Idősorok elemzése adatbányászati módszerekkel

Deák Szilárd

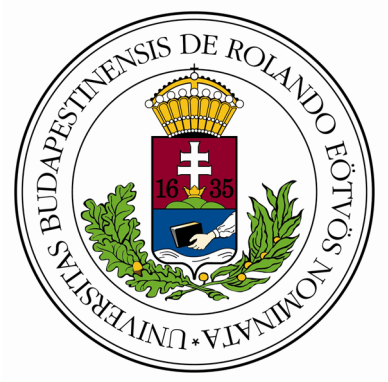
Matematika BSc, Matematikai elemző szakirány

Témavezető:

Lukács András, tudományos főmunkatárs

Számítógéptudományi Tanszék

Eötvös Loránd Tudományegyetem, Természettudományi Kar



2013

Tartalomjegyzék

1. Bevezető **3**

2. Idősor reprezentációk **4**

2.1. Eredeti idősor 4

2.2. Diszkrét Fourier-transzformált (DFT) 4

2.3. Haar wavelet transzformáció 7

2.4. Symbolic Aggregate Approximation (SAX) 8

2.4.1. Normálás 8

2.4.2. Dimenziócsökkentés 9

2.4.3. Diszkretizáció 10

2.5. A SAX egy lehetséges módosítása 11

3. Idősorok távolsága **12**

3.1. Euklidészi távolság 12

3.1.1. Komplex adatok esetén 12

3.1.2. SAX reprezentációban 13

3.2. Dynamic Time Warping (DTW) 13

4. Klasszifikáció **16**

4.1. A k-NN klasszifikátor 17

5. Mérések előkészítése **19**

5.1. Az adatok, és a mérési környezet 19

5.2. Mérőszámok 20

5.3. Az alkalmazott modellek 22

6. Eredmények **24**

6.1. Pontosság 24

6.2. Futásidő 25

6.3. Egyéb mérőszámok 27

1. Bevezető

A mai világban keletkező egyre nagyobb mennyiségű adat többsége valós értékű, időparaméterrel indexelhető véges sorozat, azaz idősor. Ezen adatokból való értékes információk kinyerésének egyik technikája az adatbányászat. Az adatbányászati alap feladatok (klaszterezés, klasszifikáció, kívülálló keresés, mintázatok keresése) közül a továbbiakban idősorok klasszifikációval fogok foglalkozni. Ennek célja az idősorok osztályokba sorolása.
A dolgozat fő célja a adatbányászati modellt építő algoritmusok használatát megelőző előkészítő lépések, az adatok átalakítása, a klaszterezéshez szükséges távolság függvény kiválasztása, amelyek nagyban befolyásolják a modellek predikciós erejét.
Dolgozatom első felében a főbb idősor reprezentációkat, és távolság kiszámítási módszereket ismertetem. Ezek után bemutatom a klasszifikáció feladatát és az alkalmazott modellt, a k-NN klasszifikátort. A fenti adatbányászati módszerek bemutatása után valós teszt adatokon számítógéppel elvégzett mérésekkel értékelem ki azokat, vonok le következtetéseket a használt módszerekre vonatkozóan.

2. Idősor reprezentációk

Az idősorok adatbányászati módszerekkel való feldolgozása előtt, nagyon fontos lépés, hogy megfelelő alakra hozzuk az adatokat. Vannak esetek amikor az idősor eredeti formájában is alkalmas a további feldolgozásra, de általában egy jobb reprezentáció választásával javíthatunk az alkalmazandó adatbányászati módszerek hatékonyságán. Egy jó reprezentációval például lehetséges az adatok méretének csökkentése a lehető legtöbb információ megtartása mellett, vagy az egyes jellemző tulajdonságok (pl.: periodicitás) elkülönítése. [1]

2.1. Eredeti idősor

A legegyszerűbb, és legtermészetesebb reprezentáció az, ha meghagyjuk az idősort eredeti formájában. Használhatósága viszont függ az adatok jellegétől. Ennek ellenére nem hagyjatjuk figyelmen kívül, ugyanis sok esetben ezzel is jó eredményeket kaphatunk. Sőt, bizonyos távolság számításához használt algoritmushoz is erre a reprezentációra van szükség.

2.2. Diszkrét Fourier-transzformált (DFT)

A Fourier-transzformáció során, egy időtől függő függvényből, melyben $f(t)$ a jel erőssége egy adott időponban, egy másik függvényt állítunk elő, ahol a $dft(x)$ értékek egy adott x frekveciájú jel erősségét mutatják. Más megközelítéssel azt is mondhatjuk, hogy a függvényt periodikus függvények lineáris kombinációjára bontja fel. A Fourier-transzformáció folytonos függvényekre is értelmezve van, de a gyakorlatban főként ennek diszkrét változatát használják. [2]
Jelen esetben is a diszkrét változatot alkalmazzuk, mivel az idősoraink ilyen értékekből állnak. A diszkrét Fourier-transzformáció (DFT) egy $C^n \rightarrow C^n$ függvény, azaz a bemenő idősor és az eredmény is komplex számokból áll, kiszámítása a következő módon történik:

$$dft(X) = (c_0, c_1, .., c_{n-1}),$$

$$c_k = \sum_{t=0}^{n-1} x_t e^{-2\pi i t k / n}$$

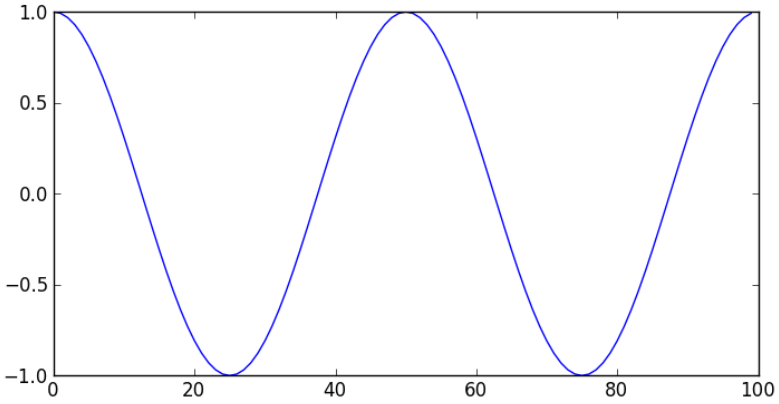
Az idősoraink legtöbbször valós számok sorozatai, így érdemes megjegyezni, hogy ebben a speciális esetben igaz az alábbi összefüggés:

$$c_{n-k} = c_k^*,$$

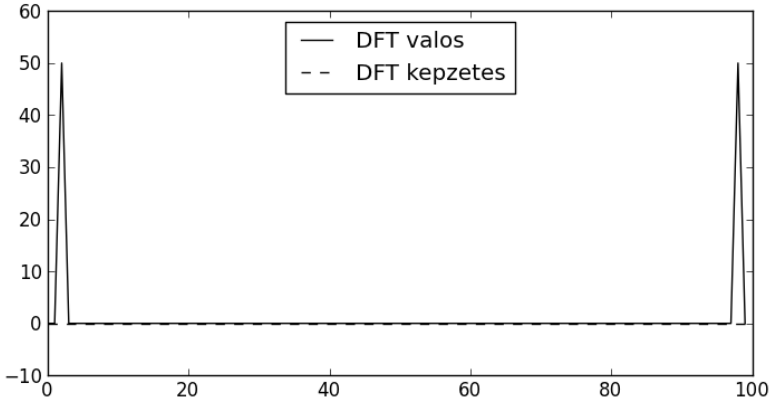
ahol c^*a komplex konjugálást jelenti.
Kiszámításához az Euler-formulát alkamazhatjuk, így külön kiszámíthatjuk a valós és képzetes részeket:

$$e^{ti} = \cos t + i \sin t$$

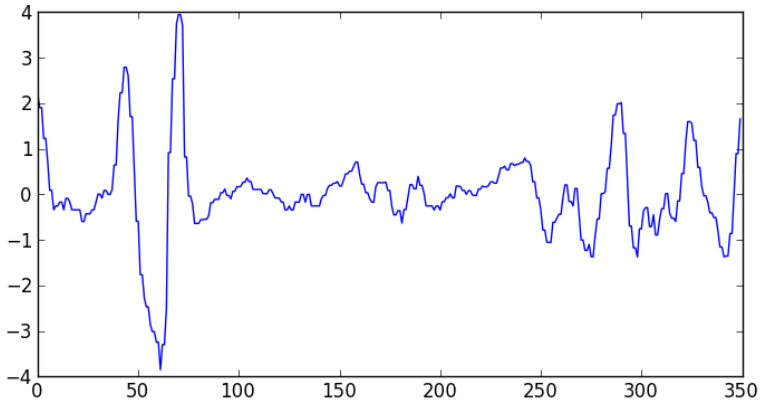
Az alábbi két, ábrákkal szemléltetett példán jól látszik a DFT működése, és az a tulajdonsága, hogy a függvényeket szinuszok és koszinuszok összegére bontja fel.



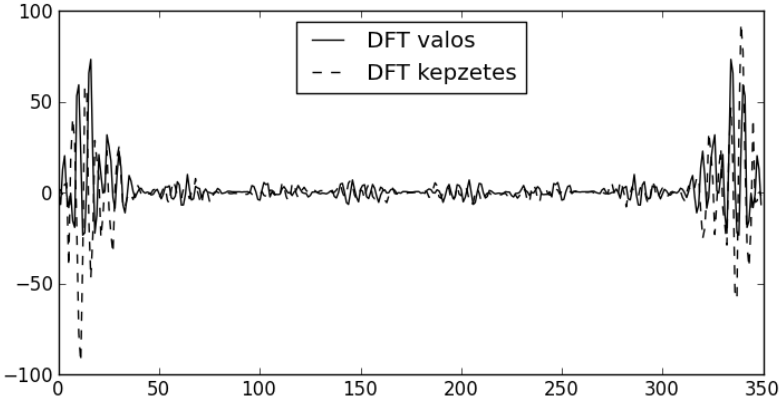
1.ábra: Sima koszinusz jel



2.ábra: Koszinusz Fourier-transzformáltja



3.ábra: Egy módosítatlan idősor



4.ábra: Az előbbi idősor Fourier-transzformáltja

2.3. Haar wavelet transzformáció

Az úgynevezett waveletek, a digitális jelfeldolgozásban, és így az idősorok adatbányászati reprezentációjában is fontos szerepet játszanak. A DFT során az idősort különböző frekvenciájú periodikus függvények lináris kombinációjaként állítottuk elő. A waveletekkel való reprezentáció ehhez nagyon hasonlóan működik, csak ebben az esetben korlátos tartójú függvényekből készítjük a bázist periodikus függvények helyett, és ezek lineáris kombinációját adjuk meg. Ezek a mesterséges függvények azonosan 0 függvényekként indulnak, majd egy adtott t időpontban valamilyen kitérérés után, oda is térnek vissza. Nagyon sok féle wavelet alapú módszer létezik, és ezeket a bázisként használt függvények különböztetik meg egymástól. [4]
Az ilyen típusú reprezentációk legegyszerűbb, és legelső típusa, a Haar wavelet. Az ebben használt alapfüggfény:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & \textit{egyébként} \end{cases}$$

A módszer arra alapszik, hogy bármely valós függvény közelíthető mint

$$\psi(t), \psi(2t), \psi(4t), .., \psi(2^k t), ..$$

függvények lineáris kombinációja.
A reprezentáció kiszámításának egyik módja, hogy a Haar transzformációt mátrix műveletekként adjuk meg. Ehhez először a használt alapfüggvényt vektorok formájában írjuk fel. A legegyszerűbb vektorok melyek megfelelő wavelet bázisok lesznek:

$$\left\langle \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right\rangle, \left\langle \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right\rangle$$

Ezekből a bázisvektorokból elkészítjük a Haar mátrixot:

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

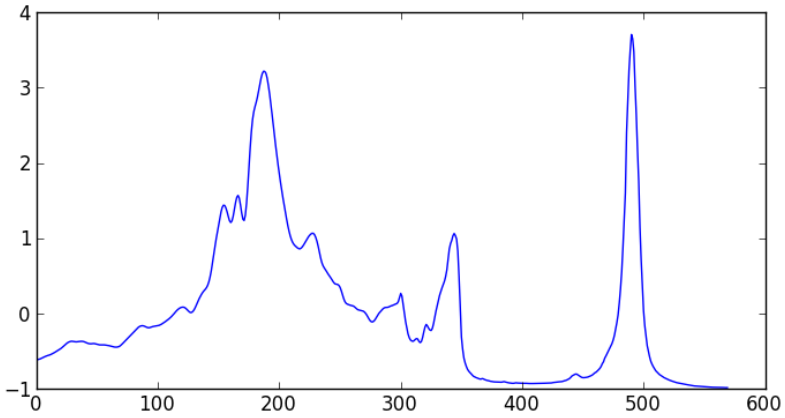
Ezek után elvégezhetjük egy vektor Haar transzformációját, mint ezzel a mátrixal való szorzás. Mivel ez a mátrix 2×2-es, ezért ezzel csak 2 hosszú vektorokat tudunk egy lépésben transzformálni, azonban hosszabb vektorok esetében sem lesz szükség másik bázisra. A transzformálandó idősort kisebb részekre osztjuk fel, és rekurzió segítségével végezzük el az átalakítást H_2-vel. [3]

2.4. Symbolic Aggregate Approximation (SAX)

Ebben a reprezentációban az idősort, egy előre megadott hosszúságú ($lsax$) karakter sorozattá alakítjuk. A karaktereket egy ugyancsak előre adott hosszúságú ($msax$) ábécéből vesszük. Ezzel a módszerrel az idősor dimenzióját is csökkentjük, n dimenziós idősorból lsax dimenziósat hozunk létre. A reprezentáció fontos tulajdonsága, hogy lehetséges az egyes karakterek közti távolság kiszámítása, így ki tudjuk számítani két SAX-al kapott reprezentáció távolságát, amely szükséges a legtöbb adatbányászati módszer alkalmazásához. Kiszámítására részletesen az idősorok távolságáról szóló részben (3.1) térek ki. [5]
A SAX reprezentáció létrehozásának lépései a következők:

2.4.1. Normálás

Az idősor minden eleméből kivonjuk az elemek átlagát, és elosztjuk a szórással. Így 0 várható értékű, és 1 szórású idősorokat kapunk. Ezzel a lépéssel összehasonlíthatóvá válnak az eltérő várható értékkel, és szórással rendelkező adatok is.



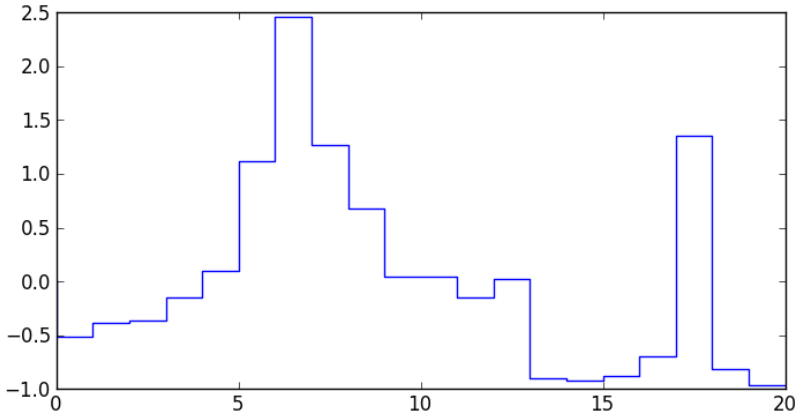
5.ábra: Idősor normálás után

2.4.2. Dimenziócsökkentés

A már normált $X = (x_1, x_2, ..., x_n)$, n dimenziós idősort PAA (Piecewise Aggregate Approximation) segítségével, $lsax(\ll n)$ dimenzióssá alakítjuk. Ennek célja, hogy drasztikusan csökkentse az idősoraink hosszát, és ezzel a későbbi műveletek kiszámításához szükséges időt, a lehető legtöbb hasznos információ megtartása mellett. A módszer finomhangolásához szükséges az $lsax$ érték helyes megválasztása. Túlságosan kicsi $lsax$ esetén nagy lesz az információ veszteség, nagy $lsax$ esetén viszont fölösleges információkkal lassítjuk a számításokat.

$$C = (c_1, c_2, ..., c_{lsax})$$

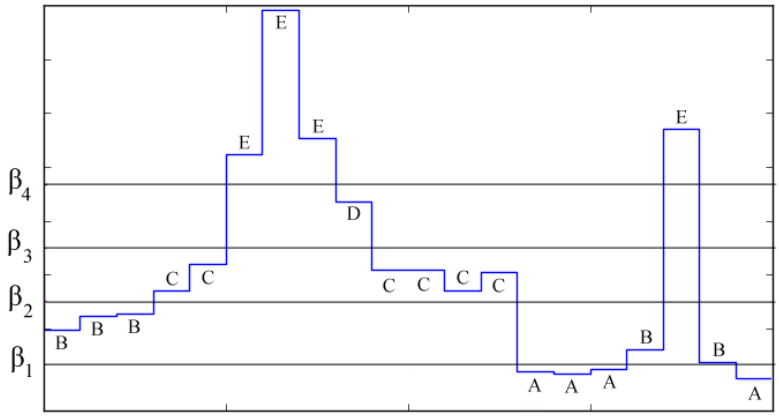
$$c_i = \frac{lsax}{n} \sum_{j=\lfloor \frac{n}{lsax} \rfloor i}^{\lfloor \frac{n}{lsax} \rfloor (i+1)-1} x_j$$



6.ábra: Idősor PAA után

2.4.3. Diszkretizáció

Ebben a lépésben rendeljük az $msax$ féle különböző karaktert, az előbb kapott C idősor értékeihez.
Ehhez először $msax - 1$ darab β_1,..,β_{msax-1} osztópontot kell megadni, melyek később meghatározzák az egyes intervallumokat, amelybe eső értékek az ábécé azonos betűjét kapják az új reprezentációban. Azt szeretnénk, hogy minden szimbólum azonos valószínűséggel fordulahasson elő, ezért feltételezve az értékek normális eloszlását, az osztópontokat úgy határozzuk meg, hogy a Gauss eloszlás sűrűségfüggvényét egyenlő területű részekre osszák fel. Ez után minden c_i-re meghatározzuk mely osztópontok közé esnek. A c_i-t az ábécé j-edik betűjével reprezentáljuk, ha: $\beta_{j-1} \leq c_i < \beta_j$.



7.ábra: Az idősor diszkretizálása

Az így kapott karakter sorozat lesz az idősor SAX reprezentációja. (A példában: BBBCCEEEDCCCCAAABEBA)

2.5. A SAX egy lehetséges módosítása

Az előbbiekben ismertetett, alap SAX reprezentáció létrehozása során feltételeztük, hogy az adataink Gauss eloszlásúak. Ezt a feltevést használtuk fel, a β osztópontok meghatározására. De felvetődik a kérdés, mi van ha valamilyen más eloszlás alapján határozzuk meg őket? Ez az ötlet lehetőséget ad a SAX módosítására. Használjuk fel az egyébként is rendelkezésre álló adatokat arra, hogy annak tapasztalati eloszlása szerint hozzuk létre az osztópontokat!
Ehhez első lépésként, a train adatokon elvégzem a SAX első két lépését, ezzel a diszkretizáció előtti állapotba hozva a idősorokat. Ezután minden idősor összes értéket egyetlen tömbként kezelve, nagyság szerint rendezem. Ezzel megkapom az összes előforduló érték rendezett tömbjét. Ha ezt felosztjuk a kívánt $msax$ darab olyan intervallumra, melyek egyenlő darabszámú értéket tartalmaznak, az egyes intervallumok határait véve β_1,..,β_{msax-1}osztópontoknak olyan értékeket kapunk, amellyel a SAX elvégzése után az egyes karakterek előfordulásának valószínűsége közel azonos.
Nagyobb train adatsorok, és hosszú idősorok esetén gyorsíthatjuk ezt a módszert úgy, hogy limitáljuk a fenti módszerhez felhasznált idősorok számát.

3. Idősorok távolsága

Sok alapvető adatbányászati módszerhez elengedhetetlen, hogy ki tudjuk számítani két idősor távolságát. Egy $d(X, Y)$ távolság függvény, az X és Y idősorokhoz egy pozitív valós számot rendel. Minél nagyobb ez a szám, annál jobban különbözik X és Y egymástól. A távolság függvény meghatározásakor is sok lehetőség közül választhatunk, és ahogyan a jó reprezentáció kiválasztása, úgy a megfelelő távolság mérték is befolyásolja az erre épülő adatbányászati algoritmusok eredményét. Bizonyos esetekben a választott reprezentáció megváltoztathatja a távolság kiszámításának módját.

3.1. Euklidészi távolság

A szokásos L^2 távolság, kiszámítása:

$$d(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

3.1.1. Komplex adatok esetén

Mivel a Fourier-transzformáció eredménye képpen komplex számokat kapunk, ezért defeiniálnunk kell ezek távolságát is. A távolság kiszámításának elve változatlan, a képletben csak kis változtatásokra van szükség:

$$d_{dft}(X, Y) = \sqrt{\sum_{i=1}^{n}(|x_i - y_i|)^2}$$

Így a komplex vektorok abszolútértékét véve az eredmény egy valós szám lesz.

3.1.2. SAX reprezentációban

SAX reprezentáció esetén az Euklideszi távolság egy módosított változatával számolhatunk:

$$d_{sax}(X_{sax}, Y_{sax}) = \sqrt{\frac{n}{lsax}} \sqrt{\sum_{i=1}^{lsax} (dist(x_i, y_i))^2}$$

A módosításokra a SAX speciális tulajdonságai miatt van szükség. Mivel ebben a reprezentációban nem számok, hanem karakterek szerepelnek az idősorok egyes koordinátáiban, ezért külön definiálnunk kell két karakter távolságát: $dist(x_i, y_i)$-t. Ehhez egy D távolság mátrixot hozunk létre, melyben $D(i, j)$ az ábécé i-edik és j-edik karakterének minimális távolsága. Ezek az értékek a β osztópontok távolságaiból adódnak, az alábbi módon:

$$D(i,j) = \begin{cases} 0, & |i-j| \leq 1 \\ \beta_{max(i,j)-1} - \beta_{min(i,j)}, & egyébként \end{cases}$$

Előzők alapján kitöltött távolság mátrix, $msax = 5$ esetben:

	A	B	C	D	E
A	0	0	0,59	1,09	1,68
B	0	0	0	0,5	1,09
C	0,59	0	0	0	0,59
D	1,09	0,5	0	0	0
E	1,68	1,09	0,59	0	0

A kapott távolságot ezen kívül még meg kell szorozni $\sqrt{\frac{n}{lsax}}$-el, erre a dimenzió csökkentése miatt van szükség. Az így kapott távolság érték egy alsó becslés lesz az idősorok eredeti reprezentációjban vett távolságára.

3.2. Dynamic Time Warping (DTW)

A valóságban sokszor előfordulhat, hogy két idősor valójában ugyan azt az eseményt írja le, viszont az események bekövetkezésének ideje, vagy hossza

különböző. Ezeket az Euklidészi távolság nem képes korrigálni, így két egyébként hasonló idősor távolsága mégis nagyobb lesz. A DTW ezzel szemben tudja kezelni ezeket, ezért gyakran használják például beszéd, vagy kézírás felismerésre, ahol tipikusak az ilyen típusú eltérések az adatok között. Egy beszéd felismerő esetében szeretnénk, hogy azonos szavak hangmintái közt kicsi legyen a távolság akkor is, ha éppen az egyiket más hangszínnel, vagy sebességgel mondtuk is ki.
A DTW működése során, az egyik (X) idősort a másikba (Y) transzformálja, a lehető legkisebb költséggel. Az algoritmus a dinamikus programozás módszerével halad, egyre növelve a transzformált idősor hosszát, és közben kiszámítja a transzformáció összköltségét. A már kiszámolt értékeket egy táblázatban rögzíti:

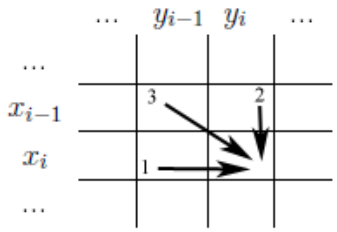
	y_1	y_2	$\cdots$	y_{n-1}	y_n
x_1					
x_2					
$\cdots$			$dtw(i,j)$		
x_{n-1}					
x_n					

A táblázatban oszloponként halad y_1-től kezdve, ahol:

$$dtw(i,j) = c(x_i, y_j) + min \left\{ \begin{array}{c} dtw(i,j-1), \\ dtw(i-1,j), \\ dtw(i-1,j-1) \end{array} \right\}$$

Az algoritmus egy-egy cella kiszámításakor az alábbi három lehetséges eset közül, a minimális költségűt választja:

1. Az X előző elemével hasonlítja össze, az Y következő elemét.
2. Az X következő elemével hasonlítja össze, az Y előző elemét.
3. Az X következtő elemét, az Y következő elemével hasonlítja össze.



A kapott minimumhoz adja hozzá a transzformáció költségét $c(x_i, y_j)$-t, ami általában $|x_i - y_j|$.

Az 1. és 2. eset adja a DTW lényegét. Ezekben az esetekben, amennyiben ennek a költsége a legkisebb, az algoritmus az idősorok különböző indexű elemeit hasonlítja össze, ellentétben az Euklideszi távolsággal, mely mindíg két azonos indexű elem távolságát veszi. A DTW ezzel képes az idősorok eltéréseit rugalmasabban kezelni.

Amikor $dtw(n, n)$ értékét, azaz a teljes X idősor Y-ba transzformálásának költségét megkaptuk, ez lesz maga a DTW távolság.

Nagy méretű idősorok esetén a táblázat mérete, és így a számításhoz szükséges idő is igen nagy lesz. Ilyen esetekben elkerülhető a teljes táblázat kitöltése, így gyorsítva az algoritmust. Mivel csak a $dtw(n, n)$-re van szükségünk, ezért megoldható, hogy csak a főátlóhoz közeli értékeket számítsuk ki.

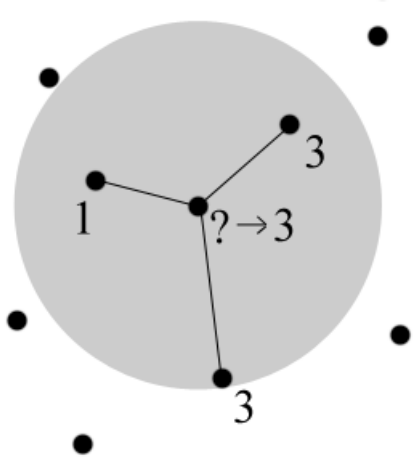
4. Klasszifikáció

A klasszifikációs feladat során, minden idősort egy-egy kategóriába sorolunk. Célunk, hogy egy új, ismeretlen kategóriájú idősort a lehető legnagyobb pontossággal soroljuk be a helyes osztályba. Ezt akkor nevezzük helyesnek, ha az adott adat a valóságban is abba a kategóriába tartozik. Például orvosi EKG adatok esetében akkor jó az osztályozás, ha egy egészségesnek klasszifikált mérés valóban egészséges emberhez tartozik. [6]
Ahhoz hogy ezt megtehessük, szükségünk van olyan idősorokra, melyeknek már tudjuk a helyes kategóriáját. Ebből az adatsorból lesz a *tanító (train)* adat, ami alapján a klasszifikációt elvégezzük. Mielőtt azonban alkalmaznánk egy választott klasszifikációs algoritmust, meg kell győződnünk arról, hogy az megfelelően az elvárásaink szerinti pontossággal végzi-e el a feladatot. Hogy ezt a pontosságot mérni tudjuk, az ismert adatoknak csak egy részét használjuk fel tanításra, a többit pedig *teszt (test)* adatnak tartjuk fenn.
A pontosság mérése során a teszt részbeli idősorokat klasszifikáljuk a tanító adatok alapján, és összevetjük a kapott eredményeket a valós osztályokkal. A klasszifikáció pontossága a helyesen besorolt idősorok százalékos aránya lesz. Tehát ha ez az érték például 80%, akkor várhatóan 0,8 valószínűséggel fogunk a helyes osztályba sorolni egy ismeretlen idősort.
Külön feladat a klasszifikáció előtt, a kapott adatok tanító és teszt részre való szétválasztása. Ennek is sok különböző módja létezik, legegyszerűbb az, ha a teljes adatot használjuk egyszerre test és train adatnak. Ez viszont a legrosszabb választás is, ugyanis a legtöbb klasszifikációs algoritmus esetében a valós predikciós erejéhez képest magasabb pontosság értékeket eredményez. Egy jó megoldás lehet viszont, ha egy megadott arányban véletlenszerűen osztjuk két részre az adatot. Továbbá alkalmazhatunk kereszt-validációt is, amely adott k számú darabra vágja fel az adatot, és mindíg az egyik darabot teszt adatnak , a többit pedig train adatnak használja, így k-szor végezve el a klasszifikációt.
A klasszifikációs feladat egy speciális esete, a bináris klasszifikáció, amikor is az osztályok száma pontosan 2. A gyakorlati alkalmazásokban is gya-

kori, ezért érdemes ezzel külön foglalkozni. Bináris klasszifikációról beszélhetünk minden igen-nem válaszlehetőségekkel rendelkező feladat estén, például: "Esni fog az eső?".

4.1. A k-NN klasszifikátor

A k-NN, azaz k-Nearest Neighbor elég egyszerű, mégis igen hatékony klasszifikációs módszer. Az algoritmus az osztályozandó idősorhoz megkeresi azt a k db tanító adatbeli idősort, melyek tőle való távolsága a legkisebb. Az így kapott k db szomszéd kategóriájából pedig többségi döntéssel választja ki a klasszifikáció eredményét. Legegyszerűbb változata az 1-NN például megkeresi a kérdéseshez legközelebbi tanító idősort, és ennek címkéje lesz egyben a jóslat is.



8.ábra: Példa egy 3-NN klasszifikációra

A k paraméter megválasztása lényeges hatással van a klasszifikáció végeredményére. A legjobb k érték természetesen az, amivel a klasszifikáció a legpontosabb lesz, de meghatározása nem egyértelmű. Ha kis értéket választunk, azzal a tanító adatban található zaj hatását erősíthetjük, ha túl nagyot, azzal viszont túlságosan összemoshatjuk az elkülönülő osztályokat. Mindezen felül befolyásoló tényező a tanító idősorok száma, ez minél több, annál nagyobb k-t választhatunk. Továbbá mivel az algoitmus többségi döntést használ, célszerű páratlan értéket választani, ezen felül pedig olyat, amely lehetőleg

relatív prím az osztályok számához. Ezekkel elkerülhetőek a "döntetlen" helyzetek.
Mivel az algoritmus lényegében a távolságok kiszámítására épül, így annak módszere itt jut igazán nagy szerephez. Ahogyan az előző fejezetben ismertettem, erre a leginkább magától adódó módszer a megszokott Eulkideszi távolság, és annak variációi.

5. Mérések előkészítése

A konkrét mérések előtt ismertetem a használt adatokat, a mérőszámokat melyek az eredmények elemzését szolgálják, és a modelleket melyekkel a méréseket végzem.

5.1. Az adatok, és a mérési környezet

A mérésekhez Eamonn Keogh által ingyenesen hozzáférhetővé tett adatsorokat használok. [7] Ezen adatok mindegyike valamilyen valós alkalmazási területről származik, és célja éppen a klasszifikációs és klaszterező módszerek vizsgálata. Az adatsorban a tanító és teszt adatokra való szétválasztás már előre megtörtént, így ennek módjával nem kell foglalkoznunk. Ez segít abban is hogy konzisztensebb eredményeket kapjunk, hiszen így minden mérést ugyan azokon az adatokon végzünk. Keogh az adatsor leírásában azt javasolja, hogy az egész adatsorra végezzük el a méréseket, így általános képet kaphatunk a vizsgált módszer hatékonyságáról, több különböző adat esetén. Ez természetesen csak akkor értendő így, ha a célunk nem egy speciális tulajdonságú idősor klasszifikációja. Jelen dolgozatban a bináris klasszifikációval foglalkozom, ezért alábbi adatokat használom a továbbiakban:

	Osztályok	Train (db)	Test (db)	Idősor hossz
Gun-Point (Gun_Point)	2	50	150	150
Lightning-2 (Lighting2)	2	60	61	637
ECG (ECG200)	2	100	100	96
Yoga (yoga)	2	300	3000	426
Coffee (Coffee)	2	28	28	286

A használt algoitmusokat, a klasszifikátort, és az adatok beolvasását, Python programnyelven valósítottam meg. A program helyenként optimalizálatlan, néhány módszernek létezik gyorsabb vátozata. Célom főként az algoritmusok helyes használata, és az ezzel kapott eredmény volt, ezért a futásidők is így tekintendők.
A kód Python 2.7.2 alatt készült, 64 bites Windows 7 operációs rendszeren.

Az álltalam használt hardver: 3,2 GHz-es 4 magos AMD processzor, 6 GB RAM.

5.2. Mérőszámok

Ahhoz hogy a modellek jobban összehasonlíthatóak legyenek, és messzemenőbb következtetéseket tudjunk levonni a kapott eredményekből, a klasszifikáció jóságán felül (pontosság) újabb mérőszámokat célszerű bevezetni. Egy bináris klasszifikátor futása során az alábbi mérőszámokat tudjuk rögzíteni:

- **TP** (True Positive): A helyesen „1” címkéjűnek klasszifikált idősorok száma.
- **TN** (True Negative): A helyesen „0” címkéjűnek klasszifikált idősorok száma.
- **FP** (False Positive): A tévesen „1” címkéjűnek klasszifikált idősorok száma. Ennek megfelelője a statisztikai hipotézisvizsgálatban az elsőfajú hiba.
- **FN** (False Negative): A tévesen „0” címkéjűnek klasszifikált idősorok száma. Ennek megfelelője a statisztikai hipotézisvizsgálatban a másodfajú hiba.

Ezeket egy tévesztési mátrixban is ábrázolhatjuk:

		valós	
		0	1
jósolt	0	TN	FN
	1	FP	TP

Ebből a négy adatból számos információ nyerhető ki a modellek működésére vonatkozóan. Ezek vizsgálatához az alábbi mérőszámkat származtathatjuk:

- **Pontosság** (Accuracy): A helyesen klasszifikált idősorok aránya. Ez az egyik legfontosabb mérőszám, amely a teljes klasszifikációs modell hatékonyságát mutatja meg.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Szenzitivitás** (True Positive Rate): Az „1” címkéjűek közül helyesen klasszifikált idősorok aránya. Nagyobb érték esetén a másodfajú hiba csökken.

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

- **Specificitás** (True Negative Rate): A „0” címkéjűek közül helyesen klasszifikált idősorok aránya. Nagyobb érték esetén az elsőfajú hiba csökken.

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

- **Precízitás** (Positive Predictive Value): Az „1” -nek klasszifikált értékek ilyen arányban helyesek.

$$PPV = \frac{TP}{TP + FP}$$

- **F-mérték**: Mesterséges mérőszám a pontosság mérésére, a precízitás és szenzitivitás harmonikus közepe.

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

- **Matthews korrelációs együttható**: Ugyancsak egy mesterséges mérőszám a pontosság mérésére, amely már a TN értékeket is figyelembe veszi.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Ezeken felül nem elhanyagolható szempont egy modell alkalmazása során a futásidő sem:

- **Futási idő**: A teljes klasszifikációs modell lefutásához szükséges idő, a megadott test és train adatokon, a már ismertetett mérési környezetben. Az idők az alábbi formátumban értendők:

$$[perc] : [másodperc].[ezredmásodperc]$$

5.3. Az alkalmazott modellek

A 2-es és 3-as pontokban definiált reprezentációkból, és távolság mértékekből, meg kell alkotni azokat a modelleket melyeket használni fogunk a klasszifikációs feladat megoldására. Erre azért van szükség, mert ezek nem mind kompatibilisek egymással, például két SAX-al reprezentált idősor távolságát nem tudjuk DTW segítségével kiszámítani. Ezért a mérésekhez az alábbi reprezentáció-távolság párokat fogom használni:

- 1. Modell (M1): Eredeti idősor és Euklideszi távolság
- 2. Modell (M2): Eredeti idősor és DTW távolság
- 3. Modell (M3): Fourier-transzformált idősor és Euklideszi távolság komplex változata
- 4. Modell (M4): Haar wavelet és Euklideszi távolság
- 5. Modell (M5): SAX és Euklideszi távolság SAX-beli változata
- 6. Modell (M6): SAX módosított osztópontokkal

A SAX típusú módszerek alkalmazásához meg kell határoznunk az $msax$ és az $lsax$ állandókat. Az $msax$, vagyis a használt ábécé méretét minden esetben 8-nak választottam. Így kevesebb információt veszítünk el mint kisebb érték esetén, tehát nagyobb pontosságot érünk el, de az algoritmus futásideje továbbra is alacsony marad. Az $lsax$, vagyis a PAA után keletkező idősor hosszának megadását egy fix érték helyett dinamikusan végzem. Minden egyes adatsorra a benne található idősorok hosszának 1/5 része lesz az $lsax$ értéke. Így az idősorok nagyban eltérő hossza miatt átlagosan sokkal jobb eredményeket kapunk.
A SAX esetén tapasztalataim szerint különböző adatokra más-más beállításokkal érhető el a legnagyobb pontosság, én azonban ezekkel a beállításokkal egy általánosan jól teljesítő SAX reprezentációt szerettem volna kapni. Bizonyos esetekben tehát az esedmények részben kapott pontosság javítható pár %-al, az előbb említett változók finomhangolásával.
A klasszifikációhoz minden esetben 3-NN klasszifikátort használtam. Mivel

dolgozatomban kisebb hangsúlyt fektetek a konkrét klasszifikációs módszerre, ezért azt az értéket választottam, amellyel megfigyeléseim szerint átlagosan a legjobb pontosságértékek születtek.

6. Eredmények

Ebben a részben ismertetem, és elemzem a mérések eredményeit. Az eredmények ismertetése során összefüggéseket keresek az adatok egyes jellemzői, és a modellek alkalmazhatósága között.
A mérések eredményeit, a kiszámolt származtatott mérőszámokkal együtt, a program csv fileokban tárolja el, az egyszerűbb kezelhetőség érdekében. Mivel az összes mérési eredmény megtalálható a csatolt cvs fileban (eredmenyek.cvs), ezért a továbbiakban csak a fontosabb mérőszámokat tartalmazó táblázatokat jelenítem meg, az átláthatóság kedvéért. Az egyes adatok esetén elért legjobb eredményeket kivastagítottam.
A számítások elvégzése során, a legnagyobb, "Yoga" adatra, az M2 modell futásideje már több nap lett volna, így ezt a mérést nem végeztem el. Ennek oka, hogy a DTW távolságok kiszámítására álltalam használt alap algoritmus futásideje $O(n^2)$ (ahol n az összehasonlítandó idősorok hossza). Az adat méretéből adódóan túl sok DTW távolság kiszámítására lenne szükség.

6.1. Pontosság

A klasszifikációs feladat során, az elsõdleges célunk a minél nagyobb pontosság elérése volt. Bináris klasszifikáció esetén, az 50%-os pontosság egyenértékű azzal, mint ha pénzfeldobással döntenénk el, hogy az egyes idősorok mely osztályokba kerüljenek. Az ennél magasabb pontosság értékek tekinthetőek tehát valamilyen szinten eredményesnek. A mi méréseink a legtöbb esetben lényegesen meghaladták az 50%-os pontosságot, tehát ebből a szempontból mindegyik használt modell alkalmas a feladat elvégzésére.

	M1	M2	M3	M4	M5	M6
Gun_Point	87,3%	86,7%	87,3%	86,7%	74,0%	**88,7%**
Lighting2	77,1%	**80,3%**	77,1%	52,5%	67,2%	75,4%
ECG200	**90,0%**	82,0%	**90,0%**	78,0%	87,0%	89,0%
yoga	**79,2%**	-	**79,2%**	75,4%	73,4%	74,8%
Coffee	82,1%	85,7%	82,1%	**92,9%**	53,6%	57,1%

Láthatjuk, hogy az M1 modell mindegyik adatsorra átlagon felüli pontosságértékeket ért el. Mondhatjuk tehát, hogy ez az a modell, mellyel az összes többi versenyez, hiszen miért használnánk bonyolultabb modellt, amikor a lehető legegyszerűbb is jobban teljesít. A dolog itt válik összetettebbé, ugyanis megfigyelhetjük, hogy az egyes adatokra más-más modell bizonyult a legideálisabbnak.
Az M2, azaz a DTW távolság két esetben elõzte meg az M1-et, és megfigyelhetjük, hogy ez éppen azokban az esetekben történt, amikor az egyes idősorok hossza nagy. Ez azzal magyarázható, hogy a hosszabb idősorokban nagyobb eltérést okozhatnak a távolság kiszámításánál az olyan típusú eltérések, melyeket a DTW távolság kezelni képes.
Az M3 modellel kapott TP-TN-FP-FN értékek minden esetben megegyeztek az M1 modell értékeivel. Megfigyeltem, hogy az M1 és M3 modellek esetén kapott távolságok egy adatonként változó konstans szorzóban térnek el egymástól, ez a jelenség a Parseval-egyenlőség miatt következik be. [8]
Az M4 modell pontossága az előzőekhez képest jóval nagyobb szórást mutat. Ebben az esetben nem figyelhető meg összefüggés az idősorok hossza és a pontosság közt, az eltérő eredmények okát máshol kell keresnünk. A Haar wavelet reprezentáció az idősorok belső viselkedésétől függően javít, vagy ront a predikciós erőn.
A SAX modellekkel kapott értékeket vizsgálva láthatjuk, hogy az M6-ban tett kiegészítés, miszerint a β értékeket a kapott tanító adattól függően határozzuk meg, eredményesnek bizonyult. Minden esetben a pontosság javulását láthatjuk az alap SAX-hoz képest, volt ahol a különbség elérte a 14,7%-ot. A többi modellel való összehasonlítás során a módosított SAX, a Coffee adat kivételével minden esetben csak pár %-al lemaradva áll a legjobb eredményt produkálók mögött, sőt egy esetben ez produkálta a legjobb eredményt.

6.2. Futásidő

A futásidők önmagukban nem mutatnak sokat, de egymással összehasonlítva, és az adatsorok jellemzőit ismerve rengeteg következtetést levonhatunk a modellekre vonatkozóan.

	M1	M2	M3	M4	M5	M6
Gun_Point	**00:00.320**	02:41.576	00:05.361	00:00.361	00:00.589	00:00.598
Lighting2	00:00.687	26:33.370	00:49.912	00:00.757	**00:00.615**	00:00.647
ECG200	**00:00.307**	01:27.867	00:02.525	00:00.325	00:00.720	00:00.722
yoga	01:46.601	-	13:53.522	01:48.421	01:33.268	**01:32.811**
Coffee	**00:00.099**	01:01.764	00:04.955	00:00.115	00:00.126	00:00.135

A futásidőket tekintve is igen jól szerepelt az alap M1 modell, mivel az adatokon semmilyen átalakítást nem végzünk, és a távolságok kiszámítása is gyors. Az M1-től minimális futásidőkülönbségekkel (5-10%-os lassulás) végzett az M4 modell. A Haar reprezentáció előállítása tehát a gyakorlatban rendkívül gyorsnak bizonyult.

A Fourier és a Haar-transzformáció közti hasonlóságok ellenére a futásidők drasztikusan megnövekedtek az M3 modell esetén. Ez a sok lebegőpontos számítás következménye. A megvalósítás során a Python alap komplex osztályát (cmath) használtam.

Az általam használt k-NN klasszifikátor működése során rendkívül sok távolság számítást végez, ez az optimalizálatlan DTW algoritmussal együtt nagyon nagy futásidőket eredményezett az M2 modell esetén. Hiba lenne természetesen ebből messzemenőbb következtetéseket levonni, csupán annyi mondható el, hogy ebben az optimalizálatlan nyers formában a DTW csak speciális esetben lehet a megfelelő választás.

Az M5 és M6, azaz a SAX típusú modelleink esetében igen érdekes eredmények születtek. Ezek futásideje minden adatra a gyors módszerek közé tartozik, két adatsorra viszont még az M1 idejét is felülmúlják! A kapott eredmény jól szemlélteti a SAX dimenziócsökkentő tulajdonságának hatását, hiszen a gyorsulás éppen a két leghosszabb idősorokat tartalamazó adatok esetében volt megfigyelhetõ. A két módszer futásideje egyébként minden esetben közel azonos, tehát elmondhatjuk, hogy az M6-ban tett kiegészítés nem csökkenti jelentősen a módszer futásidejét sem.

6.3. Egyéb mérőszámok

A gyakorlati alkalmazások során a tárgyalt két mérőszámon túl, más tulajdonságok is fontosak lehetnek a modell kiválasztása során. Például egy orvosi alkalmazás esetén nem mindegy, hogy a programunk a negatív, vagy a pozitív eredményeket klasszifikálja pontosabban. Hiába nagy ugyanis a pontosság egy ilyen esetben, ha ezt úgy értük el, hogy mindenkit egészségesnek klasszifikáltunk, azokat is, akik valójában betegek. Erre vonatkozó információkat árulnak el a szenzitivitás (TPR) és specificitás (TNR).
A TPR és TNR eredményeket vizsgálva nem tudunk a modellekről általános állítást megfogalmazni. Azt látjuk, hogy inkább főként az adatokra jellemző tulajdonság az, hogy a TPR vagy a TNR értékek lesznek-e nagyobbak. Például a Lighting2 adatsor esetén látható, hogy míg a TPR értékek igen magasak mindegyik modell esetén, az igazi nehézséget a negítív értékek helyes klasszifikációja jelenti.
Az F-mérték, és a Matthews korrelációs együttható segítségével hasonló pontossági sorrendet állíthatunk fel a modellek között mint a klasszikus pontosság esetében. Eltérés viszont, hogy ezek a mérőszámok érzékenyebbek a klasszifikáció egyes résztulajdonságára.

Hivatkozások

[1] Krisztian Antal Buza: Fusion Methods for Time-Series Classification (2011), Background fejezet
http://www.ismll.uni-hildesheim.de/pub/pdfs/Buza_thesis.pdf

[2] Discrete Fourier transform, Wikipedia,
http://en.wikipedia.org/wiki/Discrete_Fourier_transform

[3] Musawir Ali: An Introduction to Wavelets and the Haar Transform,
http://www.cs.ucf.edu/~mali/haar/

[4] Haar wavelet, Wikipedia,
http://en.wikipedia.org/wiki/Haar_wavelet

[5] Lin, J., Keogh, E., Lonardi, S. & Chiu, B. (2003) A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. San Diego, CA. June 13.
http://www.cs.ucr.edu/%7Eeamonn/SAX.pdf

[6] Jiawei Han és Micheline Kamber: Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers (2000) ISBN 1558604898

[7] Keogh, E., Zhu, Q., Hu, B., Hao. Y., Xi, X., Wei, L. & Ratanamahatana, C. A. (2011). The UCR Time Series Classification/Clustering Homepage:
http://www.cs.ucr.edu/~eamonn/time_series_data/

[8] Parseval's theorem, Wikipedia,
http://en.wikipedia.org/wiki/Parseval%27s_theorem