Algoritmi poravnavanja vremenskih nizova

Jukić-Bračulj, Mirjana

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Mirjana Jukić-Bračulj

ALGORITMI PORAVNAVANJA VREMENSKIH NIZOVA

Diplomski rad

Voditelj rada:
izv. prof. dr. sc. Luka Grubišić

Zagreb, srpanj, 2017.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Ovaj rad je posvećen mojoj obitelji.

Sadržaj

Uvod

Vremenski niz je niz kronološki uređenih vrijednosti mjerenja neke pojave. Primjeri takvih nizova mogu se pronaći u financijama, prepoznavanju uzoraka, prognozi vremena i potresa, astronomiji i ostalim područjima koja zahtijevaju mjerenja određenih vrijednosti u vremenskim razmacima. Zbog široke primjene vremenskih nizova u praksi potrebne su različite metode za njihovu analizu, uspoređivanje i klasifikaciju.

Najpoznatiji način mjerenja udaljenosti između dva niza je računanjem euklidske udaljenosti. Potrebno je da nizovi budu iste duljine i ukupna udaljenost ovisi o udaljenosti točaka zadanih nizova koje imaju iste indekse. Kod vremenskih nizova udaljenost se može mjeriti na već opisani način, ali za točnije rezultate potrebno je uzeti u obzir varijacije u brzini promjene mjerenih vrijednosti i vremenske pomake. Jednostavan primjer je potpisivanje osobe. Prilikom više različitih davanja potpisa brzina pisanja varira pa pripadni vremenski nizovi nastali bilježenjem koordinata potpisa mogu naizgled biti jako različiti. Zato je potrebno pronaći lokalno slične dijelove i poravnati dobivene nizove tako da se pri mjerenju udaljenosti računa udaljenosti između tih dijelova. Na taj način se dobije udaljenosti koja je bliža čovjekovu intuitivnom načinu uspoređivanja dvaju potpisa. Jedan od najpoznatijih algoritama za poravnanje vremenskih nizova je algoritam dinamičkog poravnavanja vremena.

Algoritam je po prvu puta predstavljen 1978. godine u radu "Dynamic Programming Algorithm Optimization For Spoken Word Recognition" autora Hiroaki Sakoe i Seibi Chiba [12]. Uspješno je riješen problem izgovaranja iste riječi različitom brzinom te je ubrzo našao primjenu i u ostalim područjima usko povezanim s vremenskim nizovima. Pomoću metoda dinamičkog programiranja algoritam računa optimalno poravnanje dvaju vremenskih nizova te udaljenost dobivenu tim poravnanjem. Prilikom računanja zadaju se različita ograničenja na način poravnavanja nizova pa se tako dobivaju različite varijacije algoritma od kojih je potrebno izabrati najbolju za pripadni problem. Algoritam ima kvadratnu složenost što predstavlja problem kod računanja s velikim skupovima podataka. Zbog toga su istražene mogućnosti postavljanja donjih granica čiji izračun je jeftiniji za provođenje, a pomoću tih vrijednosti može se ocijeniti postoji li potreba za računanje algoritma na pripadnim nizovima. Najpoznatije su Keoghova i poboljšana Keoghova donja granica. Za potrebe klasifikacije i grupiranja nizova koristi se metoda pod nazivom bari-

centrično usrednjavanje koja pomoću navedenog algoritma omogućuje dobivanje srednje vrijednosti skupa vremenskih nizova. Ova metoda također ima i efekt smanjenja dimenzije skupa podataka, budući da se u daljnjem izračunu udaljenost novog podatka od cijele klase (ponekad nazvane i klasterom podataka) zamjenjuje računanjem udaljenosti do baricentra.

Cilj ovog rada je pokazati da je vrijeme klasificiranja uzorka uspoređujući ga s baricentrima klasa bitno manje od vremena potrebnog za klasificiranje prilikom kojeg se dani uzorak uspoređuje sa svakim uzorkom iz svih klasa te da korištenjem brže klasifikacije ne dolazi do značajnih gubitaka u točnosti algoritma.

Poglavlje 1

Algoritam dinamičkog poravnavanja vremena

1.1 Vremenski nizovi

Definicija 1.1.1. Vremenski niz je preslikavanje $x\colon [0,T] \to \mathbb{R}^n$, $T \in \mathbb{R}$.

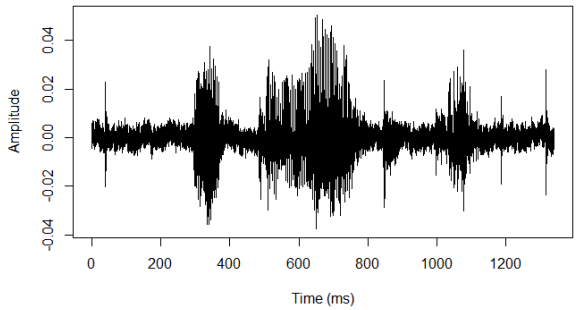
Napomena 1.1.1. Višedimenzionalni vektori nastaju zbog bilježenja vrijednosti različitih parametara modela. Na primjer, vremenski niz nastao bilježenjem korisnikova potpisa sastoji se od dvodimenzionalnih vektora koji predstavljaju koordinate točaka koje pripadaju potpisu. Osim toga moguće je pamtiti i informaciju o tome je li u pojedinoj točki došlo do podizanja ili spuštanja olovke prilikom pisanja. Tada će se vremenski niz sastojati od trodimenzionalnih vektora.
Podatci se najčešće bilježe u jednakim vremenskim razmacima, ali to nije nužno.

Primjeri vremenskih nizova

Vremenski nizovi imaju široku primjenu. U medicini nastaju praćenjem vrijednosti parametara u ljudskom organizmu kroz vrijeme, dok su u financijama potrebni za prognozu dinamike tržišta. Osim što nose informacije o podatcima kroz proteklo vrijeme, korištenjem statističkih metoda, mogu se dobiti predviđanja za budućnost koja služe kao temelj za buduće akcije. Za lakšu analizu vremenskih nizova postoji nekoliko matematičkih alata kao što su *Matlab* [6] i *R* [11] koji olakšavaju postupak analize i grafičkog prikaza podataka.

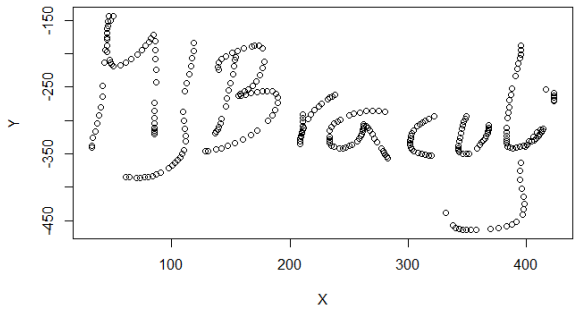
Primjer 1.1.1. *Prva primjena algoritma dinamičkog poravnavanja vremena bila je u prepoznavanju izgovorenih riječi (vidi [12]). Pri izgovaranju riječi nastaje vremenski niz koji se sastoji od amplituda zvuka zabilježenih u jednakim vremenskim razmacima čija veličina*

ovisi o uređaju koji snima zvuk. Prikaz jednog takvog vremenskog niza nastalog izgovaranjem riječi ***matematika*** *nalazi se na slici 1.1.*



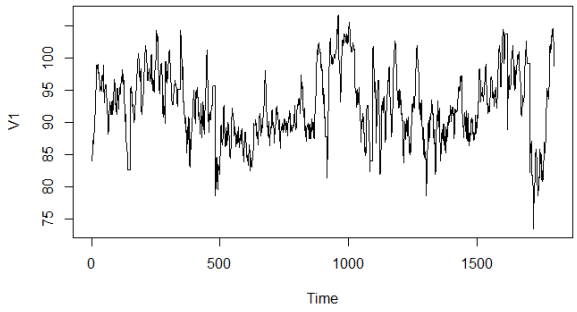
Slika 1.1: Zvučni signal riječi matematika

Primjer 1.1.2. *Tijekom korisnikova potpisivanja prstom na ekranu mobilnog uređaja bilježenjem točaka nastalih tijekom potpisivanja dobije se jedan vremenski niz. Njegov grafički prikaz vidi se na slici 1.2.*



Slika 1.2: Korisnikov potpis prstom na ekranu mobilnog uređaja

Primjer 1.1.3. *Sljedeći vremenski niz dobiven je bilježenjem trenutnog broja otkucaja čovjekova srca svakih 0.5 sekundi. Na slici 1.3 su prikazani dobiveni podatci.*



Slika 1.3: Brzina otkucaja čovjekova srca

1.2 Algoritam dinamičkog poravnavanja vremena

Za zadana dva vremenska niza $X := (x_1, x_2, ..., x_N) \in F^N$ i $Y := (y_1, y_2, ..., y_M) \in F^M$, N i $M \in \mathbb{N}$, potrebno je pronaći njihovu udaljenost. Označimo s $c\colon F \times F \to \mathbb{R}_{\geqslant 0}$ funkciju koja zadovoljava svojstva norme na prostoru F. Neka je $C \in \mathbb{R}^{N \times M}$ definirana kao $C_{n,m} := c(x_n, y_m)$ matrica udaljenosti.

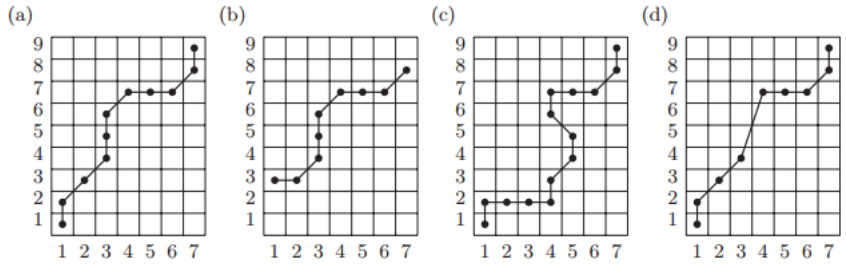
Definicija 1.2.1. *(N, M)-poravnanje nizova* je niz $p = (p_1, p_2, ..., p_L)$ gdje je $p_l = (n_l, m_l) \in \{1, ..., N\} \times \{1, ..., M\}$, $l \in \{1, ..., L\}$, koji zadovoljava sljedeće uvjete:

1. Rubni uvjeti: $p_1 = (1, 1)$ i $p_L = (N, M)$
2. Monotonost: $n_1 \leqslant n_2 \leqslant ... \leqslant n_L$ i $m_1 \leqslant m_2 \leqslant ... \leqslant m_L$
3. Pomak: $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ za $l \in \{1, ..., L - 1\}$.

Poravnanje nizova pridružuje svakom elementu $x_{n_l} \in X$ barem jedan element $y_{m_l} \in Y$ i obratno. Prvi uvjet definicije osigurava da poravnanje uzme u obzir cijele nizove odnosno da je prvom (zadnjem) elementu niza X pridružen prvi (zadnji) element niza Y. Zbog monotonosti poravnanja ne može se dogoditi "povratak u prošlost". Dakle, ako je i-tom elementu niza X pridružen j-ti element niza Y onda znamo da bilo kojem elementu iz niza X s indeksom većim od i ne može biti pridružen element niza Y s indeksom manjim od j. Treći uvjet sprječava da element niza bude izostavljen iz poravnanja. Na slici 1.4 nalazi se ilustracija uvjeta iz definicije 1.2.1.

Neka je p zadano poravnanje. S $c_p(X, Y)$ označimo udaljenost nizova X i Y dobivenu iz poravnanja p. Tada vrijedi sljedeća jednakost:

$$c_p(X, Y) := \sum_{l=1}^{L} c(x_{n_l}, y_{m_l}). \tag{1.1}$$



Slika 1.4: Primjeri poravnanja nizova X duljine 7 i Y duljine 9. **(a)** Poravnanje zadovoljava 1., 2. i 3. uvjet iz definicije 1.2.1. **(b)** Poravnanje ne zadovoljava rubne uvjete. **(c)** Monotonost poravnanja je narušena. **(d)** 3. uvjet iz definicije 1.2.1 nije ispunjen.

Optimalno poravnanje nizova X i Y, označeno s p_{min}, je poravnanje koje daje najmanju udaljenost nizova. Označimo s $DTW(X, Y)$ udaljenost nizova dobivenu tim poravnanjem.

$$DTW(X, Y) := c_{p_{min}}(X, Y) = \min\{c_p(X, Y) \mid \text{p je (N, M)-poravnanje nizova}\} \tag{1.2}$$

Lema 1.2.1. *DTW funkcija nije pozitivno definitna.*

Dokaz. Za nizove $X := \{x_1, x_2\}$ i $Y := \{x_1, x_1, x_2, x_2, x_2\}$ vrijedi $DTW(X, Y) = 0$ budući je $c(x_i, x_i) = 0$, $i = 1, 2$. □

Lema 1.2.2. *DTW funkcija ne zadovoljava nejednakost trokuta.*

Dokaz. Neka je $F = \{\alpha, \beta, \gamma\}$. Definiramo normu $c\colon F \times F \to \{0, 1\}$ s $c(x, y) := 0$ ako $x = y$ i $c(x, y) := 1$ ako $x \neq y$, $x, y \in F$. Za nizove $X = (\alpha, \beta, \gamma)$, $Y = (\alpha, \beta, \beta, \gamma)$ i $Z = (\alpha, \gamma, \gamma)$ dobijemo $DTW(X, Y) = 0$, $DTW(X, Z) = 1$ te $DTW(Y, Z) = 2$. Ne vrijedi nejednakost trokuta $DTW(Y, Z) < DTW(Y, X) + DTW(X, Z)$. □

Označimo s $D \in F^{N \times M}$ *matricu sumiranih udaljenosti* optimalnog poravnanja. Vrijedi $D_{n,m} = DTW(X, Y)$.

Teorem 1.2.1. *Matrica sumiranih udaljenosti D zadovoljava sljedeće jednakosti:*

1.

$$D_{n,1} = \sum_{k=1}^{n} c(x_k, y_1), \ za\ n \in \{1, ..., N\} \tag{1.3}$$

$$D_{1,m} = \sum_{k=1}^{m} c(x_1, y_k), \ za\ m \in \{1, ..., M\} \tag{1.4}$$

2.

$$D_{n,m} = \min\{D_{n-1,m-1}, D_{n-1,m}, D_{n,m-1}\} + c(x_n, y_m) \quad za\ n \in \{2, ..., N\}\ i\ m \in \{2, ..., M\} \tag{1.5}$$

Dokaz. Jednakosti 1.3 i 1.4 trivijalno slijede iz uvjeta definicije poravnanja dvaju nizova 1.2.1. Za $n > 1$ i $m > 1$ neka je $q = (q_1, q_2, ..., q_{L-1}, q_L)$ optimalno poravnanje za X i Y. Iz rubnog uvjeta znamo da je $q_L = (n, m)$. Neka je $(i, j) := q_{L-1}$. Zbog uvjeta na pomak poravnanja znamo da je $(i, j) \in \{(n-1, m-1), (n-1, m), (n, m-1)\}$. Označimo s $N[s : t]$, $s, t \in \mathbb{N}$, $s \leq t$, podniz proizvoljnog niza N, $(N_s, N_{s+1}, ..., N_t)$. Vrijedi da je $(q_1, ..., q_{L-1})$ optimalno poravnanje nizova $X[1 : a]$ i $Y[1 : b]$ zato što je q optimalno poravnanje nizova X i Y. Iz navedenih tvrdnji slijedi 1.7. □

Iz prethodnog teorema slijedi rekurzivna relacija na kojoj se temelji algoritam.

$$D_{i,j} = c(i, j) + \min\{D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}\} \tag{1.6}$$

Inicijalizacija matrice D može se pojednostavniti dodavanjem novog retka i stupca čiji se elementi inicijaliziraju na $+\infty$. Formalno, $D_{n,0} := +\infty$ za $n \in \{1, ..., N\}$, $D_{0,m} := +\infty$ za $m \in \{1, .., M\}$ i $D_{0,0} := 0$. Osim toga može se primijetiti da je za računanje DTW udaljenosti dovoljan prostor za pohranu veličine $O(N)$ ($O(M)$) jer se matrica može popunjavati po recima (stupcima). Za dobivanja poravnanja potrebno je pamtiti cijelu matricu D pa je stoga prostorna složenost veličine $O(N \times M)$.

Pri implementaciji algoritma koriste se metode dinamičkog programiranja. U nastavku slijedi implementacija osnovnog oblika algoritma.

```
Ulaz: S niz duljine N, Q niz duljine M.
Izlaz: DTW udaljenost.

1     for i := 1 to N
2        DTW[i, 0] := +∞
3     for i := 1 to M
4         DTW[0, i] := +∞
5
6     DTW[0, 0] := 0
7
8     for i := 1 to N
9        for j := 1 to M
10           cost := d(S[i], Q[j])
11           DTW[i, j] := cost + minimum(DTW[i-1, j  ],
12                                       DTW[i  , j-1],
13                                       DTW[i-1, j-1])
14    return DTW[N, M]
```

Nameće se pitanje kako iz matrice akumuliranih vrijednosti D dobiti optimalno poravnanje $p_{min} = \{p_1, .., p_l\}$. Ako je zadana matrica D do poravnanja se dolazi obrnutim

redoslijedom od dobivanja matrice. Pretpostavimo da je $p_l = (n, m)$. Ako je $(n, m) = (1, 1)$ onda je $l = 1$ pa smo gotovi. Inače vrijedi rekurzivna formula

$$p_{l-1} := \begin{cases} (1, m-1), & \text{ako je } n = 1 \\ (n-1, 1), & \text{ako je } m = 1 \\ \text{argmin}\{D_{n-1,m-1}, D_{n-1,m}, D_{n,m-1}\}, & \text{inače.} \end{cases} \tag{1.7}$$

Vremenska složenost algoritma je $O(N^2)$. Zbog toga postoje različiti prijedlozi za ubrzanja algoritma. Neke od njih navodimo u sljedećem odjeljku.

1.3 Varijacije algoritma dinamičkog poravnavanja vremena

Veličina koraka

Pomak opisan u definiciji 1.2.1 osigurava da je u poravnanju nizova X i Y svakom elementu prvog niza pridružen element iz drugog niza i obratno. Mana tog uvjeta je da jednom elementu niza može biti pridružen veliki broj elemenata iz drugog niza pa poravnanje ima vertikalne odnosno horizontalne dijelove. To znači da će se poravnanje prilagoditi nastaloj lokalnoj jednoličnosti jednog od nizova i njoj pridružiti jedan element iz drugog niza.

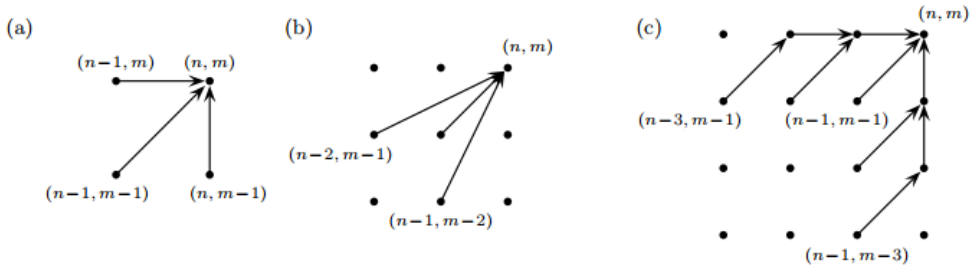
Kako bi se izbjegli takvi dijelovi potrebno je promijeniti veličinu pomaka. Jedna opcija je da 3. uvjet iz definicije 1.2.1 zamijenimo s $p_{l+1} - p_l \in \{(2, 1), (1, 2), (1, 1)\}$ za $l \in \{1, ..., L\}$. Element matrice akumuliranih vrijednosti se sada računaju po formuli

$$D_{n,m} = \min\{D_{n-1,m-1}, D_{n-2,m-1}, D_{n-1,m-2}\} + c(x_n, y_m) \tag{1.8}$$

za $n \in \{2, ..., N\}$ i $m \in \{2, ..., M\}$. Početne vrijednosti su $D_{0,0} := 0$, $D_{1,1} := c(x_1, y_1)$, $D_{n,0} := +\infty$ za $n \in \{1, ..., N\}$, $D_{n,1} := +\infty$ za $n \in \{2, ..., N\}$, $D_{0,m} := +\infty$ za $m \in \{1, ..., M\}$ i $D_{1,m} := +\infty$ za $m \in \{2, ..., M\}$.

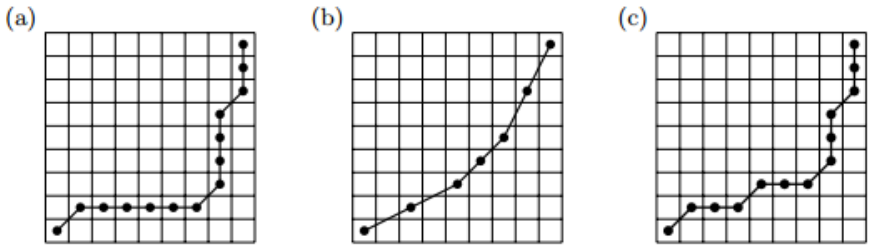
Navedeni uvjet ne osigurava da će svakom članu prvog niza biti pridružen barem jedan član drugog niza i obrnuto. Jedan takav primjer poravnanja može se vidjeti na 1.6b). Zato je potrebno modificirati rekurzivnu formulu tako da svi elementi nizova X i Y utječu na konačnu sumu udaljenosti. Dobije se sljedeća formula

$$D_{n,m} = min \begin{cases} D_{n-1,m-1} + c(x_n, y_m) \\ D_{n-2,m-1} + c(x_{n-1}, y_m) + c(x_n, y_m) \\ D_{n-1,m-2} + c(x_n, y_{m-1}) + c(x_n, y_m) \\ D_{n-3,m-1} + c(x_{n-2}, y_m) + c(x_{n-1}, y_m) + c(x_n, y_m) \\ D_{n-1,m-3} + c(x_n, y_{m-2}) + c(x_n, y_{m-1}) + c(x_n, y_m). \end{cases} \tag{1.9}$$



Slika 1.5: Prikaz tri različita načina pomaka pri traženju optimalnog poravnanja. **(a)** Odgovara pomaku iz definicije 1.2.1 **(b)** Prikaz poravnanja koje odgovara rekurzivnoj formuli 1.8 **(c)** Prikaz poravnanja koje odgovara rekurzivnoj formuli 1.10

za $(n, m) \in \{1, ..., N\} \times \{1, ..., M\}$. Sada su početne vrijednosti sljedeće: $D_{1,1} := c(x_1, y_1)$, $D_{n,-2} := D_{n,-1} := D_{n,0} := +\infty$ za $n \in \{-2, ..., N\}$ i $D_{-2,m} := D_{-1,m} := D_{0,m} := +\infty$ za $m \in \{-2, ..., M\}$



Slika 1.6: Prikaz poravnanja dobivenih primjenom različitih, prethodno opisanih rekurzivnih funkcija. **(a)** Poravnanje dobiveno korištenjem rekurzije s pomakom iz definicije 1.2.1 **(b)** Poravnanja dobiveno korištenjem rekurzivne formule 1.8 **(c)** Poravnanje dobiveno korištenjem rekurzivne formule 1.10

Lokalne težine

Ako preferiramo pojedini smjer u kreiranju poravnanja više od nekog drugog možemo mu pridružiti veću težinu uvođenjem vektora težina $(w_d, w_h, w_v) \in \mathbb{R}^3$. Tako dobivamo sljedeću

rekurziju

$$D_{n,m} = min \begin{cases} D_{n-1,m-1} + w_d c(x_n, y_m) \\ D_{n-1,m} + w_h c(x_n, y_m) \\ D_{n,m-1} + w_v c(x_n, y_m) \end{cases} \tag{1.10}$$

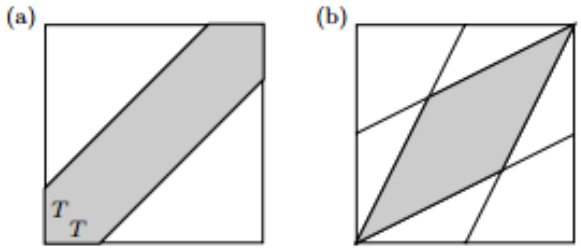
za $n \in \{2, ..., N\}$ i $m \in \{2, ..., M\}$. Nadalje, vrijedi da je $D_{n,1} := \sum_{k=1}^{n} w_h c(x_k, y_1)$ za $n > 1$, $D_{1,m} := \sum_{k=1}^{m} w_v c(x_1, y_k)$ za $m > 1$ te $D_{1,1} := c(x_1, y_1)$. Osnovna rekurzija algoritma poravnavanja vremena ima vektor težina $(1, 1, 1)$. Na taj način se preferira dijagonalni pomak u kreiranju poravnanja jer jedan dijagonalni korak odgovara jednom koraku u lijevo pa gore (ili jednom koraku gore pa lijevo). Da bismo ujednačili sve poteze potrebno je da vektor težina bude $(2, 1, 1)$.

Globalna ograničenja

Za eliminaciju pojedinih poravnanja i smanjenje broja računskih operacija potrebno je uvesti globalne konstante. Tako se smanjuje složenost algoritma i izbacuju rubni slučajevi poravnanja koji u većini slučajeva nisu dobra rješenja. Formalno, neka je $R \subseteq \{1, ..., N\} \times \{1, ..., M\}$ podskup koji odgovara globalnom ograničenju. Kažemo da poravnanje p_R^* *pripada* podskupu R ako svaki njegov element pripada podskupu R. Dva najpoznatija ograničenja na domenu poravnanja su *Sakoe-Chiba površina* i *Itakura paralelogram*.

Sakoe-Chiba površina ima konstantnu širinu T i vrijedi da elementu x_n može biti pridružen y_m za $m \in \{\frac{M-T}{N-T}(n-T), ..., \frac{M-T}{N-T}(n+T)\} \cap \{1, ..., M\}$. Dobivena površina je simetrična s obzirom na dijagonalu pravokutne mreže $N \times M$ kao na slici 1.7.

Itakura paralelogram za $S \in \mathbb{R}_{>0}$ definira R kao skup čiji elementi na pravokutnoj mreži $N \times M$ čine paralelogram definiran pravcima s nagibom $\frac{1}{S}$ i S.



Slika 1.7: Prikaz osnovnih oblika globalnih ograničenja na poravnanje. **(a)** *Sakoe-Chiba površina* **(b)** *Itakura paralelogram*

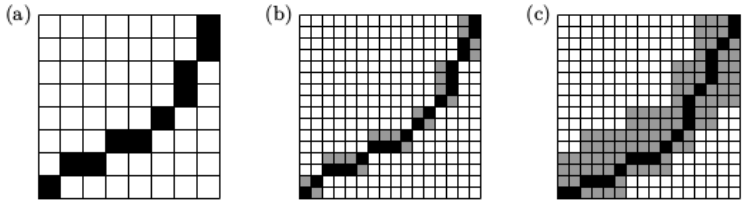
Za primjenu globalnih ograničena u računanju ukupne udaljenosti dovoljno je definirati da je $c(x_n, y_m) := +\infty$ za $(n, m) \in \{1, ..., N\} \times \{1, ..., M\} \setminus R$. Na taj način pri računanju

udaljenosti nizova potrebno je računati samo one elemente matrice D koji pripadaju skupu R što kod korištenje *Sakoe-Chiba površine*, kada je $T \ll N$, može dovesti do velikog ubrzanja algoritma.

Glavni nedostatak uvođenja globalnih ograničenja pri izvođenju algoritma je u tome što optimalno poravnanje p^* ne mora nužno pripadati podskupu R pa optimalno poravnanje dobiveno algoritmom uz ograničenje p_R^* neće biti jednako optimalnom poravnanju p^*.

Višeslojni algoritam poravnavanja vremena

Neka su zadana dva niza X i Y s pripadnim duljinama $N_1 := N$ i $M_1 := M$. Polazni nizovi se zbog potreba ubrzanja algoritma generaliziraju tako da se njihov broj elemenata smanji za određeni faktor f_2. Dobiju se nizovi s duljinama $N_2 := N_1/f_2$ i $M_2 := M_2/f_2$ (pretpostavljamo da f_2 dijeli N_1 i M_1). Računa se optimalno poravnanje p_2^* na dobivenim nizovima. Pomoću tako dobivenog poravnanja i faktora f_2 dobijemo podskup R pravokutne mreže $N_1 \times M_1$ kao u primjeru na slici 1.8. Tek sada se računa optimalno poravnanje p_R^* na potpunom sloju podataka koje pripada dobivenom podskupu R. Uspješnost postupka ocjenjuje se prema jednakosti dobivenog optimalnog poravnanja p_R^* i optimalnog poravnanja bez ograničenja p^*.



Slika 1.8: Postupak generalizacije podataka u svrhu ubrzanja algoritma **(a)** Poravnanje dobiveno na generaliziranom skupu podataka **(b)** Postupak dobivanja podskupa R koji predstavlja globalno ograničenje na poravnanje polaznih nizova.**(c)** Prošireni podskup R^δ

Neka L_2 označava duljinu poravnanja p_2^*. Tada je podskup R veličine $L_2 \times f_2^2$. Za računanje optimalnog poravnanja treba ukupno $N_2 \times M_2 + L_2 \times f_2^2$ što je znatno manje nego $N_1 \times M_1$.

Zbog mogućnosti da optimalno poravnanje p^* ne pripada dobivenom podskupu R može ga se proširiti koeficijentom $\delta \in \mathbb{N}$ tako da se za svaki element u R doda δ elemenata koji su na mreži pozicionirani gore, dolje, lijevo i desno od polaznog elementa.

Poglavlje 2

Metode ubrzanja algoritma dinamičkog poravnavanja vremena

Algoritam dinamičkog poravnavanja vremena je jedan način određivanja mnogo prema mnogo poravnanja dvaju nizova uz ograničenje na monotonost poravnanja. Neka je $NDTW$ oznaka za algoritam koji ispušta uvjet monotonosti. Vrijedi tvrdnja da je njegova složenost u slučaju jednodimenzionalnih nizova $O(n \log n)$, a ako su sortirani onda je linearna (dokaz se može pronaći u [2]). Označimo s DTW_p ($NDTW_p$) algoritam dinamičkog poravnavanja vremena (s ispuštanjem uvjeta monotonosti) koji koristi p-normu za računanje udaljenosti nizova. Vrijedi sljedeća rekurzivna formula

$$D_{i,j} := \begin{cases} 0, & \text{ako je } i = j = 0 \\ \infty, & \text{ako je } i = 0 \oplus j = 0 \\ & \text{ili } |x_i - y_j| > \omega \\ |x_i - y_j|^p + & \\ \min\{D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}\}, & \text{inače.} \end{cases} \tag{2.1}$$

pri čemu je ω globalno ograničenje na poravnanje, a $\oplus$ označava operaciju *ekskluzivno ili*. Formula vrijedi za $0 < p < \infty$. Za $p = \infty$ početni uvjeti ostaju kao prethodno definirani, a rekurzija je sljedeća

$$D_{i,j} = \max\{|x_i - y_j|, \min\{D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}\}\}. \tag{2.2}$$

Za zadane nizove X i Y duljina redom M, N računamo udaljenost nizova na način $DTW_p(X, Y) = \sqrt[p]{D_{M,N}}$
Očito je da vrijedi sljedeća nejednakost

$$NDTW_p(X, Y) \leq DTW(X, Y) \leq \|X - Y\|_p \text{, za sve } 0 < p \leq \infty. \tag{2.3}$$

Neka je A proizvoljan niz iz $\mathbb{R}^n$ i $c \in \mathbb{R}$. Definiramo relaciju "$\leq$" na $\mathbb{R}^n \times \mathbb{R} \cup \mathbb{R} \times \mathbb{R}^n$, $n \in \mathbb{N}$ na sljedeći način:

$$\begin{aligned} A \leq c \text{ ako i samo ako } \forall i \in \{1, ..., n\}\ A_i \leq c \\ c \leq A \text{ ako i samo ako } \forall i \in \{1, ..., n\}\ c \leq A_i. \end{aligned} \tag{2.4}$$

Propozicija 2.0.1. *Neka su X i Y jednodimenzionalni vremenski nizovi realnih brojeva takvi da postoji $c \in \mathbb{R}$ takav da vrijedi $X \geq c \geq Y$ ili $X \leq c \leq Y$ tada vrijedi $DTW_1(X, Y) = \|X - Y\|_1$*

Dokaz. Bez smanjenja općenitosti možemo pretpostaviti da je $X \geq c \geq Y$. Pretpostavimo da su X' i Y' dva jednodimenzionalna vremenska niza konstruirana iz poravnanja dobivenog $NDTW_1$ algoritmom tako da za svaki element poravnanja proširimo X' i Y' s elementima iz nizova X i Y koji se nalaze na koordinatama određenim trenutnim elementom poravnanja. Tada vrijedi $NDTW_1(X, Y) = \left\|X' - Y'\right\|_1$. Jasno je da vrijedi $X' \geq c \geq Y'$ zato što su elementi od X' i Y' ujedino i elementi nizova X i Y. Vrijedi sljedeća nejednakost

$$\begin{aligned} NDTW_1(X, Y) &= \sum_i |x_i' - y_i'| = \sum_i (|x_i' - c| + |c - y_i'|) \\ &= \left\|X' - c\right\|_1 + \left\|c - Y'\right\|_1 \\ &\geq \|X - c\|_1 + \|c - Y\|_1 \\ &= \|X - Y\|_1 . \end{aligned} \tag{2.5}$$

Korištenjem nejednakosti 2.3 dobijemo tvrdnju teorema. □

Napomena 2.0.1. Tvrdnja propozicije 2.0.1 ne vrijedi za $p > 1$. Primjerice, za nizove $X = (0, 0, 1, 0)$ i $Y = (2, 3, 2, 2)$ vrijedi $DTW_2(X, Y) = \sqrt{17}$ dok je $\|X - Y\|_2 = \sqrt{18}$.

Definicija 2.0.1. Neka je $\omega \in \mathbb{N}$ globalno ograničenje i X vremenski niz. Neka su $U(X)_i = \max_k\{x_k \mid |k - i| \leq \omega\}$ i $L(X)_i = \min_k\{x_k \mid |k - i| \leq \omega\}$ za $i = 1, ..., n$. Nizovi $U(X)$ i $L(X)$ čine *omotač* oko niza X.

Lema 2.0.1. *Za $b \in [a, c]$ vrijedi $(c - a)^p \geq (c - b)^p + (b - a)^p$ za $1 \leq p < \infty$*

Dokaz. Za $p = 1$ tvrdnja očito vrijedi. Za $p > 1$ deriviranjem funkcije $f(b) = (c - b)^p + (b - a)^p$ dobije se da funkcija postiže minimum u točki $b = \frac{c+a}{2}$. Na rubovima intervala za $b \in \{a, c\}$ funkcija postiže maksimum iz čega slijedi tvrdnja. □

Teorem 2.0.1. *Neka su zadana dva vremenska niza jednake duljine X i Y i neka je $1 \leq p < \infty$. Za proizvoljan vremenski niz H takav da je $x_i \geq h_i \geq U(Y)_i$ ili $x_i \leq h_i \leq L(Y)_i$ ili $h_i = x_i$ za sve indekse i vrijedi sljedeća nejednakost*

$$DTW_p(X, Y)^p \geq NDTW_p(X, Y)^p \geq \|X - H\|_p^p + NDTW_p(H, Y)^p. \tag{2.6}$$

Za $p = \infty$ vrijedi $DTW_\infty(X,Y) \geq NDTW_\infty(X,Y) \geq max(\|X-H\|_\infty, NDTW_\infty(H,Y))$.

Dokaz. Pretpostavimo da je $1 \leq p < \infty$. Označimo s $\mathcal{P}$ poravnanje određeno $NDTW$ algoritmom odnosno vrijedi $NDTW_p(X,Y)^p = \sum_{(i,j)\in\mathcal{P}} |x_i - y_i|^p$. Za svaki i vrijedi da je $h_i \in [\min(x_i, y_i), \max(x_i, y_i)]$ pa korištenjem tvrdnje iz Leme 2.0.1 vrijedi $|x_i - y_i|^p \geq |x_i - h_i|^p + |h_i - y_j|^p$ za proizvoljan $(i,j) \in \mathcal{P}$. Iz toga slijedi nejednakost

$$NDTW_p(X,Y)^p \geq \sum_{(i,j)\in\mathcal{P}} (|x_i - h_i|^p + |h_i - y_i|^p) \geq \|X-H\|_p^p + \sum_{(i,j)\in\mathcal{P}} |h_i - y_j|^p. \tag{2.7}$$

Tvrdnja teorema slijedi iz prethodne tvrdnje i nejednakosti $\sum_{(i,j)\in\mathcal{P}} |h_i - y_j|^p \geq NDTW_p(H,Y)$. Za $p = \infty$ vrijedi nejednakost

$$\begin{aligned} NDTW_\infty(X,Y) = \max_{(i,j)\in\mathcal{P}} |x_i - y_j| &\geq \max_{(i,j)\in\mathcal{P}} \max(|x_i, h_i|, |h_i, y_j|) \\ &\geq \max(\|X-H\|_\infty, NDTW_\infty(H,Y)). \end{aligned} \tag{2.8}$$

□

Korolar 2.0.1. *Neka su zadana dva vremenska niza X i Y jednake duljine. $\|X-H\|_p$ aproksimira $DTW_p(X,Y)$ i $NDTW_p(X,Y)$ s greškom manjom ili jednakom $\|H-Y\|_p$.*

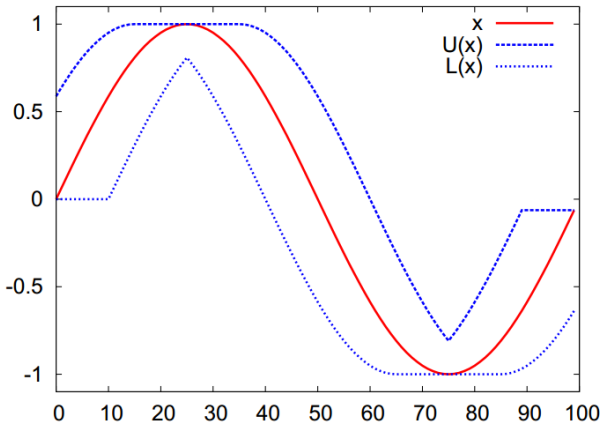
Dokaz. Zbog nejednakosti trokuta $\|X-H\|_p + \|H-Y\|_p \geq \|X-Y\|_p$ i $\|X-Y\|_p \geq DTW(X,Y)$ slijedi da je $\|X-H\|_p + \|H-Y\|_p \geq DTW_p(X,Y)$ odnosno $\|H-Y\|_p \geq DTW_p(X,Y) - \|X-H\|_p$. □

U nastavku slijedi algoritam za pronalaženje nizova $U(Y)$ i $L(Y)$ koji zahtjeva najviše $3 \times n$ usporedbi gdje je n duljina niza Y.

Ulaz: $Y = (y_1, ..., y_n)$, ω globalna ograda DTW algoritma
Izlaz: nizovi U(Y) i V(Y)

```
1     u = queue(), l = queue()
2     u.append(1), l.append(1)
3     for i ∈ {2,...,n} do
4         if i ≥ ω+1 then
5             U_{i-ω} = y_{front(u)}, L_{i-ω}=y_{front(l)}
6         if y_i ≥ y_{i-1} then
7             u.pop_from_back()
8             while y_i > y_{back(u)} do
9                 u.pop_from_back()
10        else
11            l.pop_from_back()
12            while y_i < y_{back(l)} do
13                l.pop_from_back()
14        u.append(i), l.append(i)
```

```
15          if i = 2ω +1 + front(u) then
16              u.pop_from_front()
17          else if i = 2ω+1+front(l) then
18              l.pop_from_front()
19      for i ∈ {n+1,..., n+ω} do
20          U_{i−ω} = y_{front(u)}, L_{i−ω} = y_{front(l)}
21          if i−front(u) ≥ 2ω +1 then
22              u.pop_from_front()
23          if i−front(l) ≥ 2ω+1 then
24              l.pop_from_front()
```



Slika 2.1: Grafički prikaz nizova $U(X)$ i $L(X)$

2.1 Keoghova donja granica za DTW algoritam

Neka su X i Y vremenski nizovi i neka je $H(X, Y)$ vremenski niz definiran na sljedeći način

$$H(X,Y)_i = \begin{cases} U(Y)_i, & \text{, ako je } x_i \geq U(Y)_i \\ L(Y)_i, & \text{, ako je } x_i \leq L(Y)_i \\ x_i, & \text{inače} \end{cases} \tag{2.9}$$

za $i = 1, ..., n$. H predstavlja *projekciju niza X na niz Y* i nalazi se u omotaču oko niza Y. Korištenjem tvrdnji iz teorema 2.0.1 dobijemo sljedeću nejednakost

$$NDTW_p(X,Y)^p \geq ||X - H(X,Y)||_p^p + NDTW_p(H(X,Y),Y)^p \text{ za } 1 \leq p < \infty. \tag{2.10}$$

Definiramo $LB\_Keogh(X,Y) := \|X - H(X,Y)\|_p$. Iz Teorema 2.0.1 i prethodnih tvrdnji proizlazi sljedeći korolar.

Korolar 2.1.1. *Neka su zadana dva vremenska niza jednake duljine X i Y i neka je* $1 \leq p < \infty$. *Tada vrijedi*

1. *LB_Keogh je donja granica za DTW algoritam*

$$DTW_p(X,Y) \geq NDTW_p(X,Y) \geq LB\_Keogh_p(X,Y). \tag{2.11}$$

2. *Točnost donje granice je određena širinom omotača niza Y*

$$DTW_p(X,Y) - LB\_Keogh_p(X,Y) \leq \left\| \max_i \{U(Y)_i - y_i, y_i - L(Y)_i\} \right\|_p . \tag{2.12}$$

Pomoću sljedećeg algoritma za zadani niz Y moguće je pronaći niz iz S koji je najmanje udaljen od njega. Računa se donja granica te ako je ona dovoljno velika onda se preskače računanje DTW algoritma. U ovom primjeru korištena je l_1 norma i prethodno definirani nizovi $U(Y)$ i $L(Y)$

Ulaz: $Y = (y_1, ..., y_n)$, S skup vremenskih nizova
Izlaz: niz X_min koji ima najmanju udaljenost s obzirom na niz Y

```
1   min_distance = ∞
2   U, L = envelope(Y)
3   for X ∈ S do
4     distance = 0
5     for i ∈ {1, 2..., n} do
6         if x_i > U_i then
7             distance = distance + x_i − U_i
8         else if x_i < L_i then
9             distance = distance + L_i − x_i
10    if distance < min_distance then
11        t = DTW_1(X, Y)
12        if t < min_distance then
13            min_distance = t
14            X_min = X
```

2.2 Poboljšana Keoghova donja granica za DTW algoritam

Poznato nam je da vrijedi sljedeća nejednakost

$$NDTW_p(X,Y)^p \geq LB\_Keogh_p(X,Y)^p + NDTW_p(H(X,Y),Y)^p \text{ , za } 1 \leq p < \infty. \tag{2.13}$$

Osim toga vrijedi

$$NDTW_p(H(X,Y),Y) \geq LB\_Keogh_p(Y,H(X,Y)). \tag{2.14}$$

Iz prethodnih nejednakosti slijedi

$$NDTW_p(X,Y)^p \geq LB\_Keogh_p(X,Y)^p + LB\_Keogh_p(Y,H(X,Y))^p \text{ , za } 1 \leq p < \infty. \tag{2.15}$$

Ovime smo dobili još jednu donju granicu za DTW algoritam. Definiramo

$$LB\_Improved_p(X,Y)^p := LB\_Keogh_p(X,Y)^p + LB\_Keogh_p(Y,H(X,Y))^p. \tag{2.16}$$

Sljedeći algoritam koristi prethodno definiranu donju granicu za DTW algoritam kako bi dodatno ubrzao. Prvo se računa donja granica *LB_Keogh* za zadani niz Y i trenutno izabrani X iz zadanog skupa. Ako je ona dovoljno velika onda se uzima novi X, inače se računa *LB_Improved* granica i provjerava se je li moguće pomoću te granice odbaciti X, ako nije onda se računa cijeli DTW algoritam. I u ovom primjeru korištena je l_1 norma i prethodno definirani nizovi $U(Y)$ i $L(Y)$

Ulaz: $Y = (y_1, ..., y_n)$, S skup vremenskih nizova
Izlaz: niz X_min koji ima najmanju udaljenost s obzirom na niz Y

```
1   min_distance = ∞
2   U, L = envelope(Y)
3   for X ∈ S do
4       X' = X
5       distance = 0
6       for i ∈ {1, ..., n} do
7           if x_i > U_i then
8               distance = distance + x_i − U_i
9               x'_i = U_i
10          else if x_i < L_i then
11              distance = distance + L_i − x_i
12              x'_i = L_i
13      if distance < min_distance
14          U', L' = envelope(X')
15          for i ∈ {1, 2, ..., n} do
16              if y_i > U'_i then
17                  distance = distance + y_i − U'_i
18              else if y_i < L'_i then
19                  distance = distance + L'_i − y_i
20          if distance < min_distance then
21              t = DTW_1(X, Y)
22              if t < min_distance then
23                  min_distance = t
24                  X_min = X
```

2.3 Metoda *Podijeli pa vladaj*

Još jedan oblik računanja poravnanja dva vremenska niza može se dobiti korištenjem metode *Podijeli pa vladaj*. Algoritam je predstavio Dan Hirschberg 1975. godine u radu [4]. Korištenjem dinamičkog programiranja algoritam je tražio najdulji zajednički podniz dvaju nizova znakova. Vremenska i prostorna složenost algoritma su redom $O(m \times n)$ i $O(n + m)$ pri čemu su m i n duljine zadanih nizova.

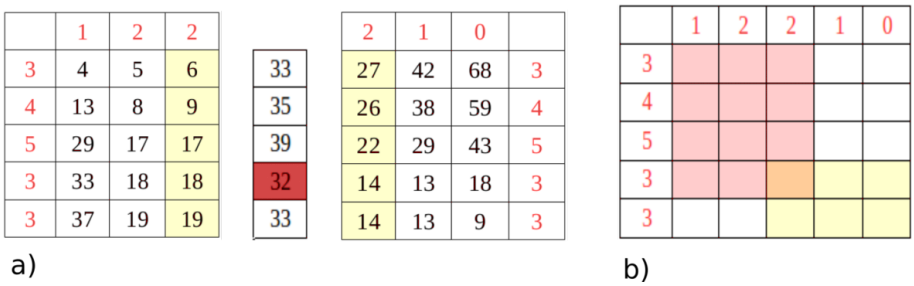
Navedeni algoritam može se upotrebljavati i za dobivanje optimalnog poravnanja dva vremenska niza X i Y duljine m i n. Algoritam je rekurzivan i sastoji se od dva glavna koraka. U prvom koraku potrebno je izračunati tablice sumiranih vrijednosti za niz X i prvu odnosno drugu polovicu niza Y. Kod računanja tablice za drugu polovicu niza Y ishodište je polje (m, n). Za dobivanje elementa konačnog poravnanja potrebno je još zbrojiti stupce tablica koji pripadaju središnjem elementu niza Y kao na slici 2.2 i naći indeks retka s minimalnom sumom. Drugi korak, pomoću dobivene točke iz prvog koraka, dijeli nizove na dva dijela i na svakom od njih ponavlja isti postupak. Slijedi pseudokod opisanog algoritma.

```
Ulaz: vremenski nizovi X i Y
Izlaz: DTW poravnanje preko globalnog polja P

1  Divide_and_conquer(X, Y){
2      m = |X|
3      n = |Y|
4      middle = ⌈n/2⌉
5      if n ≤ 2 or m ≤ 2 then
6          tmp = compute_DTW_alignment(X, Y)
7          add tmp to global array P
8      else
9          left = alignment(X, Y[1:middle])
10         right = alignment(X, Y[middle:n])
11         min_index = 0
12         for i ∈ {1,...,n}
13             if left(i, middle)+right(i, middle) <
14                 left(min_index, middle)+right(mid_index, middle)
15                     min_index = i
16         add (min_index) to global array P
17         Divide_and_conquer(X[1:min_index], Y[1:middle])
18         Divide_and_conquer(X[min_index, m], Y[middle, n])
19 }
```

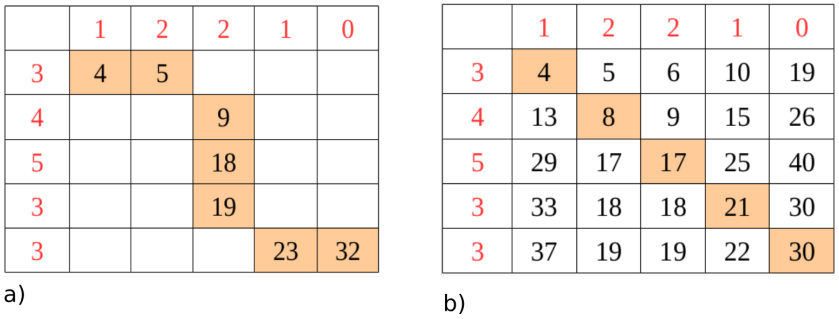
U sljedećem primjeru prikazana su dva osnovna koraka algoritma i pokazano je da algoritam ne daje uvijek optimalno poravnanje.

Primjer 2.3.1. *Neka su zadani nizovi X=1, 2, 2, 1, 0 i Y = 3, 4, 5, 3, 3. Slika 2.2 ilustrira prvi i drugi korak prethodno navedenog algoritma na početnim nizovima.*



Slika 2.2: **a)** Prikaz tablica sumiranih vrijednosti niza *X* i prve odnosno druge polovice niza *Y* i središnjeg stupca dobivenog sumiranjem zadnjeg stupca prve tablice i prvog stupca druge tablice **b)** Drugi korak *Podijeli pa vladaj* algoritma u kojem se početni problem pomoću točke iz prvog koraka dijeli na dva istovrsna podproblema

Rekurzivnim računanjem poravnanja na podnizovima dobivenim podjelom prema indeksu retka minimalne sume unutrašnjih stupaca lijeve i desne tablice dobije se poravnanje koje daje udaljenost 14 dok je rezultat DTW algoritma optimalno poravnanje s udaljenosti 12. Dakle, ovaj algoritam ne daje nužno optimalno poravnanje. Usporedba poravnanja vidi se na slici 2.3



Slika 2.3: **a)** Poravnanje dobiveno algoritmom *Podijeli pa vladaj* **b)** Poravnanje dobiveno *DTW algoritmom*

2.4 Reducirani DTW algoritam

Još jedan način razmišljanja o ubrzanju DTW algoritma temelji se na ideji da se računaju samo ona polja matrice čiju je vrijednost nužno znati da bi se izračunalo optimalno poravnanje. Postavlja se pitanje kako odrediti koja polja matrice će biti popunjena, a koji elementi su dovoljno različiti da njihovu udaljenost nije potrebno računati. Temelji ovog

pristupa su normalizacija zadanih nizova na interval [0, 1] te korištenje rijetko popunjene matrice.

Definicija 2.4.1. Matrica $A \in \mathbb{R}^{n \times n}$ je *rijetko popunjena matrica* ukoliko postoji algoritam takav da se Ax, za vektor $x \in \mathbb{R}^n$ može računati u $O(n)$ operacija.

Definicija 2.4.2. *Donji susjedi* elemenata matrice s indeksom c su elementi s indeksima iz skupa $\{c-1, c-n, c-(n+1)\}$. *Gornji susjedi* elementa matrice s indeksom c su elementi s indeksima iz skupa $\{c+1, c+n, c+(n+1)\}$.

Definicija 2.4.3. *Blokirani element* rijetko popunjene matrice je svaki element koji ima vrijednost nula.

Za lakše utvrđivanje sličnih dijelova nizova potrebno ih je normalizirati korištenjem formule

$$S_{norm} = \frac{S_i^k - \min(S^k)}{\max(S^k) - \min(S^k)} \tag{2.17}$$

pri čemu S_i^k označava i-ti element k-tog niza.

Algoritam se sastoji od 3 dijela. U prvom dijelu se pomoću ulaznih parametara računaju broj odnosno širina intervala I_k te širina dijela u kojem se preklapaju. Pomoću njih određuje se koji elementi nizova pripadaju istom intervalu. Ako je taj uvjet zadovoljen računa se euklidska udaljenost ta dva elementa i vrijednost se sprema u rijetko popunjenu matricu SM. Ako elementi ne pripadaju istom intervalu onda je vrijednost matrice na pripadnom indeksu nula. Opisanom postupku odgovara sljedeća formula

$$SM_{i,j} = \begin{cases} \|S(i), Q(j)\|_2, & \text{ako je } S(i) \in I_k \text{ i } Q(j) \in I_k \\ 0, & \text{inače.} \end{cases} \tag{2.18}$$

Ako je euklidska udaljenost jednaka nuli onda se vrijednost tog elementa postavlja na -1 tako da ga se može razlikovati od blokiranog elementa matrice.

Nakon incijalizacije matrice SM slijedi računanje DTW udaljenosti. Za element na indeksu c traži se njegov donji susjed s najmanjom vrijednosti i njegova vrijednost se zbraja s vrijednosti trenutnog elementa. Postupak je isti kao kod osnovnog oblika DTW algoritma. Osim toga provjerava se jesu li gornji susjedi trenutnog elementa blokirani, ako jesu, računa se euklidska udaljenost pripadnih elemenata nizova i ti elementi nisu više blokirani.

Treći dio zadužen je za traženje optimalnog poravnanja. Krećući od indeksa koji odgovara paru elemenata (m, n) rekurzivno se dodaju indeksi donjih susjeda trenutnog elementa koji imaju najmanju vrijednost. Postupak se ponavlja sve dok se ne dođe do početnog elementa matrice koji odgovara paru elemenata (1, 1). Slijedi pseudokod opisanog algoritma. *Res* vrijednost služi kod određivanja broja odnosno širine intervala koji se koriste kod inicijalizacije rijetko popunjene matrice SM. Matrica SM je indeksirana linearno po stupcima.

```
Ulaz: vremenski nizovi S i Q duljine m odnosno n, Res vrijednost
Izlaz: optimalno DTW poravnanje i pripadna udaljenost nizova

S' = Normalise(S)
Q' = Normalise(Q)
LowerBound = 0
UpperBound = Res
for all 0 ≤ LowerBound ≤ 1 − Res/2 do
    S'_indices = find(LowerBound ≤ S' ≤ UpperBound)
    Q'_indices = find(LowerBound ≤ Q' ≤ UpperBound)
    LowerBound = LowerBound + Res/2
    UpperBound = LowerBound + Res
    for all s'_i ∈ S'_indices do
        for all q'_j ∈ Q'_indices do
            Add EuclideanDistance(s'_i, q'_j) to SM
        end for
    end for
end for
for all c ∈ SM do
    LowerNeighborus = {c−1, c−n, c−(n+1)}
    minCost = min(SM(LowerNeighbours))
    SM(c) = SM(c) + minCost
    UpperNeighbours = {c+1, c+n, c+n+1}
    for all N_i ∈ UpperNeighbours
        if N_i == 0 then
            SM ∪ EuclideanDistance(pair(N_i))
        end if
    end for
end for
WarpingPath = ∅
hop = SM(n × m)
WarpingPath ∪ hop
while hop ≠ SM(1) do
    LowerNeighbours = {hop−1, hop−n, hop−(n+1)}
    [minCost, index] = min[Cost(LowerNeighbours)]
    hop = index
    WarpingPath ∪ hop
end while
WarpingPath ∪ SM(1)
return WarpingPath, SM(n×m)
```

Prednost ovog algoritma nad metodom *Podijeli pa vladaj* je što uvijek daje optimalno poravnanje zadanih nizova zato što se od standardnog DTW algoritma razlikuje samo po tome što ne računa cijelu matricu akumuliranih vrijednosti nego samo ona polja koja na početku imaju veliku vjerojatnost da će pripadati optimalnom poravnanju jer pripadaju istom intervalu i ona čija vrijednost postane nužna prilikom računanja matrice akumuliranih

vrijednosti. Tako se prostor potreban za spremanje matrice može znatno smanjiti. Vremenska složenost algoritma je u najgorem slučaju $O(n \times m)$ gdje su m i n duljine nizova.

Primjer 2.4.1. *U ovom primjeru prikazati ćemo glavne korake na nizovima S = [3, 4, 5, 3, 3] i Q = [1, 2, 2, 1, 0]. Prvi korak zahtjeva normalizaciju podataka pa se tako dobiju sljedeći nizovi:* $S^{'}$ *= [0, 0.5, 1.0, 0.0, 0.0] i* $Q^{'}$ *= [0.5, 1.0, 1.0, 0.5, 0]. Neka je Res parametar jednak 0.5. Iz toga slijedi da je broj intervala jednak* $\frac{2}{Res} = 4$*, a to su [0, 0.5], [0.25, 0.75], [0.5, 1], [0.75, 1.25]. Preklapanje dobivenih intervala pridonosi smanjenju odblokiranih elemenata prilikom provođenja drugog koraka algoritma kada se računa tablica akumuliranih vrijednosti. Sljedeći korak je računanje euklidske udaljenosti onih elemenata nizova koji pripadaju istom intervalu. Tako se dobije matrica prikazana na slici 2.4 a).*

Pri računanju matrice akumuliranih vrijednosti koristi se opća rekurzivna formula kojom se odabire donji susjed s najmanjom vrijednosti i zbraja se s trenutnim elementom. Ako element koji nije blokiran ima blokirane susjede potrebno ih je odblokirati računajući euklidsku udaljenost pripadnih elemenata. U ovom primjeru su to označeni elementi matrice s vrijednostima 34, 35 i 38 na slici 2.4 b).

Zadnji dio algoritma konstruira optimalno poravnanje krećući od indeksa zadnjeg elementa matrice koji odgovara paru (m, n) dodajući rekurzivno indekse donjeg susjeda s najmanjom vrijednosti.

	1	2	2	1	0
3	4	B	B	4	9
4	9	4	4	9	16
5	16	9	9	16	B
3	4	B	B	4	9
3	4	B	B	4	9

a)

	1	2	2	1	0
3	4	B	B	4	13
4	13	8	12	13	20
5	29	17	17	28	38
3	33	B	B	21	30
3	37	34	35	25	30

b)

Slika 2.4: **a)** Matrica euklidskih udaljenosti elemenata nizova koji pripadaju istim intervalima. Ako elementi ne pripadaju istom intervalu onda je polje matrice blokirano (B) odnosno ima vrijednost 0. **b)** Matrica akumuliranih vrijednosti u kojoj su neka polja odblokirana

Poglavlje 3

Baricentrično usrednjavanje

3.1 Uvod

Zbog sve većeg prisustva vremenskih nizova u praksi potrebno je prilagoditi algoritme koji se danas koriste u analizi podataka, prepoznavanju uzoraka i strojnom učenju takvoj vrsti podataka. Veliki broj podataka često je potrebno klasterirati odnosno podijeliti podatke u skupine tako da se u istoj skupini nalaze podatci koji dijele slična svojstva prema kojima ih razvrstavamo. Uspješan rad velikog broja algoritma koji se koriste u prethodno navedenim granama temelji se na metodama pronalaska srednje vrijednosti skupa ulaznih podataka.

Pronalazak srednje vrijednosti skupa vremenskih nizova nije jednostavan posao jer je potrebno uzeti u obzir ovisnost podataka u nizu o vremenu. Zbog toga je jasno da traženje srednje vrijednosti po koordinatama ne dolazi u obzir i potrebno je iskoristiti uspješnost algoritma poravnavanja vremena za uspoređivanje vremenskih nizova. Pri tome treba paziti da se ne naruši konvergencija algoritama za klasteriranje. U praksi se često koristi *medoid* umjesto srednje vrijednosti. Medoid je element čija je prosječna udaljenost od drugih elemenata skupa najmanja, a od srednje vrijednosti skupa razlikuje se po tome što je medoid uvijek element danog skupa dok srednja vrijednost općenito ne mora biti.

Označimo s $\mathbb{S} = \{S_1, S_2, ..., S_N\}$ skup vremenskih nizova te s $C^{(T)}$ niz duljine T. Srednju vrijednost danog skupa definiramo na sljedeći način.

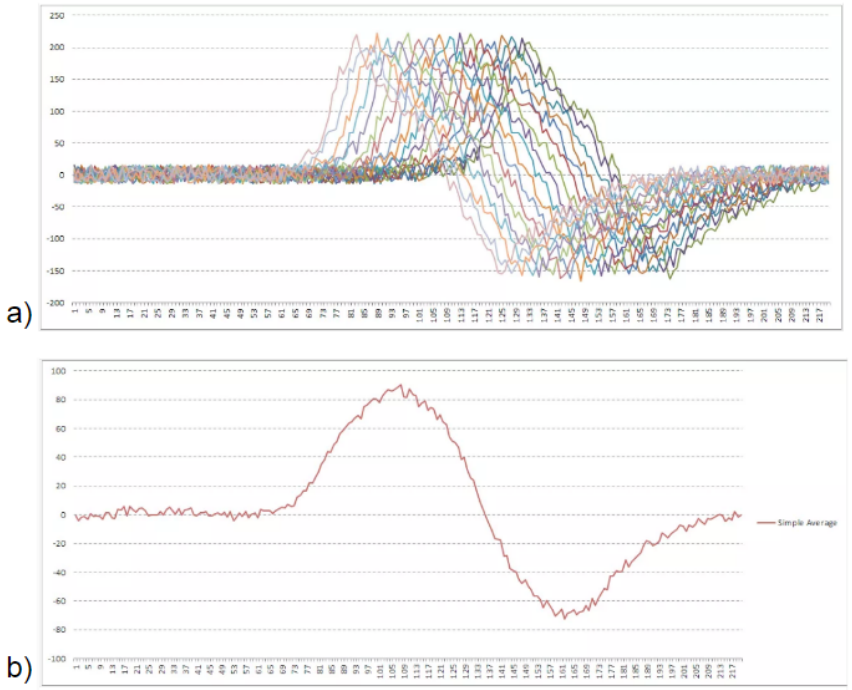
Definicija 3.1.1. Neka je $\mathbb{S} = \{S_1, S_2, ..., S_N\}$ skup vremenskih nizova nad prostorom E duljine T. Označimo s E^T prostor svih nizova duljine T. Srednja vrijednost danog skupa je niz $C^{(T)}$ koji zadovoljava sljedeću nejednadžbu:

$$\forall X \in E^T, \sum_{n=1}^{N} DTW^2(C^{(T)}, S_n) \leq \sum_{n=1}^{N} DTW^2(X, S_n). \tag{3.1}$$

Budući niz koji predstavlja srednju vrijednost ne mora nužno biti duljine T potrebno je u prethodnu nejednakost uvesti sve moguće slučajeve na duljinu niza. Stoga se zahtjeva

ispunjenost sljedećeg uvjeta:

$$\forall t \in [1, +\infty], \left(\forall X \in E^t, \sum_{n=1}^{N} DTW^2(C^{(t)}, S_n) \leq \sum_{n=1}^{N} DTW^2(X, S_n) \right). \tag{3.2}$$



Slika 3.1: Na ovoj slici se vidi da računanje srednje vrijednosti skupa vremenskih nizova uzimanjem srednjih vrijednosti koordinata s istim indeksom ne čuva vrijednosti amplitude nizova. **a)** Skup vremenskih nizova kojem je potrebno naći srednju vrijednost. **b)** Srednja vrijednost zadanog skupa vremenskih nizova dobivena računanjem srednje vrijednosti koordinata

U teoriji su poznata dva načina traženja srednje vrijednosti zadanog skupa vremenskih nizova. Oba su usko povezana s problemom višestrukog poravnavanja vremenskih nizova. Umjesto računanja poravnanja dva niza tako da se uzima minimalna vrijednost prethodna tri susjedna elementa u dvodimenzionalnoj mreži, DTW algoritam se proširuje tako da se primjerice za 3 vremenska niza traži minimalan element od prethodnih sedam susjednih elemenata u kocki. Za N vremenskih nizova dobije se struktura N-dimenzionalne kocke. *I*-ta koordinata niza srednje vrijednosti skupa može se dobiti računanjem srednje vrijednosti koordinata određenih *i*-tom koordinatom višestrukog poravnanja. Primjerice, ako imamo N nizova *i*-ta koordinata vektora poravnanja biti će *N*-torka $(s_{1_{i_1}}, s_{2_{i_2}}, ..., s_{N_{i_N}})$ koja predstavlja

indekse elemenata koji su međusobno pridruženi. I-ta koordinata niza srednje vrijednosti će biti jednaka

$$\frac{s_{1_{i_1}} + s_{2_{i_2}} + ... + s_{N_{i_N}}}{N}. \tag{3.3}$$

Vrlo lako se može uočiti glavni problem kod ovog pristupa, a to je preveliki broj koraka potreban za računanje višestrukog poravnanja nizova. Složenost tog algoritma je $O(T^N)$ što vodi do neupotrebljivosti za imalo veće vrijednosti broja N. Osim toga algoritam prelazi i granice dostupne količine memorije.

Druga metoda temelji se na pretraživanju prostora rješenja. Najveća moguća duljina višestrukog poravnanja je $T^N - 2T$ pa pretraživanje prostora mogućih rješenja završava sa složenosti $O(T^N)$ u slučaju diskretnog konačnog prostora. U slučaju neprekidnog prostora rješenja metoda je neupotrebljiva.

Zbog neuspješnosti pronalaska točnog prosječnog niza razvio se skup metoda za pronalazak njegove aproksimacije. Jedna od najpoznatijih metoda je iterativno usrednjavanje parova nizova. Ako su zadane srednje vrijednosti C_1 i C_2 skupova $\mathbb{S}_1$ i $\mathbb{S}_2$ onda se za srednju vrijednost unije $\mathbb{S}_1 \cup \mathbb{S}_2$ uzima niz dobiven računanjem srednje vrijednosti nizova C_1 i C_2. Po ovom principu, za zadani skup nizova, srednju vrijednost možemo dobiti tako da na početku izaberemo dva niza, nađemo njihovu srednju vrijednost koju nakon toga prilagođavamo svakom preostalom elementu tog skupa odnosno računamo srednju vrijednost skupa koji se sastoji od prethodno izračunate srednje vrijednosti i neiskorištenog elementa zadanog skupa. Metode se međusobno razlikuju po redoslijedu uparivanja elemenata iz skupa i načinu računanja srednje vrijednosti dvaju nizova.

Jedan od načina uparivanja elemenata zadanog niza temelji se na principu igranja utakmica na turniru. Prvo se od početnog broja elemenata N dobije $\frac{N}{2}$ nizova srednje vrijednosti. Isti postupak se primjenjuje na dobiveni skup srednjih vrijednosti. Tako se dobije skup veličine $\frac{N}{4}$. Postupak se nastavlja dok se ne dobije skup koji ima točno jedan element koji označava srednju vrijednosti polaznog skupa. Ovim pristupom postupak računanja srednje vrijednosti dva niza se ponavlja točno N-1 puta. Drugi način je da se spajaju elementi skupa koji imaju najmanju udaljenost. Prvo se izračuna matrica udaljenosti skupa, pronađu se dva niza koji imaju najmanju udaljenost i izračuna se njihova srednja vrijednost. Postupak se ponavlja sve dok skup ne sadrži točno jedan element. Na ovaj način računanje srednje vrijednosti se ponavlja točno N-1 puta, ali dodatan posao je računanje matrice udaljenosti prije svakog uparivanja nizova.

Računanje srednje vrijednosti iz dobivenog poravnanja može se provesti na nekoliko načina. *Jedna koordinata po pridruživanju* daje srednju vrijednost čija je duljina jednaka duljini vektora pridruživanja. Za svaku koordinatu poravnanja, koordinata niza srednje vrijednosti se dobije kao srednja vrijednost odgovarajućeg para elemenata iz zadanih nizova. Glavni nedostatak ovog postupka računanja srednje vrijednosti je što duljina dobivenog niza može biti čak $|S_1| + |S_2| - 2$ što je maksimalna duljina poravnanja nizova S_1 i S_2.

Drugi način se temelji na principu *jedna koordinata po komponenti povezanosti*. Svako poravnanje definira jedan graf. Koordinatu niza srednje vrijednosti dobijemo kao srednju vrijednost svih vrhova pripadne komponente povezanosti. Ovim načinom duljina niza srednje vrijednosti može biti između 1 i $\min\{|S_1|, |S_2|\}$. Očito je da svaki pristup ima svojih mana i prednosti. Kod prvog je problem što se duljina konačnog niza srednje vrijednosti može povećati sve do $N \times T$ dok kod se kod drugog može dogoditi da dobijemo samo jednu komponentu povezanosti pa tako imamo niz srednje vrijednost duljine 1 koji sigurno ne može dobro opisivati zadane nizove. Zato je potrebno pronaći metodu koja će uspješno izbjeći mane i jednog i drugog pristupa. Jedna od njih se zove baricentrično usrednjavanje.

3.2 O metodi

Baricentrično usrednjavanje je iterativna metoda usrednjavanja skupa vremenskih nizova koja računanjem DTW poravnanja između trenutnog niza srednje vrijednosti i svakog od preostalih nizova iz skupa približava trenutnu srednju vrijednost stvarnoj srednjoj vrijednosti skupa. Neka je $\mathbb{S} = \{S_1, S_2, ..., S_N\}$ zadani skup vremenskih nizova duljine T te neka je s $C^i = (c_1^i, c_2^i, ..., c_{T'}^i)$ označen niz srednje vrijednosti skupa nizova $\mathbb{S}$ u i-toj iteraciji. Glavna ideja je minimizirati udaljenosti svake od koordinata niza srednje vrijednosti od koordinata nizova iz zadanog skupa koji su joj pridruženi računanjem DTW poravnanja. To se postiže tako da se svaka koordinata niza C_i računa kao srednja vrijednost skupa koordinata nizova iz skupa $\mathbb{S}$ koje su joj pridružene. Označimo s $C^{i+1} = (c_1^{i+1}, c_2^{i+1}, ..., c_{T'}^{i+1})$ niz srednje vrijednosti u *i+1*-oj iteraciji. Tada je veza između koordinata nizova C^i i C^{i+1} dana s

$$c_j^{i+1} = barycenter(assoc(c_j^i)) \tag{3.4}$$

gdje je

$$barycenter(\{a_1, a_2, ..., a_N\}) = \frac{a_1 + a_2 + ... + a_N}{N} \tag{3.5}$$

te je *assoc* funkcija koja za koordinatu c_j^i računa skup kooridnata nizova iz $\mathbb{S}$ koji su joj pridruženi DTW poravnanjem između pojedinog niza skupa i niza srednje vrijednosti.

Primijetimo, redoslijed odabira elemenata iz zadanog skupa pri računanju poravananja ne utječe na promjenu niza srednje vrijednosti jer prvo računa DTW poravnanje svakog niza iz zadanog skupa i trenutnog niza srednje vrijednosti, a nakon toga se mijenjaju koordinate niza srednje vrijednosti. Slijedi pseudokod algoritma koji računa jedan korak metode baricentričnog usrednjavanja.

```
Ulaz: C_0, S = {S_1, S_2, ..., S_N} skup nizova duljine T
Izlaz: niz srednje vrijednosti C

1 asocciationTable = polje reda T' koje sadrzava skup kooridnata
    pridruzenih svakoj od koordinata niza C
```

```
m[T, T] = tablica DTW tablica iz koje se moze konstruirati DTW
    poravnanje nizova
for niz in 𝕊 do
    m = DTW(C, niz)
    i = T'
    j = T
    while i ≥ 1 and j ≥ 1 do
        associationTable[i] = associationTable[i] ∪ niz_j
        (i, j) = second(m[i, j])
    end while
end for
for i = 1 to T' do
    C_i^i = barycenter(associationTable[i])
end for
return C'
```

Uspješnost metode ovisi o početnoj inicijalizaciji niza srednje vrijednosti čija su dva glavna svojstva duljina odabranog niza i vrijednosti koordinata. Vrijednosti koordinata se mogu odabrati na mnogo načina od kojih su najčešće ispitivana sljedeća dva: slučajno odabrani brojevi i koordinate slučajno odabranog niza iz skupa $\mathbb{S}$. Prema mjerenjima iz [10] vrijedi da se najbolji rezultati postižu za početnu duljinu niza veličine T gdje je T duljina nizova iz zadanog skupa $\mathbb{S}$ i slučajno odabrani niz iz tog skupa.

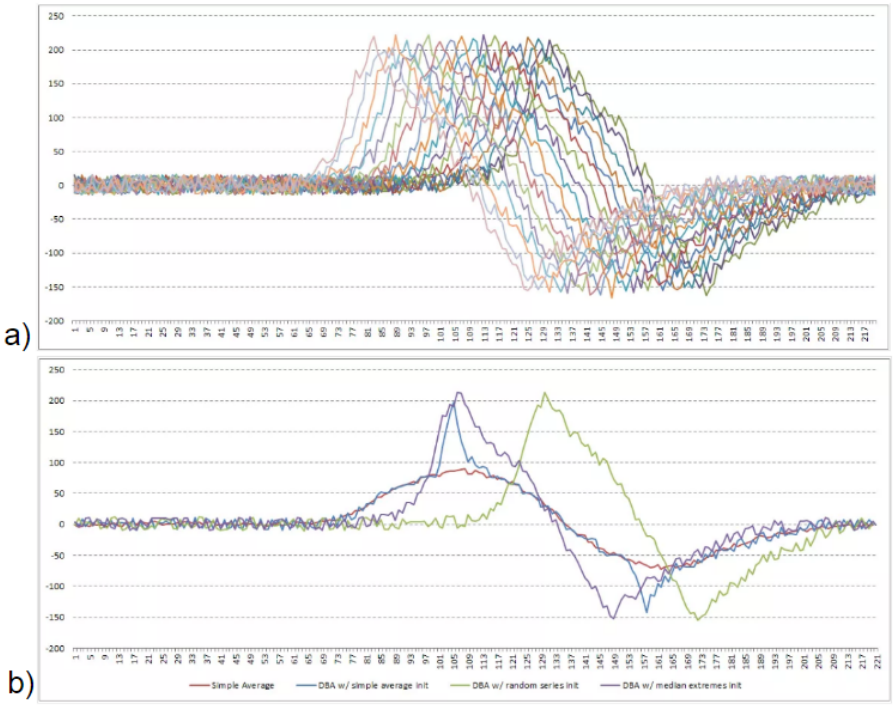
Lema 3.2.1. *Neka je $X = x_1, ..., x_n$ niz realnih brojeva duljine n. Neka je $\bar{x} = \frac{x_1,...,x_n}{n}$ aritmetička sredina niza X. Za svaki $c \in \mathbb{R}$ vrijedi*

$$\sum_{i=1}^{n}(x_i - \bar{x})^2 \leq \sum_{i=1}^{n}(x_i - c)^2 \tag{3.6}$$

Dokaz.

$$\begin{aligned}
\sum_{i=1}^{n}(x_i - c)^2 &= \sum_{i=1}^{n}(x_i - \bar{x} + \bar{x} - c)^2 = \sum_{i=1}^{n}[(x_i - \bar{x}) + (\bar{x} - c)]^2 \\
&= \sum_{i=1}^{n}(x_i - \bar{x})^2 + 2\sum_{i=1}^{n}(x_i - \bar{x})(\bar{x} - c) + \sum_{i=1}^{n}(\bar{x} - c)^2 \\
&= \sum_{i=1}^{n}(x_i - \bar{x})^2 + 2(\bar{x} - c)\sum_{i=1}^{n}(x_i - \bar{x}) + \sum_{i=1}^{n}(\bar{x} - c)^2 \\
&= \sum_{i=1}^{n}(x_i - \bar{x})^2 + \sum_{i=1}^{n}(\bar{x} - c)^2
\end{aligned} \tag{3.7}$$

Vidimo da za c=$\bar{x}$ zadana suma poprima minimum $\sum_{i=1}^{n}(x_i - \bar{x})^2$. □



Slika 3.2: Na slici **b)** se vide različite srednje vrijednosti zadanog skupa nizova prikazanog na **a)**

Svakom novom iteracijom vrijednost $\sum_{i=1}^{N} DTW_2(C, S_i)$ je manja ili jednaka toj istoj sumi u prethodnoj iteraciji. Za proizvoljan i $\in$ 1, ..., N vrijedi:

$$DTW_2(C, S_i)^2 = \sum_{s \in S_1} (c_1 - s)^2 \tag{3.8}$$

pri čemu je S_i skup koordinata koje su u računanjem DTW poravnanja pridružene i-toj koordinati niza C. Dakle ukupna suma jednaka je zbroju kvadrata udaljenosti koordinata niza C s elementima skupa kojima je pridružen nakon računanja svih DTW poravnanja sa svakim od nizova iz $\mathbb{S}$. Po Lemi 3.2.1 vrijedi da će svaka od koordinata niza C minimizirati pripadni dio sume $\sum_{i=1}^{N} DTW^2(C, S_n)$ pa će dobiveni niz srednje vrijednosti biti još bliže konačnom rješenju.

3.3 Vremenska složenost

U svakoj iteraciji metode potrebno je izračunati DTW algoritam N puta. Složenost DTW algoritma je $O(T^2)$ gdje je T duljina zadanih nizova. Iz toga slijedi da je vremenska

složenost prvog dijela iteracije $O(N \times T^2)$. Drugi dio se sastoji od računanja srednjih vrijednost, a budući da koordinatama niza srednje vrijednosti može biti pridruženo najviše $N \times T$ koordinata (skup $\mathcal{S}$ se sastoji od N nizova duljine T) njegova složenost je $O(N \times T)$. Ako s I označimo broj iteracija potrebnih za određivanje niza srednje vrijednosti onda je konačna složenost metode

$$O(I \times (N \times T^2 + N \times T)) = O(I \times N \times T^2). \tag{3.9}$$

Poglavlje 4

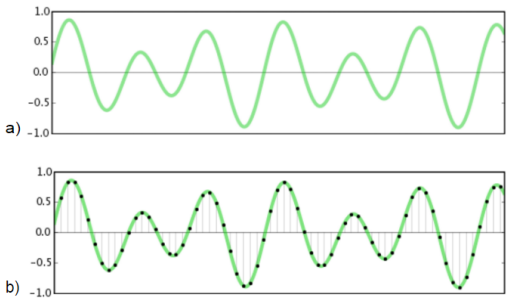
Prepoznavanje izgovorene riječi

4.1 Reprezentacija zvuka u računalu

Zvuk je val frekvencija 16Hz do 20kHz koji nastaje periodičnim titranjem izvora zvuka. Matematički bi se zvuk mogao opisati kao sljedeća funkcija vremena

$$x(t) = A f_x(2\pi n t) \tag{4.1}$$

pri čemu je t proteklo vrijeme, A amplituda , n frekvencija i f_x valna funkcija. Funkcija $x(t)$ je neprekidna pa je tako definirana za svaku vrijednost vremena t. Nažalost, u svijetu računala moguće je samo pamtiti konačan skup podataka stoga je potrebno prikazati zvuk kao konačan niz podataka uzimajući vrijednost funkcije u određenom vremenskom intervalu. Vremenski razmak između uzimanja vrijednost odabre se tako da je iz prikupljenih vrijednosti funkcije moguće interpolirati početnu funkciju.



Slika 4.1: **a)** Prikaz neprekidne funkcije koja opisuje zvuk **b)** Na slici su označene točke koje predstavljaju pripadni zvučni val u memoriji računala.

4.2 Modeliranje jednostavnog prepoznavača izgovorenih riječi

Snimanjem zvuka dobije se jedan vremenski niz koji nosi veliki broj informacija iz okoline. Neke od njih su važne jer nose informacije o izgovorenoj riječi, glasu i emociji govornika što je potrebno za povećanje razine točnosti prepoznavača. Dio podataka odgovara pozadinskom šumu što je u većini slučajeva informacija koju je potrebno ukloniti prilikom traženja odgovarajuće riječi.

Pogledajmo jedan jednostavan primjer modeliranja prepoznavača riječi - kalkulatora. Sustav snima korisnikov govor, iz njega dobiva riječi koje označavaju brojeve i osnovne računske operacije na brojevima: zbrajanje, množenje, oduzimanje i dijeljenje, prepoznaje izgovorene riječi, provodi odgovarajuću računsku operaciju na dobivenim brojevima i ispiše korisniku zadanu računicu. Postavljaju se tri osnovna pitanja:

- Kako podijeliti korisnikov govor na riječi odnosno kako prepoznati koji je dio snimljenog zvuka govor, a koji je tišina ili pozadinska buka?
- Kako izvući korisne informacije o izgovorenoj riječi iz prikupljenih podataka od strane računala?
- Kako prepoznati izgovorenu riječ?

Najjednostavniji način za klasificiranje dijela snimke kao tišina ili govor uz pretpostavku da nema pozadinskog šuma je određivanjem maksimalne amplitude tog dijela snimljenog zvuka. Potrebno je odrediti prag koji će dijeliti tišinu od govora. Ako je maksimalna amplituda ispod određenog praga onda se taj dio snimke klasificira kao tišina. Naprednji način otkrivanja sadrži li zvučni segment čovjekov govor je korištenjem algoritama za otkrivanje glasa (*eng. Voice activity detection*). Oni se temelje na računanju osnovnih svojstava zvučnog vala kao što su energija, periodičnost, dinamika i brzina prijelaza valne funkcije iz pozitivnog dijela u negativni. U našem jednostavnom prepoznavaču izgovorenih riječi uzimamo jednostavniji model traženja tišine u zvuku.

Za dobivanje korisnih informacija o snimljenom zvuku koristi se metoda računanja *mel-frekvencijskih kepstralnih koeficijenata* skraćenog naziva MFCC. Izvlačenje što kvalitetnijih parametara iz dobivenih podataka će pozitivno utjecati na točnost prepoznavača pa je zbog toga ovaj dio modela jako važan. MFCC metoda se temelji na principima ljudske percepcije zvuka i sastoji se od sljedećih koraka:

1. **Naglašavanje visoke frekvencije**

   Neka je $X(t)$ funkcija koja opisuje ulazni zvučni val. Prvi korak metode sastoji se od linearne transformacije ulaznog vala na sljedeći način

   $$y(t) = x(t) - 0.95x(t-1). \tag{4.2}$$

Ako zvuk ima veću frekvenciju onda se susjedni uzorci iz diskretnog prikaza razlikuju po svojoj vrijednosti više nego susjedni uzorci zvučnog vala s nižom frekvencijom pa će ovom transformacijom dijelovi s nižom frekvencijom imati manju amplitudu nego dijelovi s visokom frekvencijom. Tako se postiže da signali s većom frekvencijom imaju veću energiju definiranu s

$$\sum_t x(t)^2. \tag{4.3}$$

Na ovaj način se lakše razlikuje ljudski glas od pozadinskog šuma.

2. **Podjela na okvire**

   Dobiveni digitalni signal se dijeli na jedinice duljine od 20 do 30 ms trajanja koje se preklapaju za $\frac{1}{3}$ do $\frac{1}{2}$ duljine okvira. Često se vrijednost duljine okvira postavlja na neku potenciju broja dva zbog potreba sljedećih koraka metode. Ako duljina nije višekratnik odabrane potencije broja dva potrebno je zadnji okvir proširiti nulama.

3. **Prozoriranje**

   Beskonačni signal je u računalu prikazan pomoću konačnog broja uzoraka. Zbog idućeg koraka ove metode, brze Fourierove transformacije, potrebno je ukloniti prekide nastale kao posljedica opisivanja signala zvuka na navedeni način. Jedno od rješenja je korištenje funkcije naziva *Hammingov prozor* koja je opisana sljedećom formulom:

   $$w(n) = 0.54 - 0.46cos(\frac{2\pi n}{N-1}) \tag{4.4}$$

   gdje je N broj uzoraka u svakom okviru te $0 \leq n \leq N\text{-}1$. Na svaki okvir se primjenjuje prozoriranje tako da se funkcija kojom je opisan zvučni val množi s funkcijom Hammingov prozor.

4. **Brze Fourierove transformacije**

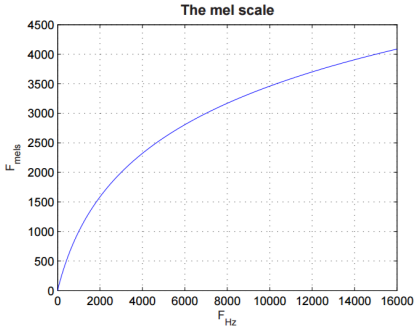
   Pomoću brzih Fourierovih transformacija iz vremenske domene prelazimo u frekvencijsku domenu signala. Iz takvog prikaza jasno se vidi koje frekvencije postoje u signalu i u kolikoj mjeri. Prednosti prikaza signala na ovakav način su: operacije s funkcijama koje opisuju signale kao što je konvolucija su puno lakše za računanje, računanje svojstava signala kao što je snaga, energija (Parsevalov teorem). Brze Fourierove transformacije se provode na svakom od okvira posebno i iz njih se računa spektar snage pojedinog okvira sljedećom formulom

   $$P = \frac{|FFT(x_i)|^2}{N} \tag{4.5}$$

   pri čemu je FFT brza Fourierova transformacija i-tog okvira diskretnog signala, a N duljina okvira.

5. **Računanje mel-frekvencijskih kepstralnih koeficijanata**

   Za računanje MFCC koeficijenata koristi se Mel skala frekvencija nastala 1940-tih godina kao rezultat provođenja pokusa na ljudskom uhu s nastojanjem određivanja mjere za frekvenciju zvuka baziranoj na pitanju kako i s kolikom razlikom čovjek čuje određene frekvencije. Jedan od rezultata istraživanja je sljedeći graf koji prikazuje odnos frekvencije mjerene u hercima i melima.

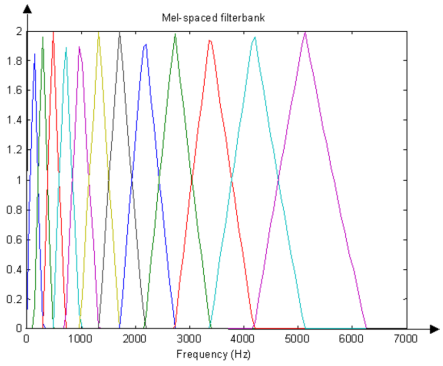


Slika 4.2: Mel skala

Formula koja se najčešće koristi za pretvorbu frekvencije iz jedne jedinice u drugu je dana jednadžbom

$$F_{mel} = \frac{1000}{\log(2)}\left[1 + \frac{F_{Hz}}{1000}\right] \tag{4.6}$$

Mel-frekvencijski kepstralni koeficijenti računaju se primjenom skupa filtera na energije signala dobivene brzim Fourierovim transformacijama. Do traženih koefici-



Slika 4.3: Skup od dvanaest filtera za računanje mel-frekvencijskih kepstralnih koeficijenata

jenata dolazi se primjenom filtera na pojedini okvir. Kao rezultat se dobije suma energija pripadnih frekvenicija pomnoženih s težinama koje su određene filterom. Koeficijenti dobiveni korištenjem filtera su visoko korelirani pa je potrebno ukloniti korelaciju. Ovaj korak je uveden zbog algoritama strojnog učenja kod kojih je točnost bolja ako su svojstva odabrana za treniranje modela nezavisna.

6. **Diskretne kosinusne transformacije**

   Diskretne kosinusne transformacije imaju široku primjenu u kompresiji podataka kao što je primjerice slika. Nakon primjene transformacija na koeficijente iz prethodnog koraka dovoljno je uzeti prvih 13 koeficijenata jer oni opisuju sva važna svojstva signala.

Nakon dobivanja važnih svojstava riječi računanjem *mel-frekvencijskih kepstralinih koeficijenata* potrebno ih je usporediti korištenjem *algoritma dinamičkog poravnanja vremena*.

4.3 Rezultati mjerenja i zaključak

Rezultati mjerenja

Skup riječi za testiranje algoritma sastoji se od snimki sljedećih riječi: nula, jedan, dva, tri, četiri, pet, šest, sedam, osam, devet, plus, minus, dijeljeno, puta, jednako. Snimke su izgovorene od 13 različitih osoba pri čemu je svaka od njih barem jednom izgovorila svaku od navedenih riječi. U trening skupu nalaze se snimke od 11 ljudi (10 ženskih i 1 muška osoba). Snimke od preostale dvije osobe koriste se za testiranje (jedna muška i jedna ženska osoba). Osim njih za testiranje se koriste i snimke jedne osobe koja se pojavljuje i u trening skupu. Testni skup sastoji se od 15 (svaka riječ je izgovorena jednom), a trening skup od 271 riječi. Točnost algoritma testirana je na tri načina. Prvi i drugi način traže riječ iz trening skupa takvu da je udaljenost dobivena algoritmom poravnanja vremena minimalna. To zahtjeva računanje udaljenost onoliko puta kolika je veličina trening skupa. Prvi način koristi opći algoritam poravnavanja vremena, a drugi jedan oblik ubrzanja algoritma. Treći način koristi baricentre dobivene metodom baricentričnog usrednjavanja. Svaka od riječi iz test skupa uspoređuje se s baricentrima dobivenim usrednjavanjem elemenata trening skupa koji pripadaju istoj riječi. Kod ovog načina mjerenja točnosti rezultati variraju ovisno o načinu odabira početnih vrijednosti baricentara. Jedna od mogućnosti je uzeti niz s proizvoljnim vrijednostima elemenata (3.a), a druga uzeti niz iz skupa kojem tražimo srednju vrijednost (3.b). Rezultati mjerenja prikazani su u tablicama 4.1, 4.2 i 4.3.

Iz dobivenih rezultata vidimo da je najlošije prepoznavanje kod muške osobe jer u trening skupu dominiraju snimke dobivene od ženskih osoba. Algoritam najčešće griješi kod pridruživanja jedan - jednako i obratno. Najbolji rezultati dobiju se ako u trening skupu

	Točnost(%)	Vrijeme(s)
1. način	86.67	65.83
2. način	80	67.48
3.a način	73.3	4.74
3.b način	87.7	4.97

Tablica 4.1: Mjerenje točnost algoritma na 15 uzoraka koji pripadaju ženskoj osobi

	Točnost(%)	Vrijeme(s)
1. način	53.3	60
2. način	47.7	61.25
3.a način	47.7	4.66
3.b način	53.3	4.7

Tablica 4.2: Mjerenje točnost algoritma na 15 uzoraka koji pripadaju muškoj osobi

	Točnost(%)	Vrijeme(s)
1. način	86.6	69.35
2. način	93.3	69.58
3.a način	80	5.1
3.b način	86.6	5.2

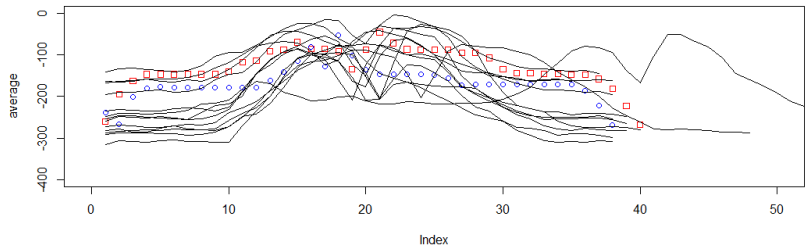
Tablica 4.3: Mjerenje točnost algoritma na 15 uzoraka koji pripadaju ženskoj osobi čije se snimke (različite od snimki iz test skupa) nalaze i u trening skupu

postoje snimke testirane osobe. Za određivanje baricentra skupa nizova bolji rezultati se dobiju kad se za početni niz uzme jedan od nizova iz skupa. Ovisnost dobivenog baricentra o početnom nizu prikazana je na slici 4.4.

Za računanje mel-frekvencijskih kepstralnih keoficijenata i udaljenost algoritma dinamičkog poravnanja vremena korišten je programski jezik *python* i paketi *dtw*, *fastdtw*, *librosa*. Za računanje baricentra skupa vremenskih nizova korišten je programski jezik *R* i paket *dtwclus*. Pripadni kodovi i primjeri datoteka koje sadrže mel-frekvencijske kepstralne koeficijenite nalaze se na `https://github.com/mjukicbraculj/Diplomski-rad`.

Zaključak

Prvi korak za modeliranje prepoznavača izgovorene riječi je kvalitetno predprocesiranje zvuka. Potrebno je ukloniti tišinu, buku, normirati podatke i obaviti složene matematičke transformacije opisane u odlomku 4.2. Nakon toga, primjenom algoritma poravnavanja



Slika 4.4: Slika donosi usporedbu baricentara dobivenih krećući od različitih početnih nizova. Crvenim kvadratićima su označene kooridnate baricentra dobivenog odabirom prvog elementa iz skupa kao početni element, a plavim kružićima je označen baricentar kod kojeg je početni niz proizvoljan

vremena je potrebno usporediti dobivene podatke. Glavni problem algoritma je kvadratna složenost što predstavlja problem kada je potrebno usporediti snimljenu riječ s uzorcima iz baze. Zbog velikog broja usporedbi prepoznavač je spor i neupotrebljiv u aplikacijama koje rade u stvarnom vremenu. Zbog toga je važno ispitati razliku u točnosti prilikom korištenja pojedinačnih uzoraka iz baze i srednjih vrijednosti klasa. Iz navedenih mjerenja vidimo da se vrijeme klasificiranja smanjilo za više od deset puta, a jako malo se izgubilo na točnosti što zasigurno predstavlja bolji način klasificiranja uzorka.

Poglavlje 5

Prepoznavanje rukopisa

U ovom poglavlju opisan je postupak prikupljanja rukopisa više različitih osoba, njihova analiza i klasifikacija. Zadatak svake osobe je napisati pet puta riječ *Matematika*. Cilj praktičnog rada je algoritmom dinamičkog poravnavanja vremena usporediti potpise mjerenjem udaljenost među njima te primijeniti metodu baricentričnog usrednjavanja na skup potpisa iste osobe te mjeriti udaljenost potpisa od tako dobivenih nizova.

5.1 Prikupljanje i pripremanje podataka

Svaka osoba je 5 puta na ekranu računala napisala riječ *Matematika*. Program za bilježenje zapisa na ekranu napravljen je u programskom jeziku *Processing* i radi na principu bilježenja koordinata točaka potpisa tijekom zapisivanja riječi. Točke se bilježe 30 puta u sekundi. Dobiveni su dvodimenzionalni vremenski nizovi prosječne duljine 500 točaka.

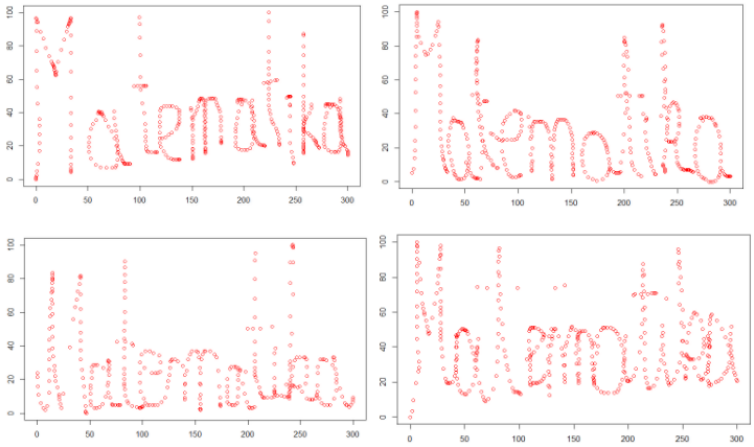
Korisniku je omogućeno potpisivanje na bilo kojem dijelu ekrana pa koordinate točaka potpisa variraju ovisno o veličini ekrana i veličini samog zapisa. Kako bi prepoznavač potpisa bio neovisan o veličini zapisa i njegovoj poziciji na ekranu potrebno je provesti sljedeća dva koraka

- Translacija - niz je translatiran u ishodište kooridnatnog sustava po sljedećoj formuli

$$\begin{aligned} x_i &= x_i - x_{min} \\ y_i &= y_i - y_{min} \end{aligned} \tag{5.1}$$

- Normalizacija - točke su svedene na pravokutnik veličine 300×100 po sljedećoj formuli

$$\begin{aligned} x_i &= \frac{x_i}{x_{max}} \times 300 \\ y_i &= \frac{y_i}{y_{max}} \times 100 \end{aligned} \tag{5.2}$$



Slika 5.1: Primjer četiri zapisa riječi *Matematika* dobiveni od različitih osoba

5.2 Rezultati mjerenja i zaključak

Rezultati mjerenja

Nakon mjerenja udaljenosti svakog zapisa sa svakim od preostalih uspoređene su dobivene udaljenosti. Zapis koji je najbliži zapisu iste osobe smatra se točno klasificiranim zapisom, inače se broji kao greška. Broj točno razvrstanih zapisa u postotku je 90%. Na slici 5.2 može se vidjeti isječak potpune tablice udaljenosti te veličine izmjerenih udaljenosti i razlike među njima.

Problem kod ovakvog pristupa je što je za klasificiranje jednog uzorka potrebno proći kroz sve zapise i za svaki izračunati udaljenost. Zbog kvadratne složenosti algoritma to može znatno usporiti aplikaciju pa je potrebno na neki način ubrzati način klasifikacije zapisa. Jedan od mogućih ubrzanja je da se od prikupljenih uzoraka, pomoću metode baricentričnog usrednjavanja, izračuna niz koji označava srednju vrijednost klase zapisa koji pripadaju istoj osobi te se onda novi zapis uspoređuje s nizovima srednjih vrijednosti. Tablica udaljenosti nizova od njihove srednje vrijednosti je prikazana na slici 5.2. Dobivena srednja vrijednost zapisa određene osobe može se profinjavati svakim novim zapisom koji pripada njenoj klasi tako da se izračuna srednja vrijednost novog zapisa i trenutne srednje vrijednosti.

Za računanje udaljenosti vremenskih nizova i srednje vrijednosti korišten je programski jezik *R* te paketi *dtw* i *dtwclust* koji sadrže implementaciju metode baricentričnog usrednjavanja i algoritma dinamičkog poravnavanja vremena.

	o1, p1	o1, p2	o1, p3	o1, p4
o1, p1	0	9285	9435	8585
o1, p2	9285	0	4088	3062
o1, p3	9435	4088	0	3946
o1, p4	8585	3062	3946	0
o1, p5	8893	3240	4155	2790
o2, p1	10600	8267	7313	7277
o2, p2	6954	7717	7781	7478
o2, p3	8293	6292	6748	5890
o2, p4	7492	7795	8120	7287
o2, p5	9822	7980	8336	7275
o3, p1	15933	16338	15083	15646
o3, p2	14226	15576	15011	14784
o3, p3	18864	19742	17944	18924
o3, p4	19095	18356	17283	18047
o3, p5	17603	18830	17742	17530
o4, p1	11416	11264	11406	10296
o4, p2	12829	12964	11969	11645
o4, p3	12628	14175	13794	12468
o4, p4	13363	14689	13676	13152
o4, p5	9815	11120	10197	9792
o5, p1	6287	7009	6884	5892
o5, p2	7296	8831	8158	7476
o5, p3	6347	8457	7024	6911
o5, p4	8065	6825	6473	5687
o5, p5	6497	7638	6765	6169
o6, p1	8857	8937	8236	7342
o6, p2	9044	8286	7113	6950
o6, p3	10290	10124	8397	7810
o6, p4	6457	8432	8000	6864
o6, p5	7315	8279	7749	6435

Slika 5.2: Isječak tablice udaljenosti zapisa na kojem se vidi razlika u udaljenostima između različitih zapisa i jedan primjer pogreške. S *oi, pj* je označen *j-ti* potpis *i-te osobe*.

Zaključak

Iz prethodnih mjerenja vidimo da algoritam dinamičkog poravnavanja vremena daje veliku točnost prilikom prepoznavanja rukopisa osobe. Na tablici 5.2 vidimo da algoritam ispravno klasificira četiri od pet potpisa iste osobe. Pogrešna klasifikacija jednog potpisa

	s0	s1	s2	s3	s4	s5
o1, p1	7371	7158	18855	10471	7298	6346
o1, p2	2596	7237	17980	11597	6305	7909
o1, p3	3791	7646	17043	10662	5927	7828
o1, p4	1397	6989	17617	10637	5233	6357
o1, p5	2702	6802	18660	12146	6358	7817
o2, p1	7101	5571	14018	7810	7113	6533
o2, p2	6685	2680	14659	8234	5929	4046
o2, p3	5356	2878	13341	8676	5782	5544
o2, p4	6575	1291	14360	7994	5959	4450
o2, p5	6544	3261	14081	8357	6449	5667
o3, p1	15088	11989	4275	11201	12585	12295
o3, p2	13941	11011	7949	12430	12609	12306
o3, p3	17888	13380	8292	14708	14951	15624
o3, p4	17283	14371	2270	13478	13913	15241
o3, p5	16603	13039	7666	13712	14498	13856
o4, p1	9050	7705	14653	5207	7721	7895
o4, p2	10768	8850	14446	3782	8049	7861
o4, p3	11369	9385	17080	5452	9526	8959
o4, p4	12291	10566	15804	2705	9932	8791
o4, p5	8925	6774	12703	5417	7403	5751
o5, p1	5458	5961	16201	8969	2518	4801
o5, p2	7004	5155	13032	6588	3426	3879
o5, p3	6299	6289	14584	7785	3501	5008
o5, p4	5183	5988	13997	7330	1314	4713
o5, p5	5496	6776	15754	8833	3006	5503
o6, p1	7055	6420	14243	6807	5520	4032
o6, p2	6242	5926	14262	7149	4572	4009
o6, p3	7559	6479	15120	7109	5079	3988
o6, p4	6516	4704	16895	6917	4842	1143
o6, p5	6318	5747	16696	7558	4775	2578

Slika 5.3: Na slici se može vidjeti tablica udaljenosti svakog potpisa od baricentara (označenih s s_i gdje je i indeks osobe)

ne znači nužno da je algoritam pogriješio već može značiti da je osoba napravila neku pogrešku prilikom pisanja pa taj potpis značajno odstupa od prosjeka. Uspješnost algoritma nije se smanjila prilikom korištenjem baricentara što je jako dobro jer to predstavlja brz način klasificiranja potpisa. Umjesto uspoređivanja sa svakim od potpisa u bazi, potpis se uspoređuje samo s baricentrima što značajno smanjuje broj računanja udaljenost nizova pa tako i pripadno vrijeme pronalaska prave klase uzorka.

Bibliografija

[1] G. Al-Naymat, S. Chawla i J. Taheri, *Sparsedtw: A novel approach to speed up dynamic time warping*, Proceedings of the Eighth Australasian Data Mining Conference-Volume 101, Australian Computer Society, Inc., 2009, str. 117–127.

[2] J. Colannino, M. Damian, F. Hurtado, S. Langerman, H. Meijer, S. Ramaswami, D. Souvaine i G. Toussaint, *Efficient many-to-many point matching in one dimension*, Graphs and combinatorics **23** (2007), br. 1, 169–178.

[3] F. Gustafsson, *Statistical sensor fusion*, Studentlitteratur, 2010.

[4] D. S. Hirschberg, *A linear space algorithm for computing maximal common subsequences*, Communications of the ACM **18** (1975), br. 6, 341–343.

[5] D. Lemire, *Faster retrieval with a two-pass dynamic-time-warping lower bound*, Pattern recognition **42** (2009), br. 9, 2169–2180.

[6] MATLAB, *version 7.10.0 (R2010a)*, The MathWorks Inc., Natick, Massachusetts, 2010.

[7] M. Meinard, *Information retrieval for music and motion*, sv. 2, Springer, 2007.

[8] H. B. Mitchell, *Data fusion: concepts and ideas*, Springer Science & Business Media, 2012.

[9] R. Niels, *Dynamic Time Warping An intuitive way of handwriting recognition?*, Master Thesis, 2004.

[10] F. Petitjean, A. Ketterlin i P. Gançarski, *A global averaging method for dynamic time warping, with applications to clustering*, Pattern Recognition **44** (2011), br. 3, 678–693.

[11] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008, `http://www.R-project.org`, ISBN 3-900051-07-0.

[12] H. Sakoe i S. Chiba, *Dynamic programming algorithm optimization for spoken word recognition*, IEEE transactions on acoustics, speech, and signal processing **26** (1978), br. 1, 43–49.

Sažetak

Vremenski nizovi imaju široku primjenu u praksi pa je potrebno osmisliti nove i prilagoditi postojeće algoritme za računanje s takvom vrstom podataka. Jedan od algoritama koji se koristi za pronalazak udaljenosti između dva vremenska niza proizvoljne duljine je algoritam poravnavanja vremenskih nizova. Algoritam uzima u obzir varijacije u vremenu prilikom nastanka niza i na taj način dobiva točniju usporedbu vremenskih nizova. Ima kvadratnu složenost pa se uvode algoritmi koji korištenjem donjih ograda daju ubrzanje početnog algoritma.

U prvom poglavlju rada opisana je ideja i osnovi oblik algoritma. Prikazane su jednostavne varijacije u koraku algoritma i ograničenja koja mogu djelomično ubrzati računanje udaljenosti nizova.

Drugo poglavlje uvodi Keoghovu i poboljšanu Keoghovu donju granicu kao oblike ubrzanja algoritma osvrćući se na njihovu složenost i usporedbu s početnim algoritmom. Osim donjih granica kao jednog načina ubrzanja algoritma, koriste se još metoda podijeli pa vladaj i upotreba rijetko popunjenih matrica.

Osim za usporedbu vremenskih nizova algoritam dinamičkog poravnavanja vremena koristi se za dobivanje srednje vrijednosti skupa vremenskih nizova. Metoda baricentričnog usrednjavanja, opisana u trećem poglavlju, nizom iteracija u kojima se računanju udaljenosti nizova pomoću algoritma dolazi do tražene srednje vrijednosti.

U praktičnom dijelu rada napravljena je primjena algoritma u prepoznavanju govora i rukopisa. Prikazan je način pripremanja podataka, dobivanje osnovnih svojstava uzorka i rezultati njihove usporedbe dobiveni korištenjem algoritma.

Summary

Time series appear in many application domains like biology, finance, image analysis, etc. It is necessary to adapt techniques and algorithms to this kind of data. Dynamic time warping algorithm is dynamic programming paradigm to compute the alignment between two time series with arbitrary lengths. Algorithm has quadratic time complexity and because of that many speed-up techniques are introduced.

In the first chapter we introduce classical dynamic time warping algorithm and give examples of step function. Simple techniques to speed up the algorithm are shown.

In the second chapter we describe algorithms that have better time and space complexity than classical dynamic time warping algorithm. Some of them are: Keogh lower bound, improved Keogh lower bound, divide and conquer and sparse dynamic time warping algorithm.

A global averaging method for dynamic time warping is presented in third chapter. The method is called dynamic time warping barycenter averaging. The method iteratively refines an initial average sequence in order to minimize its squared dynamic time warping distances to averaged sequences.

In the forth and fifth chapter we present case studies that use the algorithm for speech and handwriting recognition and measure accuracy of different types of algorithm.

Životopis

Rođena sam 15. rujna 1993. godine u Sinju. Osnovnu školu i gimnaziju sam završila u Sinju. Tijekom srednjoškolskog obrazovanja redovito sam sudjelovala na državnim natjecanjima iz matematike te osvojila nagrade za drugo i treće mjesto u svojoj kategoriji. Godine 2012. upisujem preddiplomski studij matematike na Prirodoslovno-matematičkom fakultetu. Na kraju studija dobivam nagradu za izniman uspjeh od strane Matematičkog odsjeka. Godine 2015. upisujem diplomski studij Računarstva i matematike te sam na kraju studiranja ponovno nagrađena za izniman uspjeh od strane Matematičkog odsjeka. Tijekom zadnje godine studiranja na diplomskom studiju obavila sam dvije stručne prakse u pojedinačnom trajanju od šest mjeseci.