Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00020/2010
Datum: 01.09.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko ter Fakulteta za matematiko in fiziko izdaja naslednjo nalogo:

Kandidat: **SIMON KOZINA**

Naslov: **RAZVOJ SISTEMA ZA NADZOROVANJE SPREMEMB PRI HOJI ČLOVEKA**

DEVELOPMENT OF A SYSTEM FOR MONITORING CHANGES IN HUMAN GAIT

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Na tržišču je vse več sistemov za oddaljeno varnostno nadzorovanje starejših ljudi. Večina teh sistemov je dragih in ljudem nedostopnih, saj uporabljajo zelo napredno opremo za nadzor. Osnovne funkcije nadzora bi lahko opravljali s cenovno dostopnimi senzorji, kot so pospeškomeri. Tema te diplomske naloge je preučiti možnosti uporabe podatkov o pospeških obeh nog pri ugotavljanju sprememb pri hoji človeka. Definirajte ustrezne atribute, formulirajte učni problem in eksperimentalno poiščite čim ustreznejši učni algoritem za dani problem. V okviru diplomskega dela naj bo izdelano in eksperimentalno ovrednoteno programsko orodje, ki realizira navedene funkcije.

Mentor:

akad. prof. dr. Ivan Bratko

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Franc Solina

Somentor:

prof. dr. Matjaž Gams

Dekan Fakultete za matematiko in fiziko:

prof. dr. Andrej Likar

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Simon Kozina

RAZVOJ SISTEMA ZA NADZOROVANJE SPREMEMB PRI HOJI ČLOVEKA

DIPLOMSKO DELO
NA INTERDISCIPLINARNEM UNIVERZITETNEM ŠTUDIJU

Mentor: akad. prof. dr. Ivan Bratko

Ljubljana, 2010

Rezultati diplomskega dela so intelektualna lastnina Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje Fakultete za računalništvo in informatiko ter mentorja.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Simon Kozina,

z vpisno številko 63050191,

sem avtor diplomskega dela z naslovom:

Razvoj sistema za nadzorovanje sprememb pri hoji človeka

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  akad. prof. dr. Ivana Bratka
  in somentorstvom
  izr. prof. dr. Matjaža Gamsa
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 20.09.2010 Podpis avtorja:

Zahvala

Zahvalil bi se svojemu mentorju profesorju Ivanu Bratku in somentorju profesorju Matjažu Gamsu za vse nasvete ob izdelavi tega dela. Velika zahvala gre tudi Boži Cvetkovič in Eriku Dovganu iz Odseka za inteligentne sisteme, ki sta mi pomagala pri zajemu podatkov in uporabi sistema Confidence.

Posebna zahvala tako za moralno kot tudi finančno podporo tekom celotnega študija gre staršem in starim staršem. Zahvalil bi se tudi punci Ines za potrpljenje, podporo in potrebno vzpodbudo k dokončanju študija.

Kazalo

Povzetek **1**

Abstract **3**

1 Uvod **5**
1.1 Motivacija in cilji 5
1.2 Pregled področja 6
1.3 Pregled vsebine 7

2 Uporabljene metode in orodja **9**
2.1 Strojno učenje 9
2.1.1 Naivni Bayesov klasifikator 10
2.1.2 Klasifikacijsko drevo 10
2.1.3 Naključni gozdovi 10
2.1.4 Metoda najbližjih sosedov 11
2.1.5 Metoda podpornih vektorjev 11
2.2 Mere ocenjevanja točnosti 11
2.2.1 Klasifikacijska točnost 11
2.2.2 Ploščina pod ROC krivuljo (AUC) 12

3 Zaznavanje gibanja in iskanje atributov **15**
3.1 Naprava za zaznavanje gibanja 15
3.2 Način zajemanja in opis podatkov 16
3.3 Definicija učnega primera 18
3.4 Določanje korakov 19
3.4.1 Algoritem 19
3.4.2 Predolgi koraki 21
3.5 Računanje atributov 23

4 Testiranje **27**
4.1 Opis učne in testne množice 27
4.2 Poskusi 28
4.2.1 Poskus na petsekundnem časovnem intervalu 29
4.2.2 Poskus na desetsekundnem časovnem intervalu 30
4.2.3 Poskus na tridesetsekundnem časovnem intervalu 31
4.2.4 Poskus na šestdesetsekundnem časovnem intervalu 33
4.3 Rezultati testiranja 34
4.3.1 Predstavitev modelov strojnega učenja 35
4.4 Sistem za nadzor hoje 37

5 Diskusija in zaključek **39**

Seznam slik **41**
Seznam tabel **43**

Literatura **45**

Seznam uporabljenih kratic in simbolov

kNN	k Nearest Neighbours - k najbližjih sosedov
SVM	Support Vectore Machine - metoda podpornih vektorjev
CA	Classification Accuracy - klasifikacijska točnost
AUC	Area Under Curve - ploščina pod krivuljo
ROC	Receiver Operating Characteristic - sprejemnikova operativna značilnost
CT	Classification Tree - klasifikacijsko drevo
NB	Naive Bayes Classificator - naivni Bayesov klasifikator
RF	Random forrest - množica naključnih dreves
MAC	Media Access Control

Povzetek

Na tržišču je vse več sistemov za oddaljeno nadzorovanje starejših ljudi. Večina teh sistemov je dragih in ljudem nedostopnih, saj uporabljajo zelo napredno opremo za nadzor. Osnovne funkcije nadzora bi lahko opravljali s cenovno dostopnimi senzorji, kot so pospeškomeri.

Namen diplomskega dela je bila izdelava sistema za nadzor sprememb v hoji človeka. Pri delu smo uporabljali pospeške leve in desne noge v vseh treh koordinatnih smereh. Ugotovili smo, da je najboljši pristop k rešitvi problema delitev naše množice podatkov na fiksne časovne intervale. Brez te predpostavke bi težko določili pravilne atribute potrebne za strojno učenje.

Da bi bolje razumeli podatke enega časovnega intervala, smo hojo razdelili na njene primitivne elemente - korake. S pomočjo splošne karakteristike hoje smo zasnovali dva algoritma. Skupaj nam razdelita podatke na posamezne korake. Izbrali smo osem različnih atributov značilnih za hojo. Nato smo poskusili ustvariti univerzalni model strojnega učenja, ki naj bi prepoznal razlike med normalno hojo in šepanjem. Testiranje smo opravili na različnih časovnih intervalih. Izkazalo se je, da je podatke bolje razrezati na daljše množice. Tako dobimo bolj zanesljive informacije o hoji človeka na obravnavanem intervalu. Ugotovili smo, da je SVM najboljši model strojnega učenja za predstavljen klasifikacijski problem.

Ključne besede:

nadzor hoje, strojno učenje, pospeškomer, ocenjevanje točnosti

Abstract

On the market, there are more and more different systems for monitoring elderly people. Most of the systems are expensive and inaccessible since they use very sophisticated monitoring equipment. Basic functions of monitoring could be done by reasonably priced sensors such as accelerometers.

The object of this work was to develop the system for monitoring changes in human gait. We used data from accelerometers attached to the left and right ankle. Accelerations are represented in three-dimensional coordinate system. We realized the best solution to our problem was to divide data set into fixed time intervals. Without this assumption it would have been difficult to determine the right attributes for machine learning.

To better understand the data in one interval, we divided walking into its primitive elements - steps. By using some gait characteristics we were able to conceive two algorithms. Together they enabled us to divide data into single steps. Eight different attributes distinctive for walking were chosen. The next thing was an attempt to create universal training data set that could recognize the difference between normal walking and limping. Testing was done on different time intervals. It proved successful to divide data into longer time intervals. By doing so we got better information about gait in one time interval. SVM proved to be the optimal machine learning model for the presented classification problem.

Key words:

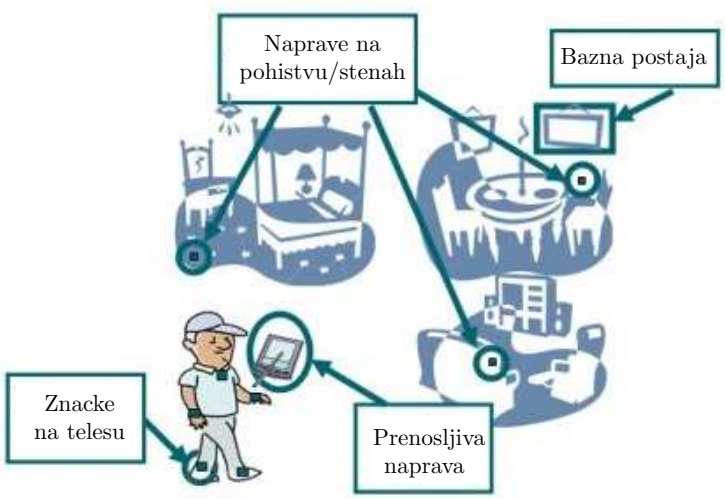
gait monitoring, machine learning, accelerometer, accuracy assessment

Poglavje 1

Uvod

1.1 Motivacija in cilji

Evropska populacija se stara. Razlog leži v dvigu življenjske dobe in zmanjševanju rodnosti. Procent ljudi starejših od 65 let naj bi se dvignil iz 17,9 % v letu 2007 na 53,5 % v letu 2060. Število starejši ljudi bo tako preraslo družbeno sposobnost za njihovo skrb. Torej moramo poskrbeti, da bodo starejši ljudje lahko dlje živeli neodvisno, seveda z minimalno podporo aktivne populacije. To je tudi primarni cilj projekta Confidence[7]. Glavni cilj projekta Confidence



Slika 1.1: Prikaz sistema Confidence.

je izdelava in integracija novih tehnologij, s katerimi bi lahko izdelali sistem

za detekcijo neobičajnih dogodkov (padci, šepanje) ali nepričakovanih vedenj, ki bi lahko bila povezana z zdravstvenimi problemi starejših ljudi. Sistem je namenjen notranji uporabi. Kot vidimo na sliki 1.1, imamo v stanovanju bazno postajo in več naprav, ki komunicirajo z njo. Naprave po stanovanju komunicirajo s prenosljivo napravo, ki jim pošilja podatke značk. Značke so lahko že všite v oblačila ali kako drugače integrirane v različne pripomočke. Bazna postaja nadzoruje gibanje človeka in v primeru nepravilnosti to sporoči zdravstveni ustanovi.

Celoten sistem temelji na značkah, ki bazni postaji sporočajo svoje koordinate v prostoru. Čeprav je sistem zasnovan tako, da je cena le-tega občutno manjša, kot cena nekaterih drugih podobnih sistemov, se še vedno giblje okoli 2000 EUR.

Namen tega diplomskega dela je pokazati, da lahko določene dele sistema nadomestimo s še cenejšimi komponentami. Skozi nalogo se bomo ukvarjali s prepoznavanjem hoje zgolj s pospeški levega in desnega gležnja. Cilja diplomskega dela sta:

- Ustvariti programsko okolje, ki bo podatke pospeškomerov leve in desne noge pretvoril v uporabne atribute, ki bodo primerni za strojno učenje.
- Poiskati najboljši klasifikator, s katerim bi s pomočjo vnaprej določene učne množice pravilno klasificirali hojo poljubnega človeka. Problem hoje človeka bomo omejili na hojo in šepanje. Torej bomo imeli v resnici binarni razred, kjer bo 1 pomenila normalno hojo z 0 pa bomo označili šepanje.

Seveda bo točnost algoritmov strojnega učenja odvisna tudi od pravilne izbire atributov.

1.2 Pregled področja

Objavljenih je kar nekaj raziskav in opravljenih več testiranj, ki so poskušale klasificirati človeške aktivnosti samo s pomočjo pospeškomerov na telesu. Ravi in ostali[6] so poskušali določiti osem različnih aktivnosti (hoja, tek, sesanje, počepi, hoja po stopnicah gor in dol, umivanje zob, biti pri miru). Uporabljali so en sam pospeškomer, ki je bil nameščen na trebušnem predelu. Za potrebe strojnega učenja so ločili štiri različne primere:

- Združili so podatke iste osebe in preverjali uspešnost s prečnim preverjanjem. Točnost je bila 98%.

- Združili so podatke vseh oseb in preverjali kot v zgornjem primeru. Tudi tukaj je bila točnost zelo visoka, kar $97,8\%$.
- Podatki ene osebe, ki so bili zbrani v enem dnevu so uporabili kot učno množico, testno množico pa so sestavljali podatki iste osebe zbrani v ostalih dneh. Točnost takšnega testiranja je bila $75,5\%$.
- Vse podatke ene osebe so uporabili za učno množico. Uspešnost so testirali na ostalih osebah in dosegli 57,7 % točnost.

Mannini in Sabatini[4] sta poskušala razlikovati med sedmimi različnimi aktivnostmi (hoja, tek, ležanje,...) s pomočjo petih pospeškomerov nameščenih po celotnem telesu. Učno množico sta pripravila za vsako osebo posebej. Tako je bilo učenje prirejeno aktivnostim samo trenutne testirane osebe. Atributi stojnega učenja so bili kar podatki o pospeških v vseh treh koordinatnih smereh. Uspelo jim je doseči $95,6\%$ klasifikacijsko točnost.

Vidimo, da so raziskave in testiranja večinoma usmerjena na testiranje ene osebe. To bi bi radi spremenili in ustvarili univerzalno učno množico. Poskušali bomo najti ustrezne atribute, ki bi posplošili problem učenja hoje in nam tako pomagali pri pravilni klasifikaciji hoje različnih ljudi.

1.3 Pregled vsebine

V drugem poglavju si na kratko ogledamo različne klasifikacijske metode uporabljene pri določanju optimalne metode strojnega učenja. Odločimo se tudi za dve meri točnosti, ki jih bomo uporabljali pri razlikovanju uspešnosti klasifikacijskih metod. V tretjem poglavju so opisani uporabljeni pospeškomeri in podatki s katerimi razpolagamo. Razvijemo dva algoritma, katera nam pomagata pri ločevanju posameznega koraka na določenem časovnem intervalu. Sledi še predstavitev atributov uporabljenih pri stojnem učenju. Različni poskusi testiranja so opisani v četrtem poglavju. Odločimo se za testiranje na štirih različnih časovnih intervalih in na vsakem izberemo optimalno metodo klasifikacije. Ogledamo si še dva različna modela za klasifikacijo novih primerov. V zadnjem poglavju sledijo še diskusija, pregled rezultatov in zaključki.

Poglavje 2

Uporabljene metode in orodja

Večina programske kode, razvite v namen te diplomske naloge, je bila napisana v programskem jeziku Python (www.python.org). Spada v skupino skriptnih jezikov in ni zelo učinkovit, saj je čas izvajanja programov precej daljši kot pri nekaterih objektno orientiranih programskih jezikih (Java, C++). To njegovo pomanjkljivost odpravi možnosti hitrega pisanja kode in sprotnega preverjanja razvitih modulov.

Pri strojnem učenju nam je bilo v veliko pomoč ogrodje za strojno učenje in odkrivanje znanj Orange[1]. Zasnovan je modularno in omogoča integracijo s programskim jezikom Python, kar pomeni preprosto in učinkovito kombiniranje uporabnikovih funkcij z že obstoječimi. V sistemu Orange so že implementirane vse metode strojnega učenja, ki so opisane v nadaljevanju.

2.1 Strojno učenje

Poznamo veliko različnih metod strojnega učenja. Kononenko[2] jih deli glede na način uporabe naučenega znanja: klasifikacija, regresija, učenje asociacij in logičnih relacij, učenje sistemov enačb in razvrščanje. Pri reševanju našega problema bomo uporabljali metode strojnega učenja, katere delujejo na principu klasifikacije. Naloga klasifikatorja je določiti razred problemu, ki je opisan z množico atributov. Atributi so neodvisne zvezne ali diskretne spremenljivke, s katerimi opisujemo primere, razred pa je odvisna diskretna spremenljivka, ki ji določimo vrednost glede na vrednost neodvisnih spremenljivk. Da lahko klasifikator določi razred, mora imeti določeno diskretno/klasifikacijsko funkcijo, ki preslika prostor atributov v razred. Glede na funkcijo ločimo več različnih klasifikatorjev.

2.1.1 Naivni Bayesov klasifikator

Naloga naivnega Bayesovega klasifikatorja (NB) je izračunati pogojne verjetnosti za vsak razred pri danih vrednostih atributov za nov primer, ki ga želimo klasificirati. Bayesov klasifikator, ki točno izračuna vse pogojne verjetnosti je optimalen, ker minimizira pričakovano napako. V splošnem ne moremo izračunati vseh pogojnih verjetnosti in moramo zato predpostaviti pogojno neodvisnost atributov. Čeprav sedaj izračunane pogojne verjetnosti niso več optimalne, nam učna množica skoraj vedno zadošča za zanesljivo oceno vseh potrebnih verjetnosti za izračun končne pogojne verjetnosti vsakega razreda.

2.1.2 Klasifikacijsko drevo

Klasifikacijsko drevo (CT) je sestavljeno iz notranjih vozlišč, vej in listov. Notranja vozlišča ustrezajo atributom, veje ustrezajo podmnožicam vrednosti atributov in listi ustrezajo razredom. Pot v drevesu od korena do lista ustreza enemu odločitvenemu pravilu. Algoritmi za gradnjo klasifikacijskih dreves in pravil izbirajo atribute glede na oceno informativnosti posameznih atributov. Atributi z višjo oceno informativnosti so bližje korenu drevesa, kot tisti z nižjo oceno. Klasifikacija novega primera poteka tako, da se uporabi ustrezno odločitveno pravilo.

2.1.3 Naključni gozdovi

Naključni gozdovi (RF) so ena izmed metod, ki zgradijo model s kombiniranjem klasifikatorjev, naučenih z enim učnim algoritmom. Osnovni algoritem, ki ga uporabljajo naključni gozdovi so klasifikacijska dreves. Ideja je generirati zaporedje klasifikacijskih dreves, tako da se pri izbiri najboljšega atributa v vsakem vozlišču naključno izbere majhno število atributov med katerimi se poišče najboljšega. V vsakem vozlišču naj bi izbirali med logaritmom števila atributov plus ena, lahko pa uporabimo tudi kakšno drugo mejo. Za razliko od odločitvenih dreves tukaj nimamo samo enega drevesa, vendar se jih generira veliko (Orange ima za privzeto vrednost število naključnih dreves 15, ponavadi pa se jih generira več, tudi 100).

Vsako zgrajeno drevo se uporabi za klasifikacijo novega primera. Končen razred primera se določi na podlagi glasovanja. Vsako drevo ima en glas, ki ga nameni enemu od razredov. Porazdelitev glasov normiramo in dobimo distribucijo verjetnosti po vseh razredih.

2.1.4 Metoda najbližjih sosedov

Najpreprostejša oblika algoritma najbližjih sosedov (kNN) kot znanje uporablja kar množico vseh učnih primerov. Učni algoritem si samo zapomni vse primere. Pri klasifikaciji novega primera iz učne množice poišče k najbližjih primerov - sosedov. Nov primer klasificira v razred kateremu pripada večina najbližjih sosedov. Pri tem je potrebno zaradi ustrezne metrike v prostoru atributov normalizirati vrednosti zveznih atributov in definirati razdaljo (metriko) med vrednostmi vsakega diskretnega atributa.

2.1.5 Metoda podpornih vektorjev

Osnovna ideja metode podpornih vektorjev (SVM) je v danem prostoru atributov postaviti optimalno hiperravnino. Optimalna hiperravnina je tista, ki je enako in najbolj oddaljena od najbližjih primerov iz obeh razredov. Najbližjim primerom optimalne hiperravnine pravimo podporni vektorji, razdalji hiperravnine do podpornih vektorjev pa rob. Naloga algoritma SVM je najti hiperravnino, ki ima maksimalen rob.

V originalnem prostoru linearna hiperravnina pogosto ne zadošča za sprejemljivo klasifikacijsko točnost. Tako lahko naredimo tudi implicitno transformacijo atributnega prostora v kompleksnejši atributni prostor. Z nelinearno transformacijo lahko postane prostor primeren za linearno ločitveno hiperravnino. Na ta način lahko z različnimi transformacijami rešujemo različne nelinearne probleme.

2.2 Mere ocenjevanja točnosti

V želji, da najdemo najboljšo metodo strojnega učenja za naš problem, se moramo vprašati, na kakšen način bomo ocenjevali točnost različnih klasifikatorjev. Odločili smo se za uporabo dveh mer za ocenjevanje točnosti.

2.2.1 Klasifikacijska točnost

Klasifikacijska točnost (CA) je ena od mer, ki se uporablja zelo pogosto. Definirana je kot delež primerov, pri katerih je klasifikator pravilno napovedal razred:

$$CA = \frac{N^{(p)}}{N} \cdot 100\%$$

N je število vseh možnih primerov in $N^{(p)}$ število pravilno klasificiranih problemov.

Klasifikacijska točnost je popolnoma neobčutljiva v primerih verjetnostne klasifikacije. Oglejmo si naslednji primer: imamo dva različna klasifikatorja in eno testno množico. Oba napovedujeta verjetnosti istega razreda. Prvi za vse primere testne množice napove verjetnost 0,51, drugi pa verjetnost 1. Klasifikacijska točnost obeh klasifikatorjev bi bila 100%. Iz tega bi lahko sklepali, da sta oba algoritma enako dobra. Seveda temu ni tako, saj je drugi klasifikator boljši kot prvi. Zaradi te pomanjkljivost smo se odločili uporabiti še dodatno mero ocenjevanja točnosti AUC.

2.2.2 Ploščina pod ROC krivuljo (AUC)

Pri iskanju optimalnega klasifikatorja nas zanima predvsem razmerje med senzitivnostjo in specifičnostjo. Senzitivnost ocenjuje odstotek pravilno klasificiranih pozitivnih primerov:

$$Senz = \frac{TP}{TP + FN}$$

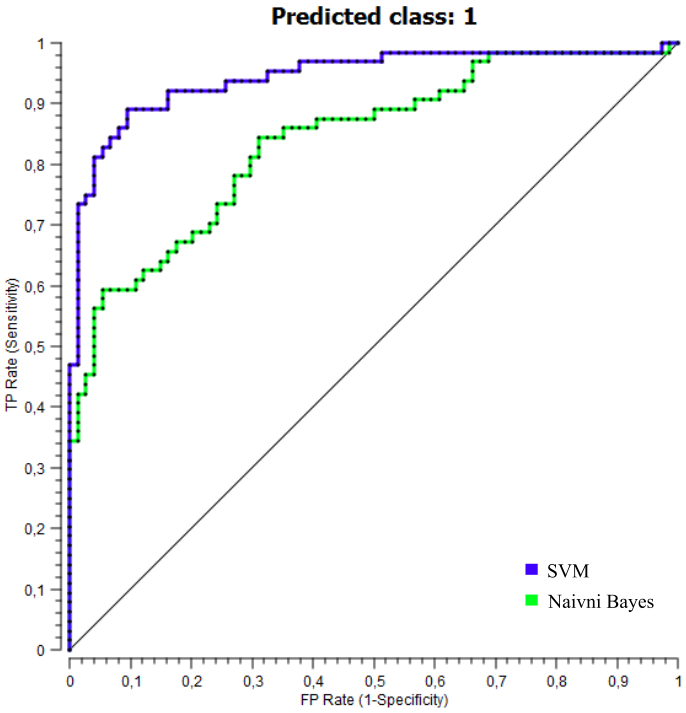
Specifičnost pa ocenjuje odstotek pravilno klasificiranih negativnih primerov:

$$Spec = \frac{TN}{TN + FP}$$

TP	-	število pravilno klasificiranih pozitivnih primerov
FP	-	število napačno klasificiranih negativnih primerov
TN	-	število pravilno klasificiranih negativnih primerov
FN	-	število napačno klasificiranih pozitivnih primerov

Tabela 2.1: Vsi možni izidi pri napovedovanju binarnega razreda.

Razmerje med specifičnostjo in senzitivnostjo lahko analiziramo s pomočjo ROC krivulje. Primer takšne krivulje je prikazan na sliki 2.1. Na vodoravni osi prikažemo relativno število napačno klasificiranih negativnih primerov (FP). Na navpični osi pa prikažemo relativno število pravilno klasificiranih pozitivnih primerov (TP). Če narišemo krivuljo za dva algoritma strojnega učenja, ki ju želimo primerjati, nam o kvaliteti posameznega klasifikatorja govori ploščina pod ROC krivuljo. Boljši je tisti klasifikator, ki ima večji AUC. Na primeru iz slike 2.1 je torej SVM boljši od naivnega Bayesa.



Slika 2.1: Primer ROC krivulje.

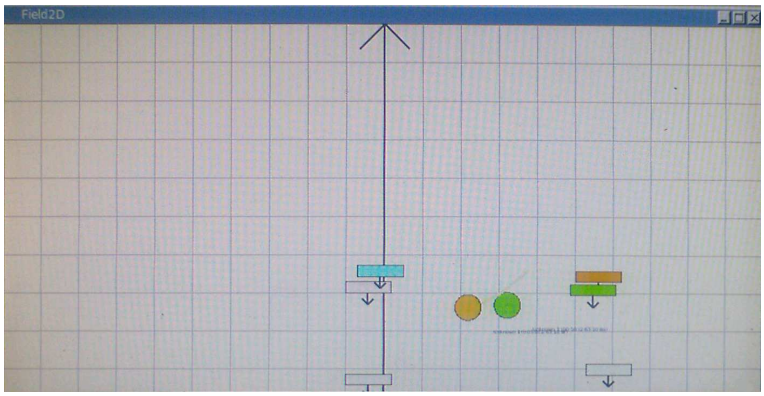
Poglavje 3

Zaznavanje gibanja in iskanje atributov

3.1 Naprava za zaznavanje gibanja

Slika 3.1: Pospeškomer.

Na sliki 3.1 vidimo napravo s katero merimo pospeške. Gre za pospeškomer, ki ga je podjetje Fraunhofer izdelalo prav za uporabo na projektu Confidence. Prav tako je izdelalo tudi namenski strežnik za komunikacijo s pospeškomeri. Vsaka naprava mora imeti svoj unikaten MAC naslov, da ga strežnik uspešno prepozna. Programska oprema le-tega je zaščitena in se je ne da spreminjati, tako so nam dostopni samo podatki, ki nam jih posreduje strežnik. Grafični vmesnik programske opreme je predstavljen na sliki 3.2. Z oranžnim in zelenim krogom sta označeni trenutni poziciji dveh pospeškomerov. Spreminjanje pospeškov lahko tako opazujemo v realnem času.



Slika 3.2: Prikaz grafičnega vmesnika programske opreme.

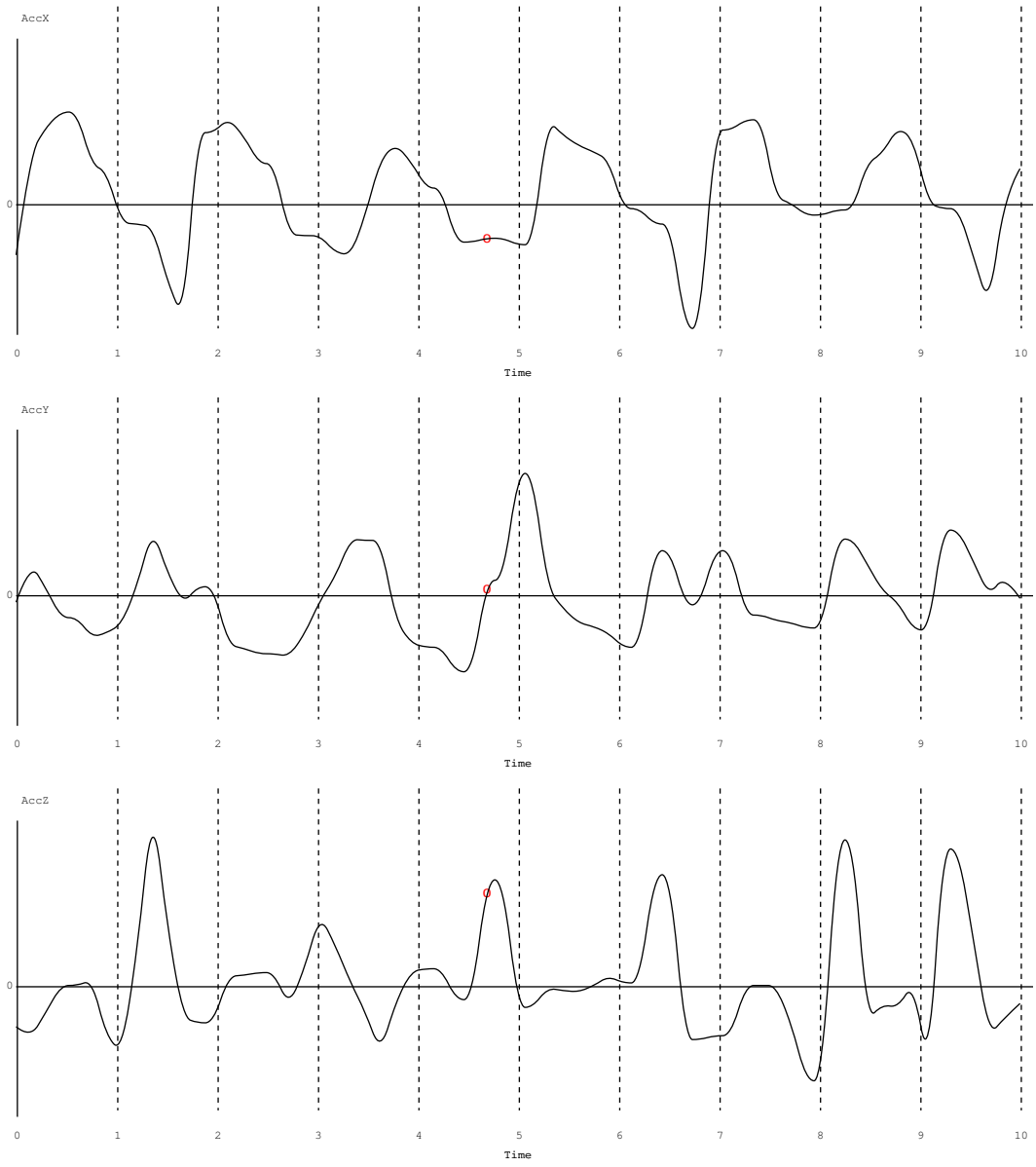
3.2 Način zajemanja in opis podatkov

Pospeškomeri komunicirajo s strežnikom, ki pošilja podatke klientom. Posamezen paket, ki je prikazan v tabeli 3.1, je dolg 99 bajtov.

Ime polja	**Dolžina**	**Opis**
Dolžina naslova naprave	1 bajt	Dolžina MAC naslova.
Naslov naprave	6 bajtov	MAC naslov pošiljatelja.
Časovni žig	4 bajte	Čas predstavljen v urah, minutah, sekundah in milisekundah.
Dolžina podatkov	2 bajta	Dolžina prihajajočih podatkov.
Povprečni pospešek	3×4 bajte	Struktura: $[[a_x], [a_y], [a_z]]$ Enota: $\frac{mm}{s^2}$ Tip: 32 bitna predznačena cela števila
Ostalo	74 bajtov	Ostane še 8 polj, ki vsebujejo različne podatke. Za nas niso pomembna, saj nas zanimajo samo pospeški.

Tabela 3.1: Prikaz enega podatkovnega paketa.

Od strežnika za vsako meritev dobimo dva podatkovna paketa. V vsakem od njiju so shranjeni podatki za eno nogo. Iz enega paketa torej izluščimo vektor s štirimi različni podatki - čas in pospeške dane noge (a_x, a_y, a_z). Na sliki 3.3 vidimo meritve pospeškov v vseh treh koordinatnih smereh. Slika



Slika 3.3: Meritve pospeškov leve noge v vseh treh koordinatnih smereh v desetsekundnem časovnem intervalu.

prikazuje desetsekundni časovni interval leve noge, v katerem je bilo opravljenih 67 posamičnih meritev pospeškov. Koordinati pospeškov v x in y smeri sta vzporedni z zemeljsko površino, pospeški v z smeri pa so pravokotni na zemeljsko površino.

Če združimo oba dobljena vektorja, lahko sestavimo matriko meritev velikosti $n \times 4$:

$$Data = (t_i, L_i, R_i, C_i) \qquad i = \{1, \ldots, n\} \tag{3.1}$$

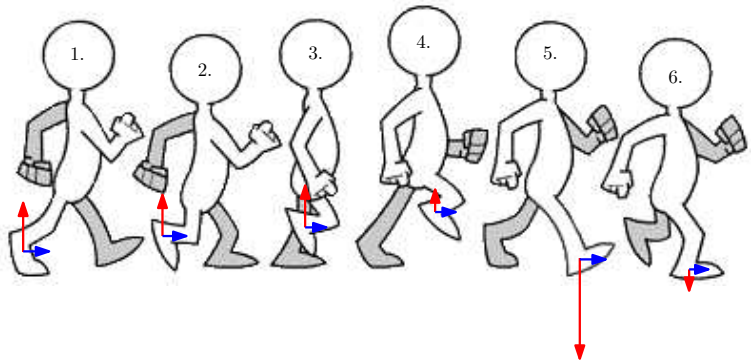
Število vseh meritev je n. Vsaka vrstica v enačbi 3.1 predstavlja eno meritev, katero zapišemo kot vektor podatkov (t, L, R, C). Čas meritve označimo s t in je podan v milisekundah. L in R sta vektorja dolžine 3, v katerih so shranjeni pospeški leve in desne noge v vseh treh koordinatnih smereh. S $C = \{0, 1\}$ smo označili razred, v katerega spada meritev, torej podatek o tem ali človek šepa ($C = 0$) ali hodi normalno ($C = 1$). Seveda potrebujemo ta podatek samo v učni množici. Primer matrike meritev:

$$Data = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ 3083 & (-71, 228, 32) & (-107, -2, 60) & 1 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{3.2}$$

Podatek iz matrike 3.2 je posebej označen tudi na sliki 3.3.

3.3 Definicija učnega primera

Kot je opisano že v uvodu, je naš cilj iz podatkov o pospeških leve in desne noge poiskati atribute, s katerimi bomo čim bolje klasificirali normalno hojo ali šepanje. Tukaj se pojavi vprašanje: Ali naj bo en primer učne množice sestavljen iz ene same meritve, torej vektorja (t, L, R, C), ali naj bo en primer neka podmnožica matrike iz enačbe 3.2 dolžine k. Lakany[3] je pokazal, da se ljudje med seboj razlikujemo po načinu hoje. Ker bi radi s pomočjo učne množice napovedovali hojo tudi za osebe, katerih podatki niso vključeni v učno množico, nam ena sama meritev ne bo dovolj za opis učnega primera. Tukaj se pojavi potreba po izbiri atributov, ki bodo opisovali splošne karakteristike hoje. V ta namen razdelimo meritve v več enakih intervalov. Vsak od teh intervalov bo sedaj osnova za naš učni primer. Ker bi dobre atribute težko izračunali iz začetnih podatkov, smo se odločili razdeliti intervale na primitivne elemente hoje - korake. To pomeni, da bomo v vsakem intervalu iskali podmnožice velikosti $\ell_1, \ell_2, \ldots, \ell_m$. Vsaka podmnožica bo vsebovala meritve enega koraka. m torej predstavlja število korakov v časovnem intervalu.



Slika 3.4: Prikaz enega koraka desne noge in pripadajočih pospeškov. Rdeča barva predstavlja pospeške v smeri z osi, z modro barvo pa so predstavljeni pospeški v smeri x osi.

Definirajmo sedaj en učni (in seveda testni) primer kot fiksno dolg časovni interval, ki vsebuje vsaj en korak leve in en korak desne noge. Korak bomo definirali kot čas od dviga stopala iz tal do ponovnega dotika celotnega stopala s tlemi. To prikazuje tudi slika 3.4, kjer sta označena še vektorja pospeškov v smereh x in z. V izbranem časovnem intervalu bi radi dobili vektor vseh korakov:

$$(t_i, L_i, R_i, C_i) \Rightarrow (S_1, S_2, \ldots, S_m) = S \tag{3.3}$$

kjer je $S_i \in S$ množica vseh meritev, ki opisujejo 1 korak.

$$S_i = (t_j, L_j, R_j, C_j) \qquad j = \{\sum_{k=1}^{i} \ell_k + 1, \ldots, \sum_{k=1}^{i+1} \ell_k\}$$

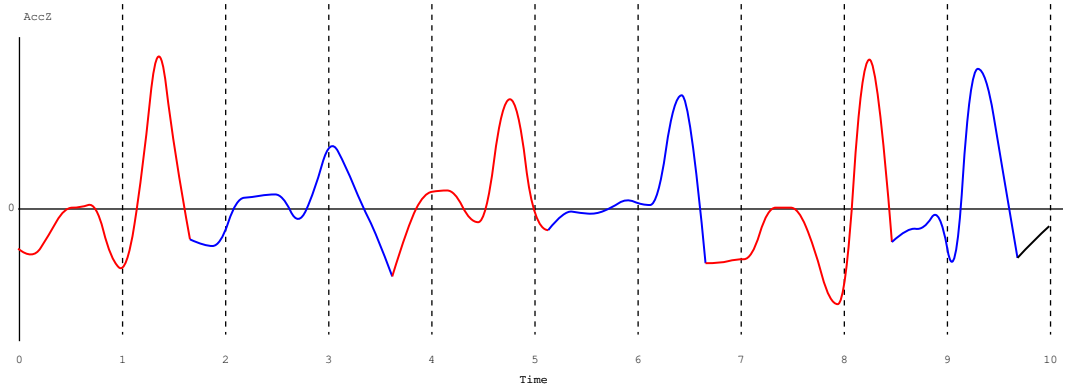
Potrebujemo torej algoritem, ki bo našo množico podatkov razdelil na posamezne korake.

3.4 Določanje korakov

Iz slike 3.3 lahko sklepamo, da sta x in y koordinati pospeškov pri hoji več ali manj naključni (odvisni od smeri hoje). Preostanejo nam torej samo še pospeški v smeri osi z.

3.4.1 Algoritem

Če si ogledamo sliko 3.4 vidimo, da so pospeški pri prvih štirih slikah koraka bolj ali manj enakovredni in obrnjeni v nasprotni smeri zemljine gravitacije.



Slika 3.5: Prikaz posameznih korakov izmenično obarvanih z modro in rdečo barvo.

V peti sliki se noga začne spuščati. Absolutna vrednost pospeška je sedaj večja kot v predhodnih primerih in se obrne navzdol. Prav to karakteristiko bomo uporabili za ločevanje korakov. Na sliki 3.5 so označeni koraki na istem intervalu kot na sliki 3.3 v smeri koordinatne osi z. Razrez intervala na korake opravimo z algoritmom 1.

Algoritem 1 išče korake na časovnem intervalu $[t_1, t_2]$. Poleg začetne in končne vrednosti intervala na vhodu podamo še matriko z vsemi podatki in spremenljivko $foot = \{"left", "right"\}$, s katero določimo algoritmu podatke katere noge naj upošteva. Na začetku s pomočjo intervala $[2t_1 - t_2, 2t_2 - t_1]$ poiščemo maksimalno in minimalno vrednost, kateri nam bosta služili za določanje praga med koraki. Pri testiranju se je izkazalo, da je najboljša enačba za prag:

$$threshold = (max - min) * 0.4$$

V nadaljevanju algoritem išče vse podmnožice podatkov, ki vsebujejo zaporedne padajoče podatke. Če je razlika med začetkom in koncem intervala večja od praga smo uspešno našli konec enega koraka. Tako algoritem najde vse korake in jih vrne v obliki podani z enačbo 3.3.

Algoritem 1 pri svojem delu uporablja še dve dodatni funkciji. Funkcija $getMinMax$ poskrbi za določanje minimalne in maksimalne vrednosti podatkov na določenem intervalu in podani nogi. Funkcija $cutData$ nam, če je podana tudi ena od vrednosti $\{"left", "right"\}$, vrne matriko, ki vsebuje podatek o času in pospešku v z smeri podane noge. Vsi podatki se nahajajo samo v podanem časovnem intervalu. Če podatek o izbiri noge ni podan, nam funkcija vrne vse podatke med podanim časovnim intervalom.

Algoritem 1: Razdeljevanje podatkov na korake.

Vhod: data[1..n][1..4], int t_1, int t_2, string foot
Izhod: steps[[step1],[step2],..]

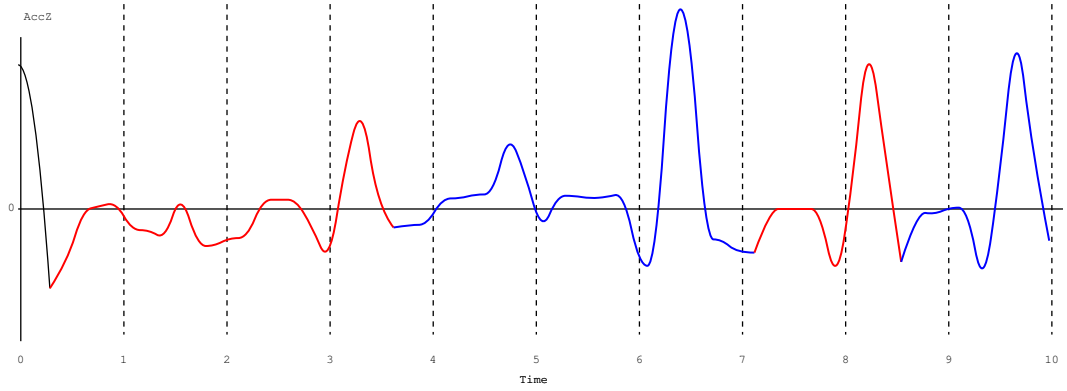
```
1  (min, max) ← getMinMax(data, 2t1 − t2, 2t2 − t1, foot);
2  timeFrame ← cutData(data, t1, t2, foot);
3  threshold ← (max - min)∗0.45;
4  steps ← [];
5  start ← timeFrame[0][0];
6  for i ← 1 to len(timeFrame) do
7  |  j ← i ;
8  |  while j <len(timeFrame) and
   |  timeFrame[j][1] ≥timeFrame[j + 1][1] do
9  |  |  j ← j + 1;
10 |  end
11 |  if timeFrame[i][1] - timeFrame[j][1] >threshold then
12 |  |  i ← j ;
13 |  |  end ← timeFrame[j][0];
14 |  |  steps.append(cutData(data, start, end));
15 |  |  start ← end;
16 |  end
17 end
18 return steps
```

3.4.2 Predolgi koraki

Algoritem 1 torej išče samo konce korakov. Nobene informacije pa ne poda o tem, ali je v končni točki enega koraka tudi začetek drugega. Vemo, da človeška hoja ni vedno sestavljena samo iz korakov, ampak lahko oseba tudi stoji, ali dela kaj drugega. Torej bi bilo smiselno preveriti, če so koraki $S_i \in S$ pravilni. V resnici bomo preverili, ali S_i vsebuje odvečne informacije.

Na sliki 3.6 vidimo graf pospeškov v smeri z osi na desetsekundnem časovnem intervalu. Koraki na intervalu so obarvani s pomočjo algoritma 1. Vidimo, da sta prva dva koraka izrazito daljša kot naslednja dva, saj vsebujeta 22 in 21 meritev. Prvi korak bi se v resnici moral začeti pri približno dveh sekundah, drugi pa pri približno petih sekundah. Če si podrobneje ogledamo sliki 3.5 in 3.6 opazimo, da se vsak korak začne v nekem lokalnem minimumu kar je tudi logično, saj je to posledica algoritma 1.

Da bi določili predolge korake, moramo izbrati mejo, ki bo določala mak-



Slika 3.6: Prikaz posameznih korakov izmenično obarvanih z modro in rdečo barvo.

simalni čas enega koraka. Meja je določena z naslednjo enačbo:

$$threshold2 = int(A + 6),$$

kjer A predstavlja povprečno število primerov za vse korake, ki so že bili izračunani. Na začetku, ko še ni nobenih primerov, je $A = 11$. Predolgi koraki so torej vsi, ki vsebujejo več kot 17 meritev. Da bi izrezali začetke teh korakov, moramo poiskati lokalni minimum v okolici meritve A. Uporabili smo uteževanje primerov in nato poiskali minimum na intervalu. Uteži naraščajo linearno z oddaljevanjem od meritve A. Če meritev iz S_i definiramo kot

$$s_j \in S_i \qquad j = 1 \ldots \ell_i$$

bi bila naša osrednja meritev, torej tista ki je na sredini okolice, enaka s_A. Razdalja med s_A in poljubno meritvijo je torej

$$d(s_A, s_j) = abs(A - j)$$

in utež za meritev s_j

$$w_j = 1 + d(s_A, s_j) \cdot 0.05$$

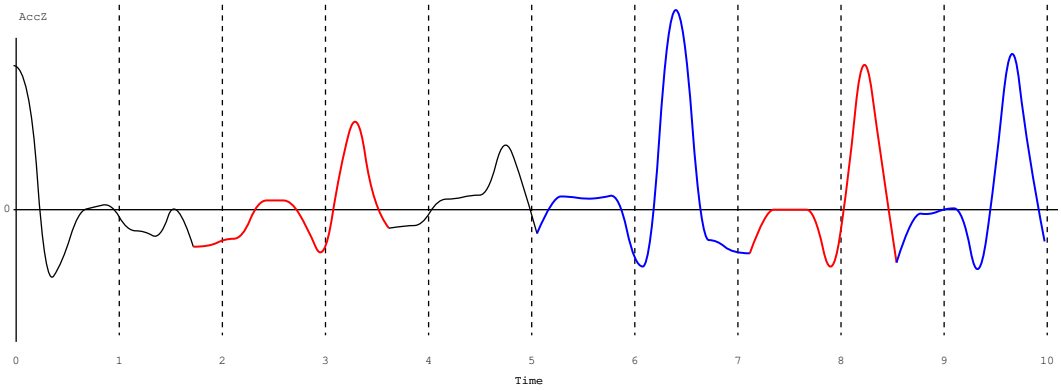
Algoritmu 2 kot vhodne podatke podamo tabelo vseh meritev enega koraka S_i in trenutno povprečje A. Za vsako meritev v S_i izračuna utež in določi novo vrednost. Na koncu uporabimo še funkcijo $minI$, ki nam vrne indeks minimalnega elementa v seznamu. Začetni seznam elementov S_i lahko sedaj skrajšamo za prvih $minI$ elementov.

Algoritem 2 bi lahko vključili v algoritem 1 med vrstici 11 in 12. S tem bi hitreje preverili, ali je število meritev preveliko. Če sedaj poženemo oba algoritma hkrati, dobimo razdelitev korakov prikazano na sliki 3.7.

Algoritem 2: Obravnavanje predolgih korakov.

Vhod: $S_i[1..\ell_i]$, int A
Izhod: subset of S_i

```
1 tmpS_i ← S_i;
2 for j ← 1 to ℓ_i do
3 |   w ← 1 + (A − j) ∗ 0.05 ;
4 |   tmpS_i[j] ← tmpS_i[j] * w
5 end
6 minIndex ← minI(tmpS_i) ;
7 return newS_i[minIndex:]
```



Slika 3.7: Prikaz posameznih korakov izmenično obarvanih z modro in rdečo barvo po popravljanju korakov.

3.5 Računanje atributov

Algoritem 1 nam torej vrne vektor $S = (S_1, S_2, \ldots, S_m)$, v katerem so podani podatki o vseh korakih leve ali desne noge v določenem časovnem intervalu. V resnici moramo za vsak izbrani interval algoritem pognati dvakrat, da dobimo podatke o obeh nogah. Za izračun atributov lahko nato uporabimo enak algoritem na obeh dobljenih množicah.

Kateri bi bili smiselni atributi za opis množice S? Odločili smo se za 8 atributov, s katerimi smo poskušali čimbolj opisati celotno množico S. Spodaj so našteti in opisani atributi leve noge. Atributi desne noge se računajo identično. Razlika je samo v uporabljeni množici S in zadnji črki imena, kjer črko L zamenjamo s črko R.

st_korakov_L - Podatek o številu korakov v izbranem časovnem intervalu je enak moči množice $|S| = m$.

cas_L - Povprečen čas vseh korakov dobimo, če izračunamo vsoto razlik med maksimalnim in minimalnim časom vsakega koraka. Rezultat nato delimo s številom korakov:

$$t = \frac{1}{m} \cdot \sum_{s \in S_i} (max(s(t)) - min(s(t))) \tag{3.4}$$

pospesek_L - Povprečni pospešek vseh korakov. Izračunati je potrebno dolžino vektorja za vsak primer posebej in nato sešteti vse dobljene dolžine. Tako dobimo pospešek vseh korakov skupaj, ki ga nato še povprečimo glede na število korakov v S.

$$a = \frac{1}{m} \cdot \sum_{s \in S_i} \sum_{j=1}^{\ell_i} \sqrt{s_j(x)^2 + s_j(y)^2 + s_j(z)^2} \tag{3.5}$$

hitrost_L - Povprečna hitrost vseh korakov. Izračunamo jo iz povprečnega pospeška, kot v enačbi 3.5, vendar izpustimo podatek o pospešku v smeri z osi.

$$a_{ground} = \frac{1}{m} \cdot \sum_{s \in S_i} \sum_{j=1}^{\ell_i} \sqrt{s_j(x)^2 + s_j(y)^2} \tag{3.6}$$

Uporabimo še čas iz enačbe 3.4 in dobimo povprečno hitrost:

$$v = a_{ground} \cdot t \tag{3.7}$$

razdalja_L - Na podoben način, kot smo izračunali povprečno hitrost, lahko izračunamo tudi povprečno razdaljo korakov. Uporabimo enačbi 3.4 in 3.6 in dobimo:

$$s = \frac{1}{2} \cdot a_{ground} \cdot t^2 \tag{3.8}$$

minAccZ_L - Izračunamo minimalno vrednost pospeška v smeri osi z:

$$a_{min} = min\{s_j(z); s \in S_i \wedge j \in \{1, \ldots, \ell_i\}\}$$

Kot posledica definicije algoritma 1 je to v resnici tudi globalni minimum na izbranem časovnem intervalu.

Atributi	Slika 3.3	Slika 3.7
st_korakov_L	6	4
cas_L	973	1572
pospesek_L	375,43	408
hitrost_L	16,62	31,85
razdalja_L	19,42	40,1
minAccZ_L	-185	-162
maxAccZ_L	407	432
visina_L	3,27	7,1

Tabela 3.2: Atributi izračunani na primerih, ki so predstavljenih na slikah 3.3 in 3.7.

maxAccZ_L - Izračunamo maksimalno vrednost pospeška v smeri osi z:

$$a_{max} = max\{s_j(z); s \in S_i \wedge j \in \{1, \ldots, \ell_i\}\}$$

Podobno kot pri atributu *maxAccZ_L* je to globalni maksimum. Oba atributa sta neodvisna od našega algoritma 1, zato bomo v poglavju 4.2 predstavili tudi rezultate učenja brez uporabe teh atributov.

visina_L - Povprečno višino koraka smo izračunali s seštevanjem absolutnih vrednosti pospeškov vseh korakov. Cilj je izračunati pot, ki jo noga opravi od najnižje do najvišje točke in nazaj ter rezultat razpoloviti.

$$a_z = \frac{1}{m} \cdot \frac{1}{2} \cdot \sum_{s \in S_i} \sum_{j=1}^{\ell_i} abs(s_j(z))$$

$$h = \frac{1}{2} \cdot a_z \cdot t^2 \tag{3.9}$$

Zapis pospeška $s_j(x)$ v smeri osi x in v ostalih koordinatnih smereh ni natančen in je uporabljen samo zaradi boljše preglednosti. Pravilen zapis pospeškov v smeri osi x bi bil $s[j][1][0]$, vendar bi to poslabšalo razumevanje formule. Prav tako je poenostavljen tudi zapis za računanje časa v enačbi 3.4.

V tabeli 3.2 so vrednosti vseh osmih atributov leve noge na primerih predstavljenih na slikah 3.3 in 3.7.

Poglavje 4

Testiranje

Cilj testiranja je poiskati optimalen učni algoritem. S pomočjo vnaprej določene učne množice želimo za poljubnega človeka napovedati ali šepa ali hodi normalno. V ta namen smo našo začetno matriko meritev razdelili na podmnožice. Uporabili smo štiri različne časovne intervale: 5 sekund, 10 sekund, 30 sekund in eno minuto. Na vsakem intervalu bomo ugotavljali uspešnost vseh algoritmov strojnega učenja naštetih v drugem poglavju.

4.1 Opis učne in testne množice

Domeno sestavlja testiranje sedmih oseb. Podatki prvih štirih oseb so bili uporabljeni za testne množice, ostalih treh oseb pa za učno množico.

	Skupni čas hoje	Čas normalne hoje	$\frac{\text{Čas normalne hoje}}{\text{Čas šepanja}}$
Oseba 1	7 min 53 s	3 min 47 s	0,93
Oseba 2	10 min 2 s	2 min 7 s	0,27
Oseba 3	11 min 48 s	5 min 32 s	0,88
Oseba 4	11 min 9 s	5 min 48 s	1,08
Oseba 5	46 min 1 s	28 min 57 s	1,70
Oseba 6	17 min 22 s	13 min 7 s	3,09
Oseba 7	40 min 33 s	26 min 43 s	1,93

Tabela 4.1: Časi hoje različnih testnih oseb.

V tabeli 4.1 so našteti časi testiranj posameznih oseb. Vidimo, da so osebe,

ki so vključene v učno množico, skupaj hodile 1 uro 43 minut in 56 sekund. Osebe testne množice pa 40 minut in 52 sekund.

V nadaljevanju bomo primere zadnjih treh oseb združili v eno učno množico. Testiranje točnosti bomo opravili na vsakem testnem primeru posebej. Med osebami, ki jih testiramo lahko poudarimo eno veliko razliko; način šepanja oseb. Osebi 1 in 2 sta šepali zelo očitno, ker sta vlekli eno nogo po tleh. Za razliko od njiju, sta osebi 3 in 4 šepali dosti bolj naravno. Lahko bi celo trdili, da je bila pri njiju razlika med hojo in šepanjem zelo zabrisana.

	Časovni intervali			
	5 sekund	10 sekund	30 sekund	1 minuta
Oseba 1	91	48	18	8
Oseba 2	118	60	20	10
Oseba 3	138	71	24	11
Oseba 4	127	65	23	12
Učna množica	1221	623	209	103

Tabela 4.2: Število primerov v učni in testni množici pri različnih časovnih intervalih.

V tabeli 4.2 je za vsak časovni interval podano število primerov v učni množici in vseh štirih testnih množicah. Vidimo, da število primerov pada obratnosorazmerno z dolžino intervala.

4.2 Poskusi

Opravili smo več različnih poskusov. V enem poskusu smo opravili vsa testiranja na določenem časovnem intervalu. Skozi celotno serijo poskusov smo merili klasifikacijsko točnost in AUC. Prva dva poskusa smo razdelili na dve fazi. V prvi fazi smo uporabili celotno množico atributov, v drugi fazi pa smo uporabili samo atribute, ki jih izračunamo s pomočjo algoritma 1. Izkazalo se je, da je originalno definirana množica atributov zelo uspešna pri napovedovanju novih primerov. Zelo velik informacijski prispevek imajo atributi, ki določajo maksimalno in minimalno vrednost pospeška v smeri koordinatne osi z na celotnem intervalu. Z reduciranjem atributov želimo preveriti tudi uspešnost našega algoritma za določanje korakov. Tako nam ostane nam še dvanajst atributov, šest za vsako nogo. Če bomo v drugi fazi dobili primerljivo

klasifikacijsko točnost in AUC s prvo fazo, bomo s tem potrdili tudi uspešnost algoritma 1 in pravilno izbiro atributov.

Pri vseh algoritmih strojnega učenja so bile uporabljene privzete nastavitve le-teh. To pomeni, da je naivni Bayes ocenjeval apriorne verjetnosti razredov z relativno frekvenco, verjetnosti razredov pri vrednostih atributov pa z metodo LOESS (ker so atributi zvezni). Klasifikacijsko drevo je za izbiro optimalnega atributa uporabilo informacijski prispevek. Algoritem uporablja rezanje naprej pri listih, ki vsebuje dva ali manj primerov, in rezanje nazaj z m-oceno m=2. Algoritem naključnih gozdov ustvari 15 dreves, podano pa ima tudi omejitev, da ne deli listov, ki imajo manj kot 5 primerov. kNN poišče pet najbližjih sosedov glede na evklidsko razdaljo med primeri. Uporablja tudi normaliziranje zveznih atributov. SVM za jedro uporablja radialne bazne funkcije s parametrom $\gamma = \frac{1}{n}$, kjer je n število učnih primerov, parameter ν pa je nastavljen na 0,5.

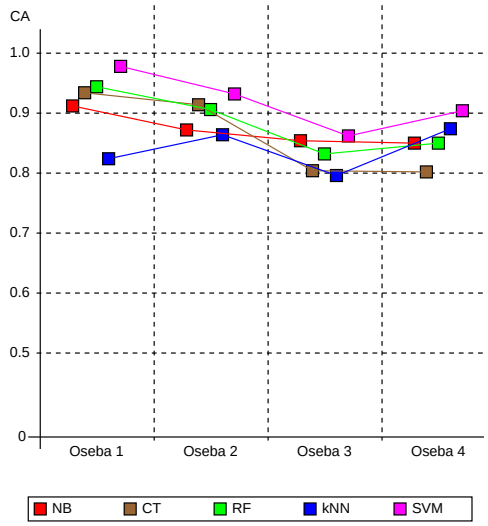
Najboljši algoritem strojnega učenja smo določili s pomočjo točkovanja klasifikatorjev na posamezni testni množici. Najboljši je prejel pet točk, najslabši pa eno. V primeru, da sta bila dva ali več klasifikatorjev izenačenih, so si točke ustrezno razdelili. Nato smo sešteli točke posameznih klasifikatorjev na različnih testnih množicah. Ker bi radi v oceni upoštevali tako klasifikacijsko točnost kot tudi AUC, smo za najboljši algoritem strojnega učenja izbrali tistega, ki je imel v skupnem seštevku največ točk.

4.2.1 Poskus na petsekundnem časovnem intervalu

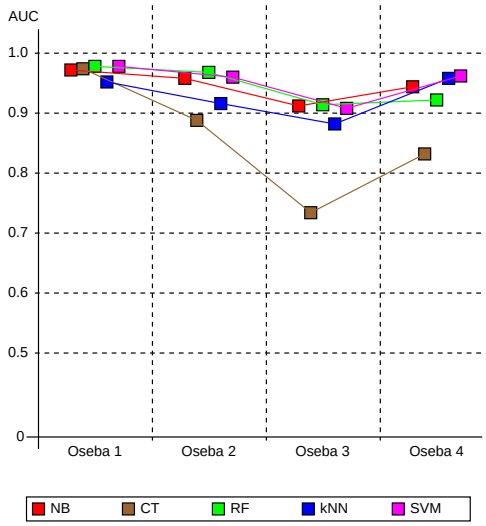
Sliki 4.1 in 4.2 prikazujeta klasifikacijsko točnost in AUC vseh klasifikatorjev na vseh primerih. Gre za prvo fazo poskusa, kjer smo uporabili celotno množico atributov.

Tako kot je opisano na začetku poglavja 4.2 iz množice atributov odstranimo maksimalne in minimalne vrednosti pospeškov. Dobimo seveda slabšo klasifikacijsko točnost in AUC. Toda naš cilj je izbrati optimalen algoritem strojnega učenja. Sliki 4.3 in 4.4 prikazujeta klasifikacijsko točnost testnih primerov na novi množici atributov. Tako kot v prvi fazi poskusa, sta tudi sedaj SVM in algoritem naključnih dreves najboljša pri klasifikaciji.

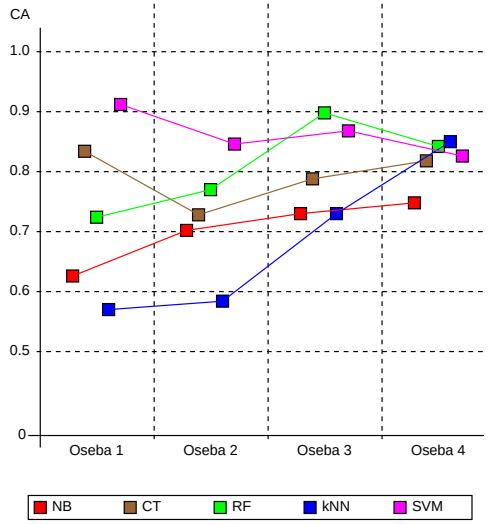
V tabeli 4.3 vidimo točkovanje posameznih algoritmov za drugo fazo poskusa. Potrdimo našo domnevo, da je v tem primeru najboljši algoritem res SVM.



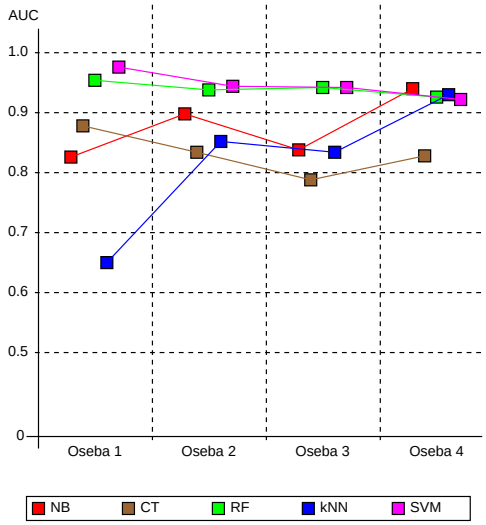
Slika 4.1: Klasifikacijska točnost.



Slika 4.2: Ploščina pod ROC krivuljo.



Slika 4.3: Klasifikacijska točnost.



Slika 4.4: Ploščina pod ROC krivuljo.

4.2.2 Poskus na desetsekundnem časovnem intervalu

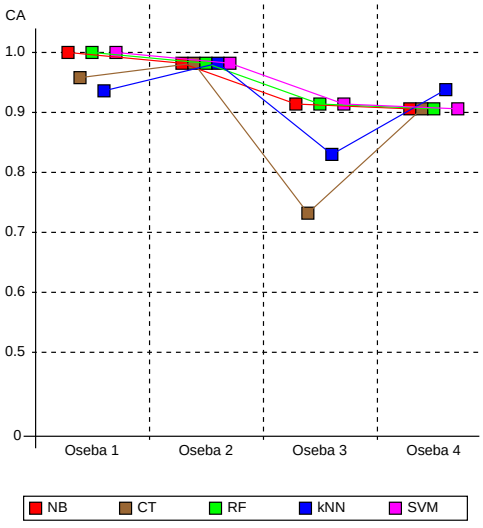
Na slikah 4.5 in 4.6 lahko res opazimo dejstvo opisano na začetku poglavja. Osebi 1 in 2 sta res šepali tako očitno, da lahko skoraj vedno napovemo pravilno vrednost. Kaj pa se zgodi, če sedaj odvzamemo 4 najboljše atribute iz množice in ponovimo testiranje? Na sliki 4.7 vidimo, da se napovedi večine algoritmov

Klasifikator	CA	AUC	Skupaj
NB	6.5	13	19.5
CT	12	6	18
RF	16	15.5	31.5
kNN	8.5	9	17.5
SVM	**17**	**16.5**	**33.5**

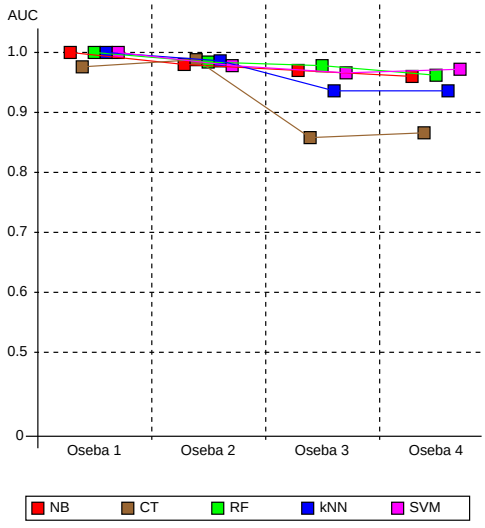
Tabela 4.3: Točkovanje klasifikatorjev na petsekundnem intervalu

.

poslabšajo, samo SVM je še vedno skoraj 100-odstoten pri svojih napovedih.



Slika 4.5: Klasifikacijska točnost.

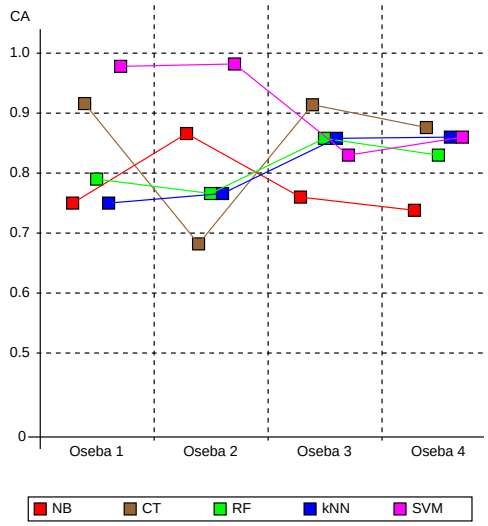


Slika 4.6: Ploščina pod ROC krivuljo.

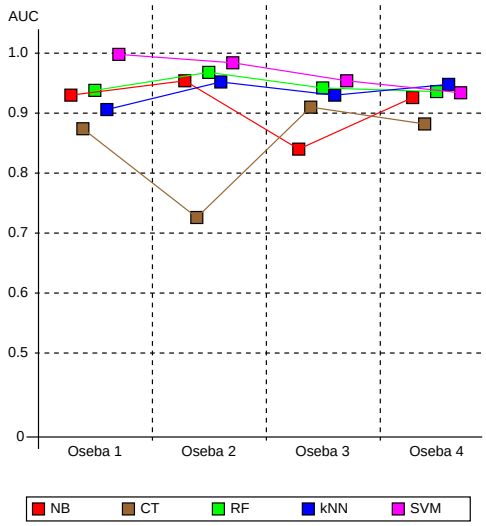
V tabeli 4.4 je navedeno točkovanje vseh uporabljenih algoritmov na desetsekundnem intervalu. Opazimo, da je najboljši algoritem zopet SVM.

4.2.3 Poskus na tridesetsekundnem časovnem intervalu

Pri prejšnjih dveh poskusih smo opazili, da sta minimalna in maksimalna vrednost intervala res dobra atributa, vendar pa iz druge faze vsakega poskusa vidimo, da smo lahko uspešni tudi brez njiju. S tem potrdimo tudi učinkovitost algoritma za odkrivanje korakov, ki nam pomaga pridobiti vse ostale atribute. V nadaljevanju bomo opustili prve faze poskusov, saj že iz poskusov pri pet-



Slika 4.7: Klasifikacijska točnost.



Slika 4.8: Ploščina pod ROC krivuljo.

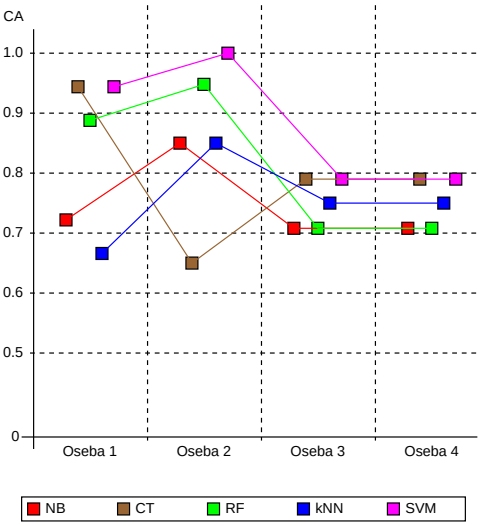
Klasifikator	CA	AUC	Skupaj
NB	7.5	9	16.5
CT	15	5	20
RF	11.0	16	27.0
kNN	11.0	12	23.0
SVM	**15.5**	**18**	**33.5**

Tabela 4.4: Točkovanje klasifikatorjev na desetsekundnem intervalu

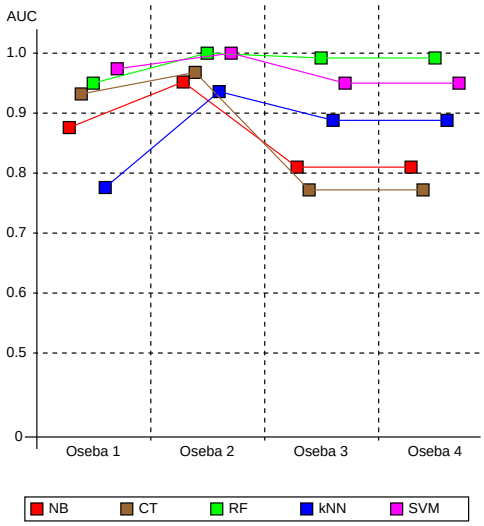
.

sekundnem in desetsekundnem intervalu vidimo, da se pri daljšem intervalu algoritmi strojnega učenja obnašajo boljše. To bomo poskušali potrditi tudi z meritvami pri naslednjih dveh intervalih. Na slikah 4.9 in 4.10 vidimo rezultate algoritmov strojnega učenja na primerih, ki zajemajo 30 sekund meritev.

Točkovanje klasifikatorjev je predstavljeno v tabeli 4.5. Tudi tukaj ni dvoma o optimalnem algoritmu. SVM je še vedno najboljši v množici petih metod strojnega učenja. Zanimivo dejstvo je, da so naključna drevesa že tretjič zapored zavzela drugo mesto pri ocenjevanju uspešnosti. Prvič se pojavi tudi primer, ko SVM ni najboljši pri obeh ocenah točnosti. Naključna drevesa imajo namreč boljši AUC, vendar imajo klasifikacijsko točnost slabšo na vseh štirih testnih primerih (slika 4.9).



Slika 4.9: Klasifikacijska točnost.



Slika 4.10: Ploščina pod ROC krivuljo.

Klasifikator	CA	AUC	Skupaj
NB	7.5	8	15.5
CT	14.5	8	22.5
RF	10.0	**18.5**	28.5
kNN	9.5	8	17.5
SVM	**18.5**	17.5	**36.0**

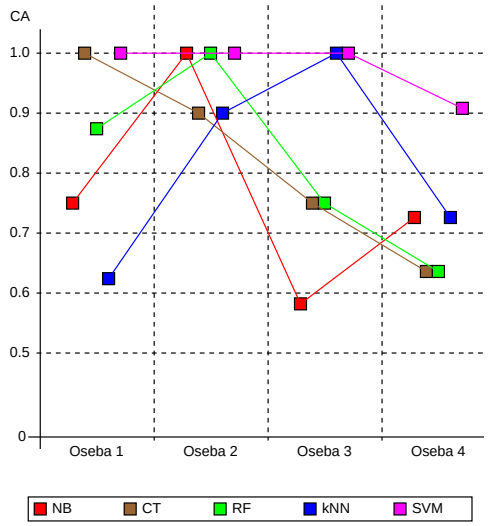
Tabela 4.5: Točkovanje klasifikatorjev na tridesetsekundnem intervalu

.

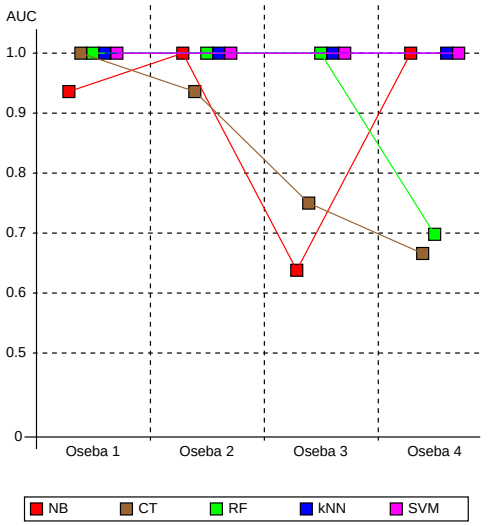
4.2.4 Poskus na šestdesetsekundnem časovnem intervalu

Šestdeset sekund je zgornja meja dolžine našega intervala. V primeru, da bi poskušali še z daljšimi časovnimi intervali, bi testna množica pri nekaterih primerih zajemala tudi samo en ali nič primerov. V resnici je že ta časovni interval na meji veljavnih poskusov, saj je naša testna množica zelo zreducirana. Ne glede na to, pa so rezultati zelo zanimivi. Na prvi pogled, bi mogoče pomislili, da sta rezultata prikazana na sliki 4.11 in 4.12 slabša od ostalih, vendar se izkaže ravno nasprotno. Ker nas v resnici zanima samo eden od klasifikatorjev, zopet vidimo, da iz množice izstopa SVM.

Iz tabele 4.6 razberemo, da je klasifikacijska točnost vseh algoritmov razen



Slika 4.11: Klasifikacijska točnost.



Slika 4.12: Ploščina pod ROC krivuljo.

Klasifikator	CA	AUC	Skupaj
NB	10.5	9.5	20.0
CT	10.0	7.5	17.5
RF	11.0	13.0	24.0
kNN	10.5	**15.0**	25.5
SVM	**18.0**	**15.0**	**33.0**

Tabela 4.6: Točkovanje klasifikatorjev na šestdesetsekundnem intervalu

.

algoritma SVM skoraj enaka. Zopet pa je točnost klasifikatorja SVM boljša od vseh ostalih. V primerjavi z ostalimi poskusi se izkaže, da so na šestdesetsekundnem intervalu njegove napovedi najboljše.

4.3 Rezultati testiranja

Zanima nas, kateri uporabljeni algoritem strojnega učenja je optimalen na naši domeni in kateri časovni interval si je dobro izbrati pri nadaljnjih testiranjih. Brez dvoma lahko rečemo, da je SVM optimalen, saj je bil najboljši pri čisto vsakem poskusu. V povprečju je bila njegova klasifikacijska točnost boljša za 12,4 %, njegova ploščina pod ROC krivuljo pa za 7,4 % od ostalih klasifika-

torjev.

Kako pa bi določili najboljši časovni interval? Odgovor ni tako enostaven kot v prejšnjem primeru, vendar pa ga vseeno lahko določimo na podlagi čisto osnovnih vrednosti. Za vsak posamezni interval lahko preštejemo kolikokrat je bil SVM najboljši (tako pri CA kot pri AUC). Samo v zadnjem - šestdesetsekundnem intervalu je bil vedno na prvem mestu. V tridesetsekundnem intervalu je bil najboljši šestkrat, v ostalih dveh pa samo petkrat. Da bi potrdili ta rezultat, lahko opravimo še en test. Sedaj nas pri vsakem intervalu zanima povprečna točnost klasifikatorja SVM. Tudi tukaj se za najboljši interval izkaže zadnji. Zaključimo lahko, da je za razrez naših začetnih meritev najboljše uporabiti interval dolžine šestdesetih sekund.

4.3.1 Predstavitev modelov strojnega učenja

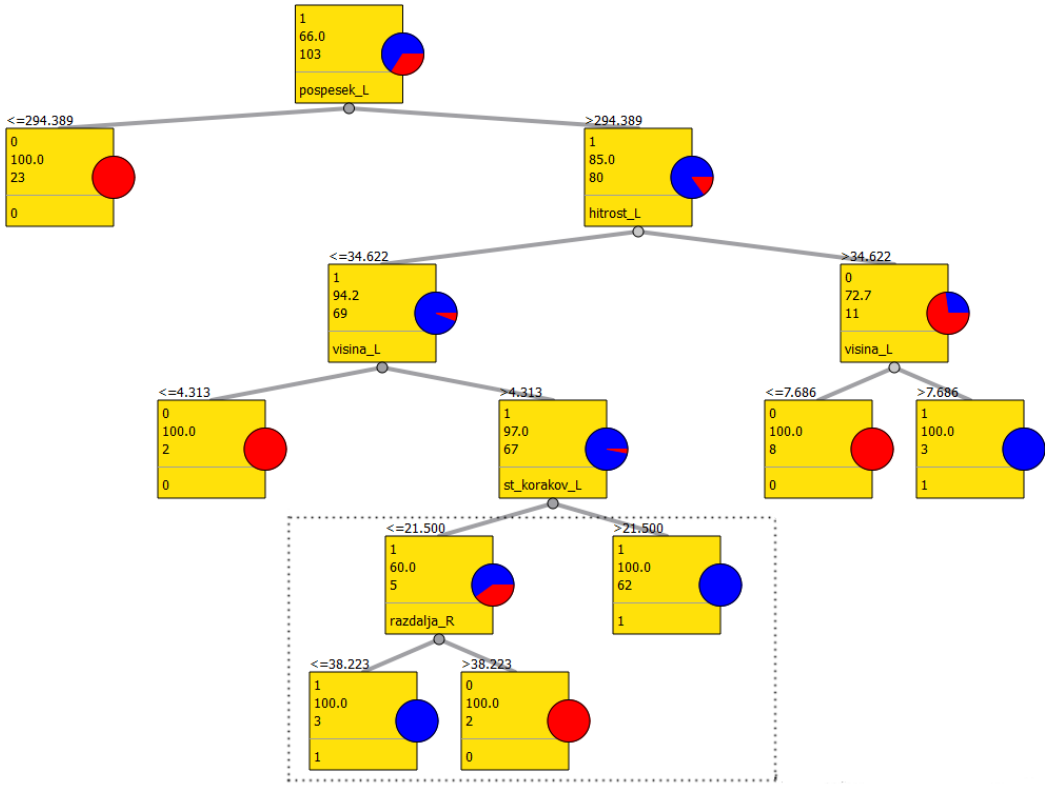
Določili smo najboljši časovni interval za našo nalogo. Kakšni so sedaj modeli, ki jih zgradijo različni algoritmi strojnega učenja? Ogledali si bomo dva različna modela strojnega učenja. Prvi je rezultat klasifikacijskega drevesa, drugi pa naivnega Bayesa.

Model klasifikacijskega drevesa je predstavljen z binarnim drevesom na sliki 4.13. V vsakem vozlišču so štirje podatki. V tabeli 4.7 so predstavljena posamezna polja na primeru korenskega vozlišča.

Vrednost	Opis
1	Večinski razred vozlišča.
66.0	Razmerje večinskega razreda z številom primerov.
103	Število primerov v vozlišču.
pospesek_L	Atribut glede na katerega se opravi naslednja delitev.

Tabela 4.7: Opis vozlišča klasifikacijskega drevesa.

Model je zgrajen iz petih različnih atributov. *pospesek_L* je najboljši atribut, saj nam razdeli večino pozitivnih in negativnih primerov. Če bi želeli še bolj poenostaviti drevo, bi uporabili rezanja naprej. To bi nehalo graditi model, ki bi imel v vozlišču več kot 96 % primerov iz večinskega razreda. Tako bi dobili binarno drevo višine štiri namesto sedanjega, katerega višina je šest. Na sliki 4.13 bi torej odrezali vsa vozlišča, ki so obkrožena s črtkano črto. Model z uporabo rezanja naprej je pri testiranju na učni množici z 10-kratnim prečnim preverjanjem dosegel boljšo klasifikacijsko točnost kot originalni model.

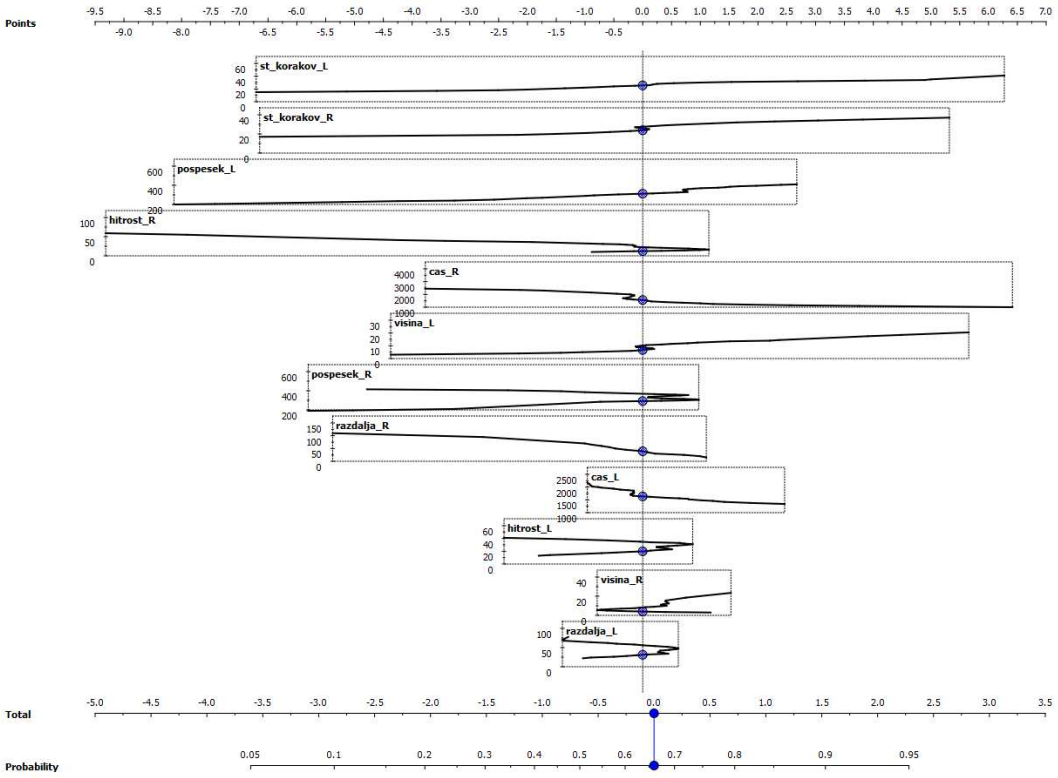


Slika 4.13: Model klasifikacijskega drevesa.

Za predstavitev modela naivnega Bayesovega klasifikatorja uporabimo nomograme [5]. Za izris moramo najprej izračunati logaritem relativnega tveganja (*log odds ratio*). Tako ocenimo kako posamezna vrednost vpliva na verjetnost ciljnega razreda. Iz dobljenih *logOR* vrednosti nato izračunamo verjetnost ciljnega razreda. Na sliki 4.14 je predstavljen nomogram, izračunan s pomočjo vrednosti iz naše učne množice. Atributi so urejeni po absolutni pomembnosti. Najboljši atribut je torej *st_korakov_L*, najslabši pa *razdalja_L*.

Vsi atributi so prikazani v dvo-dimenzionalni projekciji, tako da lahko opazujemo tudi njihov vpliv na izbrani razred, ki je v našem primeru 1 (normalna hoja). Za atributa, ki merita število korakov lahko razberemo, da bosta pozitivno vplivala na razred, če njihova vrednost večja kot 24. S pomočjo takšnega prikaza lahko izluščimo nekaj razlik med šepanjem in normalno hojo:

- Če šepamo, naredimo manj korakov v časovni enoti kot pri normalni hoji.
- Pri normalni hoji so večji pospeški in višina dviga noge, kot pri šepanju.



Slika 4.14: Model naivnega Bayesa prikazan z nomogrami.

- Pri šepanju za en korak porabimo več časa kot pri normalni hoji.

Te razlike bi verjetno vedeli našteti tudi brez uporabe nomogramov, zato smo lahko še toliko bolj prepričani v pravilnost tega modela.

4.4 Sistem za nadzor hoje

Z uporabo algoritma 1 in 2 ter zaključki iz poglavja 4.3, lahko sedaj sestavimo celoten sistem za nadzor hoje. Meritve izvajamo v realnem času. Vsakih šestdeset sekund napovemo vrednosti razreda in preverimo ali človek šepa ali ne. Pri tem uporabimo algoritem SVM. Psevdokoda sistema je podana v algoritmu 3.

Kot vidimo, je v algoritmu 3 uporabljenih kar nekaj še nedefiniranih funkcij. *SVM* ustvari klasifikacijski model iz učne množice. Ta nam za vsak testni primer vrne njegovo verjetnost razreda *normalna hoja (1)*. Funkcija *getTime*

Algoritem 3: Sistem za nadzorovanje sprememb pri hoji človeka.

Vhod: learningData, real-time data from accelerometers

```
1  while True do
2  |  SVMclassificator = SVM(learningData);
3  |  data ← [];
4  |  startTime = getTime();
5  |  while getTime() − startTime < 60 do
6  |  |  data.append(getSensorData());
7  |  end
8  |  stepsL = Algoritem1(data, 0, 60, "left");
9  |  stepsR = Algoritem1(data, 0, 60, "right");
10 |  testCase = getAtributes(stepsL, stepsR);
11 |  if SVMclassificator(testCase) > 0.5 then
12 |  |  print "normal walk";
13 |  end
14 |  else
15 |  |  print "limping";
16 |  end
17 end
```

nam vrne trenutni čas naprave v sekundah. *getSensorData* uporabimo za pridobitev trenutne meritve vseh pospeškov. *getAtributes* je funkcija, ki nam s pomočjo enačb opisanih v poglavju 3.5 izračuna vrednostih naših atributov za levo in desno nogo.

Algoritem 3 zelo poenostavljena verzija celotnega sistema. Če bi hoteli boljši nadzor nad točnostjo odločanja, bi morali sklepati tudi s pomočjo predhodnih odločitev klasifikatorja. To bi bilo potrebno v primeru napačnih napovedi, saj ne želimo takoj sprožiti alarma, v primeru šepanja. Če pa je takšna hoja konsistentna skozi nekaj zaporednih časovnih intervalov, lahko upravičeno menimo, da je osebi potrebna zdravniška pomoč.

Poglavje 5

Diskusija in zaključek

Namen diplomskega dela je bila izdelava sistema za nadzor sprememb v hoji človeka. Ugotovili smo, da je najboljši pristop k rešitvi problema delitev naše množice podatkov na fiksne časovne intervale. Brez te predpostavke bi težko določili pravilne atribute potrebne za strojno učenje. Da bi bolje razumeli podatke enega časovnega intervala, smo hojo razdelili na njene primitivne elemente - korake. S pomočjo splošne karakteristike hoje smo zasnovali dva algoritma, ki nam podatke razdelita na posamezne korake. Ob izbiri različnih atributov značilnih za hojo, smo nato poskusili ustvariti univerzalni model strojnega učenja, kateri bi prepoznal razlike med normalno hojo in šepanjem. Testiranje smo opravili na različnih časovnih intervalih. Izkazalo se je, da z rezanjem podatkov na daljše množice dobimo bolj zanesljive informacije o hoji človeka na obravnavanem intervalu. Ugotovili smo, da je SVM najboljši model strojnega učenja za predstavljen klasifikacijski problem.

Celoten sistem smo tako razdelili na dva ločena dela. Prvi del vsebuje zajem pospeškov v realnem času in posredovanje podatkov algoritmu za prepoznavo podatkov. Temu sledi še izračun atributov. Drugi del sistema skrbi za prepoznavanje hoje. Tukaj dobimo odgovor o tem, ali človek hodi normalno, ali šepa. Ker je lahko ocena pravilnosti tudi napačna, ne sprožimo alarma že ob prvi napovedi šepanja, pač pa se prepričamo še z nekaj dodatnimi meritvami.

Podoben sistem je že implementiran v projektu Confidence. Razlika je v uporabi strojne opreme. Pospeškomeri so zamenjani z napravami, ki delujejo na principu koordinat v prostoru. Tako lahko dosti lažje realizirano naše zadane cilje. Vendar pa obstajata dve veliki pomanjkljivosti v primerjavi s pospeškomeri. Prva je omejenost opreme samo na notranjo uporabo, druga pa je cena. Te naprave so dražje kot navadni pospeškomeri, katere lahko najdemo že v skoraj vsakem mobilnem telefonu. Tako postane smiseln razvoj opreme,

ki za svoje delovanje uporablja podatke o pospeških.

Idej za nadaljnje delo je veliko. Za izdelavo dobrega sistema za nadzor starostnikov, bi morali povečati število pospeškomerov nameščenih po telesu. Idealna mesta za njihovo postavitev bi bila oba gležnja, kolena, pas, prsa in tudi obe zapestji. Tako bi imeli veliko podatkov o trenutni aktivnosti določene osebe. Potrebovali bi tudi množico stanj, s katero bi lahko opisali večino človeških aktivnosti kot so plezanje, hoja, šepanje, sedenje, ležanje, tek, ... Razviti in prilagodi bi morali tudi obstoječe algoritme, ki bi sedaj morali nadzorovati tudi spremembe med aktivnostmi (hoja → sedenje). Glavni cilj nadaljnjega dela bi bil torej razvoj avtonomnega sistema, katerega bi lahko upravljal že kakšen bolj zmogljiv mobilni telefon, ki bi bil v primeru neobičajnih podatkov sposoben prepoznati nepravilnosti.

Slike

1.1 Prikaz sistema Confidence. 5

2.1 Primer ROC krivulje. 13

3.1 Pospeškomer. 15
3.2 Prikaz grafičnega vmesnika programske opreme. 16
3.3 Meritve pospeškov leve noge v vseh treh koordinatnih smereh v desetsekundnem časovnem intervalu. 17
3.4 korak 19
3.5 Prikaz posameznih korakov izmenično obarvanih z modro in rdečo barvo. 20
3.6 Prikaz posameznih korakov izmenično obarvanih z modro in rdečo barvo. 22
3.7 Prikaz posameznih korakov izmenično obarvanih z modro in rdečo barvo po popravljanju korakov. 23

4.1 Klasifikacijska točnost. (petsekundni interval, faza 1) 30
4.2 Ploščina pod ROC krivuljo. (petsekundni interval, faza 1) . . . 30
4.3 Klasifikacijska točnost. (petsekundni interval, faza 2) 30
4.4 Ploščina pod ROC krivuljo. (petsekundni interval, faza 2) . . . 30
4.5 Klasifikacijska točnost. (desetsekundni interval, faza 1) 31
4.6 Ploščina pod ROC krivuljo. (desetsekundni interval, faza 1) . . 31
4.7 Klasifikacijska točnost. (desetsekundni interval, faza 2) 32
4.8 Ploščina pod ROC krivuljo. (desetsekundni interval, faza 2) . . 32
4.9 Klasifikacijska točnost. (tridesetsekundni interval) 33
4.10 Ploščina pod ROC krivuljo. (tridesetsekundni interval) 33
4.11 Klasifikacijska točnost. (šestdesetsekundni interval) 34
4.12 Ploščina pod ROC krivuljo. (šestdesetsekundni interval) 34
4.13 Model klasifikacijskega drevesa. 36
4.14 Model naivnega Bayesa prikazan z nomogrami. 37

Tabele

2.1 Vsi možni izidi pri napovedovanju binarnega razreda. 12

3.1 Prikaz enega podatkovnega paketa. 16
3.2 Atributi izračunani na primerih, ki so predstavljenih na slikah 3.3 in 3.7. 25

4.1 Časi hoje različnih testnih oseb. 27
4.2 Število primerov v učni in testni množici pri različnih časovnih intervalih. 28
4.3 Točkovanje klasifikatorjev na petsekundnem intervalu 31
4.4 Točkovanje klasifikatorjev na desetsekundnem intervalu 32
4.5 Točkovanje klasifikatorjev na tridesetsekundnem intervalu 33
4.6 Točkovanje klasifikatorjev na šestdesetsekundnem intervalu . . . 34
4.7 Opis vozlišča klasifikacijskega drevesa. 35

Literatura

[1] J. Demšar, B. Zupan in G. Leban, "Orange: From experimental machine learning to interactive data mining," *White paper*, Dostopno na: www.ailab.si/orange, Fakulteta za računalništvo in informatiko, Ljubljana, 2004.

[2] I. Kononenko, "Strojno učenje," *Založba fakultete za elektrotehniko in fakultete za računalništvo in informatiko*, Ljubljana, 2005.

[3] H. Lakany, "Extracting a diagnostic gait signature," *Pattern Recognition*, št. 41, zv. 5, str. 1627-1637, 2008.

[4] A. Mannini, A.M. Sabatini, "Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers", *Sensors*, št. 10, zv. 2, str. 1154-1175, 2010.

[5] M. Možina, J. Demšar, M. Kattan, B. Zupan, "Nomograms for Visualizing of Naive Bayesian Classfier", *Knowledge Discovery in Databases: PKDD 2004*, str. 337-348, 2004.

[6] N. Ravi, N. Dandekar, P. Mysore, M.L. Littman, "Activity recognition from accelerometer data", *In Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence*, USA, str. 1541–1546, 2005.

[7] (2010) *Projekt Confidence*. Dostopno na: http://www.confidence-eu.org