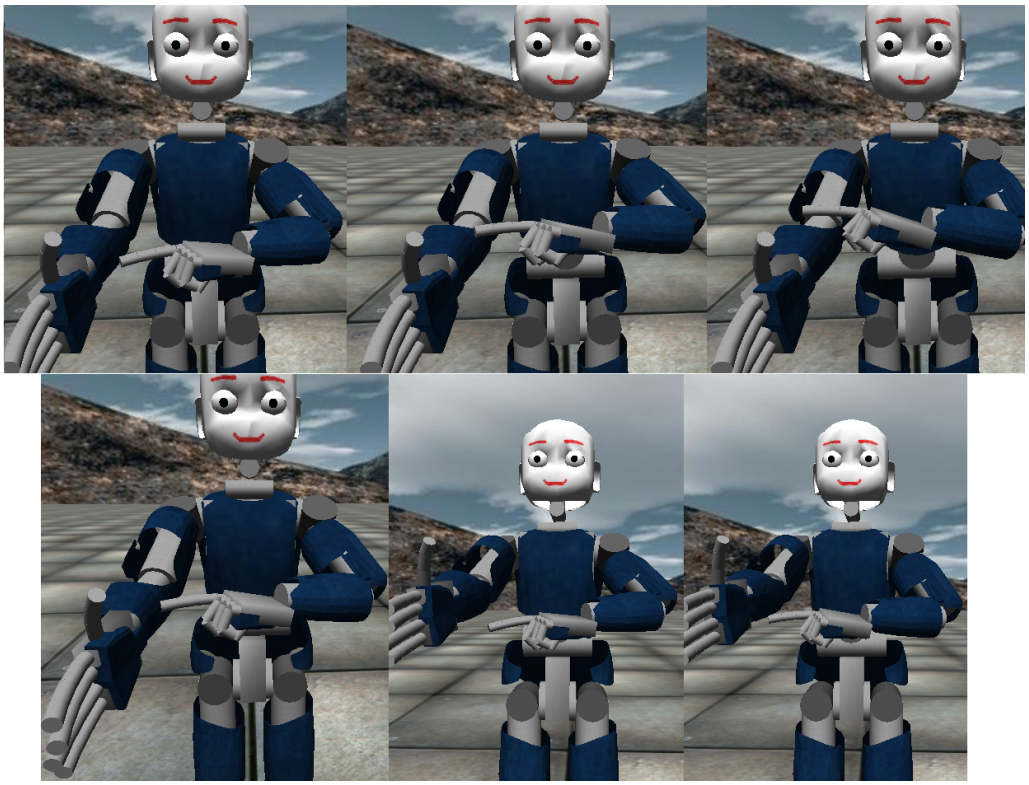
- Jedna z možností ako problém vzďaľovania rúk vyriešiť je kontrola po kroku dopredu, či nastal dotyk. Ak nastal, tak nie je problém. Ak nenastal, tak zrealizujeme lokálne prehľadávanie priestoru. To znamená, že si zvolíme zopár dôležitých stupňov voľnosti (tj. takých stupňov, ktoré veľmi výrazne posúvajú rukou, napríklad ramenný kĺb alebo kĺb v lakti), ktorými budeme hýbať oboma smermi. Toto prehliadanie môžeme realizovať pre každý stupeň samostatne alebo po niekoľkých stupňoch naraz. Takto jedno prehľadávanie priestoru nepotrvá pridlho.

- Ďalšia možnosť ako riešiť nedoliehavosť rúk alebo oddeliteľnosť rúk je zavedenie nového vektora. Tento vektor bude predstavovať zmenu vo vektore kroku vpred (obdobne môžeme zaviesť vektor pre krok späť). To v praxi znamená, že po každej aplikácii kroku vpred pripočítame krokovému vektoru jeho zmenu (po zložkách, teda $v_i(t+1) = v_i(t) + zmena_i$).

Pri zamyslení sa nad oboma vylepšenia algoritmu si uvedomíme, že sa môžu použiť súčasne. To znamená, že poloautomatický algoritmus je viac parametrizovateľný a teda vieme jednoduchšie doladiť kroky vpred, respektíve kroky vzad, ak je to nutné. Na druhej strane prehľadávacie zlepšenie nám pridáva nové konfigurácie (na obrázku 3.5 je ilustrované správanie opísaného zlepšenia). Tieto konfigurácie často nie sú dotykové. Takéto konfigurácie rozširujú vzorku o konfigurácie, ktoré sú blízko dotyku ale nie sú dotykové. Tieto polohy modelu potencionálne môžu pomôcť spresniť predikciu dotyku, pretože tréningové dáta sú veľmi podobné v zmysle propriocepcie ale líšia sa v "dotykovosti" (tj. z dvoch veľmi podobných póz v priestore, pri jednej nastáva dotyk a pri druhej nie).

3.3 Získavanie nedotykových konfigurácií

Okrem dotykových konfigurácií sú potrebné aj tie nedotykové. V prípade, že by bola použitá metóda využívajúcu kinematiku, tak množina nazbieraných konfigurácií by obsahovala viac menej len dotykové konfigurácie, čo by nebolo žiaduce. Z takouto množinou tréningových konfigurácií by neurónové modely vedeli len predikovať, *kde* má nastať dotyk, no nevedeli by predikovať, *či* dotyk má nastať. Preto je potrebné pridať aj také pozície, ktoré predstavujú konfigurácie rúk, ktoré sú veľmi ďaleko od seba. Dobrý spôsob je systematické prechádzanie priestoru. Dá sa to dosiahnuť nasledovne. Určíme konštantu, aby sme vedeli rozdeliť súvislý priestor stupňov voľnosti



Obr. 3.5: Znázornenie rôznych polôh rúk, ktoré sa preskúmajú pri prehľadávaní priestoru okolo ruky.

na diskrétne časti. To napríklad znamená, že pri konštante 10 a stupni voľnosti v rozmedzí $\langle 0, 90\rangle$, do prehľadávania prispeje hodnotami 0, 10, 20, ..., 90. Keďže na iCub-ovi používame 2×16 stupňov voľnosti, tak si nemôžeme dovoliť prehľadať úplne všetky možnosti. Rozhodli sme sa obmedziť na tie stupne voľnosti, ktoré rukami pohybujú najviac (stupne voľnosti v ramene a v lakti). Takto pokryjeme najväčšiu časť v 3D priestore.

3.4 Spracovanie surových dát z robota

Pri zbieraní proprioceptívnych a dotykových informácií z iCuba sme si všimli, že počty všetkých taxelov a počty získaných dát sú dramaticky rozdielne. Bolo to spôsobené tým, že iCub posiela pre každú skupinu trojuholníkov 192 hodnôt, napriek tomu, že niektoré skupiny sú neúplné a obsahujú menej ako 16 trojuholníkov. Trojuholník má 12 hodnôt, pričom dve sú vždy nulové (štvrtá a siedma hodnota sú rezervované pre budúcnosť a aktuálne sú nastavené na nulu). Zistili sme, že na portoch `/icub/skin/left_forearm_comp` a `/icub/skin/right_forearm_comp` iCub posiela 384 hodnôt namiesto 276 hodnôt. ICub na tento port pridal aj veľa nulových hodnôt, ktoré sú rezervované. Sú to hodnoty na pozíciach: 193–204, 217–252, 265–288, 325–336, 361–384. Po odstránení zbytočných nulových hodnôt sme mali

Ruka	min	max
prsty - od seba	0	60
palec - do dlane	10	90
palec - k prstom	0	90
palec - ohnúť	0	180
ukazovák - spodná časť	0	90
ukazovák - horná časť	0	180
prostredník - spodná časť	0	90
prostredník - horná časť	0	180
malíček + prstenník	0	270

Rameno	min	max
rameno - vertikálne	-90	10
rameno - horizontálne	0	160
rameno - otáčanie	-35	80
lakeť	15	105
lakeť - otáčanie	-90	90
zápästie - horizontálne	-90	0
zápästie - vertikálne	-20	40

Tabuľka 3.1: Rozsahy stupňov voľnosti ruky robota.

dotykovú informáciu dlhú 108 + 276 = 384 hodnôt pre každú ruku. Následne sme premenili všetky hodnoty na booleovské hodnoty (iCub posiela hodnoty v rozmedzí 0-255). Po dlhšej analýze dát sme zistili, že ak taxel signalizuje dotyk, tak ho signalizujú všetky taxely v trojuholníku. Preto sme sa rozhodli zjednodušiť dotykové dáta a použiť dotykové informácie len na úrovni trojuholníkov. To znamená, že ak taxel (v praxi to boli všetky taxely) signalizuje dotyk, tak aj trojuholník bude signalizovať dotyk. Týmto krokom sme zjednodušili dotykové informácie z 384 hodnôt na 32 hodnôt pre každú ruku, pričom sme nestratili žiadnu informáciu o dotykoch.

Spracovanie propriocepcie bolo jednoduchšie. Z každej ruky sme získali 16 hodnôt - stupňov voľnosti. Keďže rôzne stupne voľnosti majú rôzny rozsah (tabuľka 3.1), tak sme sa rozhodli ich preškálovať na hodnoty z rozsahu $\langle 0, 1 \rangle$ pomocou vzťahu:

$$val_i := \frac{val_i - min_i}{max_i - min_i} \tag{3.1}$$

Skutočnosť, že sme zafixovali jednu ruku a hýbali len druhou nevnáša do meraní chybovosť, pretože vďaka symetrii by sme dostávali rovnaké výsledky pri fixácii opačnej ruky. Preto sme rovno vygenerovali dáta pre opačnú konfiguráciu rúk.

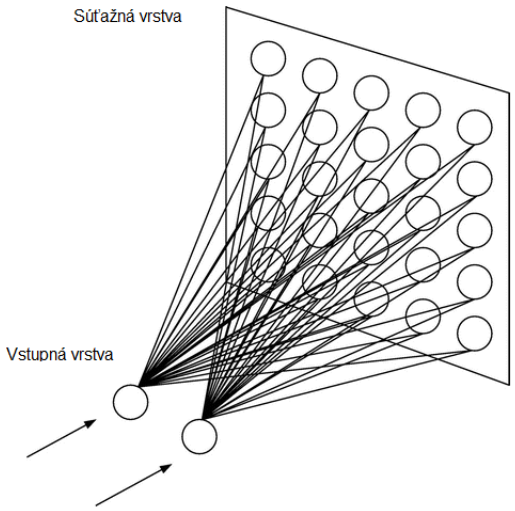
Kapitola 4

Použité modely neurónových sietí

V tejto kapitole sa budeme venovať modelom neurónových sietí, ktoré využívame v našom riešení predpovedí dotykov na základe polohy rúk robota iCub. V kapitole predstavíme jednu z metód slúžiacu na usporiadávanie trénovaných dát patriacu do kategórie učenia sa bez učiteľa. Ide o samoorganizujúce sa mapy (SOM) (Kohonen, 1982). Okrem nich predstavíme ich variáciu používanú na organizáciu a klasterizáciu dotykových informácií (Hoffmann et al., 2017) používajúcu maximálne receptívne polia (MRF-SOM). Z modelov patriacich do kategórie učenia s učiteľom si predstavíme viacvrstvový perceptron. Následne opíšeme model obojsmerného učenia sa siete BAL (Farkaš and Rebrová, 2013) a jeho univerzálnejšiu verziu model UBAL (Malinovská et al., 2018).

4.1 Samoorganizujúce sa mapy

Samoorganizujúce sa mapy patria medzi najpopulárnejšie modely. Podľa názvu modelu SOM, tieto mapy patria do kategórie modelov, ktoré sa učia bez učiteľa, to znamená, že SOM k svojmu učeniu nemá žiadne informácie o hodnotách výsledných neurónov počas procesu učenia. Podobne ako v ostatných samoorganizovaných modeloch, aj v SOM, sa váhy adaptujú za účelom zachytenia štatistických vlastností trénovaného súboru, ktorý je sieti ukázaný vo forme vektorov. Špecifickou vlastnosťou modelu SOM je, že umožňuje zachovávať topológiu trénovaných vzorov a zobrazuje tak ich charakteristické črty. Za týmto účelom sú neuróny umiestnené do pravidelnej mriežky (často obdĺžnikového alebo hexagonálneho tvaru). Takto umiestnené neuróny predstavujú priestor siete, v ktorej sa vzdialenosť dvoch neurónov meria pomocou euklidovskej vzdialenosti vektorov ich súradníc v danej mriežke. Zobrazenie rešpektujúce topológiu trénovaných vzorov, ktoré vznikne natrénovaním modelu SOM, má dôležitú vlastnosť (Kvasnička et al., 1997): ľubovoľné dva vzory, ktoré sú blízko seba v priestore trénovaných vzorov majú v sieti zastúpenie, ktoré sa taktiež



Obr. 4.1: Schéma samoorganizujúcej sa mapy.

nachádza blízko seba (v mriežke).

SOM schopnosťou samoorganizácie dát, aproximácie, funkcie hustoty dát a klasterizácie vstupov sú tieto mapy vhodné pre aplikácie klasifikácie. Rôzne aplikácie SOM: rozpoznávanie reči, odstránenie neznámeho šumu, rozpoznávanie ručného textu, formovanie hierarchických reprezentácií, klasifikácia.

Učenie SOM

SOM vychádza z Kohonenového učenia (Kohonen, 1982). Ide o jednovrstvovú sieť s úplným prepojením medzi vstupnou a "súťažnou vrstvou", teda každý neurón má informáciu o hodnote všetkých vstupných dát (obr. 4.1). Každému neurónu sú priradené jeho pozície v priestore trénovaných dát a v mriežke. Trénovanie siete spočíva v súťažení neurónov o to, kto sa adaptuje na predložený vzor najviac, pričom ostatné neuróny sa posunú v priestore trénovaných vzorov v závislosti od susednej vzdialenosti od výhercu.

Jednotlivé kroky trénovania SOM (Kohonen, 1982):

1. Inicializácia váh neurónov:
   Váhy (polohy) neurónov inicializujeme na malé hodnoty. Ak už poznáme trénovacie dáta, tak vyberieme náhodné pozície z rozsahu trénovaných vzorov a nastavíme rýchlosť učenia $\alpha < 1$. Určíme počet epoch učenia $t_{\max}$.

2. Výber vzoru na trénovanie:
   Náhodne vyberieme jeden z ešte nevybraných vzorov v danej epoche.

3. Výber víťazného neurónu:
   Vyberie sa neurón i^* s najmenšou vzdialenosťou od vzoru. Ako metriku vzdialenosti použijeme Euklidovskú vzdialenosť.

$$i^* = \arg\min_i ||x - w_i|| \tag{4.1}$$

   pričom x je daný vzor a w_i je poloha i-teho neurónu v priestore.

4. Úprava váh neurónov v súťažnej vrstve:
   Víťaz sa posunie smerom ku vzoru a jeho susedia sa ako sieť posunú s ním. Čím vzdialenejší sused, tým menej sa posunie.

$$w_i(t+1) = w_i(t) + \alpha(t) \cdot h(i^*, i) \cdot [x(t) - w_i(t)] \tag{4.2}$$

   kde $w_i(t)$ váha i-teho neurónu, $h(i^*, i)$ je funkcia vzdialenosti susedov medzi víťazom a daným neurónom. Počíta sa buď ako diskrétna:

$$h(i^*, i) = 1 \quad \text{ak} \quad i \in N_{i^*} \quad \text{inak} \quad h(i^*, i) = 0 \tag{4.3}$$

   alebo ako spojitá gaussovská susednosť (gaussian neighborhood):

$$h(i^*, i) = \exp(-\frac{d^2(i^*, i)}{\lambda^2(t)}) \tag{4.4}$$

   kde $d(i^*, i)$ je vzdialenosť neurónov i^* a i v mriežke. Posun váh sa uskutočňuje po každom určení víťaza. Podľa vzoru vyššie sa upravujú váhy všetkým neurónom pre definované okolie N_{i^*}. Počas procesu učenia dochádza k postupnému znižovaniu rýchlosti učenia α a taktiež k zmenšovaniu okolia víťaza parametrom λ. Učenie neurónovej siete sa dá zhruba rozdeliť na dve fázy. Prvá fáza robí veľké posuny váh, slúži na "rozbalenie siete" (rozprestretie siete po celej trénovanej množine). Druhá fáza robí jemné úpravy, čím sa pokúša doladiť sieť do čo najlepšieho stavu. Parametre α a λ sa určujú nasledovne.

$$\alpha(t) = \alpha_s \left(\frac{\alpha_f}{\alpha_s}\right)^{\frac{t-1}{t_{\max}}} \tag{4.5}$$

$$\lambda(t) = \lambda_s \left(\frac{\lambda_f}{\lambda_s}\right)^{\frac{t-1}{t_{\max}}} \tag{4.6}$$

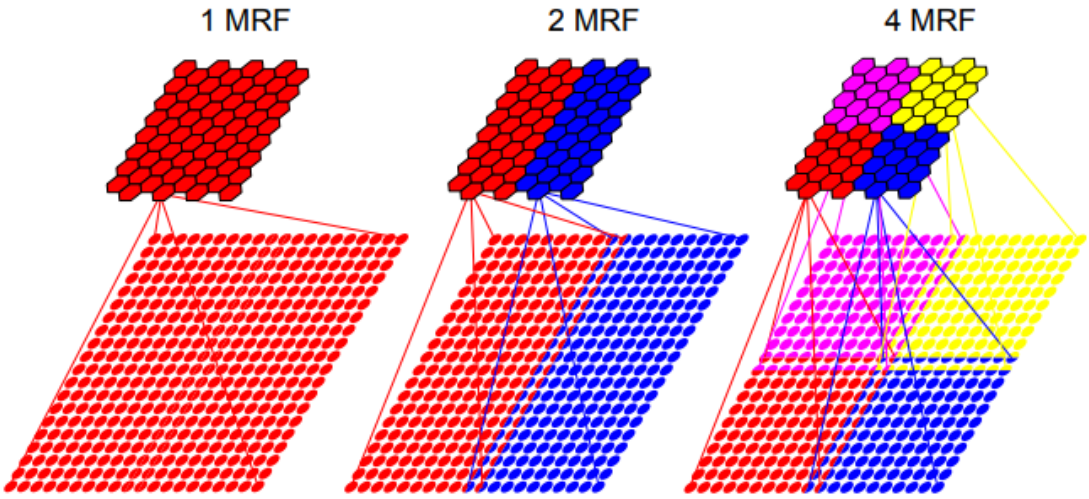
5. Dĺžka trénovania siete:
   Neurónová sieť sa bude učiť $t_{\max}$ epoch.

Na začiatku sme všetky pozície neurónov v súťažnej vrstve rozmiestnili náhodne medzi trénované hodnoty. Nastavenie parametru α (rýchlosti učenia) má veľký vplyv na rýchlosť a kvalitu učiaceho procesu neurónovej mapy. Na začiatku je jeho hodnota nastavená blízko jednotke. Taktiež na začiatku učenia sa inicializuje parameter okolia neurónov λ. Oba parametre počas procesu učenia sa znižujú s pribúdajúcim počtom epoch. Medzi neurónmi v definovanom okolí dochádza k vzájomnému ovplyvňovaniu. Víťaz na seba pôsobí pozitívne/excitačne zväčšuje svoju váhu.

4.2 Model MRF-SOM

Použili sme aj úpravu modelu SOM, ktorá používa maximálne receptívne polia (MRF-SOM) založenú na biologických pozorovaniach na somatosenzorickej kôre mozgu primátov (Leyton and Sherrington, 1917). Tieto pozorovania pritiahli veľkú pozornosť kvôli ich nespornému významu v prepojení mozgu s telom. Somatotopy týchto oblastí mozgu sa často zobrazujú vo forme „homunkula“ (obr. 1.1), tým sa zjednodušuje prezentácia a pomáha stimulovať výskum, ktorý skúma pôvod korešpondencie kortikálnych reprezentácií v somatosenzorickom systéme. Vytvorenie týchto reprezentácií sa stalo dôležitou témou diskusie ako sa príroda dokáže učiť. Dva protichodné názory sú založené na pohľade na závislosť alebo nezávislosť od aktivity neurónov. Jeden tvrdí, že vytvorenie topografických máp je následkom vzorovania, ktoré je vlastné nervovému systému a nevyžaduje špecifickú neurálnu aktivitu. Druhý názor pripisuje kľúčovú rolu vzorom neurálnej aktivity v procese tvorby somatosenzorického neurálneho okruhu. Túto úvahu o protichodných pohľadoch vypracoval Crair (1999). Na obrázku 1.1 môžeme vidieť prierez somatosenzorickej kôry. Sú na ňom znázornené oblasti reprezentujúce jednotlivé časti tela. Podľa obrázka môžeme vidieť, že jednotlivé časti sú štruktúrovane a to tak, že časti tela, ktoré sú blízko seba sú blízko seba aj v somatosenzorickej kôre. Preto sa autori článku o MRF mapách (Hoffmann et al., 2017) rozhodli vytvoriť metódu ako túto skutočnosť dosiahnuť v SOM, ktorá bude slúžiť na mapovanie kože robota. Rozhodli sa ísť cestou, pri ktorej závisí na štruktúre neurálnej aktivity. Záverom článku bolo použitie masiek určujúcich citlivosť rôznych neurónov na rôzne časti vstupného vzoru.

Maska MRF siete je binárny vektor rozmeru trénovaného vzoru. Každý neurón dostane svoju masku, ktorá mu hovorí aké hodnoty má ignorovať. Takéto masky nie sú priraďované neurónom náhodne, pomocou nich sa označujú jednotlivé skupiny neurónov tak, aby sa sústredili na jednu časť vstupu. Je nevhodné, aby jednotlivé časti boli disjunktné z dôvodu, aby nedochádzalo k veľkým deformáciám. Jednotlivé časti sa prekrývajú, tak ako to vidieť na obrázku 4.2.



Obr. 4.2: Farebne znázornené masky MRF rozdeľujú vstupný priestor (Hoffmann et al., 2017).

Učenie MRF-SOM máp

Učiaci algoritmus MRF máp obsahuje rovnaké kroky ako klasický učiaci algoritmus SOM. Na rozdiel od SOM je zameraný na konkrétne dáta (informácie o dotykoch), takže obsahuje zmeny vo vzťahoch. Učenie MRF máp opisuje nasledovný postup:

1. Inicializácia váh neurónov:
   Váhy (polohy) neurónov inicializujeme na malé hodnoty. Ak už poznáme trénovacie dáta, tak vyberieme náhodné pozície z rozsahu trénovacích vzorov a nastavíme parameter $\alpha < 1$. Inicializujú sa masky pre každý jeden neurón s prekrytím oblasti. Určíme počet epoch učenia $t_{\max}$.

2. Výber vzoru na trénovanie:
   Náhodne vyberieme jeden z ešte nevybratých vzorov v danej epoche.

3. Výber víťazného neurónu:
   V MRF-SOM sa nehľadá víťaz klasicky pomocou euklidovskej vzdialenosti ale víťaz i^* sa nájde pomocou skalárneho súčinu:

   $$i^* = \arg\ \max(w_i^\top \cdot x(t)) \tag{4.7}$$

   Dôvodom úpravy v porovnaní s klasickým modelom SOM sú MRF masky a charakteristika dát. MRF masky nulujú hodnoty mimo svojho rozsahu, tak zložky vektora mimo MRF neurónu neovplyvňujú určenie víťaza.

4. Úprava váh neurónov v súťažnej vrstve:

   $$w_i(t+1) = \frac{w_i(t) + h(i^*, i)x(t)}{||w_i(t) + h(i^*, i)x(t)||} \tag{4.8}$$

Celá adaptácia váh sa skladá z 3 krokov:

(a) $tmp = w_i(t) + h(i^*, i)x(t)$

(b) $tmp = tmp * maska_i$
kde * znamená násobenie vektorov po zložkách

(c) $w_i(t+1) = \frac{tmp}{||tmp||}$

5. Dĺžka trénovania siete:
Neurónová sieť sa bude učiť $t_{\max}$ epoch.

4.3 Viacvrstvový perceptron

Pri spomenutí slovného spojenia model neurónovej siete, tak sa často ako prvé v hlave vynorí pojem perceptron. Perceptron je vďaka svojej nelineárnej zložke (prostredníctvom aktivačnej funkcie) univerzálnym aproximátorom (Hornik et al., 1989). Perceptron je typ doprednej neurónovej siete učiacej sa s učiteľom. Patrí k najpopulárnejším modelom sietí.

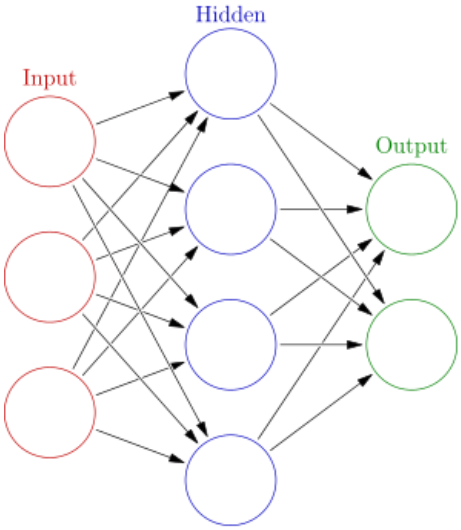
Perceptron sa skladá z vrstiev neurónov, pričom medzi dvoma susednými vrstvami je úplné synaptické prepojenie neurónov (obr. 4.3). Každé synaptické spojenie má svoju váhu. Vnútorné vrstvy (nie prvá ani posledná vrstva) sa nazývajú skryté vrstvy. Hinton et al. (2006) priniesli zrýchlenie metódy trénovania viacvrstvového perceptronu a tým znovu rozpútali záujem o umelú inteligenciu a okolo takzvaného hlbokého učenia. Ak perceptron má viac ako jednu skrytú vrstvu, tak sa nazýva viacvrstvový perceptron alebo aj hlboká sieť. Prechod cez perceptron sa vypočíta nasledovne: Postupne cez všetky vrstvy prejde vstupný vektor. Neurón v ďalšej vrstve bude nadobúdať aktivačnú hodnotu:

$$h_k(n+1) = f\left(\sum_{j=0}^{n-1} w_{kj} h_j(n)\right) \tag{4.9}$$

kde $h_k(n+1)$ je k-ty neurón $n+1$ vrstvy, f je diferencovateľná funkcia, w_{kj} je váha konkrétnej synapsy a $h_j(n)$ je aktivačná hodnota j-teho neurónu n vrstvy. Perceptron svoju nelinearitu získava vďaka aktivačnej funkcii. Takáto funkcia nadobúda hodnoty z rozsahu $[-1, 1]$ alebo aj $[-0, 1]$. Aktivačných funkcií je viacero, medzi najznámejšie patria sigmoida a hyperbolický tangens:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{4.10}$$

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{4.11}$$



Obr. 4.3: Perceptron s jednou skrytou vrstvou.[1]

Učenie perceptronu

V prvom inicializačnom kroku zvolíme malé náhodné hodnoty váhy a nastavíme rýchlosť učenia $\alpha < 1$. Perceptron sa učí zlepšovaním. To znamená, že vypočítať výsledok a následne mu učiteľ spočíta o koľko sa pomýlil a ako si má upraviť svoje váhy, tak aby dosahoval lepšie výsledky. Váhy siete sa upravujú smerom od výstupnej vrstvy ku vstupnej. Tento proces sa nazýva spätné šírenie chyby. Úprava váh sa počíta minimalizáciou chyby pomocou derivácie aktivačnej funkcie. Posledná vrstva sa zmení:

$$w_{ik}(t+1) = w_{ik}(t) + \alpha(d_i - y_i) f_i' h_k \tag{4.12}$$

kde $w_{ik}(t+1)$ je váha synapsy v $t+1$ epoche, α je rýchlosť učenia, d_i je očakávaný výsledok, y_i je vypočítaný výsledok, h_k je aktivácia k-teho neurónu. Následne sa postupne po vrstvách spätné rozšíri chyba:

$$\delta_k = \sum_{i=0}^{n-1} (w_{ik}\delta_i)\, f_k' \tag{4.13}$$

kde δ_k je chyba k-teho neurónu na konkrétnej vrstve, δ_i sú chyby nasledovnej vrstvy, šíriace sa späť. Následne sa upravia váhy danej vrstvy pre epochu $t+1$:

$$w_{kj}(t+1) = w_{kj}(t) + \alpha\delta_k x_j \tag{4.14}$$

4.4 Obojsmerné aktivačné učenie – BAL

Je známe, že učenie spätným šírením chyby je biologicky neprijateľné (O'Reilly, 1996) z dôvodu, že nepoužíva lokálne dostupné informácie. Preto existuje snaha

[1] `https://en.wikipedia.org/wiki/Artificial_neural_network`

o vytvorenie modelov neurónovej siete, ktoré sú z biologického hľadiska uskutočniteľné. Medzi takéto typy učenia siete patrí Dvojsmerné aktivačné učenie (BAL - Bidirectional activation-based learning) (Farkaš and Rebrová, 2013). Ide o obojsmernú sieť, ktorá učí vykonáva asociáciu oboma smermi.

Architektúra modelu BAL je veľmi podobná perceptrónu s jednou skrytou vrstvou. Ale na rozdiel od perceptronu má každé synaptické prepojenie neurónov priradenú váhu smerom dopredu a aj smerom dozadu. BAL počíta prechod cez každú vrstvu rovnako ako perceptron (rovnica 4.9). Ak chceme spätne asociovať výsledky na vstupy, tak otočíme smer postupu v modeli BAL, a obdobne vypočítame ako prechod dopredu, samozrejme s opačnými dimenziami vstupných a výstupných vektorov.

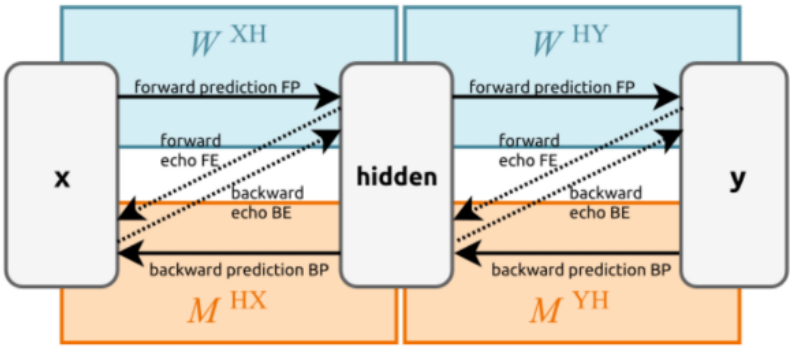
Učenie modelu BAL

Ak máme trénovaciu množinu, ktorá obsahuje viac rovnako asociovaných hodnôt, tak sa BAL nemôže naučiť tieto vzory asociovať v spätnom smere (kvôli nejednoznačnosti). Pretože BAL nemôže deterministicky rozhodnúť, na ktorý vzor sa práve počítaná hodnota má zobraziť. Testovania ukazujú, že sa v takomto prípade BAL naučí vytvorí priemer (Farkaš and Rebrová, 2013).

Na rozdiel podobnosti k perceptronu, BAL sa učí na základe lokálnych informácií. Myšlienka trénovania spočíva v zachovávaní informácie. To znamená, že ak máme dve modality, ktoré chceme asociovať (jednu na druhú a naspať), tak by jednotlivé dvojice mali mať zakódovanú spoločnú informáciu. Túto informáciu sa bude BAL snažiť extrahovať do skrytej vrstvy. Podobne ako perceptron, aj BAL obsahuje jeden parameter a to rýchlosť učenia. Jedna epocha učenia spočíva v postupnom náhodnom poradí prejdení všetkých trénovacích dát. Každá dvojica asociovaných dát sa prejde smerom dopredu a smerom dozadu. Smerom dozadu sa predikuje podľa vzorového výsledku, nie sieťou predpovedaného. Všetky aktivačne hodnoty sa zapamätajú a následne sa prikročí k úprave váh. Každej vrstve (skrytej a výslednej smerom dopredu a skrytej a prvej smerom dozadu) sa upravia váhy nasledovne:

$$w_{ij}^F = w_{ij}^F + \alpha b_i^F (a_j^B - a_j^F) \tag{4.15}$$

pričom w_{ij}^F je váha neurónu smerom dopredu, α je rýchlosť učenia, b_i^F je presynaptická aktivácia smerom dopredu (hodnota z predchádzajúcej vrstvy), a_j^B je postsynaptická aktivácia smerom dozadu a a_j^F je postsynaptická aktivácia smerom dopredu. Váhy smerom dozadu sa upravujú analogicky (vymenia sa 'F' za 'B' a opačne).



Obr. 4.4: Schéma modelu univerzálneho dvojsmerného aktivačného učenia (Malinovská et al., 2018).

4.5 Univerzálne obojsmerné aktivačné učenie – UBAL

Z dôvodu, že model BAL nie je schopný skonvergovať pri niektorých úlohách, ako je úloha 4-2-4 enkódera, priniesli Malinovská et al. (2018) zlepšenia modelu BAL, nazvané UBAL. Univerzálne obojsmerné aktivačné učenie (UBAL - universal bidirectional activation-based learning) je pokračovaním učenia BAL a zároveň pokusom o zvýšenie sily BAL, teda aj biologicky prijateľných modelov neurónových sietí. UBAL používa dve matice W a M pre každý smer aktivácie. UBAL sa od modelu BAL líši pridaním fázy echo aktivácie v každom smere (obr. 4.4).

Predikcia dopredu	$q_j^{FP} = f\left(\sum_{j=0}^{n-1} w_{ij} p_i^{FP}\right)$
Echo dopredu	$p_i^{FE} = f\left(\sum_{j=0}^{n-1} m_{ji} q_j^{FP}\right)$
Predikcia naspať	$p_i^{BP} = f\left(\sum_{j=0}^{n-1} m_{ji} q_j^{BP}\right)$
Echo naspať	$q_j^{BE} = f\left(\sum_{j=0}^{n-1} w_{ij} p_i^{BP}\right)$

Tabuľka 4.1: Vzťahy pre výpočet vnútorných premenných modelu UBAL.

V doprednej fáze predikcie sa vstupné signály šíria do vrstvy p a následne sa rozšíria do vrstvy q. Pri spätnej aktivácií sa signály šíria opačným smerom. Aktivačná echo fáza sa skladá z aktivačných hodnôt predchádzajúcej vrstvy q^{FP} respektíve p^{BP} a váhovej matici M respektíve W (tab. 4.1).

Učiace pravidlo adaptácie váhových matíc W a M:

$$w_{ij}^F(t+1) = w_{ij}^F(t) + \alpha t_i^B (t_j^F - e_j^F) \tag{4.16}$$

$$w_{ij}^B(t+1) = w_{ij}^B(t) + \alpha t_i^F (t_j^B - e_j^B) \tag{4.17}$$

kde α je rýchlosť učenia a ostatné termy sú vysvetlené v tabuľke 4.2:

Učiaci algoritmus UBAL je viac parametrizovaný ako model BAL. Model má 3 hyper-parametre: rýchlosť učenia α, predikčnú silu výsledku $0 \leq \beta \leq 1$ a predikčnú

Dopredný cieľ	$t_j^F = \beta_q^F q_j^{FP} + \beta_q^B q_j^{BP}$
Dopredný odhad	$e_j^F = \gamma_q^F q_j^{FP} + \gamma_q^B q_j^{BE}$
Spätný cieľ	$t_i^B = \beta_p^B p_i^{BP} + \beta_p^F p_i^{FP}$
Spätný odhad	$e_i^B = \gamma_p^B p_i^{BP} + \gamma_p^F p_i^{FE}$

Tabuľka 4.2: Tabuľka pomocných vzťahov učenia modelu UBAL.

silu odhadu $0 \leq \gamma \leq 1$. Tieto nové hyper-parametre sú vzhľadom na smer vo vzťahu $a^F = 1 - a^B$.

UBAL bol experimentálne testovaný na úlohe 4-2-4 enkódera, pri ktorom BAL neuspel. Experimenty ukazujú, že model UBAL túto úlohu zvláda, no na druhú stranu tento model ešte nebol vyskúšaný na iných komplexnejších úlohách. Na priek tomuto faktu autori naznačili, že model môže byť biologicky dôveryhodnejším učením, než spätné šírenie chyby.

Kapitola 5

Experimenty

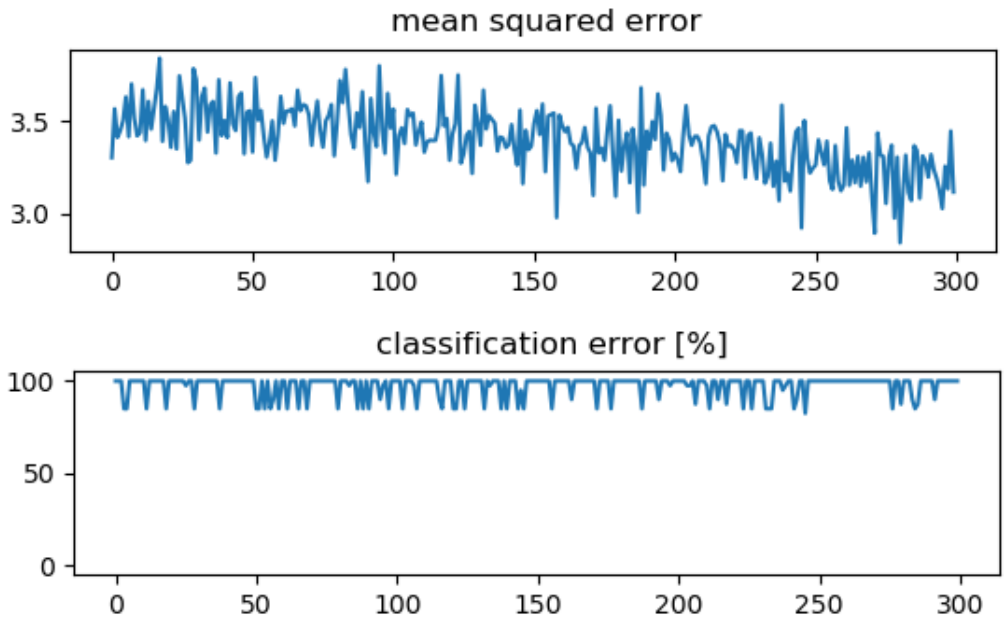
V tejto kapitole si opíšeme myšlienkové postupy pri vytváraní modelu akvizície proprioceptívnych reprezentácií s dotykovými reprezentáciami, ktorý sa učí na dátach získaných a opísaných v kapitole 3. Vytvoríme výsledný komplexný predikčný model. Pri všetkých experimentoch modelov budeme monitorovať priebeh učenia a znázorňovať ho prostredníctvom strednej kvadratickej odchýlky predikovaného výsledku so vzorom:

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n}(d_i - y_i)^2 \tag{5.1}$$

kde n je dimenzia výstupu, d_i je očakávaný výsledok a y_i je predikovaná hodnota na i-tom komponente. Okrem strednej kvadratickej odchýlky budeme monitorovať počet klasifikačných chýb vyjadrený v percentách.

5.1 Použitie modelu BAL na surové dáta

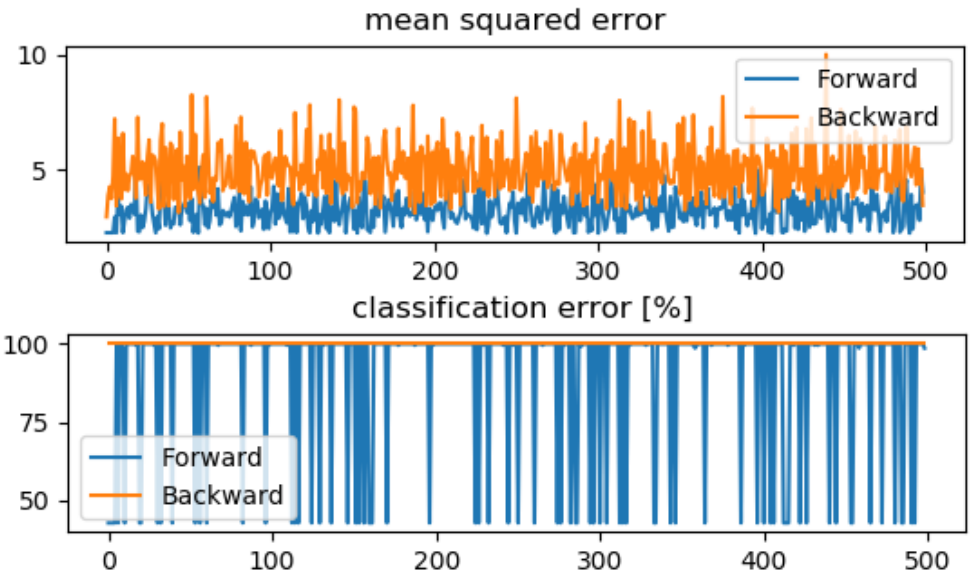
Ako prvý krok sme otestovali BAL (časť 4.4) na surových dátach tj. dátach z robota preškálovaných do intervalu $[0, 1]$, aby sme zistili ako sa model zachová, keď mu prikážeme naučiť sa asociovať nami nazbierané dáta (kap. 3). Ukázalo sa, že model BAL nedokázal asociovať dostatočne dobre, dokonca modelu sa nepodarilo ani začať sa učiť (obr. 5.1). Preto sme sa chceli uistiť, či je vôbec možné takéto dáta asociovať. Vyskúšali sme použiť perceptron s dvoma skrytými vrstvami, ktorý to zvládol dobre. Dané vzory sa bol schopný naučiť na 97%. Následne sme sa chceli presvedčiť, či je model BAL dostatočne univerzálny. Začali sme hľadať hranice, kde je model BAL možné použiť a keď sa už stáva nevhodný. Vyskúšali sme BAL naučiť ľahkú generickú úlohu. Asociovanie náhodných riedkych vektorov obsahujúcich len jednotky a nuly. Pri skúmaní správania modelu pri učení, sme prišli na to, že ak na jednej strane (či už vstupnej alebo výstupnej) sú vektory obsahujúce práve jednu jednotku, tak BAL sa zvláda naučiť takéto dáta veľmi jednoducho. Problém učenia sa modelu nastáva vtedy, ak zadáme úlohy obsahujúce vektory s veľa jednotkami



Obr. 5.1: Grafy vývoja učenia sa modelu BAL na škálovaných dátach z iCuba. Horný graf predstavuje strednú kvadratickú odchýlku. Spodný graf zobrazuje neúspešnosť modelu BAL klasifikovať dotyky.

alebo úlohy s trénovacou množinou pokrývajúcou veľkú časť zo všetkých možností. Napríklad ak máme riešiť úlohu asociácie päť rozmerného vektora s tromi jednotkami na päť rozmerný vektor s dvomi jednotkami a ako trénovacie dáta použijeme osem nekonfliktných dvojíc. Použitie modelu BAL na riešenie podobných úloh sa ukázalo ako nevhodné. Pri dátach, kde sa BAL ukázal ako vhodný sme spozorovali, že rýchlosť učenia $\alpha \approx 0.25$ priaznivo vplýva na rýchlosť a kvalitu učenia sa modelu.

Uvedomili sme si, že neúspech predchádzajúcich pokusov súvisí so skutočnosťou, že v trénovanej množine je veľa vzorov konfigurácií, ktoré sú nedotykové. Veľa dát sa teda má asociovať na nulový vektor. Keď si uvedomíme ako BAL funguje a ako vyzerá trénovacie pravidlo (časť 4.4), všimneme si, že v spätnom kroku BAL vygeneruje iba nulové vektory v každej vrstve. V tomto prípade učenie smerom vzad nebude fungovať, keďže sa v modeli násobí nulou. Tento fakt negatívne vplýva na celý proces učenia modelu BAL. Pri ďalšom postupe máme na výber dve možnosti. Buď vymeníme model za iný alebo budeme musieť predspracovať dáta získané z iCuba tak, aby neobsahovali nulové vektory. Vhodné by bolo, ak by tieto spracované dáta boli riedke vektory, najlepšie s jedinou jednotkou.



Obr. 5.2: Grafy vývoja učenia sa modelu UBAL na škálovaných dátach z iCuba.

5.2 Použitie modelu UBAL na surové dáta

Po neúspešnom pokuse modelu BAL sme sa rozhodli aplikovať UBAL (kapitola 4.5), keďže UBAL dokáže riešiť úlohy, na ktoré je BAL nevhodný. Prvé výsledky testov modelu UBAL dopadli čiastočne lepšie ako pri modeli BAL. UBAL sa začal učiť ale nebol schopný sa vzory naučiť dostatočne dobre, aby výsledok učenia mohol byť považovaný za úspešný. UBAL začal oscilovať medzi 100 - 42 percentnou chybovosťou klasifikácie dotykov (obr. 5.2). Z grafu priebehu učenia modelu UBAL vidíme, že aj modelu UBAL prekážajú nulové dotykové vektory (nenastal dotyk). Rozdiely medzi učením dopredu a dozadu sú viditeľné. UBAL nebol schopný ani raz priradiť správny výsledok smerom dozadu. Je to pochopiteľné, pretože v tomto smere sa dotyky (nula-jednotkové vektory) mapujú na vektory obsahujúce reálne čísla $\langle 0, 1 \rangle$ a pri vyhodnocovaní by bolo potrebné zložitejšie zaokrúhľovanie. Ak UBAL aj BAL majú asociovať viac hodnôt na jeden vzor, tak vytvárajú prototypy (kapitola 4.4). Čo spôsobí ťažkosti pri testovaní. Pri vzniknú nových proprioceptívnych konfigurácií robota je jediný spôsob ako overiť výsledky (dotyky) použiť robota, nastaviť mu všetky stupne voľnosti na správne hodnoty a získať informácie z umelej kože. Tento postup je veľmi nepraktický a pre učenie nepoužiteľný, lebo robotovi trvá presun rúk asi jednu sekundu. Tom môže potencionálne znamenať, že 300 vzorovú epochu by UBAL alebo BAL vykonal za 300 sekúnd ale na natrénovanie modelu UBAL potrebujeme rádovo stovky epoch. Trénovanie by sa neprijateľne predĺžilo.

Nakoľko UBAL má viac hyper-parametrov, testovali sme ho s rôznymi hodnotami hyper-parametrov β, γ, pričom sme hodnoty posúvali o 0.25 postupne v každom z parametrov. Rýchlosť učenia α sme nastavovali postupne na 0.1, 0.2, 0.3 a model sme trénovali 1000 epoch. Po uskutočnení testov sme sa dozvedeli, že žiadne z nastavení nepomohlo naučiť model UBAL dosahovať dobré výsledky. Pri všetkých testovania model dosahoval strednú kvadratickú odchliku okolo hodnoty 2.22, čo je neprijateľný výsledok. Z týchto dôvodov sme sa rozhodli zmeniť stratégiu a namiesto hľadania iného modelu siete sme sa rozhodli hľadať spôsob, ako uskutočniteľným spôsobom predpripraviť dáta z robota. Inšpirovali sme sa reprezentáciami somatosenzorickej kôry u primátov.

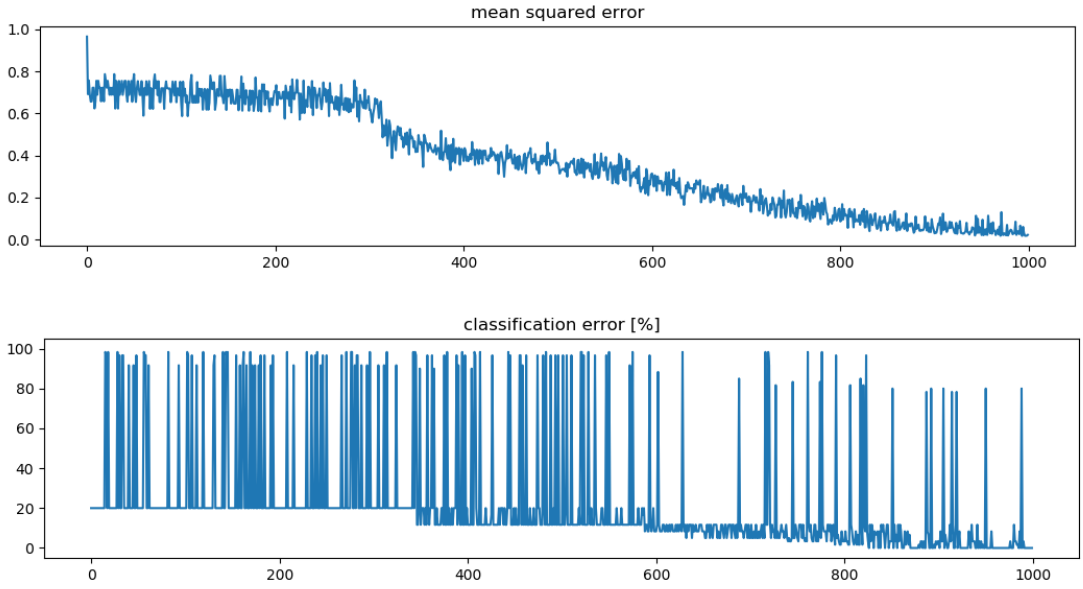
5.3 Biologicky inšpirovaná transformácia dát

Po neúspešných pokusoch trénovať modely BAL aj UBAL na dátach z iCuba sme sa rozhodli transformovať vstupné dáta tak, aby boli vo vhodnom tvare, obsahovali vektory s malým počtom jednotiek a zároveň sme dbali na relevanciu v biológii.

Inšpiráciu sme našli v neurobiológii somatosenzorickou kôrou. Inšpirovali sme sa časťou mozgu, ktorá sa venuje propriocepcii. Presnejšie povedané, využili sme vedomosti o Brodmannovej oblasťou 3a. Somatosenzorická kôra má za úlohu vyhodnocovať fyzické vnemy, čo znamená, že táto kôra dostáva cez takzvané aferentné vlákna informácie o tom, ktorý sval kontrakciu vykonáva a informáciu o miere kontrakcie. Vo svete robotiky sú pre podobné informáciám ekvivalentom informácie, aké uhly zvierajú jednotlivé stupne voľnosti robota. Táto analógia je vhodná pre tvorbu našich vstupných dát. Potrebujeme vytvoriť model, ktorý údaje z motorov iCuba vie zorganizovať a vytvorí štruktúrované reprezentácie reprezentatívnych polôh rúk na spôsob somatosenzorickej kôry. Inšpirovali sme sa dvoma prácami: Bednárová (2015), ktorá sa venovala vytváraniu reprezentácií rúk robota pomocou samoorganizujúcich sa máp (časť 4.1) a Hoffmann et al. (2017), ktorí riešili reprezentáciu umelej kože u robota pomocou MRF-SOM máp (časť 4.2). Dáta sme spracovali z oboch pohľadov. Dotykové informácie sú reprezentované riedkymi vektormi, preto by ich nebolo nutné predspracovať ale pre snahu priblížiť sa prírode sme spracovali aj túto časť vstupu.

5.3.1 Využitie SOM na transformáciu dát

Pri použití modelu SOM chceme využiť jeho vlastnosť klasterizácie. Klasterizáciu využijeme na nájdenie reprezentantov rôznych polôh rúk robota. Predstavme si, že máme dobre natrénovaný model SOM. Každý neurón mapy predstavuje reprezentatívnu pozíciu ruky robota. Pri spracovaní vstupných konfigurácií sa pre každú



Obr. 5.3: Priebeh učenia modelu BAL na dátach spracovaných modelmi SOM a MRF-SOM.

konfiguráciu nájde taký neurón, ktorý má najpodobnejšiu reprezentatívnu pozíciu ruky. Následne sa nahradí konfigurácia jej reprezentatívnym neurónom. To znamená, že vektor $[0, 1]^{16}$ nahradíme vektorom s jednou jednotkou, podľa toho, ktorý neurón je aktívny.

Na spracovanie proprioceptívnych informácií sme použili model SOM (časť 4.1). Natrénovali sme jednu mapu zvlášť pre každú ruku, keďže je známe, že reprezentácie rúk sú uložené na dvoch rozdielnych miestach v mozgu (Longo et al., 2010). Na trénovanie týchto máp sme použili proprioceptívnu časť z nazbieraných dáta z iCuba. Na spracovanie dotykových informácií sme sa rozhodli použiť MRF-SOM (časť 4.2). Pre každú ruku sme natrénovali jednu mapu veľkosti 10×8 neurónov. Ako trénovaciu množinu sme použili nami vygenerované vektory reprezentujúce dotyky namiesto vektorov získaných z iCuba (94 vzorov). Pre tento krok sme sa rozhodli z dôvodu aby mapa mala reprezentácie aj takých dotykov, ktoré nemáme v našej sade z iCuba. Modely BAL a UBAL sme trénovali na predikciu dotykov každej ruky zvlášť. Teda vstupné hodnoty boli SOM ľavej ruky a SOM pravej ruky. Výstupom bola MRF-SOM mapa ľavej ruky a pri druhom trénovaní MRF-SOM mapa pravej ruky. Po získaní natrénovaných máp sme spustili proces učenia sa modelov BAL aj UBAL z využitím týchto máp. Zistili sme, že nami navrhnuté predspracovanie dát pomohlo. Proces učenia sa modelov BAL aj UBAL bol úspešný.

Priebeh strednej kvadratickej odchýlky počas učenia (obr. 5.3 hore) ukazuje dobré výsledky. Vývoj klasifikačnej chyby (obr. 5.3 dole) ukazuje náhodne vysoké hodnoty aj pri malej kvadratickej odchýlke. Klasifikačná chyba zahŕňa tri možnosti chyby (obr. 5.4):

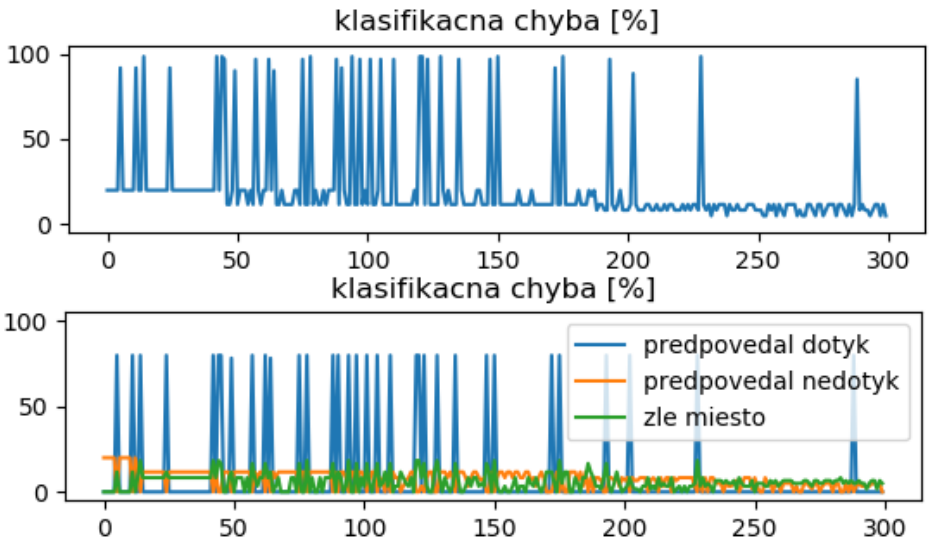
- model predikoval dotyk pri danej konfigurácii a nenastal
- model nepredikoval dotyk a dotyk nastal
- model predikoval dotyk na nesprávnom mieste

Tento jav je vysvetliteľný skutočnosťou, že všetky nedotykové konfigurácie sú asociované s jedným vektorom reprezentujúcim neurón signalizujúcim nedotyk. Fakt, že nedotykových konfigurácií je viac než dotykových je prirodzený. Pri pohľade na priestor všetkých konfigurácií je jasné, že pri dotykových konfiguráciách sú ruky v špeciálnej vzájomnej polohe a preto je nedotykových nepomerne viac. Samotný dôvod náhodnej vysokej chybovosti spočíva v tom, že ak sieť zle vyhodnotí nedotyk v danej epoche, tak potom všetky nedotykové konfigurácie sa zle klasifikujú rámci tejto jednej epochy. Na obrázku 5.3 vidieť, že po dostatočnom počte epoch (800) frekvencia týchto náhodných chýb sa významne zmenší. Po aplikovaní ďalších vylepšení učenia modelu BAL (vysvetlených v kap. 6) dokázal úplne eliminovať túto náhodnú chybovosť a trénovaciu množinu sa mu podarilo generalizovať. Pri modeli UBAL tieto vylepšenia neprinášali rovnako dobré výsledky.

Použitie modelu SOM na zjednodušenie vstupných konfigurácií je dvojsečná zbraň, pretože najväčšia výhoda SOM, ktorou je klasterizácia môže spôsobovať nepresnosti. Predstavme si, že máme dobre natrénovaný model SOM, ktorý určuje reprezentantov. Zoberme si takú konfiguráciu rúk, že nastal dotyk napríklad ľavého ukazováka na pravom predlaktí a následne sa posunieme o pár centimetrov preč od pravej ruky. Pri použití modelu SOM sa obe tieto konfigurácie rúk pravdepodobne namapujú na ten istý neurón. Tieto konfigurácie sú veľmi podobné. Líšia sa iba v malej zmene jedného zo stupňov voľnosti. Máju však rôzny výsledok dotykovosti. Z tohto dôvodu je rozumné takéto konfliktné konfigurácie riešiť tak, že ich definujeme ako dotykové. Týmto sme znepresnili hranicu medzi dotykom a tesným nedotykom, preto sme ďalej s týmto modelom nepracovali.

5.4 Filtrovanie nedotykových konfigurácií

Vrátime sa k problematike náhodných vysokých hodnôt klasifikačnej chyby opísanej v závere časti 5.3.1. Eliminácia nedotykových konfigurácií rúk v rámci súboru učiacich sa dát môže odstrániť túto chybovosť. Na obrázku 5.5 vidíme priebeh učenia



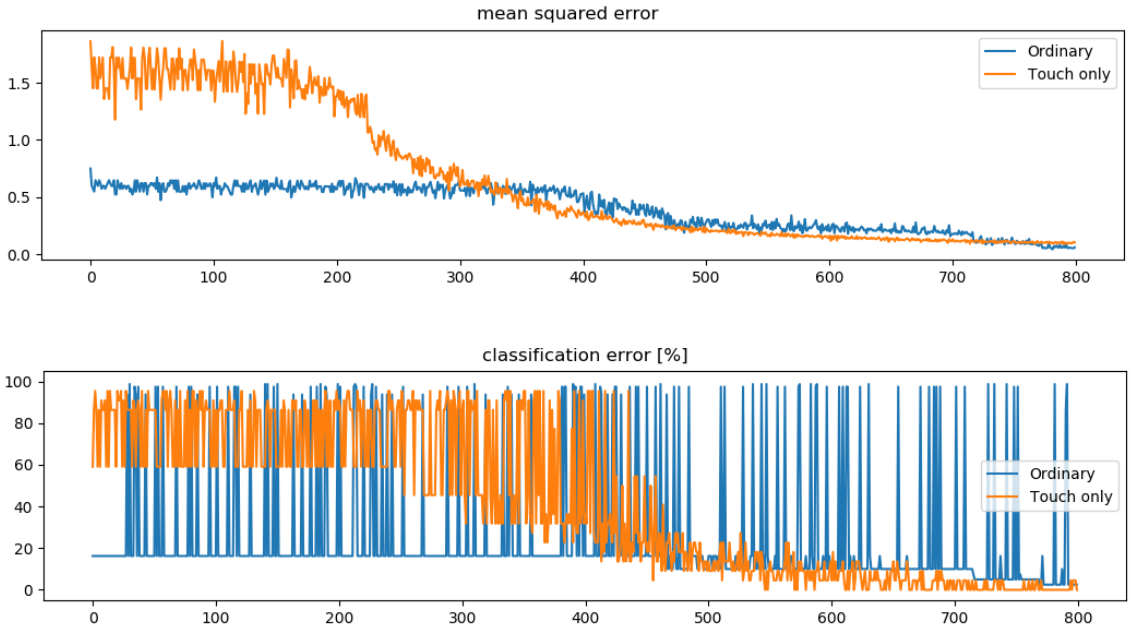
Obr. 5.4: Vývoj klasifikačnej chyby modelu BAL učiaceho sa na transformovaných dátach. Horný graf znázorňuje celkovú klasifikačnú chybu. Spodný graf znázorňuje klasifikačnú chybu rozdelenú do skupín: model predikoval dotyk ktorý nebol, model nepredikoval skutočný dotyk a dotyk nastal a model predikoval dotyk na nesprávnom mieste.

modelu BAL, ktorý sa učil len na množine dotykových konfigurácií rúk. Výsledok ukazuje, že takéto trénovanie modelu BAL bolo rýchlejšie, dokonca sa podarilo BAL naučiť 100% trénovaných vzorov, čo významne vplýva na schopnosť generalizácie. Ak sieť je schopná naučiť sa všetky vzory, tak existuje predpoklad, že bude dobre klasifikovať aj dáta, ktoré predtým nepoznala.

Odstránením nedotykových konfigurácií rúk z trénovanej množiny dát môžeme docieliť definovaním filtra. Filter by v ostrej prevádzke už natrénovanej siete vedel odhaliť "nedotyk" ešte pred tým, než sa údaj o tejto konfigurácií dostane do modelu BAL. Takéto konfigurácie model BAL už nevyhodnocuje. Filter zabezpečí, že ak nastane dotyk táto konfigurácia sa ďalej vyhodnotí modelom BAL, ktorý následne určí polohu dotyku. Filter je vhodné umiestniť pred klasterizáciu modelu SOM, vyhneme sa tak nepresnosti aspoň pri určovaní dotyku.

Na vytvorenie filtra, ktorý bude predpovedať či nastane dotyk, nepoužijeme model BAL ani UBAL. Pri trénovaní filtra použitím modelov BAL alebo UBAL by dochádzalo k nežiadúcemu trénovaniu opačným smerom, od dotyku ku konfigurácií rúk. Rozhodli sme sa na tento účel použiť jednosmerný model siete. Použili sme viacvrstvový perceptron. Perceptron predstavujúci filter sme učili na surových proprioceptívnych dátach tak, že sa asociujú s jednou z dvoch tried, na ktoré stačí vektor dĺžky jeden. Nula, ak nenastal dotyk a jednotka ak nastal. Keďže perceptron je veľmi silný nástroj, tak trénovanie filtra prebehlo bez väčších problémov.

Za zmienku stojí spomenúť nasledovný experiment. ICub nazbieral také špeci-



Obr. 5.5: Vývoj trénovania modelu BAL po odstránení nedotykových konfigurácií. Náhodné vysoké chyby sa stratili. Taktiež BAL dosahuje lepšie výsledky, ako keď sa učí aj s nedotykovými konfiguráciami.

fické nedotykové konfigurácie, ktoré sú pomerne blízke k dotykovým. Ako sme spomínali vyššie v tejto kapitole, nedotykových konfigurácií je nepomerne viac. Experiment spočíva v pozorovaní, či tieto nedotykové konfigurácie určovanie nedotykov všeobecne, pri akýchkoľvek konfiguráciách rúk. Napríklad pri rozpažení rúk robota. Vyhodnotenie ukázalo, že takáto reprezentácia dát stačí pre všetky nedotykové konfigurácie. Napriek tomu mať aj reprezentantov konfigurácií vzdialených rúk pomôže zlepšiť generalizáciu.

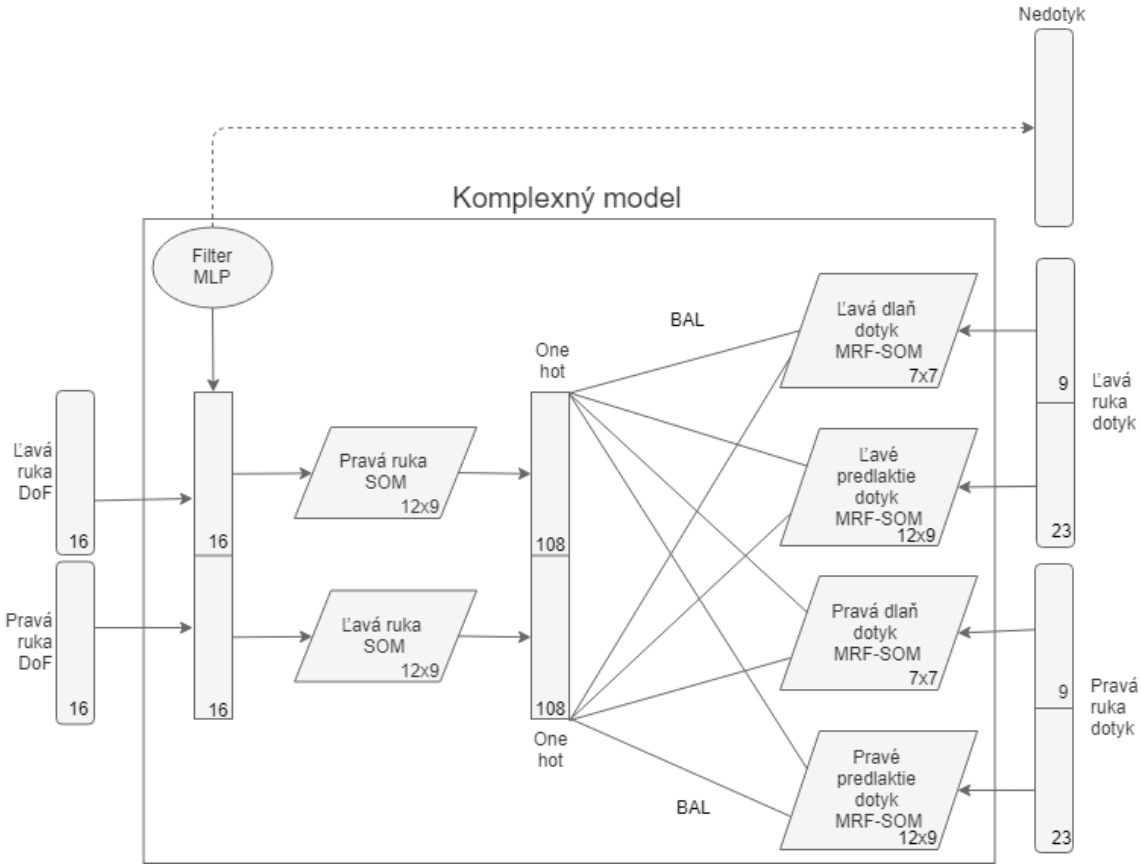
5.5 Výsledný model

Našim cieľom bolo vytvoriť model, ktorý asociuje proprioceptívne vnemy iCuba s jeho umelou kožou. To znamená, že model vie predpovedať, či nastane dotyk na základe konfigurácie, ktorú dostane ako vstup. Vytvorený model je pomerne komplexný. Schému nášho modelu vidíme na obrázku 5.6. Model obsahuje tri časti:

1. **Filter na detekciu dotykov**, ktorého úlohou je zistiť, či daná konfigurácia rúk je dotyková alebo nedotyková. Tiež slúži ako prostriedok na spresnenie hranice medzi dotykovými a nedotykovými konfiguráciami. Preto, že v ďalšom kroku náš model bude vstupnú konfiguráciu nahrádzať jej reprezentantom. Filterom je natrénovaný viacvrstvový perceptron klasifikujúci, či daná konfigurácia je dotyková alebo nie.

2. **Mapovanie surových dát pomocou SOM a MRF-SOM** predstavuje biologicky motivované zjednodušenie vstupu, ktoré zabezpečí klasterizácia trénovaných konfigurácií. Vstupná konfigurácia sa nahradí jej reprezentantom. Konkrétne nulovým vektorom s jednou jednotkou podľa toho, ktorý neurón reprezentuje vstup. Lebo neskôr bude použitý model BAL, ktorý sa lepšie učí na riedkych vektoroch. Na proces klasterizácie sme použili samoorganizujúce sa mapy. Proprioceptívna časť sa zjednodušuje modelom SOM. Dotyková časť sa zjednoduší pomocou MFR-SOM máp.

3. **Asociácia zjednodušených oboch modalít**, pre realizáciu asociácie sa nám BAL osvedčil ako vhodnejšia voľba, pretože na ňom fungovali následné vylepšenia učiaceho algoritmu.



Obr. 5.6: Celková schéma navrhnutého modelu na akvizíciu proprioceptívnych informácií s dotykovými vnemami. Model je zložený z troch častí. Prvú časť predstavuje filter, ktorého úlohou je odhaliť nedotykové konfigurácie. Druhou častou je transformácia vstupných a výstupných vzorov pomocou modelu SOM respektíve MRF-SOM. Treťou časťou je asociácia transformovaných dát vykonávaná pomocou modelu BAL. BAL vždy asociuje konfigurácie oboch rúk s jednou MRF-SOM predstavujúcou dotyk časti ruky.

Kapitola 6

Trénovanie výsledného modelu

V tejto kapitole opíšeme trénovanie jednotlivé časti výsledného modelu a vizualizáciu výsledkov. Definujeme hodnoty hyper-parametrov a ukážeme navrhnuté zlepšenia trénovania modelu BAL.

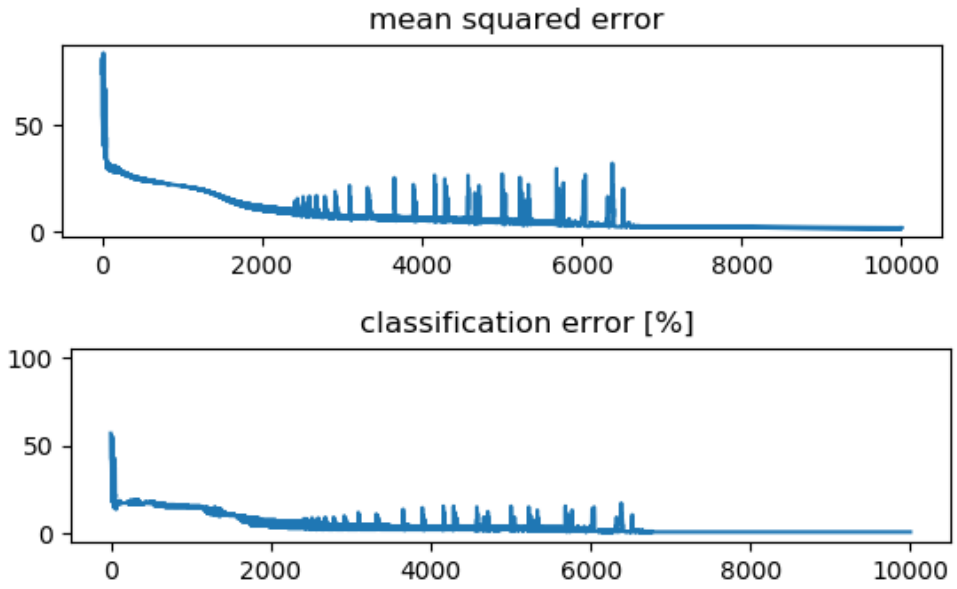
6.1 Trénovanie filtra

Opíšeme postup pri trénovaní perceptronu za účelom odfiltrovania nedotykových konfigurácií robota. Prvým krokom pri učení filtra je vytvorenie trénovaciej množiny. Takáto množina sa dá zostrojiť jednoducho. V každej dvojici prorpriocepcia-dotyk získanej z robota nahradíme dotykový vektor jedným číslom. Nulou, ak výsledkom je nedotyk. Jednotkou, ak výsledkom je dotyk. Následne sme takúto množinu dát rozdelili v pomere 75:25 čím sme získali trénovaciu a testovaciu množinu. Natrénovali sme perceptron s vrstvami 32, 50, 10 a 1 (obr 6.1) s 98% úspešnosťou klasifikácie. Po trénovaní sme otestovali jej schopnosť generalizácie na menšej časti. Priemerná presnosť modelu na testovacích vzorkách bola 94%.

6.2 Mapovanie vstupných dát

V druhom kroku sme trénovali proces mapovania vstupných dát Táto úloha je rozdelená do dvoch častí. Na trénovanie proprioceptívnych reprezentantov a na trénovanie dotykových reprezentantov.

Na trénovanie propriocepcie sme použili klasický model SOM. Trénovali sme ho na proprioceptívnych informáciach nazbieraných z iCuba. Každú ruku sme trénovali zvlášť (obr. 6.2). Mapa má rozmery 12×9 neurónov. Vzdialenosti susedov sme počítali vzťahom pre gaussouskú susednosť (vzťah 4.4). Na trénovanie dotykov sme použili MRF-SOM mapu. Trénovali sme ju na nami vygenerovanými dátami pre dotyky. Vytvorili sme vektory dotykov s dĺžkou 32 pre každú ruku. Každý vektor



Obr. 6.1: Vývoj trénovania perceptronu na surových dátach pre účely filtrovania nedotykových konfigurácií.

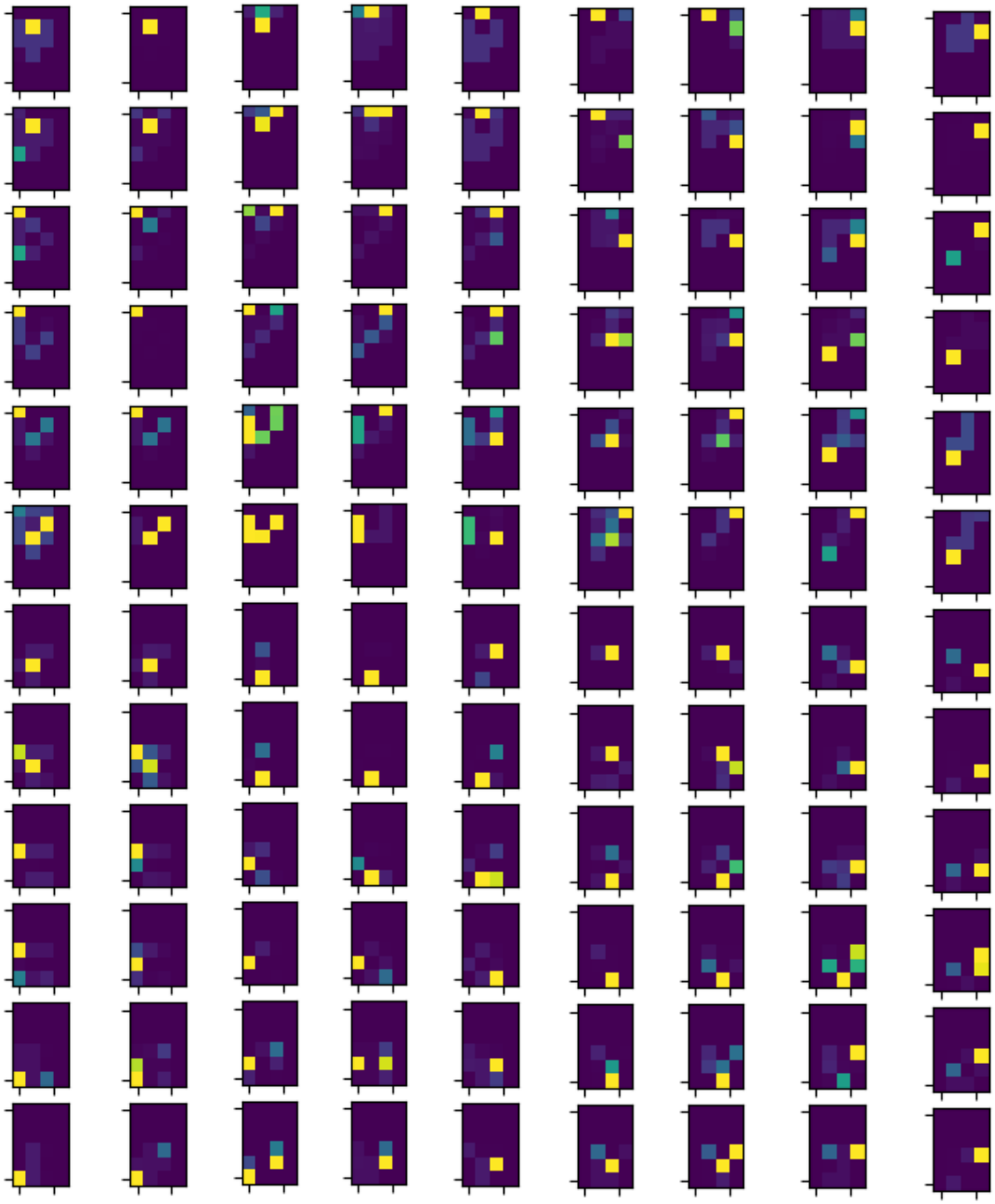
reprezentoval informácie o dotyku na ruke. Generovali sme dotyky na jednom a na dvoch trojuholníkoch modelu umelej kože vedľa seba. MRF-SOM mapu sme rozdelili po vzore homonkula (obr. 1.1) na dve samostatné mapy. Dlaň s prstami (veľkosť 9 senzorov) a predlaktie (veľkosť 23 senzorov). Mapu dlane sme trénovali: maskou sme oddelili prsty od samotnej dlane a trénovali sme ju na generovaných dátach s jedným alebo dvoma (45 vzorov) (obr. 6.3). Použili sme mapu s rozmermi 7×7 neurónov.

Predlaktie sme trénovali rovnako na vygenerovaných dátach s jedným alebo dvoma dotykmi vedľa seba (obr. 6.4). Sieť sme rozdelili maskou na štyri kvadranty. Samotná sieť má rozmery 12×9 neurónov.

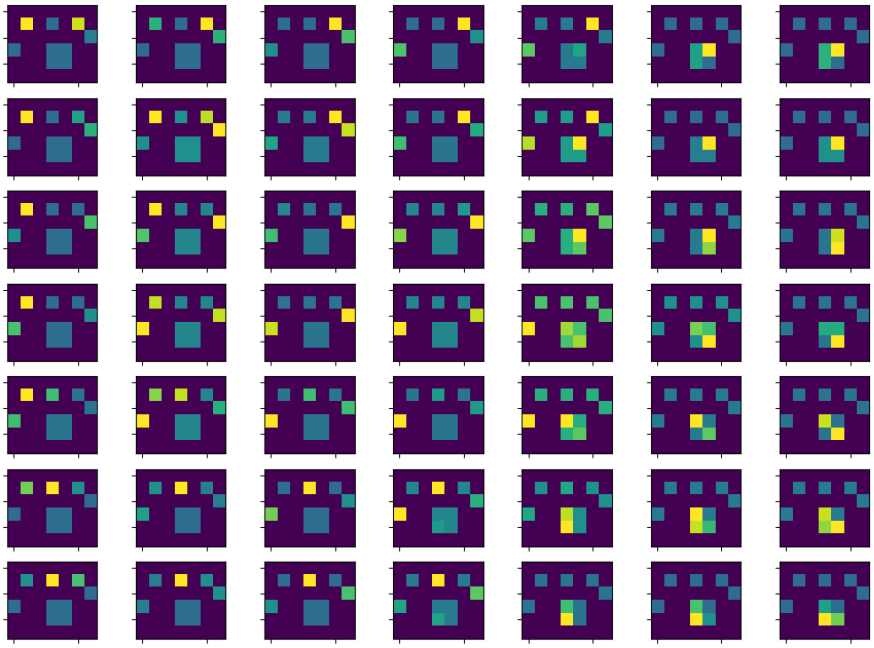
6.3 Trénovanie asociátora

Ako tretie sme trénovali model BAL vo funkcii asociátora na spracovaných dátach pomocou modelov SOM. Pred použitím filtra sme BAL trénovali na množine obsahujúcej dotykové aj nedotykové konfigurácie rúk. Trénovacia množina spôsobovala náhodných výskyt vysokých hodnôt klasifikačných chýb opísaných v časti 5.3.1. Táto skutočnosť bola dôvodom na zaradenie filtra do nášho komplexného modelu. Na ďalšie zlepšenie procesu učenia asociátora (model BAL) sme použili nasledovné 3 postupy:

Obr. 6.2: Výsek 8×8 neurónov natrénovanej SOM, so znázornením naučených póz ľavej ruky robota iCub. Zrejmá je topologická organizácia váh víťazov (reprezentujúcich preferované konfigurácie).

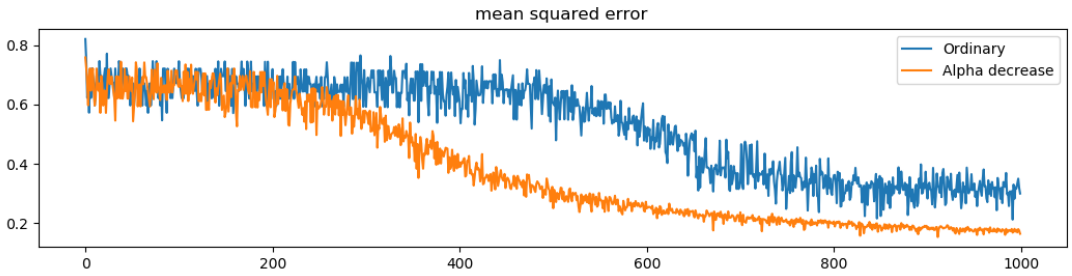


Obr. 6.3: Znázornenie naučených reprezentácií v natrénovanej MRF-SOM (12×9 neurónov) pre jednotlivé dotyky na predlaktí. Každý obrázok obsahuje 23 trojuholníkov taxelov. Každý bod je farebne znázornený, svetlosť farby predstavuje intenzitu dotyku. Žltá farba predstavuje dotyk. Zelená farba predstavuje možný dotyk a modrá farba predstavuje nedotyk. Jednotlivé dotyky sú topologicky usporiadané. MRF maska rozdeľuje mapu na dotyky štyroch kvadrantov.



Obr. 6.4: Znázornenie naučených reprezentácií v natrénovanej MRF-SOM pre jednotlivé dotyky na dlani. Každý obrázok obsahuje 9 trojuholníkov taxelov (5 na prstoch, 4 v dlani). Každý bod je farebne znázornený, svetlosť farby predstavuje intenzitu dotyku. Žltá farba predstavuje dotyk. Zelená farba predstavuje možný dotyk a modrá farba predstavuje nedotyk. Jednotlivé dotyky sú topologicky usporiadané. MRF maska rozdeľuje mapu na dotyky prstov a dlaní (hranica medzi maskami je v štvrtom stĺpci).

1. Postupné zmenšovanie parametra rýchlosti učenia α vedie k zlepšeniu učenia sa modelu BAL. Z predchádzajúcich testov sme zistili, že $\alpha = 0.25$ je vhodná hodnota, pri ktorej sa BAL učí najlepšie. Časom je táto hodnota však priveľká na jemné zmeny. Preto sme sa rozhodli ju zmenšovať a to postupne po 40 epochách na 95% predchádzajúcej hodnoty až na hodnotu 0.01. Proces zmenšovania tohto parametra zjavne pomáhalo pri procese učenia (obr. 6.5). BAL sa bol schopný po veľa epochách naučiť viac vzorov.

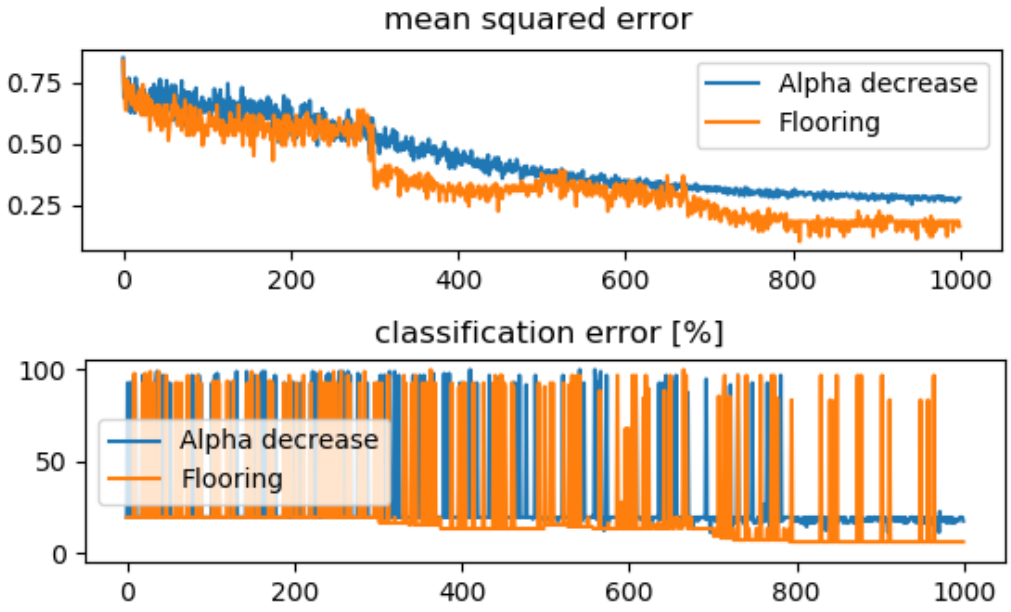


Obr. 6.5: Ukážka pozitívneho vplyvu znižovania rýchlosti učenia pri trénovaní modelu.

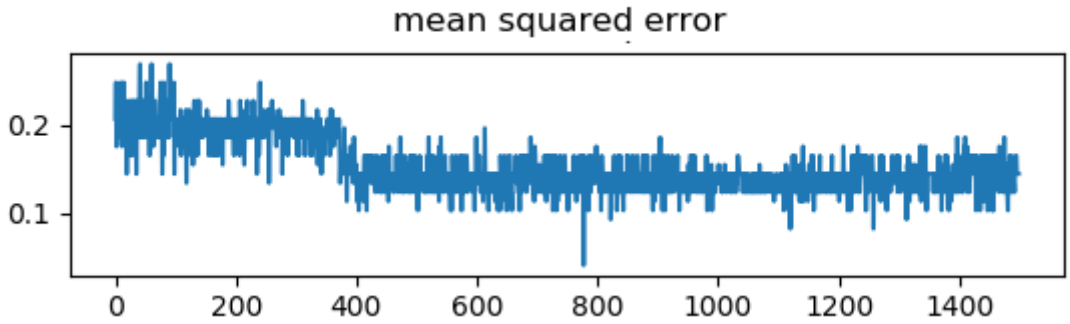
2. BAL používa sigmoidu (vzťah 4.10) ako aktivačnú funkciu. Preto nás napadlo použiť ďalšie zaokrúhľovanie dát pri predikcii. Dôvodom je, že sigmoida má najväčšiu deriváciu v bode 0 (s hodnotou 0.5). Preto sme posunuli zaokrúhľovanie na 0.4, kde nie je derivácia tak veľká a mála zmena hodnoty veľmi nezmení výsledok. Táto zmena však priniesla len malé zlepšenie.

3. Netrénovať BAL do extrémov. Rozhodli sme sa predikčné hodnoty zaokrúhľovať. Takto sme dosiahli, že pri "jasných" hodnotách sa model BAL už ďalej neučí. Zaokrúhľovali sme $y > 0.75$ na 1 a $y < 0.25$ na 0. Táto zmena významne pomohla procesu učenia (obr. 6.6). Výrazne zrýchlila trénovanie a zvýšila aj úspešnosť trénovania. Pred aplikáciou tohto vylepšenia BAL dosahoval 85-90% úspešnosť. Po zaokrúhľovaní model dosahuje 95% úspešnosť pri učení. Na druhú stranu sme si všimli, že ak zaokrúhľujeme príliš agresívne, tak chyba siete začne oscilovať (obr. 6.8). Totižto ak sa sieť pomýlila, tak naše zaokrúhľovanie spôsobilo zväčšenie chyby a následné prehnané korekcie. Tento fakt spôsobuje, že sa model nezastabilizuje na jednom mieste. Preto sme hľadali iný spôsob. Namiesto zaokrúhľovania sme posilňovali. To znamená, ak hodnota bude väčšia ako 0.65, tak k nej pripočítame konštantu β ale neprekročíme jednotku. Ak hodnota je menej ako 0.35 tak od nej odčítame túto konštantu β ale číslo bude najmenej nula. Ukázalo sa, že pri malej konštante β sa BAL síce učí pomalšie ako pri zaokrúhľovaní, ale nespôsobuje oscilovanie chyby (obr. 6.7), pričom úspešnosť modelu zostáva rovnaká. Pri trénovaní sme použili hodnotu $\beta = 0.2$.

Pri trénovaní asociátora sa nám podarilo dosiahnuť v niektorých prípadoch aj 100% úspešnosť. V rámci testov asociácie dotykov sme dosiahli 92% úspešnosť.



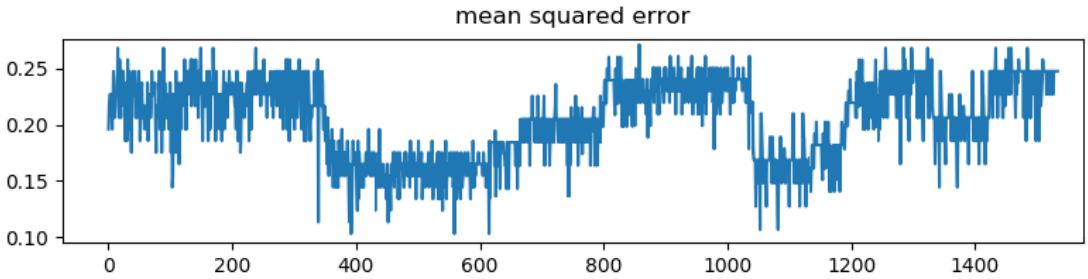
Obr. 6.6: Ukážka pozitívneho vplyvu netrénovania modelu BAL do extrémov.



Obr. 6.7: Správanie modelu BAL, keď sa naučí trenovanú množinu pomocou posilňovania učenia. Graf zobrazuje úsek dlhý 1500 epoch (nie od začiatku).

6.4 Testovanie finálneho modelu

Po natrénovaní všetkých častí sme testovali komplexný model. Na testovanie celého modelu sme použili úplne nové dáta, ktoré sme nazbierali ručne pomocou `motorGui`. Tieto nové dáta sme zozbierali tak, aby sme mali širokú paletu konfigurácií rúk, aby sme pokryli veľké množstvo možností. Testovacia vzorka obsahuje 30 nedotykových a 20 dotykových konfigurácií. Nedotykové konfigurácie obsahovali zástupcov vzdialených rúk a aj tesných nedotykov. Dotykové dáta obsahovali dotyky jedného alebo dvoch prstov (ukazovák, ukazovák a prostredník).



Obr. 6.8: Správanie modelu BAL, keď sa naučí trenovanú množinu pomocou zaokrúhľovania. Graf zobrazuje úsek dlhý 1500 epoch (nie od začiatku).

Pri testovaní "dotykovosti" filter správne označil 47 konfigurácií(z 50) pričom 2 dotykové konfigurácie označil za nedotykové a 1 nedotykovú konfiguráciu označil za dotykovú konfiguráciu. Táto nedotyková konfigurácia bola tesným nedotykom. Asociáciu dotykov sme trénovali tak, že obe modality sme transformovali pomocou SOM respektíve MRF-SOM máp a kontrolovali sme, či asociátor správne priradí transformované dáta. Znepresnenie transformáciou sme nebrali do úvahy tj. reprezentant obsahoval viac dotykov než pôvodné dáta (takéto znepresnenie nepredstavuje veľkú chybu). Pri testovaní asociácie dotykov sme zistili, že model BAL správne označil 17 konfigurácií, pričom nesprávne asocioval dva jednodotyky a jeden dvojdotyk. Všetky tri zlé asociácie boli asociácie s modelom MRF-SOM transformujúcim dlaň robota. Teda výsledný test dopadol tak, že z 50 testovacích dát model správne asocioval 44 konfigurácií, čo predstavuje 88% úspešnosť. Dôvodom nižšej schopnosti generalizácie celého modelu oproti testovaniu jednotlivých modelov je ten, že používame dva modely za sebou, a teda ich chyby sa akumulujú. Okrem toho naše testovanie prebehlo na pomerne malej množine dát, kde jedna chyba predstavuje veľkú relatívnu chybu.

Záver

Prvým výsledkom práce je vytvorenie postupu na zber proprioceptívno-dotykových informácií z humanoidného robota iCub. Pre tento účel sme predstavili algoritmus, ktorý za pomoci spätnej kinematiky vytvára nové dotykové konfigurácie. Následne z dôvodu nevhodnosti modulu počítajúceho inverznú kinematiku sme tento algoritmus upravili do podoby, ktorá nepotrebuje inverznú kinematiku a mohli sme ho aplikovať na robota iCub.

Druhým výsledkom práce je nami navrhnutý model na akvizíciu porprioceptívno-dotykových informácií tela humanoidného robota iCub. Model sa skladá s troch častí. Prvá časť predstavuje filter, ktorého úloha spočíva v odhaľovaní nedotykových konfigurácií rúk robota. Filter sme implementovali pomocou viacvrstvového perceptronu s dvoma skrytými vrstvami. Druhú časť predstavuje transformácia vstupných hodnôt pre zjednodušenie asociácie. Vstupné vzory sú transformované pomocou samoorganizujúcich sa máp za účelom nahradenia vzoru jeho reprezentantom. Treťou častou model je samotná asociácia reprezentantov vstupných vzorov. V úlohe asociátora sme použili model BAL, na ktorý sme aplikovali nami navrhnuté vylepšenia. Jednotlivé časti komplexného modelu sme učili samostatne. Nakoniec sme celý model testovali na nových ručne nazbieraných dátach, za pomoci modulu `motorGui`, aby sme otestovali rôzne konfigurácie rúk, predstavujúce veľkú časť priestoru, do ktorého sa môžu dostať ruky robota. Náš model v tomto testovaní preukazuje schopnosť generalizácie. Testovanie ukázalo, že pozície nazbierané v prvej časti práce boli dostatočné aj na generalizáciu veľmi vzdialených pozícií rúk robota.

Literatúra

Bednárová, N. (2015). Reprezentace proprioceptivních vstupu humanoidního robota iCub pomocí samoorganizujích se map. Bakalárska práca. ČVUT Praha.

Crair, M. C. (1999). Neuronal activity during development: permissive or instructive? *Current Opinion in Neurobiology*, 9(1):88–93.

Farkaš, I. and Rebrová, K. (2013). Bidirectional activation-based neural network learning algorithm. In *International Conference on Artificial Neural Networks*, pages 154–161. Springer.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hoffmann, M., Straka, Z., Farkaš, I., Vavrečka, M., and Metta, G. (2017). Robotic homunculus: Learning of artificial skin representation in a humanoid robot motivated by primary somatosensory cortex. *IEEE Transactions on Cognitive and Developmental Systems*, 10(2):163–176.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., of Biochemistry, D., Jessell, M. B. T., Siegelbaum, S., and Hudspeth, A. (2000). *Principles of Neural Science*, volume 4. McGraw-hill New York.

Kim, S. S., Gomez-Ramirez, M., Thakur, P. H., and Hsiao, S. S. (2015). Multimodal interactions between proprioceptive and cutaneous signals in primary somatosensory cortex. *Neuron*, 86(2):555–566.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

Krubitzer, L., Huffman, K. J., Disbrow, E., and Recanzone, G. (2004). Organization of area 3a in macaque monkeys: contributions to the cortical phenotype. *Journal of Comparative Neurology*, 471(1):97–111.

Kvasnička, V., Beňušková, L., Pospíchal, J., Farkaš, I., Tiňo, P., and Král', A. (1997). *Úvod do teórie neurónovỳch sietí*. IRIS Bratislava.

Leyton, A. S. and Sherrington, C. S. (1917). Observations on the excitable cortex of the chimpanzee, orang-utan, and gorilla. *Quarterly Journal of Experimental Physiology*, 11(2):135–222.

Longo, M. R., Azañón, E., and Haggard, P. (2010). More than skin deep: body representation beyond primary somatosensory cortex. *Neuropsychologia*, 48(3):655–668.

Malinovská, K., Malinovskỳ, L., and Farkaš, I. (2018). Towards more biologically plausible error-driven learning for artificial neural networks. In *International Conference on Artificial Neural Networks*, pages 228–231. Springer.

Metta, G., Fitzpatrick, P., and Natale, L. (2006). Yarp: yet another robot platform. *International Journal of Advanced Robotic Systems*, 3(1):8.

Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., Von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., et al. (2010). The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 23(8-9):1125–1134.

Nori, F., Traversaro, S., Eljaik, J., Romano, F., Del Prete, A., and Pucci, D. (2015). iCub whole-body control through force regulation on rigid non-coplanar contacts. *Frontiers in Robotics and AI*, 2:6.

O'Reilly, R. C. (1996). Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, 8(5):895–938.

Proske, U. and Gandevia, S. C. (2012). The proprioceptive senses: their roles in signaling body shape, body position and movement, and muscle force. *Physiological Reviews*, 92(4):1651–1697.

Rochat, P. (1998). Self-perception and action in infancy. *Experimental Brain Research*, 123(1-2):102–109.

Roncone, A., Hoffmann, M., Pattacini, U., and Metta, G. (2014). Automatic kinematic chain calibration using artificial skin: self-touch in the iCub humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2305–2312. IEEE.

Sherrington, C. S. (1907). On the proprioceptive system, especially in its reflex aspect. *Brain*, 29(4):467–482.

Tikhanoff, V., Cangelosi, A., Fitzpatrick, P., Metta, G., Natale, L., and Nori, F. (2008). An open-source simulator for cognitive robotics research: the prototype of the iCub humanoid robot simulator. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 57–61. ACM.

Elektronická príloha

Diplomová práca obsahuje okomentované zdrojové kódy v programovacom jazyku C++ a Python3, ako aj zdrojové súbory textov a grafov použitých v práci.