UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Strle

Prepoznavanje človeških gibov s pospeškomeri in strojnim učenjem

DIPLOMSKO DELO UNIVERZITETNEGA ŠTUDIJA

Mentor: akad. prof. dr. Ivan Bratko

Ljubljana, 2008

Zahvala

Zahvaljujem se akad. prof. Ivanu Bratku, ki me je navdušil za področje Umetne Inteligence in mi je omogočil tako opravljanje diplomskega dela kot tudi delo na drugih zanimivih projektih.

Zahvaljujem se tudi vsem sodelavcem Laboratorija za Umetno Inteligenco, ki so mi s svojimi idejami in kritikami pomagali pri odkrivanju področja Umetne Inteligence in pri nastajanju diplomskega dela. Še posebno bi se rad zahvalil Juretu Žabkarju, ki je že v času študija vestno prenašal svoje znanje name, in Janezu Demšarju, ki mi je nemalokrat pomagal pri uporabi orodja Orange.

Kazalo

Seznam uporabljenih kratic

CA	klasifikacijska točnost
DT	odločitveno drevo
DTW	dynamic time warping
KNN	k-najbližjih sosedov
TSDT	odločitveno drevo časovnih vrst

Povzetek

Prepoznavanje človeških gibov je sposobnost računalnika oz. sistema, da opazuje in prepozna človekov gib. Problemov, ki jih sistemi za prepoznavanje gibov rešujejo, je veliko in se med seboj močno razlikujejo. Najdemo jih na področju varnosti, športa, vadbe, medicine, upravljanja robotov, računalniških vmesnikov, navidezne resničnosti, iger… Načinov, kako zajemati podatke o človeškem gibanju, je več. V zadnjem času smo v ta namen priča velikemu porastu uporabe pospeškomerov v napravah široke potrošnje. Razloge za to lahko pripišemo predvsem majhni velikosti, visoki natančnosti, nizki porabi energije ter nizki ceni teh naprav. Kljub velikemu porastu pospeškomerov v napravah široke potrošnje, pa so aplikacije prepoznave gibov še vedno precej preproste (obračanje oken, štetje korakov …) oz. zelo omejene na problem, ki ga rešujejo. Smiselno bi bilo razviti metodo, ki bi omogočala učenje prepoznavanja človeških gibov, saj bi s tem rešili splošen problem prepoznave človeških gibov. V ta namen sem preučil dve metodi strojnega učenja: k-najbližjih sosedov ter odločitveno drevo časovnih vrst. Zanimalo me je predvsem, kako dobro sta se metodi sposobni naučiti prepoznave človeških gibov. Metodi sta bili preverjeni na dveh domenah prepoznave človekovih gibov: pisanje velikih črk angleške abecede na ravni podlagi ter ročne signalizacije v zraku. Uporabljeni so bili tako podatki brez šuma, kot tudi podatki z dodanimi naključnimi atributi, rezultati pa so pokazali, da metodi uspešno rešujeta dana problema.

Ključne besede: prepoznavanje človeških gibov, strojno učenje, pospeškomer, k-najbližjih sosedov, odločitveno drevo časovnih vrst.

Abstract

Human gesture recognition is the ability of a machine to recognize human gestures. It is used in various fields such as security, sports, exercise, medicine, robotics, computer interfaces, virtual reality, games… There are many different ways of obtaining human motion data. Recently we have seen increased usage of accelerometers in mobile phones and consumer electronics for this purpose. Reasons for that can be found in advances in MEMS technology which resulted in accelerometers that are as small as 3mm x 5mm x 0.9mm, operate on less than 1 milliwatt of power, and cost less than one dollar. Despite the explosion of accelerometers usage in consumer electronics, most of the gesture recognition applications are still quite basic (window rotation) or very problem specific. This indicates that there is a lack of general approach to gesture recognition using accelerometers that would enable training of the gestures. For this purpose two machine learning methods have been examined in this project: k-nearest neighbors and decision tree induction from time series. Both methods were evaluated on two gesture recognition domains: handwritten letter recognition (letters were written on the table using two-axis accelerometer) and hand signals (signals were given by moving three-axis accelerometer in the air). The methods were evaluated on data without noise as well on data with added noise. Experimental results have shown that both methods perform very well on given domains of gesture recognition.

Keywords: human gesture recognition, machine learning, accelerometer, weighted k-nearest neighbors, decision tree induction from time series.

Uvod

Prepoznavanje človeških gibov je sposobnost računalnika oz. sistema, da opazuje in prepozna človekovo gibanje. Problemov, ki jih sistemi za prepoznavanje gibov rešujejo, je veliko in se med seboj močno razlikujejo. Najdemo jih na področju varnosti, športa, vadbe, medicine, upravljanja robotov, računalniških vmesnikov, navidezne resničnosti, iger… Zaradi raznolikosti samih problemov in okolja, v katerem se pojavljajo, so načini opazovanja (zajema podatkov) in prepoznavanja gibanja različni. V splošnem jih lahko ločimo na sisteme, ki gibanje opazujejo od zunaj (razne kamere) in sisteme, ki zaznavajo gibanje na sami opazovani osebi (razni senzorji – pospeškomeri, žiroskopi, kamere). V tem delu se bom posvetil predvsem prepoznavanju gibanja, zajetega s pospeškomeri, ki jih nosi oz. upravlja oseba, pri kateri želimo prepoznati gibanje.

Pospeškomer je naprava, ki meri pospeške (in vpliv gravitacije) v eni ali več smereh. Njegovo delovanje lahko ponazorimo z zelo majhno maso, ki je napeta med dvema vzmetema. Ko pospeškomer premikamo, se premika tudi masa glede na svojo okolico. Ti premiki se zaznajo in pretvorijo v ustrezne pospeške [1].

Rezultati razvoja zadnjega desetletja na področju senzorjev za zaznavanje gibanja in splošno sprejemanje ljudi takih naprav, so pripeljali do velikega porasta uporabe takih senzorjev [1]. Zelo majhna velikost (3mm x 5mm x 0.9mm), visoka natančnost, nizka poraba energije, nizka cena (< 1$) ter splošno sprejemanje ljudi je pripeljalo do eksplozije uporabe pospeškomerov v mobilnih telefonih, igralnih konzolah, mp3 predvajalnikih, dlančnikih, športni obutvi (Slika 1)…

Worldwide Accelerometer Forecast by Application Area

$M	2005	2006	2007	2008	2009	2010	2011	2012	2013	08-13 CAGR%
Automotive	831	908	917	1,103	1,212	1,332	1,589	1,708	1,866	11%
Computers	154	187	203	232	248	286	291	327	324	7%
Video Games	26	34	87	122	148	205	202	267	288	19%
MP3	0	45	114	159	234	348	522	589	666	33%
Other Consumer	97	90	98	87	110	149	133	162	184	16%
Mobile Phones	0	0	62	144	217	317	402	564	688	37%
Industrial	541	630	722	704	818	691	835	749	818	3%
Total	1,649	1,894	2,203	2,551	2,987	3,328	3,974	4,366	4,834	14%

databeans estimates

Slika 1 Pregled in napoved uporabe pospeškomerov po področjih; vir: EE Times India [2]

Kot najbolj prepoznavne izdelke (široke potrošnje), ki uporabljajo senzorje za zaznavanje gibanja lahko navedemo:

- Apple je v prenosnike vgradil tri-smerni pospeškomer [3] (osnovna funkcija je zaščita trdega diska pred padci) in razvil vrsto iger, ki se igrajo s premikanjem prenosnika ter aplikacijo za obračanje oken operacijskega sistema tako, da je prava stran vedno zgoraj.
- Mnogi proizvajalci mobilnih telefonov (Nokia, Samsung) so vgradili pospeškomere v mobilne telefone za obračanje oken, tako, da je prava stran vedno zgoraj, in za merjenje korakov, ki se uporablja v različnih aplikacijah vadbe.
- Pojavljajo se številne igre za mobilne telefone, ki se igrajo z nagibanjem telefona.
- Aplikacije obračanja oken in vadbe se pojavljajo v številnih mp3 predvajalnikih.
- Nike je v tekaške čevlje vgradil pospeškomere in skupaj z Applom razvil aplikacijo za vodenje vadbe [4].
- Sony in Nintendo (Slika 2) sta v svoji igralni konzoli PS3 in Wii vgradila senzorje gibanja in s tem povečala zadovoljstvo ob igranju iger [5].

Slika 2 Nintendo Wii Remote

Velik porast uporabe pospoškomerov in vse večja potreba po prepoznavanju gibanja kažeta, da bi bilo smiselno razviti metodo, ki bi omogočala učenje prepoznavanja človekovih gibov. Človek bi najprej nekajkrat ponovil gib in ga označil, računalnik/sistem pa bi se iz danih primerov giba naučil prepoznave. Naslednjič, ko bi človek naredil naučeni gib, bi ga računalnik/sistem prepoznal (določil oznako). Tak način učenja je značilen za metode strojnega učenja, ko gre za učenje konceptov s primeri [6]. Koncept je v primeru prepoznavanja gibov neka podmnožica primerov, ki pripada določenemu gibu. Učenje koncepta torej pomeni, da se računalnik/sistem iz podanih primerov sam nauči, kateri primeri pripadajo določenemu konceptu.

Namen dela je torej preučiti, ali lahko s pomočjo metod strojnega učenja prepoznamo človekov gib, ki je zajet s tro-smernim pospeškomerom.

Za prepoznavanje gibov bom uporabil dve metodi strojnega učenja in ju med seboj primerjal. Metodi bosta preizkušeni na dveh potencialno zanimivih domenah prepoznavanja človekovih

gibov: pisanje velikih črk angleške abecede na mizi ter ročna signalizacija v zraku. V primeru pisanja črk na mizi bosta uporabljeni le meritvi iz pospeškomerov, ki sta postavljena vzporedno z ravnino mize. Pri ročni signalizaciji v zraku bodo uporabljeni vsi trije pospeškomeri. Ker v splošnem ni nujno, da za prepoznavo uporabljamo samo en tro-smerni pospeškomer, za katerega vemo, da je smiselno postavljen (lahko bi imeli več pospeškomerov na različnih delih telesa), bosta metodi preverjeni tudi na podatkih z dodanimi naključnimi meritvami.

Delo je razdeljeno v pet poglavij. V prvem poglavju so predstavljeni naprava za zaznavanje gibanja, splošen opis načina zajemanja meritev ter opis podatkov, ki jih pri tem dobimo. V drugem poglavju sta predstavljeni uporabljeni metodi strojnega učenja za prepoznavo človekovih gibov. V tretjem poglavju sta predstavljena problema prepoznavanja na mizi napisanih velikih črk angleške abecede ter prepoznavanja ročne signalizacije v zraku. V četrtem poglavju so predstavljeni rezultati obeh metod na danih problemih iz tretjega poglavja. V petem poglavju je podana končna ocena ustreznosti uporabljenih metod za prepoznavo človekovih gibov.

1 Zaznavanje gibanja

1.1 Naprava za zaznavanje gibanja - Xsens MTI

Za zaznavanje gibanja sem uporabil napravo Xsens MTI (Slika 3) [7]. Naprava vsebuje tri pravokotne pospeškomere, tri pravokotne žiroskope ter tri pravokotne magnetomere. Kljub temu, da naprava Xsens MTI vsebuje tri različne tipe senzorjev za zaznavanja gibanja, so bili uporabljeni le pospeškomeri. Razlog za tako izbiro je v popularnosti uporabe pospeškomerov v napravah široke potrošnje, medtem ko se žiroskopi in magnetomeri redkeje uporabljajo.

Slika 3 Naprava Xsens MTI; vir: Xsens [7]

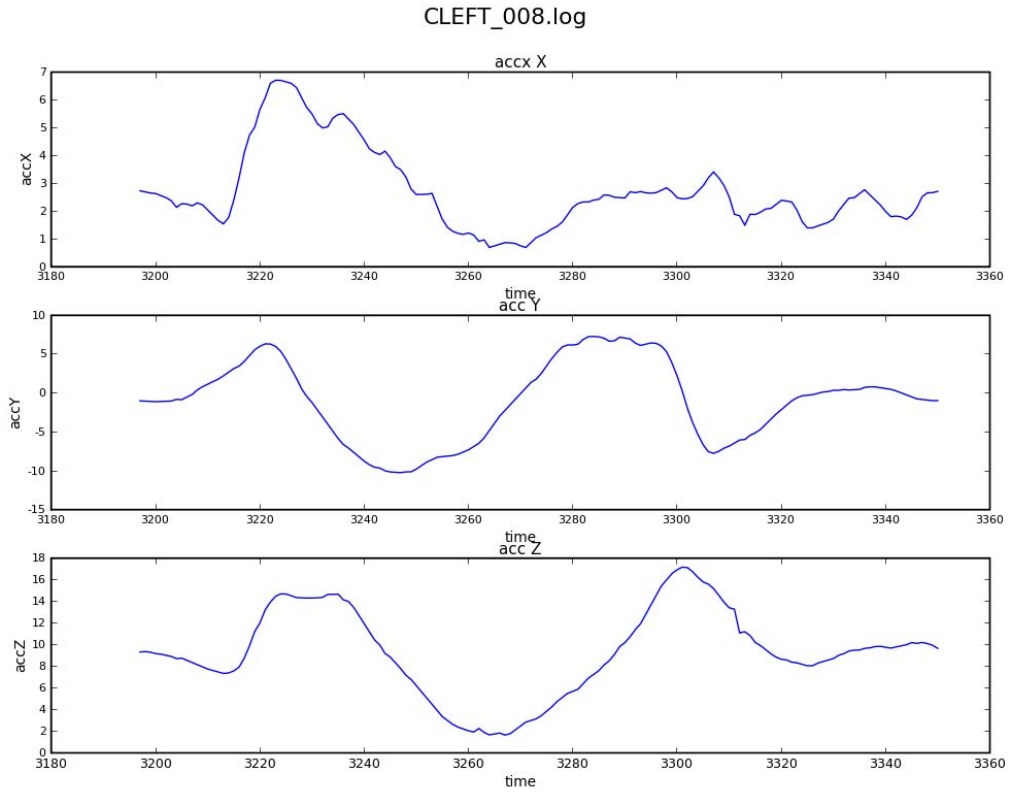
Če ne uporabimo žiroskopov in magnetomerov, je naprava Xsens MTI sestavljena iz treh pravokotnih pospeškomerov, ki merijo pospeške v območju +/- 17m/s^2. Podatki vseh treh pospeškomerov so med seboj časovno poravnani in se zajemajo s frekvenco 100 Hz (vsake 0.01 sekunde). Rezultat meritve je tabela, kjer prvi stolpec predstavlja čas, naslednji trije pa pospeške v treh pravokotnih smereh.

1.2 Opis postopka zajemanja podatkov

Podatki so bili zajeti za vsak gib posebej in sicer tako, da sem z napravo Xsens MTI, ki sem jo držal v roki, naredil izbrani gib. Vse meritve sem zajel sam in so bile zajete v enakem okolju. Začetek in konec giba je bil določen s pritiskom na gumb. Po končanem zajemu meritev za en gib, se meritve iz vseh pospeškomerov naraščajoče uredijo po času. Zajetim meritvam se doda oznaka, ki enolično določa tip zajetega giba (razred).

1.3 Opis podatkov

Primer je predstavljen z zajetimi meritvami za en gib ter enolično oznako tipa giba. Meritve iz pospeškomerov so med seboj časovno poravnane in so časovno urejene. Vse meritve iz pospeškomerov imajo znotraj enega primera enako število časovnih korakov (Slika 4), med primeri pa se lahko (po navadi tudi se) število korakov razlikuje.



Slika 4 Meritve iz treh pospeškomerov (v treh pravokotnih smereh)

Formalno je en primer sestavljen iz več časovnih vrst (meritev pospeškomerov) ter enolične oznake giba (razreda). Zajete podatke (časovne vrste) za en primer lahko definiramo kot:

Naj bo $T = (t_1, \ldots, t_n)$ časovna vrsta časov zajema podatkov dolžine n, $accx = (accx_1, \ldots, accx_n)$ časovna vrsta pospeškov v x smeri dolžine n, $accy = (accy_1, \ldots, accy_n)$ časovna vrsta pospeškov v y smeri dolžine n *in* $accz = (accz_1, \ldots, accz_n)$ časovna vrsta pospeškov v z smeri dolžine n. Potem je i-ta vrstica meritev primera e podana kot $e_i = (accx_i, accy_i, accz_i)$.

Primer e, podan z atributi *Accx, Accy, Accz* in razredom, pa je definiran kot:

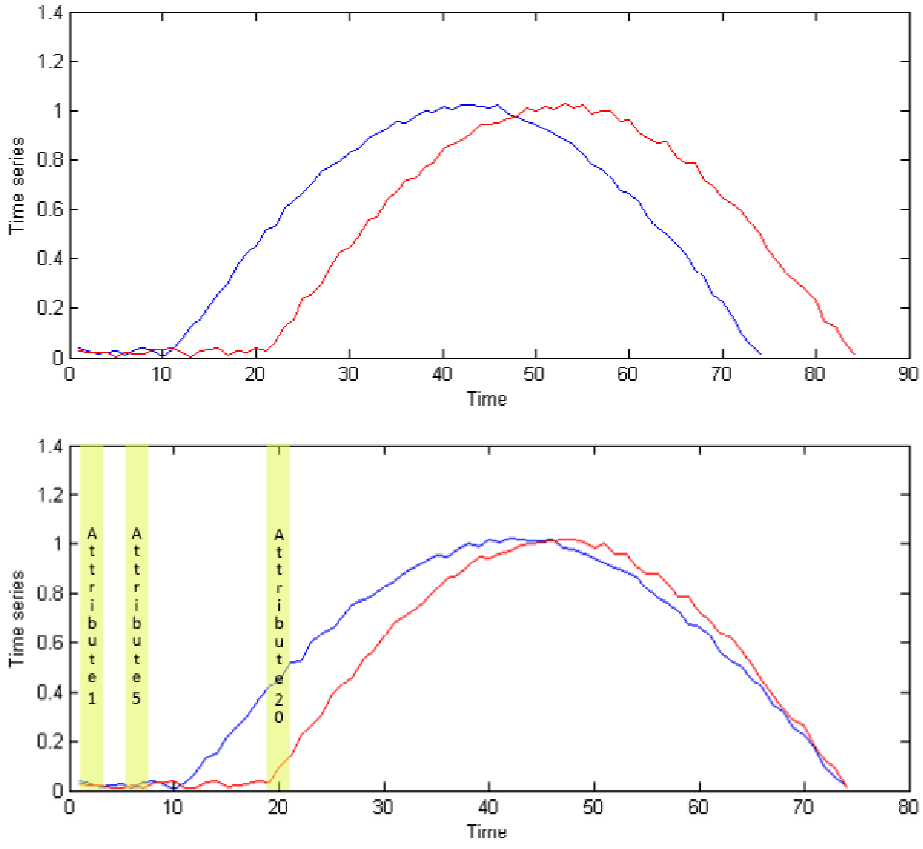
$e = (accx, accy, accz, class)$,

kjer so *accx, accy, accz* po času urejene časovne vrste atributiov *Accx*, *Accy* in *Accz*, *class* pa je enolična oznaka giba (razred) za dani primer *e*. Podobno lahko zapišemo $Accx_{ei} = accx_i$, $Accy_{ei} = accy_i$, $Accz_{ei} = accz_i$.

2 Predstavitev uporabljenih metod

Primer je v dani domeni prepoznavanja gibov predstavljen z atributi (meritve pospeškov) ter razredom (tip gibanja), kjer so atributi časovne vrste. Vse časovne vrste (atributi) določenega primera so enake dolžine in so med seboj časovno poravnane, med primeri pa to nujno ne drži. Zaradi narave uporabljenih podatkov je dano domeno težko predstaviti s klasičnim atributnim zapisom v tabeli, kjer je vsak primer v svoji vrstici, atributi in razred pa so v stolpcih. Če bi želeli podatke zapisati v taki tabelarični obliki, bi bilo potrebno poenotiti dolžine časovnih vrst preko vseh primerov in potem vsak časovni korak vsakega od atributov zapisati kot nov atribut (Slika 5). Tak zapis sicer omogoča uporabo večine znanih metod strojnega učenja, ima pa v danem primeru velike pomanjkljivosti:

- zelo veliko število atributov:
  - *št. atributov = dolžina čas. vrste × št. uporabljenih pospeškomerov*
- atributi nimajo smiselnega pomena (točke v času)
- časovne vrste med primeri niso poravnane, kar pomeni da novo zgrajeni atributi v različnih primerih predstavljajo različne smiselne dele risanja giba (na Sliki 5 bi atribut v točki 10 modre krivulje smiselno ustrezal atributu v točki 20 rdeče krivulje)



Slika 5: Pretvorba časovnih vrst v atributni zapis

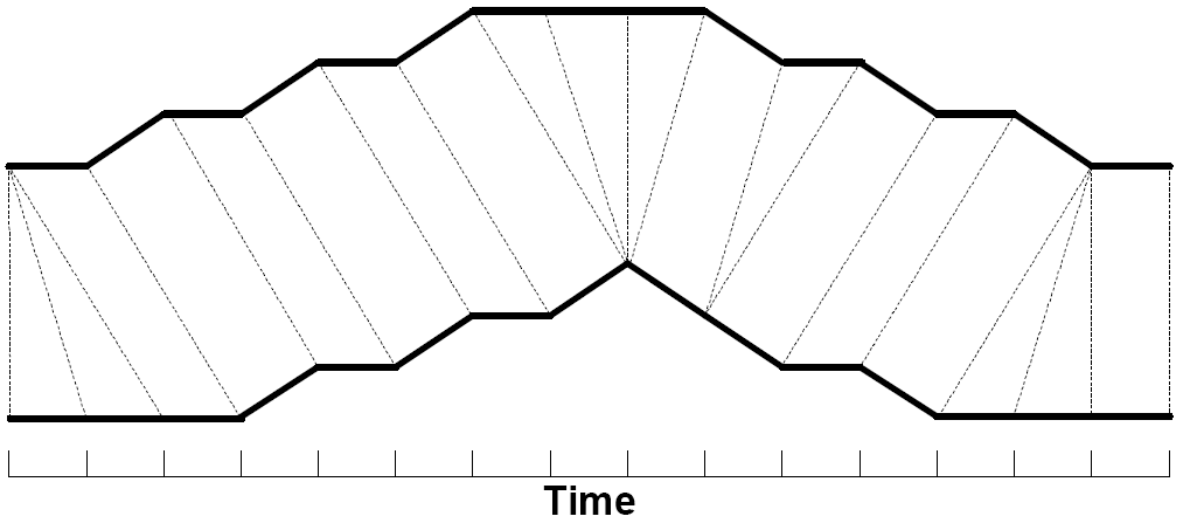
Druga možnost pretvorbe časovnih vrst v klasično tabelarično obliko je, da iz časovnih vrst izločimo značilke. Slabost takega pristopa je, da moramo vnaprej vedeti, kakšne značilke so pomembne za določen problem. Ker želimo, da metoda prepoznave gibov deluje za kakršnekoli gibe, iskanje značilk ni smiselno, saj dejanski problem vnaprej ni znan.

Zaradi težav, povezanih s pretvorbo časovnih vrst v tabelarični atributni zapis, je najbolj smiselno časovne vrste med seboj primerjati glede na njihovo oddaljenost in potem to oddaljenost uporabiti za učenje. Za računanje oddaljenosti imamo na voljo vrsto pristopov. Najenostavnejši je, da izračunamo evklidsko ali kakšno drugo razdaljo med dvema časovnima vrstama. Slabost te metode je v neupoštevanju neporavnanosti časovnih vrst med različnimi primeri. Podoben problem smo omenili že pri atributni predstavitvi in je prikazan na Sliki 5. Problem poravnanosti časovnih vrst uspešno rešuje metoda dynamic time warping (DTW), ki je opisana poglavju 2.1.

Metodi strojnega učenja, ki omogočata učenje na osnovi oddaljenosti primerov in sta bili uporabljeni, sta k-najbližjih sosedov (KNN) in odločitveno drevo časovnih vrst (TSDT). Metoda KNN je opisana v poglavju 2.2, metoda TSDT pa je opisana v poglavju 2.3.

2.1 Dynamic time warping

Metoda Dynamic Time Warping (DTW) poišče optimalno ujemanje dveh časovnih vrst, ki nista časovno poravnani. Optimalno ujemanje se doseže s poljubnim raztezanjem ali krčenjem delov časovnih vrst [8] (Slika 6). Tak način izračuna oddaljenosti je bolj primeren od izračuna evklidske ali kakšne druge razdalje, saj odpravi problem časovne poravnanosti.



Slika 6 Optimalno ujemanje časovnih vrst; vir: Intelligent Data Analysis [8]

2.1.1 Definicija DTW

DTW je definiran [8]:

Naj bosta

$X = x_1, x_2, \ldots, x_i, \ldots, x_{|X|}$ in

$Y = y_1, y_2, \ldots, y_j, \ldots, y_{|Y|}$

časovni vrsti dolžin $|X|$ in $|Y|$.

Poišči pot ujemanja W

$W = w_1, w_2, \ldots, w_K \qquad max(|X|,|Y|) \leq K < |X|+|Y|,$

kjer je K dolžina poti ujemanja in je k-ti element poti ujemanja enak

$W_k = (i,j),$

i je indeks elementa časovne vrste X, j pa indeks elementa časovne vrste Y.

Za pot ujemanja mora veljati:

- $w_1 = (1,1)$ (pot ujemanja se mora začeti na začetku obeh časovnih vrst)
- $w_K = (|X|,|Y|)$ (pot ujemanja se mora končati na koncu obeh časovnih vrst)
- $w_k = (i,j),\ w_{k+1} = (i',j'),\ i \leq i' \leq i+1,\ j \leq j' \leq j+1$ (i in j morata biti monotono naraščajoča)

Optimalna pot ujemanja $dtw(X,Y)$ med časovnima vrstama X in Y je tista pot, ki ima najmanjšo razdaljo $Dist(W)$, kjer je:

$$Dist(W) = \sum_{k=1}^{k=K} Dist(w_{ki}, w_{kj})$$

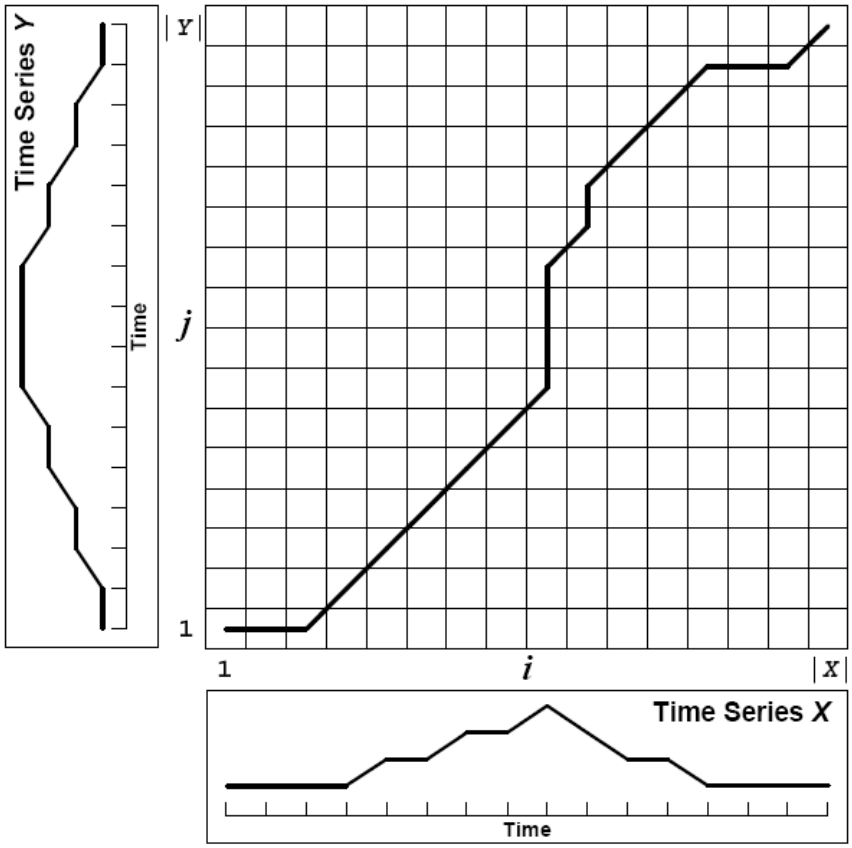
$Dist(W)$ je razdalja poti ujemanja, $Dist(w_{ki}, w_{kj})$ pa je razdalja med dvema točkama z indeksoma i in j časovnih vrst X in Y v k-tem koraku poti ujemanja.

2.1.2 Algoritem

Za izračun optimalne poti ujemanja je uporabljen pristop dinamičnega programiranja [8]. Namesto da se rešuje cel problem na enkrat, se problem razdeli na podprobleme. Najprej se reši manjše podprobleme, katerih rešitve se uporabijo za reševanje večjih podproblemov. Za reševanje DTW problema moramo zgraditi dvodimenzionalno matriko D velikosti $|X|$, $|Y|$, kjer element matrike $D(i,j)$ predstavlja minimalno razdaljo na poti ujemanja med časovnima

vrstama X in Y skozi točko x_i, y_j. Element $D(|X|,|Y|)$ matrike D vsebuje najmanjšo razdaljo poti ujemanja $Dist(W)$.

Slika 7 prikazuje matriko D in optimalno pot ujemanja. Osi x in y predstavljata časovne korake časovnih vrst X in Y, črta skozi matriko pa optimalno pot ujemanja. Tam kjer gre optimalna pot ujemanja skozi točko $D(i,j)$, se i-ta točka časovne vrste X preslika v j-to točko časovne vrste Y. Več točk časovne vrste X se lahko preslika v eno točko časovne vrste Y (vodoravna črta poti ujemanja); več točk časovne vrste Y se lahko preslika v eno točko časovne vrste X (navpična črta poti ujemanja). Če bi bili X in Y enaki časovni vrsti, bi optimalna pot ujemanja potekala po diagonali matrike D.



Slika 7 Izračun optimalne poti ujemanja; vir: Intelligent Data Analysis [8]

Za izračun optimalne poti ujemanja morajo biti izračunana vsa polja matrike D. To izračunamo s pomočjo dinamičnega programiranja, ki je mogoče zaradi naslednjih lastnosti:

- $D(i,j)$ je optimalna pot ujemanja dveh časovnih vrst dolžin i in j.
- Do $D(i,j)$ lahko pridemo iz optimalne poti ujemanja dveh časovnih vrst, ki sta za en korak krajši od i in j tako, da prištejemo razdaljo med točkama x_i in y_j.
- Ker mora biti optimalna pot ujemanja (po definiciji) monotono naraščajoča, lahko do $D(i,j)$ pridemo le iz $D(i-1,j)$, $D(i,j-1)$ ali $D(i-1,j-1)$.

Torej dinamično programiranje omogoča naslednja enačba:

$$D(i,j) = Dist(i,j) + min[D(i-1,j), D(i,j-1), D(i-1,j-1)]$$

Matriko *D* izračunamo po naslednjem algoritmu:

```
1. int DTWDistance(float s[1..n], float t[1..m], int d[1..n,1..m]) {
2. declare int DTW[0..n,0..m]
3. declare int i, j, cost
4.
5. for i := 1 to m
6.    DTW[0,i] := infinity
7. for i := 1 to n
8.    DTW[i,0] := infinity
9. DTW[0,0] := 0
10.
11. for i := 1 to n
12.   for j := 1 to m
13.         cost := d[s[i],t[j]]
14.         DTW[i,j] := cost + minimum(DTW[ i-1, j ],
15.         DTW[ i , j-1 ],
16.         DTW[ i-1, j-1 ])
17. return DTW[n,m] }
```

2.2 K-najbližjih sosedov

Metoda k-najbližjih sosedov (KNN) [9] spada med lene metode strojnega učenja in kot znanje shrani kar vse primere. Večji del učenja pravzaprav nastopi šele ob klasifikaciji novega primera. Metoda deluje tako, da danemu primeru, ki mu želimo določiti razred, poiščemo k najbližjih primerov (sosedov). Glede na vpliv in razred k najbližjih primerov se potem določi razred novega primera.

Naj bo *a* nov primer, ki ga želimo klasificirati, *dist(a,b)* oddaljenost primerov *a* in *b* ter $NN = nn_1, nn_2, \ldots, nn_k$ množica najbližjih primerov velikosti *K*. Vpliv posameznega primera iz *NN* potem izračunamo kot:

$$Vpliv(nn_i) = \frac{1/Dist(a,nn_i)}{\sum_{k=1}^{K} 1/Dist(a,nn_k)}$$

Nov primer *a* klasificiramo v razred, ki ima največji vpliv (vplive primerov po posameznih razredih seštejemo).

Napovedovanje razreda novemu primeru z metodo KNN, kjer je K=5, lahko ponazorimo z naslednjim primerom:

Recimo, da imamo na voljo učno množico desetih primerov in nek nov primer kateremu želimo napovedati razred (dejanske vrednosti razreda ne poznamo). Najprej izračunamo oddaljenost vseh primerov od novega primera, kateremu želimo določiti razred, in izberemo pet najbližjih (Tabela 1).

Primer	***Oddaljenost***	***Razred***
1	10	A
2	15	A
3	40	B
4	50	C
5	90	C
6	150	A
7	170	B
8	175	C
9	190	C
10	210	A

Tabela 1 Oddaljenost primerov od novega primera

V naslednjem koraku izračunamo vpliv posameznega primera za izbranih pet primerov (Tabela 2):

$$\sum_{k=1}^{K} \frac{1}{Dist(a, nn_k)} = \frac{1}{10} + \frac{1}{15} + \frac{1}{40} + \frac{1}{50} + \frac{1}{90} = 0.222778$$

$$Vpliv(1) = \frac{0.1}{0.222778} = 0.448878 \ldots$$

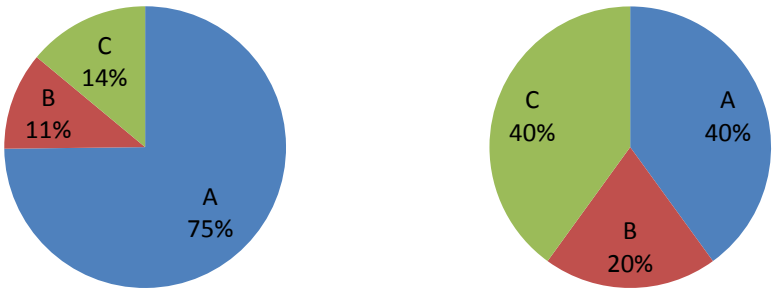
Primer	***Vpliv***	***Razred***
1	0.448878	A
2	0.299252	A
3	0.112219	B
4	0.089776	C
5	0.049875	C

Tabela 2 Vpliv najbližjih pet primerov

Zadnji korak predstavlja seštevek vplivov po posameznih razredih in klasifikacija v večinski razred A (Tabela 3).

Razred	***Vpliv***
A	0.75
B	0.11
C	0.14

Tabela 3 Porazdelitev vpliva po razredih



Slika 8 Porazdelitev razredov z upoštevanjem vpliva (levo), porazdelitev razredov brez upoštevanja vpliva (desno)

Slika 8 jasno prikazuje prednosti upoštevanja vpliva (oddaljenosti) pred enostavnejšim pristopom s katerim klasificiramo v večinski razred ne glede na oddaljenost.

2.3 Odločitveno drevo časovnih vrst

Metoda odločitveno drevo časovnih vrst (TSDT) [10] je bila razvita predvsem za reševanje problemov v medicinskih domenah, kjer so atributi predstavljeni kot časovne vrste. Osnova za metodo je metoda strojenega učenja za gradnjo odločitvenih dreves (DT), ki je spremenjena tako, da so atributi lahko predstavljeni kot časovne vrste. Odločitveno drevo in postopek gradnje sta predstavljena v poglavju 2.3.1, odločitveno drevo časovnih vrst in postopek gradnje pa v poglavju 2.3.2.

2.3.1 Odločitveno drevo

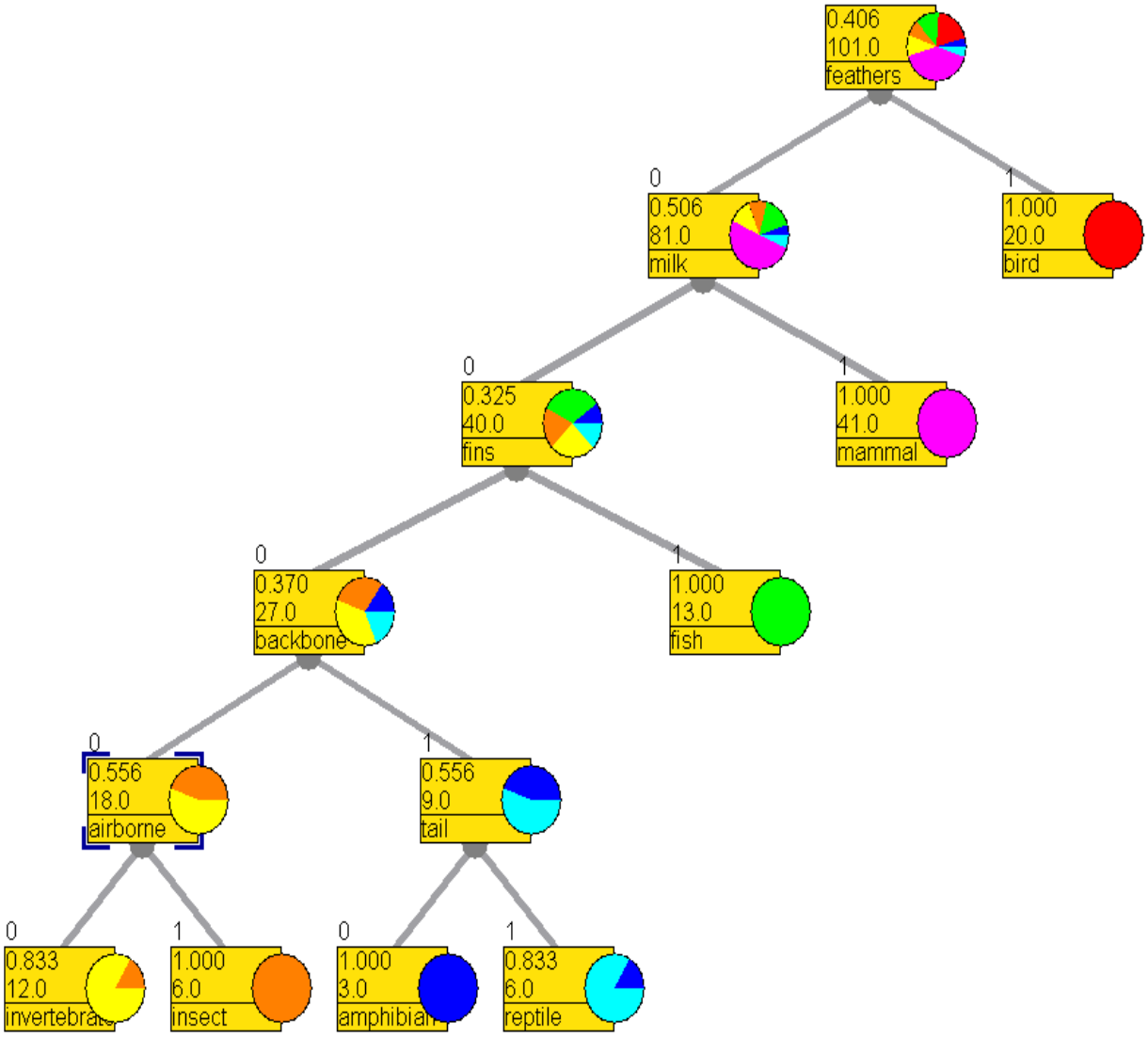
Odločitveno drevo [11,12,13] predstavlja klasifikator v obliki drevesa, kjer notranja vozlišča ustrezajo atributom, veje ustrezajo podmnožicam vrednosti atributov in listi ustrezajo razredom [14]. Klasifikacija poteka tako, da primer, ki ga želimo klasificirati (napovedati razred) potuje od korena drevesa do končnega lista v drevesu. V vsakem notranjem vozlišču se glede na vrednost atributa, ki je dodeljen vozlišču, izbere enega od sinov danega vozlišča. Tako primer potuje vse do končnega vozlišča - lista, kjer se mu določi razred.

Tudi gradnja DT poteka od korena drevesa do listov. Začnemo tako, da korenu drevesa dodelimo vse učne primere. Nadaljujemo z vejitvijo vozlišč. V vsakem vozlišču se izbere »najboljši« atribut, ki se ga dodeli vozlišču. Glede na vrednosti atributa se vozlišče razveji na več podvozlišč (sinov). Vsaki veji pripada ena vrednost atributa, vsakemu nasledniku vozlišča pa podmnožica primerov glede na vrednost atributa. Postopek vejitve rekurzivno ponavljamo, dokler ni izpolnjen ustavitveni pogoj. Ustavitveni pogoj je lahko:

- dovolj čista množica primerov,
- premajhna množica primerov,
- zmanjkalo je dobrih atributov.

Ključnega pomena za gradnjo drevesa je izbira »najboljšega atributa« [14]. Za izbiro atributa se najpogosteje uporabljajo mere: informacijski prispevek, razmerje informacijskega prispevka, gini-index in relieF.

Slika 9 prikazuje odločitveno drevo zgrajeno v orodju Orange [15]. Odločitveno drevo je zgrajeno na domeni klasifikacije živali, ki vsebuje samo diskretne atribute. Kot prikazuje slika, se v vsakem vozlišču drevesa (razen listih) vozlišče razdeli na dve podvozlišči glede na vrednost atributa, dodeljenega vozlišču. Koren drevesa vsebuje vse primere, ki se v prvem koraku razdelijo na dve podmnožici: živali, ki imajo perje (desno poddrevo) in živali, ki nimajo perja (levo poddrevo). Levo poddrevo se potem deli naprej, medtem, ko se v desnem poddrevesu (listu) vejitev ustavi (dovolj čista množica učnih primerov). Razred v listih drevesa je določen z večinskim razredom v danem listu, kar je lepo vidno v skrajnem levem in desnem listu levega poddrevesa.



Slika 9 Odločitveno drevo na domeni klasifikacije živali; vir: Orange [15]

2.3.2 Odločitveno drevo časovnih vrst

Podobno kot pri odločitvenem drevesu (poglavje 2.3.1) je tudi Odločitveno drevo časovnih vrst [10] predstavljeno kot klasifikator v obliki drevesa. Bistvena razlika med DT in TSDT vrst je v predstavitvi atributov in posledično drugačnemu pristopu pri izbiri najboljšega atributa ter vejitvi. Če pri DT nastopajo v notranjih vozliščih diskretni atributi (oz. zvezni) pri TSDT nastopajo v notranjih vozliščih časovne vrste.

Pri gradnji DT z diskretnimi atributi se v vsakem notranjem vozlišču izbere »najboljši« atribut, vejitev pa je enostavno določena z vrednostmi atributa. V primeru zveznih atributov moramo najprej poiskati mejo (diskretizacija), ki najbolje loči primere na tiste, ki imajo vrednost atributa manjšo od mejne vrednosti in tiste, ki imajo vrednost atributa večjo ali enako mejni vrednosti. Ko imamo določeno mejo, je postopek vejitve enak kot pri diskretnih atributih.

V domenah, kjer je vsak atribut predstavljen s časovno vrsto, vejitev vozlišča, glede na vrednosti atributa ni mogoča, saj ima vsak primer različno vrednost atributa. Dodatno izbiro vejitve zaplete tudi dejstvo, da primerov ne moremo urediti po vrednosti atributa (če bi to veljalo, bi lahko obravnavali atribute časovnih vrst enako kot zvezne atribute). V takih primerih moramo postopati drugače. Kot je omenjeno v poglavju 2.1, lahko z uporabo metode DTW izračunamo oddaljenost med časovnima vrstama. Ta oddaljenost služi kot osnova pri izbiri »najboljšega« atributa oz. vejitvi vozlišča. Glede na to, da je vsak primer predstavljen z več atributi (vrednost atributa je časovna vrsta) in razredom, lahko vejitev v notranjem vozlišču drevesa določimo s **primerom**, **atributom** (primer in atribut skupaj določata izbrano časovno vrsto) ter **mejo podobnosti.** Taka vejitev loči množico primerov na dve podmnožici:

- podobni primeri danemu primeru glede na izbrani atribut,
- nepodobni primeri danemu primeru glede na izbrani atribut.

Odločitveno drevo, kjer vse vejitve razdelijo množico primerov v dve podmnožici, se imenuje binarno drevo. Če smo pri DT vozlišču dodelili atribut, ki loči primere, glede na vrednosti atributa, potem pri TSDT vozlišču dodelimo primer, atribut ter mejo, ki ločijo primere glede na dano časovno vrsto in mejo. Torej vsaka vejitev vsebuje reprezentativen primer, ki po izbranemu atributu (časovni vrsti) »najbolje« loči množico primerov na podmnožico podobnih primerov ter podmnožico nepodobnih primerov glede na dano mejo.

Formalno je vsakemu notranjemu vozlišču dodeljena vejitev $\sigma(e, A, \theta)$, kjer je e izbran primer, A izbran atribut in θ meja. Naj bo A_e časovna vrsta atributa A za primer e, potem $\sigma(e, A, \theta)$ razdeli množico primerov $e_1,e_2,\ldots, e_n$ na podmnožico primerov $S_1(e, A, \theta)$, kjer za vsak primer e_i velja $dtw(A_e,A_{ei}) < \theta$ in podmnožico primerov $S_2(e, A, \theta)$, ki vsebuje ostale primere. Kvaliteta vejitve na podmnožici S_1 in S_2 je ocenjena z mero razmerje informacijskega prispevka (»gain ratio«) [16].

Najboljša vejitev $\sigma(e, A, \theta)$ je izračunana po naslednjem algoritmu:

```
Procedure: standardExSplit
Input: Set of examples e1,e2,…,en
Return value: Best split test ω
1 ω.gr = 0
2 Foreach(example e)
3     Foreach(time-series attribute A)
4           Sort examples e1,…,en using dtw(Ae,Aei)
5           Foreach(θ-cut ω')
6                 If ω'.gr > ω.gr
7                       ω = ω'
8 Return ω
```

Klasifikacija pri TSDT poteka podobno kot pri DT, s potovanjem primera od korena drevesa do lista. V vsakem vozlišču se primerja dani atribut (časovna vrsta) novega primera s časovno vrsto v vozlišču. Glede na oddaljenost časovnih vrst in dano mejo se primer uvrsti v levo ali desno poddrevo. Klasifikacija, ki poteka v listih drevesa, je enako, kot pri DT, določena z večinskim razredom, ki je v obeh primerih diskreten.

Gradnjo TSDT lahko ponazorimo z naslednjim primerom:

Recimo, da imamo na voljo učno množico petih primerov z enim atributom časovnih vrst (v splošnem imamo več atributov časovnih vrst) in diskretnim razredom (Tabela 4). Najprej izračunamo matriko oddaljenosti, ki vsebuje oddaljenosti med vsemi primeri za dani atribut (Tabela 5).

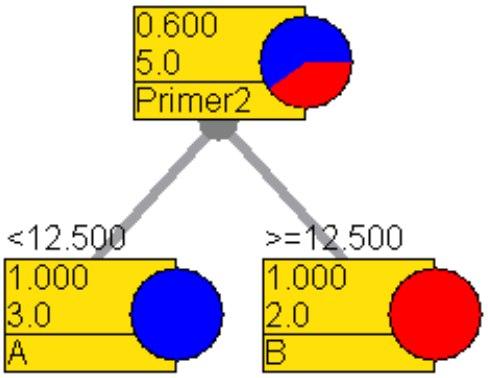
Primer	***Razred***
1	A
2	A
3	A
4	B
5	B

Tabela 4 Primeri in razred

Primer/Primer	**1**	**2**	**3**	**4**	**5**
1	0	10	30	15	25
2	10	0	5	20	15
3	30	5	0	25	50
4	15	20	25	0	20
5	25	15	50	20	0

Tabela 5 Matrika oddaljenost primerov

Naslednji korak je, da za vsak primer in atribut poiščemo najboljšo mejo in ocenimo kvaliteto dobljene vejitve. To naredimo tako, da za vsak primer in atribut, primere uredimo po oddaljenosti in za vsako možno mejo izračunamo razmerje informacijskega prispevka. Izberemo tisto vejitev (primer, atribut, mejo), ki ima najboljšo oceno (največje razmerje informacijskega prispevka). Tabela 6 prikazuje najboljše meje za posamezen primer in atribut. Na Sliki 10 je prikazano, na danih podatkih, zgrajeno drevo, ki ima v korenu najboljšo vejitev (vejitev po primeru 2 in mejo 12.5).



Slika 10 Odločitveno drevo časovnih vrst

Primer							***Razmerje informacijskega prispevka***
1	Primer	1	2	4	5	3	0.43
	Razred	A	A	B	B	A	
	Razdalja	0	10	15	25	30	
2	Primer	2	3	1	5	4	1
	Razred	A	A	A	B	B	
	Razdalja	0	5	10	15	20	
3	Primer	3	2	4	1	5	0.45
	Razred	A	A	B	A	B	
	Razdalja	0	5	25	30	50	
4	Primer	4	1	2	3	5	0.45
	Razred	B	A	A	A	B	
	Razdalja	0	15	20	25	20	
5	Primer	5	2	4	1	3	0.45
	Razred	B	A	B	A	A	
	Razdalja	0	15	20	25	50	

Tabela 6 Najboljše meje za posamezen primer (modra barva določa mejo) in ocena kvalitete vejitve

3 Predstavitev problemov

Za preverjanje ustreznosti pospeškomerov in izbranih metod za prepoznavanje človekovih gibov, sta bila izbrana dva praktična problema. Prvi problem, ki je opisan v poglavju 3.1, predstavlja prepoznavanje velikih črk angleške abecede, ki so napisane s premikanjem naprave Xsens MTI po mizi. Drugi problem predstavlja prepoznavanje ročne signalizacije v zraku (»risanje« šestih različnih znakov po zraku) in je opisan v poglavju 3.2.

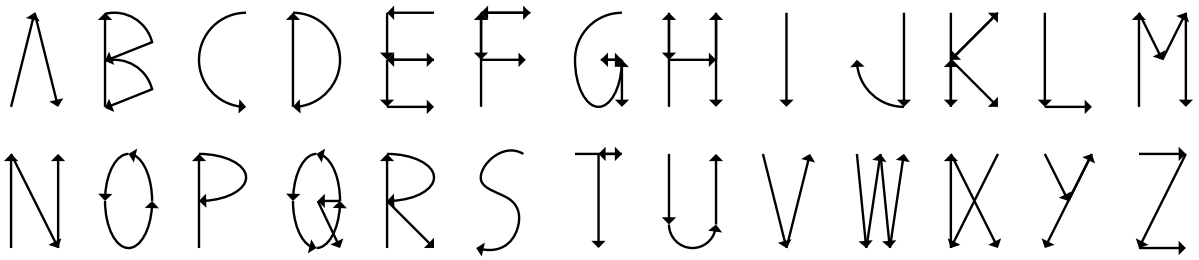
3.1 Pisanje velikih črk angleške abecede na ravni podlagi

Množico zajetih podatkov predstavlja več ponovitev pisanja velikih črk angleške abecede (26 znakov). Vseh primerov skupaj je 391, število primerov po posameznih črkah pa prikazuje Tabela 7.

Črka	*Št. prim.*	*Črka*	*Št. prim.*	*Črka*	*Št. prim.*	*Črka*	*Št. prim.*	*Črka*	*Št. prim.*
A	15	**G**	16	**M**	15	**S**	13	**Y**	13
B	18	**H**	7	**N**	13	**T**	14	**Z**	16
C	23	**I**	18	**O**	17	**U**	14		
D	18	**J**	14	**P**	15	**V**	18		
E	11	**K**	13	**Q**	20	**W**	13		
F	8	**L**	17	**R**	16	**X**	16		

Tabela 7 Število primerov za posamezen znak

Vsak primer (eno pisanje črke) je bil zajet posebej in sicer s premikanjem naprave Xsens MTI (pisanjem) po mizi. Naprava je postavljena tako, da sta oba pospeškomera vzporedna z ravnino mize (pravokotna na zemeljski pospešek). Zaradi lažjega pisanja so premiki pri pisanju posamezne črke čim bolj poenostavljeni (Slika 11).



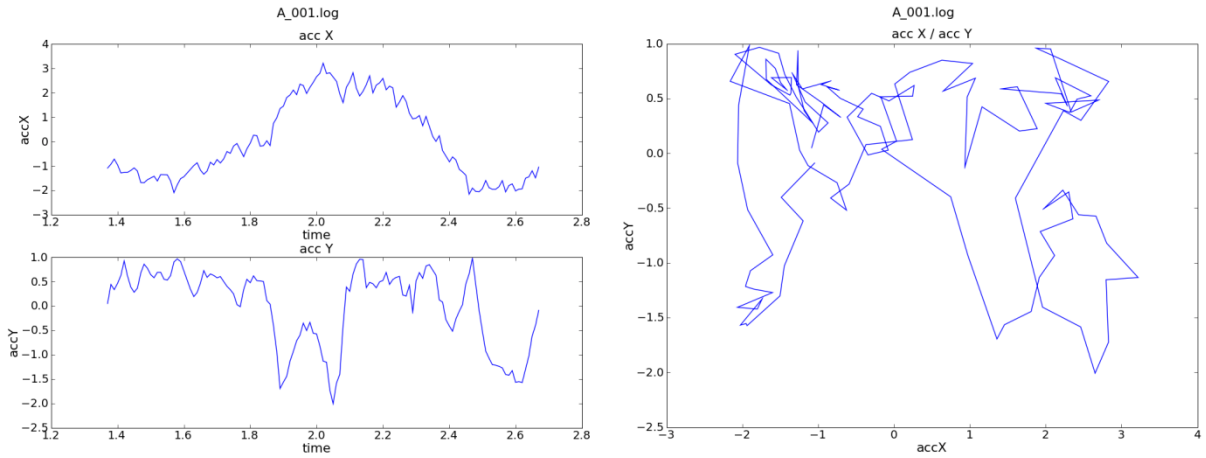
Slika 11 Sledi pisanja znakov

Vsako zaporedje premikov je zajeto z dvemi pospeškomeri, kjer je en primer *e* predstavljen kot:

$$e = (accx, accy, class),$$

accx in *accy* sta časovni vrsti meritev pospeškomerov v vodoravni in navpični smeri, *class* pa je tip znaka (črka) za dani primer *e*.

Primer meritev dobljenih iz pospeškomerov za pisanje črke 'A' prikazuje Slika 12.



Slika 12 Odvisnost pospeška od časa v *x* in *y* smeri za črko 'A'

3.2 Ročna signalizacija v zraku

Množico zajetih podatkov predstavlja več ponovitev ročne signalizacije v zraku (šest različnih gibov). Vseh primerov skupaj je 69 in zajemajo premike v levo, desno, gor, dol, risanje kroga v nasprotni smeri urinega kazalca ter risanje kroga v smeri urinega kazalca. Število primerov po posameznih gibih prikazuje Tabela 8.

Gib	***Število primerov***
Levo	12
Desno	12
Gor	9
Dol	13
Korg v nasporni smeri urinega kazalca	13
Korg v smeri urinega kazalca	10

Tabela 8 Število primerov za posamezen gib

Vsak primer (ena signalizacija z roko) je bil zajet posebej in sicer s premikanjem naprave Xsens MTI po zraku. Naprava je bila v vseh primerih podobno obrnjena (kolikor jo je možno enako držati) in sicer tako, da sta bila dva pospeškomera pravokotna na zemeljski pospešek, en pospeškomer pa je bil z zemeljskim pospeškom poravnan. Sledi posameznih gibov prikazuje Slika 13.

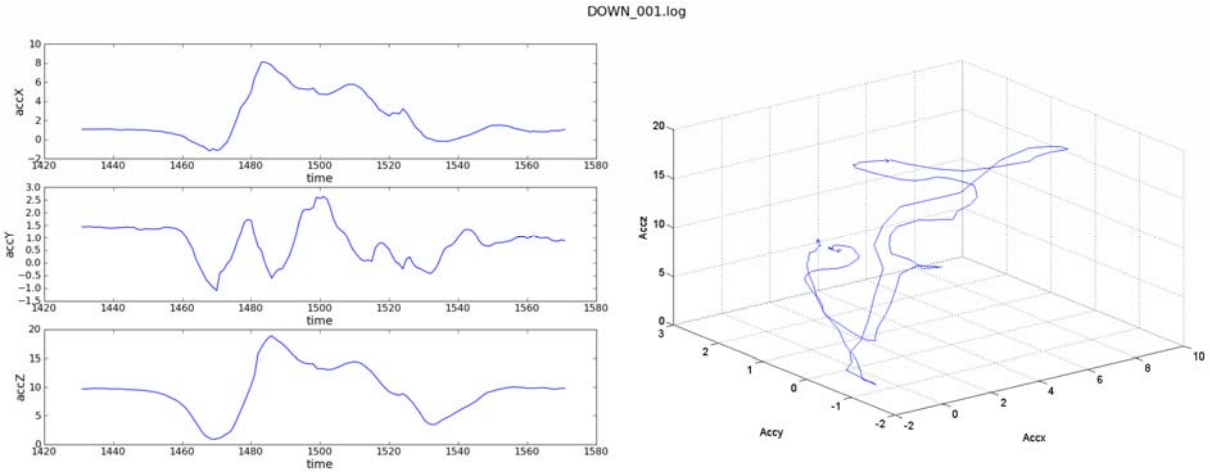
Slika 13 Sledi risanja gibov: levo, desno, dol, gor, krog v nasprotni smeri urinega kazalca, krog v smeri urinega kazalca. Črna pika predstavlja začetek risanja.

Vsako zaporedje premikov je zajeto s tremi pospeškomeri, kjer je en primer *e* predstavljen kot:

$$e = (accx, accy, accz, class),$$

accx, *accy* in *accz* so časovne vrste meritev pospeškomerov v treh pravokotnih smereh, *class* pa je tip signalizacije ('levo', 'desno', 'gor', 'dol', 'krog v smeri urinega kazalca', 'krog v nasprotni smeri urinega kazalca') za dani primer *e*.

Primer meritev dobljenih iz pospeškomerov za signalizacijo 'dol' prikazuje Slika 14.



Slika 14 Odvisnost pospeška od časa v *x*, *y* in *z* smeri za signalizacijo 'Dol'

4 Rezultati

Ustreznost uporabe pospeškomerov in metod strojnega učenja za prepoznavo človekovih gibov je bila preverjena na dveh domenah. Prva domena predstavlja prepoznavanje na mizi napisanih velikih črk angleške abecede (3.1) in uporablja dva pravokotna pospeškomera; druga domena predstavlja prepoznavanje v zraku narisanih gibov oz. prepoznavanje ročne signalizacije (3.2) in uporablja tri pospeškomere. Glede na to, da lahko za reševanje splošnih problemov prepoznave človekovih gibov uporabimo poljubno število na različnih mestih in različno obrnjenih pospeškomerov, ki niso nujno smiselno postavljeni, sta bili metodi strojnega učenja preverjeni tudi na podatkih, ki vsebujejo naključne atribute. Vsak naključen atribut je predstavljen kot časovna vrsta naključnih vrednosti med -5 in 5 in je enake dolžine kot časovne vrsta meritev iz pospeškomerov danega primera. Skupaj s podatki, ki vsebujejo naključne atribute imamo na voljo štiri različne množice testnih podatkov: črke, črke z dodanimi naključnimi atributi, signalizacija ter signalizacija z dodanimi naključnimi atributi. Različne množice testnih podatkov in prikaz strukture posameznega primera za take podatke prikazuje Tabela 9.

Testni podatki	***Struktura primera***
Črke	*(accx, accy, class)*
Črke z nak. at.	*(accx, accy, noise1, noise2, class)*
Signalizacija	*(accx, accy, accz, class)*
Signalizacija z nak. at.	*(accx, accy, accz, noise1, noise2, noise3, class)*

Tabela 9 Množice testnih podatkov. Oznake accx, accy in accz so časovne vrste meritev iz pospeškomerov za dani primer; noise1, noise2 in noise3 so časovne vrste naključnih vrednosti za dani primer; class je razred danega primera.

Vse testne množice so bile uporabljene za preverjanje obeh algoritmov strojnega učenja: KNN (poglavje 2.2) in TSDT (poglavje 2.3). Obe metodi temeljita na oceni oddaljenosti primerov oz. oddaljenosti primerov po posameznem atributu, za kar je uporabljena metoda DTW (poglavje 2.1). V primeru metode KNN se s pomočjo DTW izračuna oddaljenost med dvema primeroma preko vseh atributov, medtem ko se v primeru metode TSDT izračuna oddaljenost po posameznem atributu. Za izračun optimalnega ujemanja časovnih vrst potrebuje DTW funkcijo razdalje med dvema točkama danih vrst. V primeru KNN je ta funkcija za *i*-to in *j*-to točko meritev primerov *a* in *b* enaka:

$$Dist(a_i, b_j) = \left[\left(Accx_{a_i} - Accx_{b_j}\right)^2 + \left(Accy_{a_i} - Accy_{b_j}\right)^2\right]^{1/2},$$

oziroma v primeru dodanih naključnih atributov:

$$Dist(a_i, b_j) = \left[\left(Accx_{a_i} - Accx_{b_j}\right)^2 + \left(Accy_{a_i} - Accy_{b_j}\right)^2 + \left(Noise1_{a_i} - Noise1_{b_j}\right)^2 + \left(Noise2_{a_i} - Noise2_{b_j}\right)^2\right]^{1/2}$$

V primeru metode TSDT se podobnost med primeri računa za vsak atribut posebej in je funkcija razdalje po atributu *A* za *i*-to in *j*-to točko meritev primerov *a* in *b* enaka:

$$Dist\left(A_{a_i}, A_{b_j}\right) = \left[\left(A_{a_i} - A_{b_j}\right)^2\right]^{1/2},$$

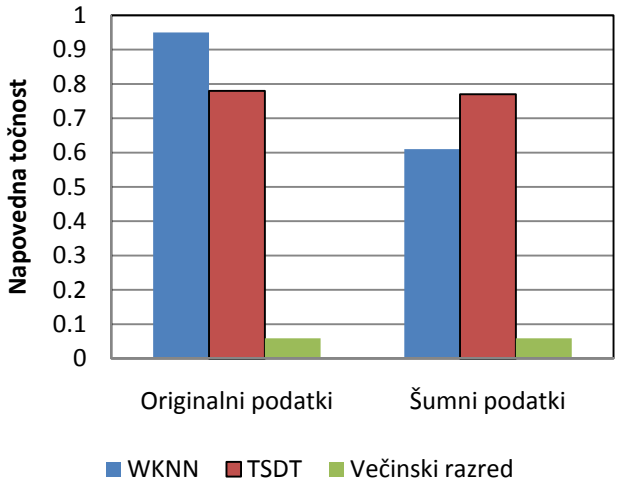
kjer je atribut *A* lahko *Accx*, *Accy* ali *Accz* oz. v primeri dodanih naključnih atributov še *Noise1*, *Noise2* ali *Noise3*.

Uspešnost prepoznave človeškega giba s pomočjo pospeškomerov in v drugem poglavju opisanih metod strojnega učenja je bila preverjena z metodo izloči enega (»leave one out«) [17]. Metoda izloči enega deluje tako, da v vsakem koraku izbere en testni primer, vsi ostali primeri pa so učni (učna množica). Na učni množici se izbrani algoritem strojnega učenja nauči napovednega modela, s testnim primerom pa se oceni kvaliteto napovedi. Kvaliteta napovedi je bila ocenjena s klasifikacijsko točnostjo (»Classification Accuracy«). Končna ocena kvalitete napovedi predstavlja klasifikacijsko točnost vseh primerov:

$$CA = \frac{\text{št.}\, pravilno\ klasificiranih\ primerov}{\text{št.}\, vseh\ primerov}$$

4.1 Rezultati na domeni pisanja črk na ravni podlagi

Preveriti sem želel, kako dobro se obneseta v poglavjih 2.2 in 2.3 opisani metodi strojnega učenja pri prepoznavi velikih črk angleške abecede. Za preverjanje ustreznosti algoritmov strojnega učenja je bil uporabljen postopek izloči enega, za oceno kvalitete učenja pa je bila uporabljena mera klasifikacijska točnost (CA).



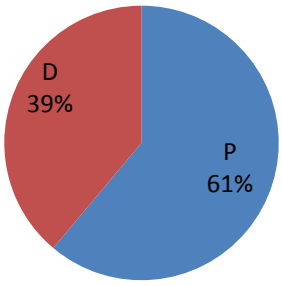
Slika 15 Napovedna točnost algoritmov strojnega učenja na domeni prepoznave črk

Na originalnih podatkih, kjer sta pospeškomera smiselno postavljena in podatki ne vsebujejo naključnih atributov, se je najbolje izkazala metoda KNN (CA = 0.95). Malo slabše se je odrezala metoda TSDT (CA = 0.78). Kljub slabšemu rezultatu metode TSDT pa je rezultat še vedno zelo dober, če ga primerjamo s spodnjo mejo klasifikacijske točnosti, ki jo lahko dosežemo z enostavno klasifikacijo v večinski razred (Slika 15).

Črka	*KNN*	*TSDT*	*KNN (šum)*	*TSDT (šum)*	*Črka*	*KNN*	*TSDT*	*KNN (šum)*	*TSDT (šum)*
A	1.00	0.80	0.73	0.80	**N**	1.00	0.69	0.23	0.77
B	1.00	0.94	0.67	0.94	**O**	0.88	0.65	0.24	0.65
C	1.00	0.74	0.70	0.74	**P**	0.87	0.67	0.67	0.73
D	0.39	0.50	0.22	0.61	**Q**	1.00	0.75	0.60	0.75
E	0.82	0.73	0.82	0.64	**R**	0.94	0.75	0.25	0.69
F	1.00	0.88	0.75	0.88	**S**	1.00	0.92	0.69	0.92
G	1.00	0.75	0.88	0.75	**T**	1.00	0.86	1.00	0.86
H	1.00	0.57	0.43	0.71	**U**	1.00	0.79	0.86	0.64
I	1.00	0.89	1.00	0.94	**V**	1.00	0.89	0.44	0.89
J	1.00	0.79	0.79	0.71	**W**	1.00	1.00	0.15	1.00
K	1.00	0.69	0.92	0.69	**X**	1.00	0.88	0.63	0.75
L	1.00	0.88	0.88	0.88	**Y**	1.00	0.54	0.38	0.54
M	1.00	0.87	0.53	0.80	**Z**	1.00	0.75	0.44	0.81
Skupaj	0.95	0.78	0.61	0.77					

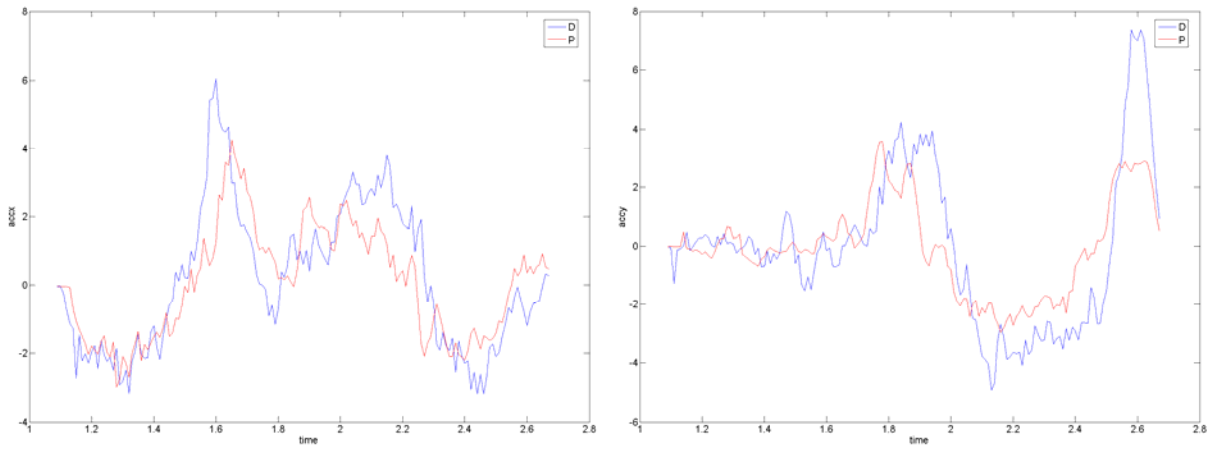
Tabela 10 Napovedne točnosti algoritmov strojnega učenja na domeni prepoznave črk, kjer »(šum)« označuje podatke z dodanimi naključnimi atributi

Zanimivo je tudi primerjati klasifikacijske točnosti po posameznih črkah. Iz Tabele 10 hitro razberemo, da metoda KNN daleč najslabše klasificira črko 'D'. Razlog je v veliki podobnosti črke 'D' s črko 'P', kar dokazuje tudi porazdelitev napovedi za črko 'D' (Slika 16).



Slika 16 Porazdelitev napovedi KNN za črko 'D'

Ta ugotovitev kaže, da je potrebno biti pri definiciji poti risanja znakov, ki jih želimo prepoznati, zelo previden, saj so lahko določeni gibi, ki so v domeni poti premikov različni, v domeni pospeškov zelo podobni. Podobnost med črkama 'D' in 'P' v domeni pospeškov lahko preverimo, če med seboj primerjamo meritve iz pospeškomerov za primera črk 'D' in 'P' (Slika 17).

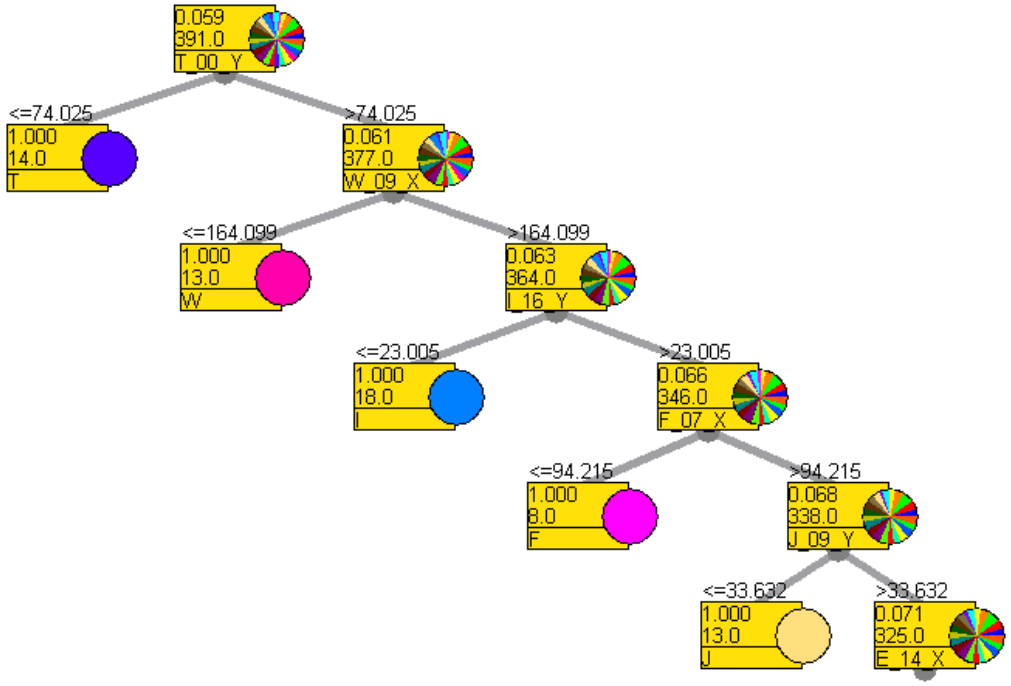


Slika 17 Primerjava meritev iz pospeškomerov v *x* in *y* smeri za primera črk 'P' in 'D'

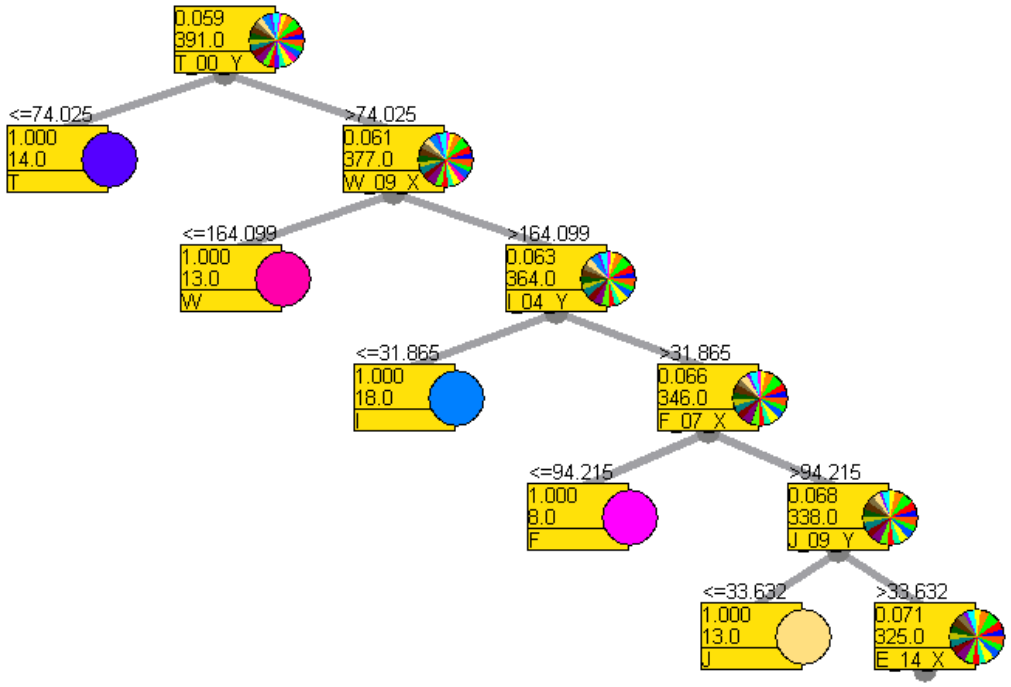
Na šumnih podatkih, kjer sta originalnim podatkom, ki vsebujejo meritve iz dveh pospeškomerov, dodana še dva naključna atributa, se je najbolje odrezala metoda TSDT (CA = 0.77). Rezultat te metode na šumnih podatkih je primerljiv z rezultatom iste metode na originalnih podatkih (CA = 0.78), kar kaže, da metoda uspešno izloči naključne atribute. Z razliko od metode TSDT pa dosega metoda KNN na šumnih podatkih precej slabše rezultate (CA = 0.61) kot na originalnih podatkih (CA = 0.95). Razlog je v nezmožnosti metode, da izloči naključne atribute, saj računa oddaljenost med primeri preko vseh atributov. Tako je pričakovati, da z večanjem števila naključnih atributov močno pada napovedna točnost te metode. Kljub slabim rezultatom na šumnih podatkih pa je metoda KNN še vedno veliko boljša od napovedovanja v večinski razred (Slika 15).

Poleg zmožnosti, da izloči šumne atribute, ima metoda TSDT tudi dobro lastnost, da je njene modele možno vizualizirati. Vizualizacija omogoča globlji vpogled v podatke ter lažjo razlago in razumevanje modela. Slika 18 prikazuje prvih pet vejitev (vseh vejitev je 30) drevesa časovnih vrst, ki je bilo zgrajeno na originalnih podatkih. Iz vizualizacije drevesa hitro razberemo, da je »najboljša« vejitev (vozlišče v korenu drevesa), vejitev po meritvi pospeškov v *y* smeri za primer »T_00«. Meja, ki loči primere na podobne in nepodobne, je podana v vejah drevesa in je v primeru vozlišča v korenu enaka 74.025. Ta meja loči primere na levo podmnožico (podobni primeri), ki vsebuje samo primere črke 'T' in na desno podmnožico, ki vsebuje ostale primere. Če se premaknemo po desni veji do naslednjega vozlišča, pridemo do nove vejitve. Podobno, kot pri vejitvi v korenu drevesa, se tudi tu primeri ločijo na levo čisto podmnožico primerov (podobni primeri) in desno množico, ki vsebuje ostale primere. Deljenje na leve čiste podmnožice in desne manj čiste podmnožice si

razložimo s tem, da so primeri, ki so blizu primeru v vozlišču (po izbranem atributu) tudi enakega razreda, kot ta primer. Zato je večja verjetnost, da bodo leve veje bolj čiste od desnih, saj vsebujejo bolj podobne primere, primeru v vozlišču, kot desne veje.



Slika 18 Prvih nekaj vozlišč odločitvenega drevesa časovnih vrst, ki je zgrajeno na domeni prepoznave črk



Slika 19 Prvih nekaj vozlišč odločitvenega drevesa časovnih vrst, ki je zgrajeno na domeni prepoznave črk z dodanimi naključnimi atributi

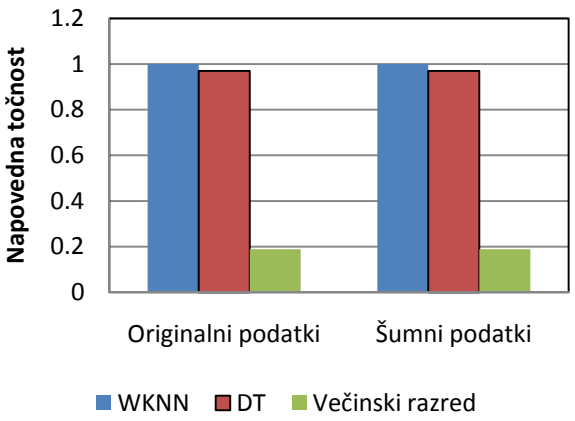
Zanimiva je tudi primerjava modelov, ki sta pridobljena z metodo TSDT na originalnih podatkih ter na podatkih z dodanimi naključnimi atributi. Če primerjamo Sliko 18 in Sliko 19 hitro opazimo, da sta enaki, kar še dodatno potrjuje, da metoda TSDT uspešno izloči naključne atribute.

4.2 Rezultati na domeni risanja znakov v zraku

Z domeno risanja znakov v zraku oz. ročne signalizacije sem želel preveriti, kako dobro se obneseta v poglavjih 2.2 in 2.3 opisani metodi strojnega učenja pri prepoznavi človekovih gibov v zraku. Za razliko od domene prepoznave velikih črk angleške abecede, kjer so bili gibi omejeni na dve dimenziji (naprava Xsens MTI drsi po mizi), v tej domeni ni več omejitve gibanja (prosto gibanje v prostoru). Podobno kot za domeno prepoznave velikih črk angleške abecede je bil, za preverjanje ustreznosti algoritmov strojnega učenja, tudi tu uporabljen postopek izloči enega, za oceno kvalitete učenja pa je bila uporabljena mera klasifikacijska točnost.

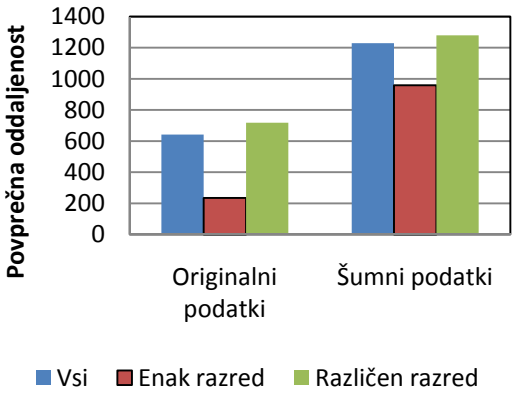
Signal	*KNN*	*TSDT*	*KNN (šum)*	*TSDT (šum)*
Levo	1	1	1	1
Desno	1	1	1	1
Gor	1	0.89	1	0.89
Dol	1	1	1	1
Krog v levo	1	0.92	1	0.92
Krog v desno	1	1	1	1
Skupaj	1	0.97	1	0.97

Tabela 11 Napovedne točnosti algoritmov strojnega učenja na domeni signalizacije v zraku, kjer »(šum)« označuje podatke z dodanimi naključnimi atributi

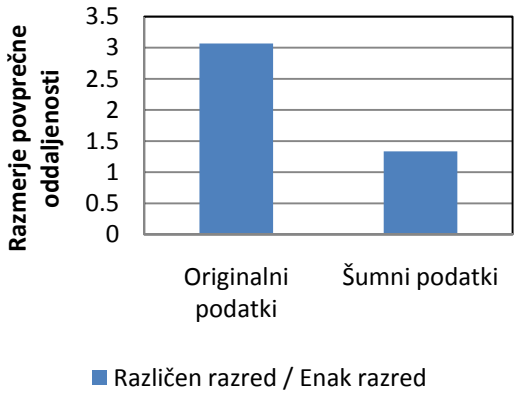


Slika 20 Napovedna točnost algoritmov strojnega učenja na domeni signalizacije v zraku

Zaradi majhnega števila različnih razredov (gibov) in velikih razlik med njimi, sta se zelo dobro odrezali obe metodi (Tabela 11). Presenetljivo dobri so tudi rezultati na podatkih z dodanimi naključnimi atributi, ki se ne razlikujejo od rezultatov na originalnih podatkih (Slika 20). Za metodo KNN bi (zaradi računanja oddaljenosti preko vseh atributov) pričakovali precej slabše rezultate na šumnih podatkih. Razlog za dobre rezultate lahko razložimo s tem, da zaradi velikih razlik med primeri različnih razredov šum (naključni atributi) ne pride do izraza. Kljub dobrim rezultatom pa ne moremo trditi, da dodani naključni atributi nimajo vpliva na rezultate dobljene z metodo KNN. Če primerjamo povprečne oddaljenosti med primeri istega razreda in primeri različnih razredov pri originalnih podatkih ter pri šumnih podatkih (Slika 21), vidimo, da so te razlike pri šumnih podatkih precej manjše kot pri originalnih podatkih. Tako je pri originalnih podatkih povprečna oddaljenost med dvema primeroma, ki pripadata istemu razredu, kar trikrat manjša od povprečne oddaljenosti med primeroma, ki pripadata različnim razredom (Slika 22). Pri šumnih podatkih je razmerje povprečnih oddaljenost med primeri istih razredov in primeri različnih razredov le še približno 1.3. Predvidevamo lahko, da bi se te razlike z dodajanjem naključnih atributov še zmanjševale, kar bi posledično pripeljalo do slabih rezultatov.



Slika 21 Primerjava povprečne oddaljenosti med primeri na domeni signalizacije v zraku



Slika 22 Razmerje povprečne oddaljenosti med primeri istega in različnega razreda na domeni signalizacije v zraku

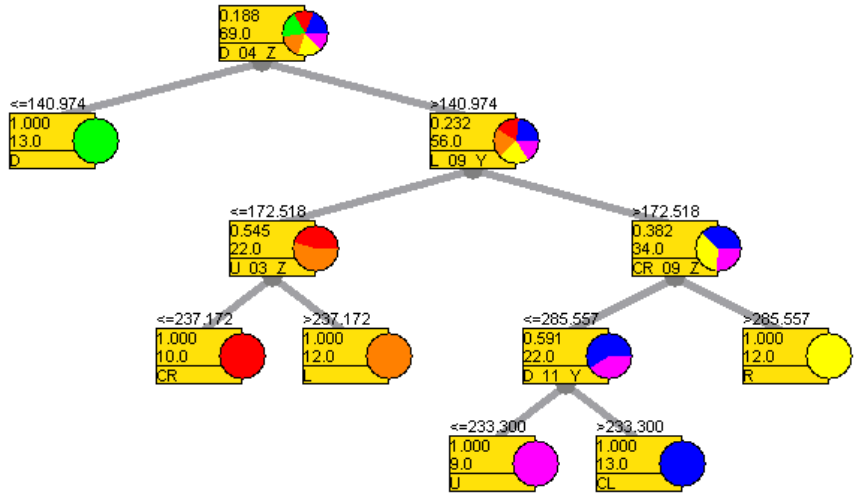
Zanimivo je tudi preučiti odločitveni drevesi časovnih vrst, ki sta zgrajeni na originalnih podatkih ter na podatkih z dodanimi naključnimi atributi, ki jih prikazujeta Slika 23 in Slika 24. Oznake v notranjih vozliščih prikazujejo po vrsticah:

- verjetnost večinskega razreda,
- število primerov v vozlišču,
- izbrani primer ter atribut;

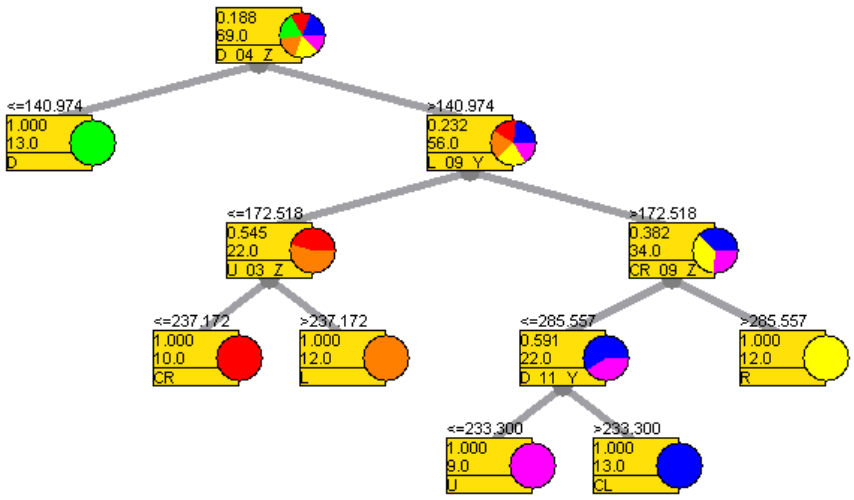
oznake na vejah pa predstavljajo izbrano mejo. Izbrani primer ter atribut razberemo iz oznake v zadnji vrstici notranjega vozlišča in sicer:

- prva dva sklopa črk označujeta primer,
- zadnji sklop (črka) označuje atribut, kjer je:
    - X meritev pospeškomera v smeri *x*,
    - Y meritev pospeškomera v smer *y*,
    - Z meritev pospeškomera v smeri *z*.

Oznake v listih so podobne oznakam v notranjih vozliščih, le da oznaka v zadnji vrstici predstavlja razred (signalizacijo), kjer je U = 'gor', D = 'dol', L = 'levo', R = 'desno', CR = 'krog v smeri urinega kazalca' in CL = 'krog v nasprotni smeri urinega kazalca'. Obarvani krogi v vozliščih predstavljajo porazdelitev razreda, kjer je: vijolična = 'gor', zelena = 'dol', oranžna = 'levo', rumena = 'desno', rdeča = 'krog v smeri urinega kazalca' in modra = 'krog v nasprotni smeri urinega kazalca'.



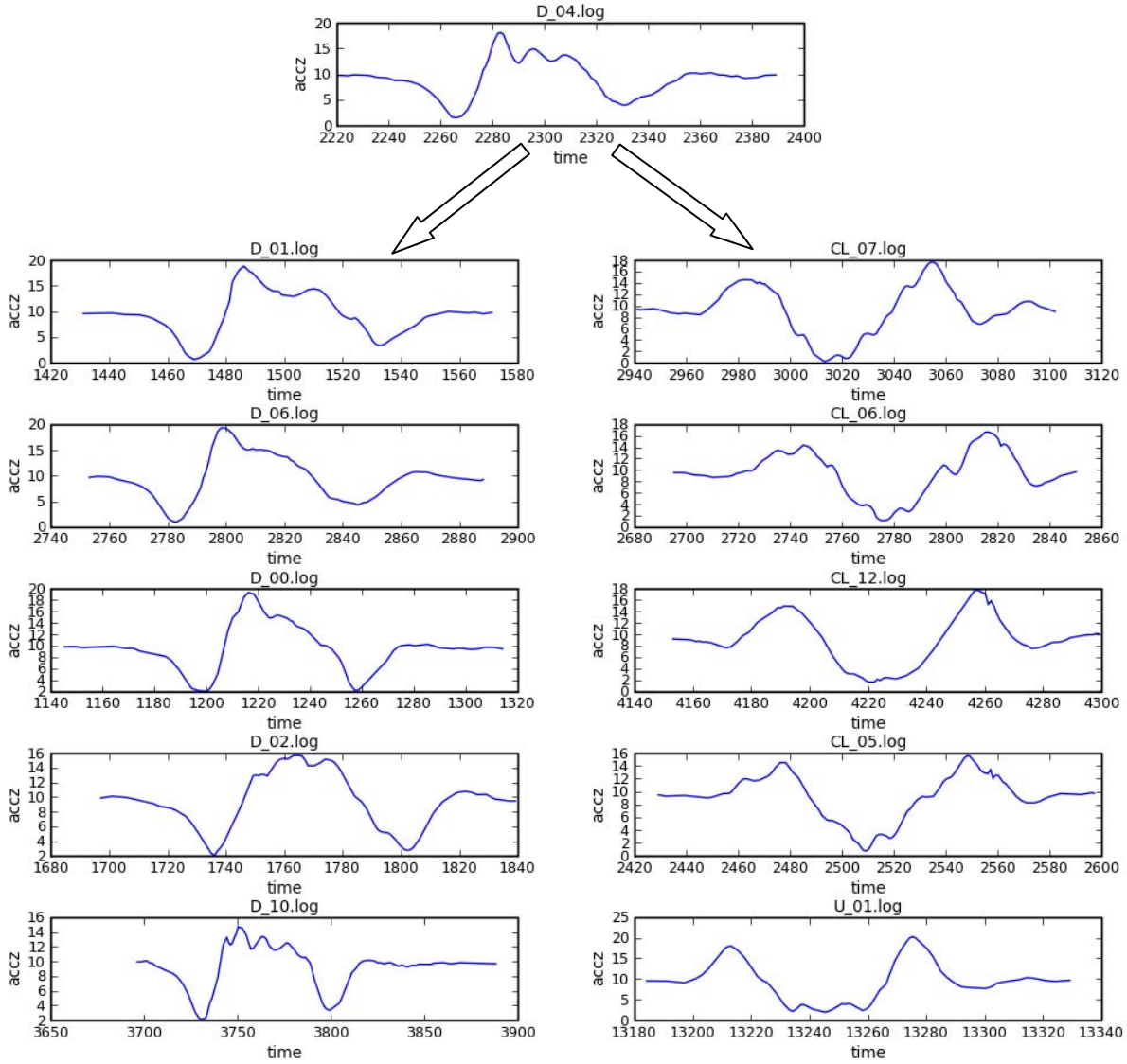
Slika 23 Odločitveno drevo časovnih vrst, ki je zgrajeno na domeni signalizacije v zraku



Slika 24 Odločitveno drevo časovnih vrst, , ki je zgrajeno na domeni signalizacije v zraku z dodanimi naključnimi atributi

Iz Slike 24 je razvidno, da nobeno vozlišče drevesa, ki je bilo zgrajeno na šumnih podatkih, ne vsebuje naključnega atributa. Če primerjamo drevo, ki je zgrajeno na originalnih podatkih (Slika 23) z drevesom zgrajenim na podatkih z dodanimi naključnimi atributi (Slika 24), hitro opazimo, da sta enaki. To dokazuje, da naključni atributi nimajo vpliva na metodo TSDT.

Polega dobre lastnosti, da TSDT izloči naključne atribute, pa omogoča vizualizacija TSDT tudi globlji vpogled v podatke in s tem lažjo razlago podatkov in samega modela. Za primer vzemimo kar prvo vozlišče (koren drevesa), na čistih podatkih zgrajenega TSDT, ki vsebuje »najboljšo« vejitev. Ta vejitev razdeli množico 69 primerov (verjetnost večinskega razreda je 0.188) na dve podmnožici glede na meritev pospeškomera v smeri *z* za primer »D_04« in mejo 140.974.



Slika 25 Vizualizacija vejitve v korenu TSDT, ki je bilo zgrajeno na domeni signalizacije v zraku

Slika 25 prikazuje vejitev v korenu drevesa. Zaradi preglednosti je prikazanih le prvih pet najbližjih primerov iz leve podmnožice (primeri, ki so oddaljeni za manj kot določa meja) in prvih pet najbližjih primerov iz desne podmnožice (primeri , ki so oddaljeni za več kot določa meja). Jasno se vidi, da so časovne vrste v levi podmnožici bolj podobne časovni vrsti v vozlišču, kot časovne vrste iz desne podmnožice. Iz Slike 15 tudi hitro razberemo, da najbolj podobne časovne vrste iz desne podmnožice pripadajo primerom signalizacije 'risanje kroga v nasprotni smeri urinega kazalca' (primeri »CL_07«, »CL_06«, …). To kaže na to, da je pri pospeških v smeri *z* signalizaciji 'dol', najbolj podobna signalizacija 'krog v nasprotni smeri urinega kazalca'.

Na podoben način, kot smo analizirali vozlišče v korenu drevesa, lahko storimo z ostalimi vozlišči. To storimo tako, da »pregledujemo« vozlišča na poti od korena do listov. Vsako vozlišče vsebuje reprezentativen primer, ki po danem atributu in dani meji »najbolje« razdeli množico primerov na dve podmnožici (podobni primeri in nepodobni primeri).

4.3 Primerjava rezultatov uporabljenih algoritmov strojnega učenja

Metoda KNN se je odrezala veliko bolje na podatkih, ki ne vsebujejo naključnih atributov oz. kjer vsi atributi smiselno opisujejo razred, od metode TSDT. Razloge za boljše rezultate lahko iščemo v samem načinu napovedovanja, računanju oddaljenosti preko vseh atributov ter uteženem računanju razredov. Po drugi strani pa se je metoda TSDT bolje obnesla na podatkih, ki vsebujejo naključne atribute. Glavni razlog za slabe rezultate metode KNN na šumnih podatkih je v računanju oddaljenosti med primeri preko vseh atributov. Če je v primeru smiselnih atributov to prednost, je v primeru naključnih oz. nesmiselnih atributov to velika pomanjkljivost, saj se pri računanju oddaljenosti med primeri upošteva tudi šum. Prednost metode TSDT je v sposobnosti izločanja naključnih atributov, kar dokazujejo tudi praktično nespremenjeni rezultati na šumnih in ne-šumnih podatkih.

Poleg same kvalitete napovedovanja lahko omenimo še ostale prednosti/pomanjkljivosti izbranih metod. Pri metodi KNN je naučeni model kar cela učna množica podatkov. Prednost tega pristopa je hitro učenje ter hitra nadgradnja modela v primeru novih učnih podatkov. Po drugi strani pa taka predstavitev modela zahteva veliko prostora v spominu in zelo počasno napovedovanje. V primeru TSDT je ravno obratno, imamo majhen model in hitro napovedovanje za ceno bolj počasnega učenja ter težje nadgradnje modela. Velika prednost TSDT je tudi v enostavni predstavitvi (vizualizaciji) in razložljivosti modela, kar domenskim strokovnjakom omogoča vpogled v podatke in v znanje, ki ga ti podatki skrivajo.

Na splošno bi lahko rekli, da je TSDT bolj primerno v primeru šumnih podatkov ter tam, kjer so pomembni razložljivi, prostorsko majhni modeli ter hitro napovedovanje. Po drugi strani je

metoda KNN boljša tam, kjer vemo, da so vsi atributi smiselni, kadar čas napovedovanja ni ključnega pomena in če potrebujemo možnost enostavne nadgradnje modela.

Rezultati nakazujejo tudi, da bi bilo mogoče preverjeni metodi še izboljšati. Eno od možnih izboljšav predstavlja kombinacija metod TSDT in KNN. Metodo TSDT (ali kakšno drugo) bi uporabili za izločanje naključnih atributov (predprocesiranje podatkov), metodo KNN pa bi kasneje uporabili za učenje oz. napovedovanje.

5 Zaključek

V diplomskem delu sem preveril kako dobro je mogoče prepoznavati človeško gibanje s pomočjo pospeškomerov in metod strojnega učenja. Preverjanje je bilo opravljeno na dveh domenah človeškega gibanja in sicer na domeni prepoznave na mizi napisanih velikih črk angleške abecede ter prepoznave v zraku podane ročne signalizacije. Za preverjanje so bili uporabljeni tako podatki brez dodanega šuma kot tudi podatki z dodanimi naključnimi atributi. Z dodajanjem naključnih atributov sem želel preveriti uspešnosti metod strojnega učenja na domenah prepoznave človeškega gibanja, kjer vsi pospeškomeri niso smiselno postavljeni oz. za določen gib niso vsi potrebni.

Za reševanje problema prepoznave človeških gibov sem uporabili dve metodi strojnega učenja in sicer k-najbližjih sosedov (KNN) ter odločitveno drevo časovnih vrst (TSDT). Rezultati so pokazali, da obe metodi uspešno prepoznavata človeško gibanje, ki je zajeto s pospeškomeri. Metoda KNN skorajda idealno prepoznava človeško gibanje na obeh domenah, kadar podatki ne vsebujejo naključnih atributov. V primeru podatkov, ki vsebujejo naključne atribute, se rezultati poslabšajo. Na drugi strani daje metoda TSDT na podatkih brez naključnih atributov slabše rezultate od metode KNN, je pa precej imuna na šum. Izkazalo se je, da se rezultati v primeru dodanih naključnih atributov ne poslabšajo glede na rezultate na podatkih brez naključnih atributov.

Glede na izvedene poskuse in dobljene rezultate lahko zaključim, da je s pomočjo pospeškomerov in metod strojnega učenja mogoče dobro prepoznavati človeške gibe. Uporabljeni metodi strojnega učenja sta se izkazali za primerni, katera od obeh metod je boljša, pa je odvisno tudi od domene prepoznave gibov. V splošnem bi lahko rekli, da je metoda KNN boljša za domene, ki ne vsebujejo naključnih atributov, metoda TSDT pa za domene, ki vsebujejo naključne atribute.

Rezultati kažejo, da bi bilo mogoče preverjeni metodi še izboljšati. Eno od možnih izboljšav predstavlja kombinacija metod TSDT in KNN. Metodo TSDT (ali kakšno drugo) bi uporabili za izločanje naključnih atributov (predprocesiranje podatkov), metodo KNN pa bi potem uporabili za učenje oz. napovedovanje.

Literatura

[1] K. Epstein and J. O'Leary, "Motion Sensing with Accelerometers – Present and Future," v *American Society for Engineering Education's 2006 St. Lawrence Section Conference*, Cornell University's College of Engineering, 2006.

[2] "Consumer market pushes accelerometers growth," *EE Times India*, October 2008.

[3] Apple. (2008, Nov.) Apple's Sudden Motion Sensor. Dostopno na: http://support.apple.com/kb/HT1935

[4] Apple. (2008, October) Nike + Ipod. Dostopno na: http://www.apple.com/ipod/nike/

[5] P. Svensson, "Tiny springs let Wii and PlayStation 3 controllers sense motion," *USA Today*, November 2006.

[6] I. Bratko, "Prolog Programming for Artificial Intelligence," v *Programming for Artificial Intelligence*. Ljubljana, Slovenia: Addison Wesley, 2001, pogl. 18.2, str. 442-446.

[7] (2008, October) Xsens MTI. Dostopno na: http://www.xsens.com/en/products/machine_motion/mti.php

[8] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, št. 11, zv. 5, str. 561-580, October 2007.

[9] I. Kononenko, "Z razdaljo uteženik k-najbližjih sosedov," v *Strojno učenje, 2 popravljena in dopolnjena izdaja*. Ljubljana, Slovenija: Založba FE FRI, 2005, pogl. 10, str. 293.

[10] Y. Yamada, E. Suzuki, H. Yokoi, and K. Takabayashi, "Decision-tree Induction from Time-series Data Based on a Standard-example Split Test," v *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, Washington DC, 2003.

[11] L. Breiman, J. Friedman, R. Olshen & C. Stone, *Classification and Regression Trees*. Belmont, CA: Chapman & Hall, 1984.

[12] S. K. Murthy, "Automatic Construction of Decision Trees from Data: A Multi-disciplinary Survey," *Data Mining and Knowledge Discovery*, št. 2, str. 345–389, December 1998.

[13] J. R. Quinlan, *C4.5: programs for machine learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[14] I. Kononenko, "Gradnja in uporaba odločitvenega drevesa," v *Strojno učenje, 2 popravljena in dopolnjena izdaja*. Ljubljana, Slovenija: Fakulteta za računalništvo in informatiko, 2005, pogl. 9, str. 250-251.

[15] J. Demsar, B. Zupan, and G. Leban, "Orange: From Experimental Machine Learning to Interactive Data Mining," Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, White Paper (www.ailab.si/orange) 2004.

[16] J. R. Quinlan, "Induction of decision trees," v *Machine Learning 1*. Boston: Kluwer Academic Publishers, 1986, str. 81-106.

[17] I. Kononenko, "Izloči enega," v *Strojno učenje, 2 popravljena in dopolnjena izdaja*. Ljubljana, Slovenija: Založba FE in FRI, 2005, pogl. 3, str. 94.

Izjava o avtorstvu

Izjavljam, da sem diplomsko nalogo izdelal samostojno pod mentorstvom akad. prof. dr. Ivana Bratka. Sodelavce, ki so mi pri delu pomagali, sem navedel v zahvali.

Blaž Strle