Debreceni Egyetem

Informatikai Kar

Szoftver Fejlesztése Tanuló Algoritmus Felhasználásával

Témavezető:
Kósa Márk
egyetemi tanársegéd

Készítette:
Kondor Gábor
programtervező matematikus
hallgató

Debrecen
2009

Tartalomjegyzék:

Köszönetnyilvánítás:

Köszönetet szeretnék mondani Kósa Márknak, aki vállalta a dolgozatom témavezetését és tanácsaival hozzájárult a munkámhoz.

Köszönettel tartozom Gincsai Tibornak, a rengeteg segítségért amit a tanulmányaim során és azon túl is nyújtott.

Továbbá szeretném megköszönni mindazoknak akik hozzájárultak a diplomamunka elkészüléséhez.

1. fejezet

Bevezetés

Tanulmányaim során az informatika számos területével ismerkedhettem meg, ezen témakörök közül a Mesterséges Intelligencia keltette fel legjobban az érdeklődésemet. A diplomamunkámban az egyetemi évek során megszerzett ismereteimet próbáltam kamatoztatni, majd a beszédfelismerés egyik lehetséges megoldását kívánom kifejteni. Továbbá bemutatom az általam készített programokat, amelyek célja a digitalizált emberi beszéd azonosítása tanuló algoritmusok segítségével, illetve a különböző tanuló algoritmusok hatékonyságának elemzése. Azonban szeretném kihangsúlyozni, hogy nem a különböző szavak és mondatok felismerése a cél, hanem az emberi beszéd illetve egyéb hangok elkülönítése, osztályozása.

A személyi számítógépek megjelenése és azok széleskörű elterjedése óta az informatika elképesztő ütemű fejlődésen ment keresztül. Az egyfolytában korszerűsödő számítógépek megjelenésével lehetővé vált a komplexebb szoftverek megjelenése és azok fejlesztése is. Ennek következtében folyamatosan születnek innovatív megoldások a különböző újonnan felmerülő problémákra, melynek következtében újabb és újabb ágak valamint kutatásra váró területek születnek az informatikában. Ezek közül a mai napig az egyik legígéretesebbnek hangzó és széles körben alkalmazható ág a mesterséges intelligencia, amely az utóbbi időben egyre inkább előtérbe került. A mesterséges intelligencia alkalmazási területe napjainkban rendkívül változatos és számos mai kutatás alapkövét képzi.

De mi is pontosan a mesterséges intelligencia? Bár a kifejezés már több, mint ötven éve megszületett, mikor John McCarthy javaslatára 1956-ban elfogadták az informatika ezen új területére általa alkotott kifejezést, azaz az *artificial intelligence*-t, az elmúlt évtizedekben felkapott új tudományág filozófiai, matematikai és biológiai alapjai már időszámításunk kezdetére visszanyúlnak. Maga a fogalom meghatározása azonban már sokkal nehezebb, de talán célszerű, ha egyszerűen csak úgy próbáljuk definiálni, mint minden olyan törekvést magában foglaló tudomány, amely a számítógép intelligens viselkedését eredményezi. Felvetődik azonban a következő kérdés, hogy mit tekintünk intelligens viselkedésnek? A

kézenfekvő válasz az, hogy minden ami „emberi”, vagyis a beszélgetés, a hétköznapi cselekvések, tevékenységek, a szakértői feladatok elvégzése és természetesen a gondolkodás. A különböző érzékelések, felismerések (kép-, arc-, hang-, beszédfelismerés) ennek az intelligens viselkedésnek az egyik alapját képzik, azonban ezek modellezése már sokkal bonyolultabb feladat.

A beszéd felismerése valamint értelmezése a mindennapi életben is egy igen lényeges tényező. Az ember rövid időintervallum alatt képes megállapítani egy hangról, hogy az emberi beszéd része vagy sem. Még akkor is közel 100%-os hatékonysággal tudja megkülönböztetni a hangot a beszédtől ha magát a nyelvet nem érti, vagyis tartalomtól függetlenül érzékeli. Azonban ugyanennek a problémának a számítógéppel történő felismerése rengeteg nehézséggel néz szembe.

A dolgozat témája azonban annyiban más a beszédfelismeréstől, hogy nem a beszéd tartalmát szeretném megállapítani valamilyen intelligens algoritmus segítségével, hanem pusztán azt a tényt, hogy egy hangfelvételen emberi beszéd található. Ezért a beszédfelismerés területe helyett a mesterséges intelligencia egy másik területét a tanuló algoritmusokat, azon belül is a neurális hálózatot tárgyalom részletesebben. Ezeknek a hálózatoknak a felhasználása széleskörű; kezdve az approximálási problémáktól, az optimalizálási módszereken át, az osztályozási feladatokig. Ezen alkalmazási területek közül a dolgozat szempontjából lényeges osztályozási feladatokra fogok koncentrálni, vagyis a neurális hálózat egy bináris osztályozást fog megvalósítani, amely megadja, hogy a bemenetére adott hanganyag tartalmaz-e beszédet vagy sem.

Azonban ahhoz, hogy ezt megtegyük, be kell tanítanunk a hálózatot jól ismert példákon keresztül. Nagy előnye ezeknek az algoritmusoknak, hogy rendkívül jól tudnak általánosítani, ami egy ilyen osztályozási feladatnál előnyös tulajdonság. A tanulás kivitelezéséhez azonban megfelelő formára kell hoznunk a hanganyagokat. A megfelelő forma egy előfeldolgozást jelent, amely valamilyen módon jellemzi az adott hanganyagot. Jelen esetben ez az előfeldolgozás a hang frekvenciatérbeli leírását jelenti, amit Fourier transzformáltjának előállításával kaphatunk meg. Azonban, mint látni fogjuk, figyelembe véve bizonyos korlátokat, mint például a tanítófolyamat idő-, tár- és erőforrás igénye, ha egy széles frekvenciatartományt vizsgálunk, akkor a transzformáció eredménye még mindig túl nagy

adathalmaz ahhoz, hogy egy neurális hálózat bemenetét képezze. A program egyik legkritikusabb része, hogy hogyan válasszuk meg azt az eljárást amely ezt az adatmennyiséget oly módon csökkenti, hogy az megfelelőn reprezentatív maradjon.

A dolgozatom témájához az alapötletet az egyik hobbim adta. Szeretek művészfilmeket nézni, amelyeket általában nem szinkronizálnak, ezért megtekintésük során feliratokra van szükség, néha azonban a feliratok nem a tökéletes időpontban jelennek meg. Ekkor született meg bennem az ötlet, hogy valamilyen eljárással megállapítsam mikor beszél egy személy a felvételen, így lényegében megkaphatjuk a feliratok pontos helyét.

2. fejezet

Hallás, hang és beszéd

2.1 A fül

A fül a hallás és egyensúlyozás érzékszerve, három fő egységre bontható: a külső, középső és belső fülre. A hallás az egyik legfontosabb érzékelési lehetőségünk. A hang levegőrezgések során eléri a fület és a hallójáratokon keresztül mozgásba hozza a dobhártyát, ezt követően a hallócsontocskák továbbítják azt a belsőfülben lévő csigának. A csigában ez ingerületet vált ki, amit a hallóideg az agynak megfelelő központjába juttat. Ekkor halljuk meg a hangot, amit az agy feldolgoz és ekkor tudjuk értelmezni azt.

A külső fül részei a fülkagyló és a külső hallójáratot ezt dobhártya választja el a középfültől. A fülkagyló feladata a hanghullámok összegyűjtése és a hallójáratba irányítása. Ezután a hallójárat a hanghullámokat továbbítja a dobhártya felé.

A belsőfül A dobhártyától befelé található a kb. 1 köbcentiméternyi dobüreg. A dobüregben a három, egymással ízülettel kapcsolódó parányi hallócsont van. Ezek a kalapács, az üllő és a kengyel. A hanginger a hallójáraton áthaladva a dobhártyának ütközik amely ezáltal rezgésbe jön. A rezgés a kalapácson és az üllőn keresztül átterjed a kengyelre, ezek közben felerősítik a hangot, amely aztán a belsőfülbe jut.

A belső fülben található a hallás és egyensúlyozás érzéksejt-rendszere. A belső fül a sziklacsont mélyén, a csontos labirintusban helyezkedik el. A hallószerv ezen belül egy előcsarnokból és az abból kiinduló csigából áll és ezeket folyadék tölti ki. A kengyel ezt a folyadékot fogja úgymond meglökni az ovális ablakon keresztül. A csigában található egy háromszög alakú hártya mely a csigát középen két részre osztja, ennek a hártyának az alsó felén találhatóak a szőrsejtek. Ezt a részt alaphártyának nevezik. A hanghullám rezgésbe hozza a csiga felső folyadékterét, melyben végigszalad a hullám, a csúcsban megfordul, visszaszalad az alsó téren a kerek ablakig. A folyamat során az alaphártya rezgésbe jön és az abban lévő különböző hosszúságú húrok rezgésbe jönnek és ott lesz a legnagyobb a kitérése ahol a hangmagasságnak megfelelő húr található. Ez az eltérés magas hangoknál a csiga

alapján, mély hangoknál a csiga csúcsánál a legnagyobb. Ahol a kilengés amplitúdója nagy, ott a szőrsejtek kapcsolatba lépnek a fedőlemezzel és ingert küld az agynak, ami aztán feldolgozza azt.

2.2 hang és beszéd

A hang a fülünk által érzékelt inger. Egy olyan mechanikai rezgés amely egy meghatározott közegben hullámként terjed. Az emberi fül által érzékelt hangok tulajdonképpen a környezetében létrejött rezgések egy hányada. Ilyen mechanikai rezgést létrehozó eszköz lehet egy hangvilla, egy megpendített gitárhúr vagy az emberi hangszálak. A rezgés egy olyan folyamat, melynek állapotai időközönként ismétlődnek. Ha ezek az időközök azonosak akkor periodikus rezgésről beszélünk. Ezeknek a hullámoknak az egyik fontos jellemzője a frekvencia. A frekvencia a másodpercenkénti teljes ciklusok számát jelenti, mértékegysége a Hertz (Hz), ami hullámmozgás esetén a másodpercenkénti teljes hullámok számával egyenlő. A hallhatóság tartománya 20 Hz és 20000 Hz közé esik, vagyis azon hangok amelyek ezen kívül esnek nem hallhatóak az ember számára. A 20 Hz alatti hangokat infrahangoknak, a 20000 Hz feletti hangokat ultrahangoknak nevezzük. Az „A” hang frekvenciája például 440Hz.

A hangok két csoportba bonthatók, az egyik csoport az egyszerű-, a másik a komplex hangok csoportja. A legtöbb hang komplex hang, ami annyit tesz, hogy egy alaphangból és annak egész számú többszöröseiből áll. Ez utóbbiakat nevezzük felharmonikusoknak. Egyszerű hangot leginkább mesterségesen lehet előállítani. Egy átlagos hangminta általában több komplex hangból tevődik össze.

Az emberi beszéd, azaz az információátvitel, a hangképző és a hangészlelő emberi szervek által jön létre. A beszéd alapját, az áramló levegőt a tüdő biztosítja. Ez a levegő átáramlik a hangszálak között a garatba, az orrnyílásba és a szájüregbe. A beszéd összetett hangokból áll. Az ember alaphangjának frekvenciája körülbelül 60Hz és 300Hz közötti és egyedi minden egyes embernél. A férfiak alaphangja körülbelül a 60Hz – 200 Hz-es frekvenciatartományon található. Átlagosan egy férfi hangja 125Hz körül van, tehát a 60Hz egy igencsak ritka eset. A nőknél ez a frekvenciatartomány a 120Hz – 300Hz közé esik. A gyerekek beszédének alaphangja 300Hz körüli. Ezek a tartományok már a szélsőséges eseteket is tartalmazzák.

Természetesen az emberi beszéd is tartalmaz felharmonikusokat, amit a hallószervünk felfog, vagyis maga a beszédértés nem csak ezen a kis tartományon történik. A beszéd nagyjából a 60 Hz és 4,000 Hz közötti frekvenciatartományba esik. A magánhangzók többnyire 1,000 Hz alatti tartományba esnek, a mássalhangzók ezzel szemben 1,000 Hz felett helyezkednek el. A beszédértésben a mássalhangzók nagyobb szerepet játszanak, de dolgozatom tárgyalásánál a beszédértés cél.

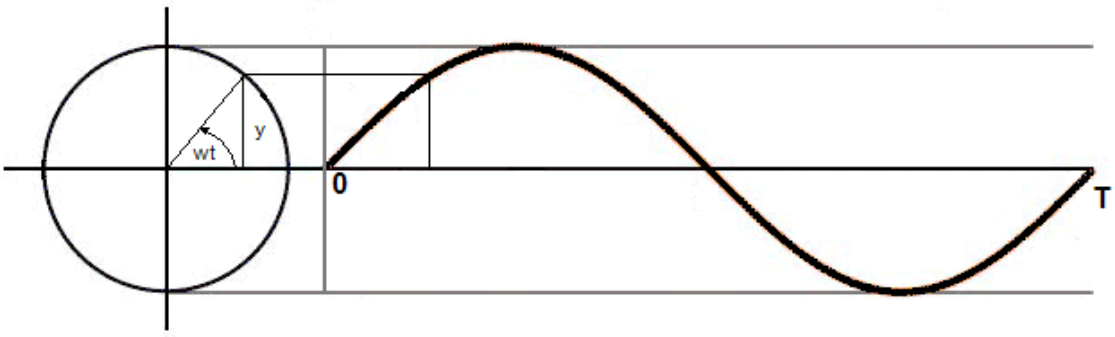
3. fejezet

Fourier Transzformáció

Egy időben változó jel előállítható különböző frekvenciájú, fázisú és amplitúdójú jelek összegeként. A Fourier-transzformáció az a művelet, amely egy adott jelhez megadja ezt a felbontást.

3.1 Függvények felbontása

Egy egyszerű mechanikai rezgést illusztrálhatunk egy rugóra függesztett testtel, amit kimozdítunk nyugalmi helyzetéből. Ha a test mozgásának nyomon követjük az időbeli változását, akkor egy szinusz hullámot kapunk. Ezt a rezgő mozgást modellezhetjük egy körmozgással, pontosabban egy *r* sugarú körpályán egyenletes ω szögsebességgel keringő ponttal, melynek ábrázoljuk a vetületét az idő függvényében. Vegyünk egy ilyen mozgásban lévő testet. Tegyük fel, hogy a rugóra felfüggesztett test éppen a nyugalmi állapotnak megfelelő pozícióban van és felfelé mozdul majd ebből az állapotból. Ekkor az ábrán látható módon egy szinusz hullámot kapunk, aminek a periódus hossza T. Ez azt is jelenti, hogy ennek a szinusz jelnek a frekvenciája f = 1/T. Vagyis 1/T idő alatt tesz meg 1 teljes periódust a hullám.

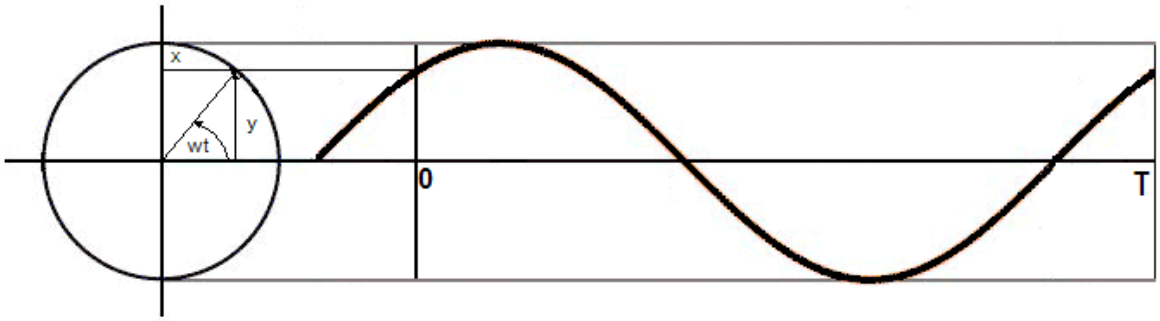


Egy ilyen szinusz hullám esetén adott t időpontban vett y kitérésnek a kiszámítása a következő:

$$y(t) = r \cdot \sin(\omega \cdot t);$$

Ha azonban egy időben változó periodikus jelet csak ilyen különböző fázisú szinusz hullámokra próbálnánk felbontani rengeteg értékes frekvenciabeli rezgés eltűnhet, ugyanis kimaradnak azok a frekvenciához tartozó hullámok amelyek periódusa nem a t_0 időpontban kezdődik.

Tekintsük meg a következő ábrát, amelyen egy eltolt fázisú szinuszos hullámfüggvényt látunk, vagyis a függvényt eltoltuk negatív irányba az x tengelyen. A frekvenciája ennek a jelnek is 1/T. A különbség csak annyi, hogy a 0. időpillanatban a rugóra felfüggesztett tárgy, illetve az őt modellező körfüggvény, most nem az előbbi nyugalmi állapotnak megfelelő pozícióban van, ahogy azt az ábrán láthatjuk. Ebből az következik, hogy az ezt leíró függvény megadásához nem elég a fenti képlet.

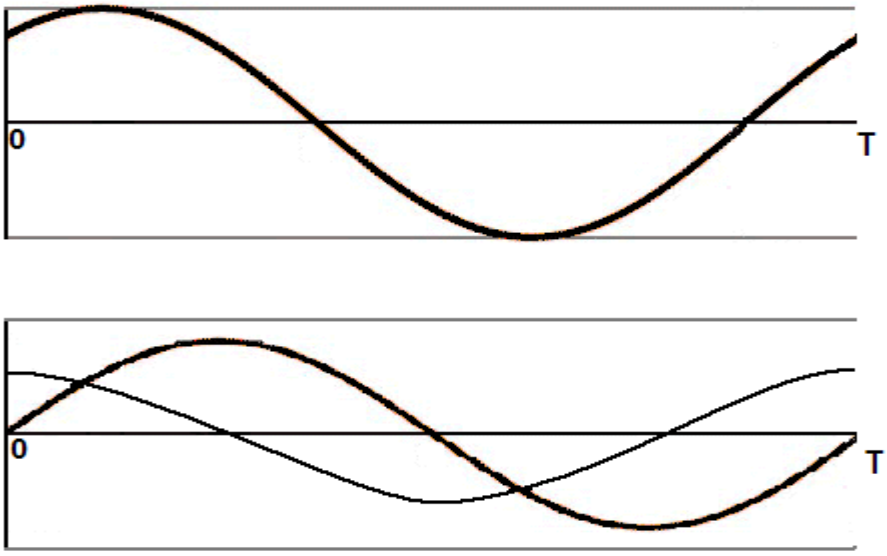


Azonban, ezt a függvényt fel lehet bontani másik két függvény összegére. A két képlet pedig az idő függvényében a következő lesz:

$$x(t) = r_1 \cdot \cos(\omega \cdot t)$$

$$y(t) = r_2 \cdot \sin(\omega \cdot t)$$

Ezt elképzelhetjük úgy, hogy két rugóra függesztett tárgyunk van. Mind a két test azonos sebességgel mozog azonban az amplitúdójuk eltérő. Valamint a kezdő időpontban az egyik test legyen a nyugalmi állapotnak megfelelő helyzetben, a másik viszont a mozgásának legmagasabb állapotában. Ha ezeket a testeket modellezzük egy-egy körmozgással és megnézzük az előzőek szerint a vetületeiket, akkor egy koszinusz és egy szinusz hullámot kapunk. Ha a vetületeket összegezzük, kapunk egy szinuszos hullámfüggvényt. Tegyük fel, hogy a két vetületi függvény-t úgy kaptuk, hogy az előző képletek szerint felbontottunk egy eltolt fázisú függvényt koszinusz illetve szinusz függvényre. Ekkor ezeket összegezve visszakapjuk az eredeti függvényt. Valahogy úgy ahogy az ábrán láthatjuk (az ábra nem pontos csupán szemléltetési célból szerepel).



Ha adott a szinusz és koszinusz hullám és az összegük amplitúdójára vagyunk kíváncsiak, akkor azt a Pitagorasz-tétel alkalmazásával könnyen kiszámíthatjuk:

$$A = \sqrt{x(t)^2 + y(t)^2} \ .$$

Ha az előző két pont körmozgását azonos origójú körökben vizsgáljuk, feltűnhet hogy az azonos szögsebesség miatt a két pont és az origó meghatároznak egy háromszöget, ami sohasem változik. A háromszög két befogója lesz a két pont sugara, vagyis a vetület által leírt

szinusz és koszinusz hullám amplitúdója, az átfogó pedig az összegük amplitúdója. Ezt a tényt egyszerűbb elképzelni ha vektorként tekintünk a két pontra.

Ezek után ha van egy időben változó függvényünk és a különböző frekvenciájú hullámok szinusz és koszinusz összetevőit keressük, akkor nem fog kimaradni azon frekvencia hullám amely nem pont a t_0 időpontban kezdődik.

3.2 Mintavételezés

Mivel a hanganyagokat számítógépes szoftver segítségével dolgoztam fel, ezért szükséges volt az analóg jelek diszkretizálására. Ha diszkrét jelekkel kívánunk dolgozni, akkor először mintavételezni kell a folytonos függvényünket. A mintavételezési eljárás során a folytonos jelből kiválasztjuk az értékek egy véges halmazát, ezek a kiválasztott értékek a minták. Hangfelvétel esetén ezek a minták diszkrét időpillanatban vett értékek, tehát a folytonos függvény egy időpillanatában meghatározott értéke. Általában ekvidisztáns mintavételezést alkalmazunk, vagyis az egymást követő két mintavételi (idő)pont távolsága azonos. Természetesen lehet ettől eltérő, azonban a jelen dolgozatban minden mintavételezés ekvidisztáns osztópontokkal történik.

H. Nyquist és Shannon-féle mintavételezési tétel:

A mintavétellel nyert diszkrét mintákból álló impulzussorozat információtartalma megegyezik az eredeti, időben folytonos analóg jel információtartalmával, az alábbi feltétel teljesülése esetén (Shannon-féle mintavételi tétel):

> A mintavétel útján nyert jelből akkor lehet az eredeti jelet információveszteség nélkül visszaállítani, ha az f_m mintavételi frekvencia értéke legalább kétszerese az eredeti analóg jelben előforduló legnagyobb $f_{\max}$ frekvenciának. A mintavételi frekvencia értékének állandónak kell lennie.
>
> Vagyis: $f_m > f_{\max}$

Ezt a megállapítást figyelembe kell venni a frekvencia összetevőkre való felbontásnál, hogy tudjuk mennyi az a maximális frekvencia érték amely veszteség nélkül számolható.

3.3 Fourier transzformáció

A diplomamunkám céljának elérése érdekében, szükség volt az egyes hangadatok különböző frekvenciájú hullámfüggvények összetevőkre bontásához. Ehhez a feladathoz kézenfekvő módszer a Fourier transzformáció.

3.3.1 Folytonos eset

Legyen $f(x)$ folytonos, és a valós számok halmazán értelmezett függvény. Ekkor a Fourier transzformációja következő módon definiált:

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

Megjegyzés: $e^{ix} = \cos(x) + i\sin(x)$

Mint látható ez egy komplex értékű függvény, amit felírhatunk az alábbi módon:

$$F(k) = \mathrm{Re}(k) + i \cdot \mathrm{Im}(k)$$

Ahol $\mathrm{Re}(k)$ és $\mathrm{Im}(k)$ a valós és képzetes része $F(k)$ -nak.

Az inverz Fourier transzformált pedig a következő:

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i k x} dk$$

3.3.2 Diszkért eset (DFT)

Ha az $f(x)$ függvényt mintavételezéssel diszkretizáljuk. Legyen ez a diszkrét függvény $g(x)$ ekkor. A diszkrét Fourier transzformáltja ennek a függvénynek a következő:

$$G(k) = \frac{1}{N} \sum_{x=0}^{N-1} g(x) e^{\frac{-2\pi ikx}{N}}$$

A $G(k)$ inverz diszkrét Fourier transzformáltja:

$$g(x) = \frac{1}{N} \sum_{k=0}^{N-1} G(k) e^{\frac{2\pi ikx}{N}}$$

3.3.3 A gyors Fourier transzformáció (FFT)

A diszkrét Fourier transzformáció (DFT) egyik nagy problémája az időigény, a rengeteg szorzásművelet miatt a műveltigénye nagyon nagy. N szám transzformálása esetén N^2 szorzást kell elvégezni. Cooley és Tukey amerikai matematikusok által kidolgozott algoritmus ezt az időigényt csökkentette radikálisan.

Az FFT alapgondolata a következő: N pont DFT-je N^2 szorzást igényel. Az adatokat két $N/2$ elemszámú részre bontjuk, akkor a két rész műveletigénye $2(N/2)^2$ lesz. N-et célszerű 2 egész kitevőjű hatványának választani.

Legyen v a kiindulási adatsor. Bontsuk fel g és h adatsorra úgy hogy az előző a páros utóbbi páratlan elemeket tartalmazza. A diszkrét Fourier transzformációt alkalmazva a két adatsorra képletének megkapjuk a *G* és *H* részspektrumokat. Ezek összekombinálásához csak w^k-nal történő szorzás kell. Ahol $w = e^{\frac{i2\pi}{N}}$.

Két részre osztjuk a kiinduló adatsort:

$$v(n) \ \ (n = 0,1,...,N-1) \ \Rightarrow g(n) = v(2n) \text{ és } h(n) = v(2n+1) \ \ (n = 0,1,...,N/2-1)$$

Melyek után a Fourier transzformált a következő alakban írható fel.

$$V_k = \sum_{n=0}^{N-1} v_n w^{-nk} = \sum_{n=0}^{N/2-1} g_n w^{-2nk} + \sum_{n=0}^{N/2-1} h_n w^{(2n+1)k} \qquad (k = 0,1,\ldots, N-1)$$

$$w_N^2 = w_{N/2}$$

$$V_k = \sum_{n=0}^{N/2-1} g_n w_{N/2}^{-nk} + w_N^{-k} \sum_{n=0}^{N/2-1} h_n w_{N/2}^{-nk} = G_k + w_N^k H_k \qquad (k = 0,1,\ldots, N/2-1)$$

Ha $k > N/2-1$ -nél, akkor $k - N/2$ -vel helyettesíthető, így végül

$$V_k = G_{k-N/2} + w_N^{-k} H_{k-N/2} \qquad N/2 \le k \le N-1,$$

$$V_{k+N/2} = G_k + w_N^{-k+N/2} H_k \qquad 0 \le k \le N/2-1.$$

Ennek a módszernek az alkalmazásához már csak $(N/2)\log_2 N$ művelt végrehajtására van szükség szemben a diszkrét Fourier transzformáció N^2 -es műveletigényével, ami jelentősen meggyorsítja az adatok előállítását.

4. fejezet

Neurális hálók

A neurális tudományok az idegrendszert, illetve az agy felépítését és annak működését tanulmányozzák. A gondolkodás mechanikája a mai tudomány számára még mindig az egyik legnagyobb feladvány, jelenleg is számos ismeretlen, feltárásra váró rejtély áll a tudósok előtt az emberi gondolkodás feltérképezését illetően. Az ma már világos a tudósok számára, hogy a tanulásért, a gondolkodásért és egyéb szellemi tevékenységek elvégzésért az agy a felelős szerv. A különböző kutatások segítették feltárni az egyes cselekvésekért, érzékelésekért és más funkciókért felelős agyi területeket. Az agy neuronokból való felépítése ekkortájt nagyjából már szintén tisztázott volt. A neurális hálózatok alapötlete, hogy az agy működését megpróbáljuk lemodellezni oly módon, hogy az egyes neuronok között kapcsolatot építünk fel amelyek akkor aktivizálódnak, ha megfelelő számú szomszédos neuron stimulálja. Ezek után jött az ötlet, hogyha a kapcsolatok módosítható súlyokat is kapnak, akkor tanulásra is képes lesz a háló. Ebben a fejezetben ezt a modellezést mutatom be. Rövid és egyszerű áttekintést nyújtva a neuronok közötti információ továbbításról, majd a lényegesebb modellezési módszerekre térek rá.

4.1 Biológiai Neuron

Az idegsejt (neuron) az idegrendszer alapvető strukturális és működési egysége. Olyan sejt, mely képes információt befogadni, feldolgozni és továbbítani más idegsejteknek, izmoknak, mirigyeknek.

Az idegsejt alkotói:

- sejttest: a sejtmagot és egyéb alkotóelemeket tartalmaz.
- axon: Az információt, továbbítja
- dendritek, amelyek (együtt a sejttesttel) a szomszédos neuronok felől érkező információt fogadják

Központi idegrendszerünk mintegy 1–10 milliárd idegsejtből áll, ezek nagyjából 1000 másik idegsejthez kapcsolódnak a szinapszisokkal. Ha az egyik idegsejt ingerületbe jön, akkor a

fogadó idegsejt dendritjei érzékelik. Majd ezeknek az idegsejteknek az axonjain továbbítódik a szomszédos idegsejtbe. Az információfeldolgozás az agyban ilyen hálózaton keresztül történik.

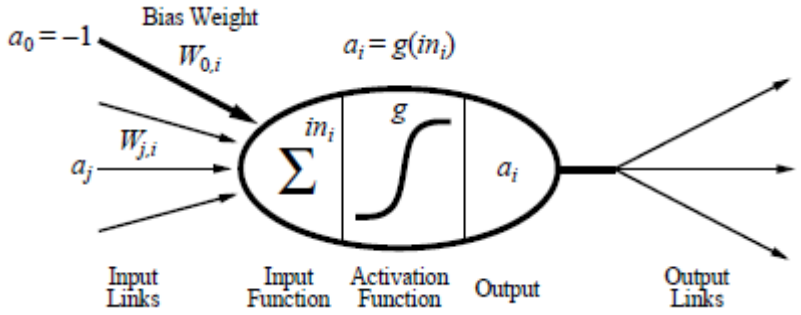
4.2 Neurális háló elemei

A neurális hálók irányított kapcsolatokkal összekötött egységekből állnak. A lenti képen látható a neuron sematikus modellje. Az i-edik neuron a_i aktivációját a következőképpen kapjuk:

$$a_i = g(\sum_{j=0}^{n} w_{j,i} a_j)$$

Ahol: a_j a j-edik egység aktivációja $w_{j,i}$ pedig a j-t az i-vel összekötő kapcsolat súlya (Input Links), illetve a $w_{0,i}$ az eltolás súly (Bias Weight) és $a_0 = -1$ rögzített érték.

Tehát a bemenetként kapott értékeket megszorozzuk az adott súllyal és ezeket összegezzük, valamint hozzá adunk még egy eltolás súlyt, amire a későbbiekben térek ki. A súlyok az adott kapcsolat erősségét határozzák meg. Ezután az a_i kimeneti értéket úgy kapjuk meg, hogy egy aktivációs függvényt (Activation Function) alkalmazunk.



Az aktivációs függvénynek olyannak kell lennie, hogy +1 körüli értéket adjon ha a jó bemenetet kapja és 0 körüli ellenkező estben. A két ilyen lehetséges aktivációs függvény küszöb függvény és a szigmoid függvény. Az első 1-et ad eredményül ha a bemenete pozitív 0-t egyébként. A szigmoid függvény képlete pedig : $f(x) = 1/(1+e^{-x})$. Az eltolás súlynak

itt van nagy szerepe. Ugyanis azzal állíthatjuk be az aktuális küszöbpontot, ami a függvény nullánál felvett értéke. Tehát a neuron akkor „tüzel” ha a $\sum_{j=1}^{n} w_{j,i} a_j$ meghaladja $w_{0,i}$ értékét.

4.3 Hálózatok felépítése

Az előrecsatolt hálózatok rétegekből épülnek fel. Három megkülönböztetett réteg van: az inputréteg, a rejtett réteg és az outputréteg. Minden réteg csak a közvetlenül megelőző rétegtől kap bemeneti jelet. Természetesen az első réteg mindig az input, az utolsó pedig mindig az outputréteg. Legyen az $i.$ réteg a $j.$ közvetlen megelőzője ekkor az $i.$ réteg összes egysége kapcsolódik a $j.$ réteg minden elemével. Ezt természetesen csak egy általános felépítési mód. Léteznek úgynevezett receptív mezők, melyek annyiban térnek el az előzőtől, hogy csak egy meghatározott halmaza van az előző réteg egységei közül, az adott réteg adott egységére rákötve. Másik lehetséges strukturális módosítás, ha az egységek kimenete nem csak a következő réteg egységeihez kapcsolható, hanem visszacsatolhatjuk őket a bemenetre, ezzel létrehozva egy sokkal bonyolultabb hálószerkezetet, amely egyfajta memorizáló szerepet játszik. Ezek a módszerek azonban túlnyúlnak jelen dolgozat keretein.

A neurális hálók a tanulást a súlyok módosításával végzik. Vegyünk olyan input értékeket, amelyek ismert a kimenete. Végigfuttatjuk őket egyesével és minden példa után az algoritmus módosítja a súlyokat a hiba csökkentése érdekében. Az összes minta végigfuttatását nevezzük 1 epochnak. Az epochokat mindaddig ismételjük, amíg valamilyen leállási feltétel nem teljesül. Többnyire az a leállási feltétel, hogy a súlyok már nagyon kis mértékben változnak a megelőző epochhoz képest. A hiba mérésére általában a négyzetes hibaösszeget használjuk.

4.4 háló struktúrák

4.4.1 Perceptron

Perceptronnak, vagy 1 rétegű neurális hálónak nevezzük azt a hálózatot, amelynek az összes bemeneti egysége közvetlenül a kimentre van rákötve. Ez azt jelenti, hogy minden súly csak

egy kimenetre van hatással. Vegyük azt az esetet, amikor egyetlen kimenti egység van. Legyen $\underline{x}$ a bementi vektor a $\underline{w}$ súlyvektor. Ekkor a perceptron csak akkor ad 1-et, ha $\underline{w} \cdot \underline{x} > 0$. A $\underline{w} \cdot \underline{x} = 0$ egyenlet egy hipersíkot határoz meg a bemeneti térben. Vagyis a hipersík egyikoldalán lesznek azok az bemenetek, amelyek 1-et adnak kimentként, a másik oldalán pedig a többi.

4.4.2 Többrétegű előrecsatolt háló

Ebben a részben olyan hálózatról lesz szó amely már rendelkezik rejtett réteggel. A legelterjedtebb változata az egy rejtett réteget tartalmazó hálózat, tehát a rejtett réteg egy neuronja küszöbfüggvényt reprezentál a bemeneti térben, majd a kimeneti réteg minden neuronja szintén küszöbfüggvényt reprezentál, azonban már a rejtett réteg kimenetén. Ha a rejtett rétegünk kellően nagy, akkor a bemenetek tetszőleges folytonos függvénye reprezentálható. Több rejtett réteget tartalmazó hálózatok nemfolytonos függvények is reprezentálhatóak. N bemenetű logikai függvény esetén $2^N / N$ db rejtett neuronra van szükség ha 1 rejtett réteget szeretnénk. A minimálisan szükséges neuronok száma azonban nehezen meghatározható. Az *Optimal Brain Damage* egy olyan algoritmus, amely segíthet a háló méretének csökkentésében. Ehhez egy teljesen összekötött hálóból indulunk ki és összeköttetéseket távolítunk el belőle. Minden tanítás után kiválasztjuk azokat az összeköttetéseket, amelyeket elhagyunk, majd újra tanítjuk és ha nem romlott a teljesítmény akkor tovább folytatjuk. Az algoritmus eltávolíthat olyan csomópontokat is amelyek nem járulnak hozzá nagyban a megoldáshoz.

4.4.3 Tartóvektor Gép (Support Vector Machine)

Az egyrétegű hálók csak lineáris döntési határokat képesek megtanulni. Ezzel szemben a többrétegű hálók már képesek általános, nem lineáris függvényeket reprezentálni, azonban betanításuk bonyolult. A tartóvektor gépek képesek nemlineáris függvények reprezentálására és tanítási algoritmusuk is hatékony. Legyen a bemeneti téren értelmezve az $\underline{x}$ vektor, melynek elemei az attribútumok. Tegyük fel továbbá, hogy a probléma megoldására nem létezik lineáris szeparátor. Azonban ha ezt a vektort átalakítjuk másik $F(\underline{x})$ vektorra az attribútumokon végrehajtott valamilyen műveletek segítségével, akkor az ilyen módú

megfelelő átalakítással az új vektor lineárisan szeparálható lesz. A tartóvektor gépek rendszerint az optimális lineáris szeparátort találják meg. Az optimális itt az, amelyik az összes lineáris szeparátor közül a legnagyobb a tartalékkal rendelkezik. Más szóval a szeparátorhoz legközelebb eső pont távolságát az egyik oldalon összegezzük a másik oldalán lévő legközelebbi pont távolságával, akkor ez az érték maximális. Ez kvadratikus programozással megoldható optimalizálási feladat.

Legyenek $\underline{x}_i$ pontjaink és ezeket szeretnénk két osztályba sorolni egy optimális lineáris szeparátorral. Ekkor a kimenet attól függően, hogy egyik vagy másik osztályba tartozik az adott bemenet: $y_i = -1 \vee y_i = 1$. A megoldandó kvadratikus programozási feladat, hogy α_i-ket meghatározzuk $\alpha_i \geq 0$ és $\sum_i \alpha_i y_i = 0$ feltételek mellett, maximalizálva a következő kifejezést:

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\underline{x}_i \cdot \underline{x}_j) .$$

Ezek után az optimális szeparátort a következő módon kapjuk:

$$h(\underline{x}) = sign\left(\sum_{i,j} \alpha_i y_i (\underline{x} \cdot \underline{x}_j) \right)$$

Ennek a szeparátor síknak fontos tulajdonsága, hogy az egyes adatpontokkal asszociált α_i súlyok mind nullák, leszámítva a szupport vektorokat, amik a szeparátorhoz legközelebb eső pontok.

Nem minden esetben van lineáris szeparátor az $\underline{x}_i$ bemeneti téren. De egy sokdimenziós $F(\underline{x})$ tulajdonságtérben találhatunk megfelelő szeparátort. Ha az egyenletben $\underline{x}_i$ vektort kicseréljük $F(\underline{x}_i)$ tulajdonságvektorra, az $\underline{x}_j$ vektort pedig $F(\underline{x}_j)$ tulajdonságvektorra, akkor $\underline{x}_i \cdot \underline{x}_j$ szorzat helyett $F(\underline{x}_i) \cdot F(\underline{x}_j)$ fog szerepelni. Ez azért lényeges mert

$F(\underline{x}_i) \cdot F(\underline{x}_j)$ gyakran kiszámítható anélkül, hogy először minden pontra kiszámítanánk külön-külön. Vagyis megadható egy olyan kernelfüggvény amelyre igaz hogy:

$$K(\underline{x}_i, \underline{x}_j) = F(\underline{x}_i) \cdot F(\underline{x}_j).$$

Tehát ha az alap képletben az $\underline{x}_i \cdot \underline{x}_j$ szorzatokat kicseréljük erre a $K(\underline{x}_i \cdot \underline{x}_j)$ kernelfüggvényre, akkor $F(\underline{x})$ tulajdonságtérben fogunk lineáris szeparátorokat találni, viszont nem kell az összes pontra kiszámítani az $F(\underline{x})$ értékét, elég csak a kernelfüggvényt kiszámítani. Így a tanulást a sokdimenziós tulajdonságtérben végezhetjük el.

5. fejezet

A program előkészítése

A továbbiakban az általam készített program működését fejtem ki, a hangadatok feldolgozásától kezdve a neurális hálózat betanításáig, továbbá megvizsgálok különböző módszereket, amelyek a tanulás sikerességét befolyásolhatják, valamint ezen különböző módszerek hatékonyságának elemzését tárgyalom részletesebben. A probléma megoldása két részből tevődik össze. Az első része egyfajta előfeldolgozás, ez a rész Java nyelven íródott. Ez az előfeldolgozás abból áll, hogy a hangadatokat megfelelő formára képezzük le, ami jelen esetben a Fourier transzformált előállítása, illetve a transzformáltak valamilyen később tárgyalt módosítása. Ennek az előfeldolgozási résznek az eredményeképpen kapunk egy fájlt, ami tartalmazza az összes feldolgozott hang megfelelő adatait. A probléma megoldásának másik része ezen eredmények elemzése. Az adatokat egy 1 rejtett réteget tartalmazó neurális hálózat, illetve egy tartóvektor gép (SVM) segítségével fogjuk feldolgozni. Ezeket az eljárásokat R nyelven írt program fogja megvalósítani, amelyhez szükségünk lesz két csomagra. Az *nnet* csomag az 1 rejtett réteget tartalmazó neurális hálózat megvalósításához szükséges. A másik csomag az *e1071* melyet az SVM használatához szükséges. Az R egy nyílt forráskódú statisztikai és grafikai környezet és mint programozási nyelv a funkcionális paradigmát valósítja meg.

5.1 WAVE file formátum

A program, ami a hangokat feldolgozza WAVE formátumú adatokat használ, ezért egy gyors áttekintést fogok adni róla a következő részben. A függelék tartalmaz egy ábrát, ami szemlélteti a felépítését és tartalmaz egyéb adatokat a különböző mezőkre vonatkozóan. A WAVE fájl formátum a Microsoft RIFF formátumának egy speciális változata. A RIFF fájl specifikusan multimédiás adatok tárolására kifejlesztett formátum. A RIFF fájl első része egy fej rész, ami specifikálja milyen információt tárolunk az adott fájlban. A wav fájl esetén az ábrán látható Format mezőben a WAVE szó található. Általában a WAVE formátum esetén a fejrész után két további rész következik: az egyik az *fmt* rész a másik a *data* rész. Az *fmt* rész

tartalmazza speciális módon, a wav fájl által tartalmazott, felvételre vonatkozó információkat. Az audioformat, a kódolásra vonatkozó információt tartalmazza, ha ez az érték 1, akkor PCM kódolású az adat. Az *fmt* rész tartalmaz még egy az ábrán NumChannels-ként feltüntetett mezőt, amely megadja, hogy hány csatornás a hanganyag (mono = 1, stereo = 2, stb). A Samplerate mező a mintavételezési frekvencia, az egy másodperc alatt vett minták számát határozza meg. Lényeges még a téma szempontjából a BitsPerSamples mező, amely arról ad információt, hogy a minták hány biten vannak tárolva. A *data* rész a tényleges hanganyagot tartalmazza.

5.2 Hanganyagok tárolása és formai követelményei

A hangadatok tárolása kulcsfontosságú a feldolgozás szempontjából. Továbbá a neurális hálózat betanítása érdekében, valamilyen módon jelezni kell azt, hogy mely felvételek tartalmaznak emberi beszédet, illetve melyek azok az adatok, amelyek zajt tartalmaznak, de emberi beszédet nem. Ennek egy igen egyszerű módja ha a két halmaz elemeit elkülönítjük egy-egy külön területre és a beolvasást ezután úgy végezzük, hogy figyeljük, melyik területről olvassuk be.

A tesztelés során a felvételeket az ingyenes Audacity nevű szoftverrel rögzítettem, amely könnyű szerkesztési lehetőséget biztosított, hogy a hangfelvételeket a megfelelő alakra tudjam hozni. Az előző témakörben leírtak alapján, a tesztelés során a feldolgozás megkönnyítése érdekében, meghatározott formátumú wav fájlokat használtam. Minden felhasznált wav file PCM kódolású, 1 csatornás (mono), 44100 mintavételi frekvenciával és az adatok 16 biten ábrázoltak. Fontos, hogy wav fájlokban az adatok *little-endian* módon vannak tárolva ezért a beolvasásuknál oda kell figyelni.

5.3 A program beállítása

Az előző részben említettem, hogy a beszédet tartalmazó, illetve nem tartalmazó fájlok két elszeparált helyen tárolandók. A program ezeknek a helyét egy, a program könyvtárán belül megtalálható setup.ini fájlból olvassa ki. A *voicefolder* és a *novoicefolder* paraméterek módosításával megváltoztathatjuk a hangot tartalmazó, illetve a hangot nem tartalmazó file-

ok helyét. A program ezeket a wav fájlokat sorban dolgozza fel, ha az elérési út nem létezik az egyiknek, akkor az eredmény azon halmazba tartozó mintát nem fog tartalmazni. Ha egyik sem létezik, vagy mind a kettő üres, akkor a program eredmény nélkül ér véget.

Lehetőség van ezeken felül egyéb paraméterek megváltoztatására is a setup.ini nevű fájlban. Az *outputfile* paraméterrel, a kimeneti fájl helyét és nevét tudjuk megadni. Amennyiben nem létezik a fájl, akkor a program létrehozza azt, ha azonban létezik, akkor egy másik paramétertől az *append*-től függ, hogy a meglévő fájlt eldobjuk, vagy az eredményeket a fájl végéhez fűzzük. Ennek az értéke lehet *true* ami azt jelenti, hogy a hozzáfűzést választjuk, vagy lehet *false*, amely a fájl törlését és új állomány létrehozását eredményezi. A végére fűzés sokat segíthet, ha nem egyszerre akarjuk feldolgozni az adatokat. Hasznos lehet ez a beállítás abban az eseben, ha a különböző hang felvételeket szeparáltan tároljuk és nem egyszerre akarjuk feldolgozni őket.

A *samplenum* paraméter arra szolgál, hogy megadjuk azt az elemszámot, amely meghatározza, hogy mennyi mintát dolgozzunk fel egyszerre. Például ha vesszük az alapbeállítást, ami 65536, akkor 65536 db mintát olvas be egyszerre a program az aktuális wav fájlból. Ennek a beolvasott mintának ezután vesszük a Fourier transzformáltját és a megfelelő alakra hozás után az eredményt egy fájlba írjuk. Ezután, amennyiben maradt még *samplenum* számú minta hátra, akkor beolvassa és végrehajtja ezen is a Fourier transzformálást és kiírja a fájlba a megfelelő módon. Ezt addig végzi, amíg be tud olvasni *samplenum* számú mintát. Ha már nem tud többet beolvasni, akkor a következő fájlra lép az algoritmus és ezen is elvégzi ez előbb leírtakat mindaddig, amíg nincs több fájl. Fontos azonban megjegyezni, hogy a mintavételezési tétel miatt pontosan csak a mintavételezési frekvencia feléig kapunk hasznos adatokat. Vagyis ha 44100 a mintavételezési frekvencia, akkor a maximális frekvencia, amit a minta tartalmazhat 22050Hz.

Könnyen kiszámolható, hogy a beállított *samplenum* esetén mennyi időt dolgozunk fel. Ha *samplenum* értékét elosztjuk a mintavételezési frekvenciával, akkor pontosan ezt az időt kapjuk. Ezek szerint ha megtartjuk az alapértéket, a 65536 mintát, akkor 65536 / 44100 ≈ 1.486 másodpercnyi adat feldolgozása történik egyszerre.

A Fourier transzformált eredménye egy komplex számokat tartalmazó tömb. A komplex érték valós része a koszinusz hullámok amplitúdóját, a képzetes része a szinusz hullámok

amplitúdóját adja meg. A tömb i-edik eleme a mintasor alatt i periódusú koszinusz, illetve szinusz hullám amplitúdóját jelenti. Fontos megjegyezni, hogy az i-edik érték nem azt jelenti, hogy az egy i frekvenciájú függvény. Ha az i-edik érték frekvenciáját akarjuk kiszámolni, akkor az i-t el kell osztanunk a feldolgozott adatok időtartamával. Például ha 44100 mintavételi frekvencia mellett az alapérték szerinti 65536 db mintát veszünk, akkor az előbb kiszámítottak alapján az időtartama 1.486 másodperc. Ha az i-t elosztjuk ezzel az időtartammal, akkor megkapjuk az i. értékhez tartozó frekvenciát. Tehát ebben az esetben tömb 20. elemének valós és képzetes része a 20 / 1.486 ≈ 13,459 Hz-es hullámoknak felelnek meg.

Ha a *samplenum* paramétert 2 valamely egész kitevőjű hatványával egyenlő, akkor a program automatikusan gyors Fourier transzformációt fog alkalmazni, ellenkező esetben simán diszkrét Fourier transzformációt. Mivel ez jelentősen felgyorsítja a program futását érdemes ezt a lehetőséget kihasználni.

Az eddigi 5 paraméter mellett (*samplenum, voicefolder, novoicefolder, ouputfile, append*) szerepel a setup.ini fájlban további három. Ezek közül a *lowerfreq* és az *upperfreq* a frekvenciatartomány szűkítését szolgálják. A *lowerfreq* az alsó frekvenciahatárt tudjuk módosítani, az *upperfreq* paraméterrel pedig a felső frekvenciahatárt. Ha a feldolgozandó elemszám 2 valamely egész kitevőjű hatványával egyenlő, tehát gyors Fourier transzformációt alkalmazunk, akkor a kiírt eredmények tartománya szűkül csak. Ha ez az elemszám nem 2 hatvány, akkor diszkrét fourier transzformáltat alkalmaz az algoritmus. Ekkor viszont nem csak a kiírás tartománya csökken, hanem a számítás is csak arra a tartományra korlátozódik, amit megadtunk. Ez valamennyivel gyorsítja ugyan a DFT-t, azonban az FFT még így is sokkal gyorsabb. Az utolsó paraméter a *numofsubdomain*. Ennek a paraméternek a segítségével egy szűkítést végezhetünk a transzformáltak eredményein. Ha ezt az értéket 0-ra állítjuk, akkor a transzformált teljes eredményét megkapjuk a frekvencia tartományon belül. Ha az értéket 0-nál nagyobb értékre állítjuk, legyen ez az érték N, akkor az előbb megadott tartományt N db egyenlő részre daraboljuk és ezeknek a résztartományoknak az átlagát vesszük.

Ezen három paraméter beállítása kulcsfontosságú az adatok méretének csökkentésében. Továbbá figyelembe kell venni, hogy ha a résztartományokat nagyon nagyra választjuk, akkor

az információ vesztés nagy lesz. Természetesen az ideális az lenne ha Fourier transzformált eredményét szimplán ráengedhetnénk a neurális hálózat bemenetére. Azonban ekkor egy jól működő hálózat felépítése túlságosan nagy lenne, ezért szükséges valamilyen módon csökkenteni az adathalmaz számosságát, erre az egyik módszer a résztartományok átlagolása. A vizsgálandó frekvencia tartomány és a pontos résztartományok számának megfelelő megválasztása kulcsfontosságú abból szempontból, hogy az adatok kellőképpen reprezentatívak maradjanak az átalakítás után is.

6. fejezet

Elemzés

Az előző fejezetben leírtam az általam készített programok beállításához szükséges információkat, illetve a program működését. Ebben a részben a beszéd azonosításának különböző lehetőségeit vizsgálom és elemzem. A 6.1. alfejezetben az elemzéshez szükséges tesztadatok előállításáról lesz szó. A 6.2. alfejezetben a különböző módszerek hatékonyságát tárgyalom.

6.1 A tesztadatok elkészítése

A tesztadatok összeállítása kulcsfontosságú a probléma szempontjából. A neurális hálók tanításához szükség van olyan adatokra melynek ismerjük a kimenetét. Jelen esetben rengeteg olyan felvételre volt szükség amin egy személy beszél illetve olyanokra amik nem tartalmaznak emberi beszédet.

Azon tesztadatok előállításában, amelyek a beszédet tartalmazzák három személy vett részt: két férfi és egy nő. Különböző szavakat és rövid mondatokat olvastak fel, melyeket ezután a fent leírtak szerint a megfelelő formára hoztam. Ezeknek a hanganyagoknak a sajátossága, hogy a beszédtartalma mindegyiknek 0 másodperc és 1,5 másodperc közötti időben találhatóak. A felvétel az algoritmus miatt hosszabb mint 1,5 másodperc de, hogy ne kerüljön a beszédet tartalmazó halmazba helytelen minta ezért nem tartanak 2,5 másodpercnél tovább. Ez azért fontos mert a tesztelés során kizárólag 65536-os elemszámmal dolgoztam és ahogy fentebb leírtam a minták mindegyike 44100-as mintavételezési frekvenciával kerültek rögzítésre, ami azt jelenti, hogy (65536 / 44100 ≈) 1.486 másodpercnyi adat spektruma készült el. Természetesen az algoritmusnak nem okoz problémát ha hosszabb a hanganyag. Ilyen esetekben, a hanganyag megközelítőleg 1.5 másodperces spektrumait kapnánk, egyfajta spektogrammot. Annak oka, hogy mégsem készítettem hosszabb hangfelvételeket az, hogy biztos legyek abban, hogy nem kerül olyan adat a beszédet tartalmazó felvételek közé, amelyek nem tartalmaznak beszédet.

A beszédet nem tartalmazó minták beszerzése az internet segítségével történt. Ennek oka a korlátozott lehetőségeim a felvételek elkészítésében és ilyen módon sokkal változatosabb zajmintákat szerezhettem be. Kizárólag ingyenes forrásokat használtam illetve szabadon felhasználható anyagokkal dolgoztam. Ezeknél a hangfelvételeknél már nem tartottam fontosnak a korlátok betartását, vagyis, hogy a felvétel ne tartson 2,5 másodpercnél tovább és hogy a zaj a felvétel elején legyen. Az egyetlen kikötés a hanganyaggal szemben csak annyi, hogy semmilyen emberi beszéd ne legyen benne.

Ezek után a Java nyelven írt programot a megfelelő paraméterekkel beállítva előállítottam a tanuló algoritmusok számára megfelelő adatokat. Ez a program a megadott helyekről folyamatosan beolvassa a megadott hanganyagokat és előállítja a Fourier transzformáltjukat. Ennek eredményeképpen kapott spektrumokat feldolgozva, olyan adatsorokat hoz létre, amelyeket könnyedén értelmezni tud az R-ben megírt másik program, ami a tanuló algoritmusok megvalósítását teszi lehetővé egyszerű és könnyen kezelhető módon. Ezek után már csak a tanulómódszerek elemzése maradt.

6.2 A tanuló algoritmusok hatékonysága

A célkitűzésem között szerepelt, hogy el tudjam különíteni az emberi beszédet a különböző zajoktól. Ezt pedig egy 1 rejtet réteget tartalmazó neurális hálóval, illetve a tartóvektor gép segítségével kívántam elérni. Több különböző beállítást próbáltam ki, amelyek módosíthatják a háló eredményességét, abból a célból, hogy elemezzem az egyes módszerek hatékonyságát. Ezek a beállítások főleg a hálózat bemenetének változtatásaira irányulnak, vagyis különböző frekvencia tartományokra adott osztályozás pontosságát követtem nyomon. A kimeneti fájl, amelyet a Java program készített, tartalmazza az összes hangfelvétel adatait. Ahhoz, hogy tesztelni tudjam a különböző módszereket, valamilyen módon tanuló halmazra illetve teszt adatokra kellett felosztani őket. Ezt úgy oldottam meg, hogy az adatok felét véletlenszerűen kiválasztottam, ezek lettek a tanuló adatok, a maradék adat pedig a tesztesetek. Szerencsére ezt nagyon egyszerű volt kivitelezni az R-ben a *sample* függvényhívás segítségével, ráadásul ezt a halmazt elég csak paraméterként megadni a *nnet* illetve az *svm* függvénynek, amelyek az 1 rejtett réteget tartalmazó neurális hálót és a tartóvektor gépet valósították meg. Mivel a halmaz kiválasztása véletlenszerű, annak érdekében, hogy a különböző tanulási eljárásokat is

össze tudjam hasonlítani fontos volt, hogy ugyan arra a tanuló és teszt halmazra futtassuk őket.

6.2.1 Elemzések a hallótartományban

Az első tesztem során azt vettem alapul, hogy az emberi fül hallótartománya a 20Hz és 20kHz-es tartományba esik, tehát lekorlátoztam a kimenetet erre a tartományra. Ha ezen a tartományt felosztjuk 15 részre akkor 1332Hz széles frekvencia tartományokat kapunk, ezen tartományok elemeit pedig átlagoljuk így kapunk minden 1,5 másodpercre vonatkozóan egy 15 elemű vektort. Ez nem igazán megfelelő figyelembe véve, hogy az emberi hang alapfrekvenciája 60Hz és 300Hz közé esik, így nagyon valószínű, hogy az alaphang szinte semmilyen szerepet nem kap a tanulás során. Valamelyik tartományban meg fog jelenni, de az átlagolás össze fogja „mosni" a többi zajjal. Tehát azt feltételezzük, hogy a ilyen beállítások mellett a tanulás során az alaphangnak kevés szerepe lesz.

Véletlenszerűen kiválasztottam a minták felét amelyekre a tanítást végeztem. Ebbe a halmazba került 73 olyan minta ami emberi beszédet tartalmaz és 71 olyan mely nem tartalmaz beszédet. A maradék adathalmaz 77 beszédet és 68 zajt tartalmazó minta.

Ezt az adathalmazt tekintve az 1 rejtett réteget tartalmazó neurális háló és a tartóvektor gép hasonló eredményt ért el. A teszt során használt neurális háló 30db rejtett rétegbeli egységgel rendelkezett. Ezt a hálót többször lefuttatva azonos tanuló- és teszthalmaz mellett átlagosan a tanítóhalmazon 85%-át sorolta be helyesen, míg a teszteseteket 82%-ban. Más adathalmazt tekintve 2-3%-ot romlott ez az eredmény.

A tartóvektor gép esetén ugyan arra tanulóhalmazra, illetve teszt halmazra, amelyet a 1 rejtett réteges hálónál használtam egyaránt 86%-os pontossággal osztályozta az adatokat. Az SVM ezen teszt alatt a radiális bázis függvény-t használta magfüggvénynek (kernelfüggvénynek). Más kipróbált magfüggvény esetén, például *szigmoid* függvénynél az eredmény sokkal rosszabb lett, tehát a jó kernelfüggvény megválasztása kulcsfontosságú a tartóvektor gép esetén. Az SVM hatékonysága más teszthalmazok esetén hasonló eredményeket mutatott.

Mindkettő esetnél elmondható, hogy az osztályozás valamilyen szinten eredményes volt, azonban az elfogadhatótól messze elmarad. Annak érdekében, hogy javítsak az eredményen a

20Hz és 20kHz közötti tartományt nem 15 hanem 30 részre osztottam. Ekkor 666Hz-es tartományokat kaptam, ezek után az előző tanuló algoritmusokat használva újra elemeztem a teljesítményt. Erre az új adathalmazra SVM minimálisan javult, de a különbség a két felosztás között gyakorlatilag elhanyagolható. Ráadásul ahogy azt a 4.4.2-es részben leírtam, a rejtett rétegbeli egységek számának ebben az esetben jóval többnek kellene lennie 30-nál. Azonban a tesztelés során használt gép teljesítménye nem volt elegendő, hogy ennél több egységgel dolgozzon.

6.2.2 Elemzések a beszédtartományban

A következő módszernél, amivel megpróbálom elkülöníteni az adatokat arra alapoztam, hogy az emberi fül a beszédhangokat körülbelül a 4000Hz alatti tartományon érzékeli. Ezért a vizsgált frekvenciatartományt a 60Hz-re állítottam, ami az emberi beszéd alaphangjainak alsó határa, a felső határt pedig a 4000Hz-re állítottam. Majd az előző módszerhez hasonló módon itt is 15 illetve 30 résztartományon átlagolva a frekvenciákat, teszteseteket generáltam a tanuló algoritmusok számára. Ez azt jelenti, hogy az egyes résztartományok nagyjából 262Hz-es illetve 131Hz-es résztartományokat készítettünk. Ennél a módszernél jelentősen csökkent a az „elmosódás”.

A 262Hz-es résztartományokon vizsgálva vagyis a beszédtartományt 15 részre osztva az 1 rejtett rétet tartalmazó neurális háló a teszt adatokon jelentősen javult Többször futtatva különböző mintákra is kipróbálva azt az eredményt kaptam, hogy nagyjából 90%-os hatékonyságot érhetünk el, ami a tesztadatokat illeti ez a javulás már nem tapasztalható. Átlagosan itt is a tesztminták 82%-át osztályozta helyesen az algoritmus.

Ha növeltem a résztartományok számát 30-ra, akkor javult ugyan az eredményessége a teszt adatokon, de a 90%-tól jócskán elmaradt. Azonban az levonható, hogy ezen szűkebb tartományt használva, az eredményessége nőtt a hálónak.

A tartóvektor gép eredményeit ugyan ezekre a beállításokra és azonos tanuló és tesztadatok mellett az tapasztalható, hogy jelentősen javult a hatásfoka a hallótartományon mért értékekhez képest. Ha a beszédtartományt vagyis a 60Hz és a 4000Hz közötti tartományt 262Hz-es résztartományokra osztottam (15 résztartomány), akkor nagyjából a tanulóadatok és tesztadatok is 91-92%-os pontossággal osztályoztak.

Megnövelve a résztartományokat a tartóvektor gép esetén, az eredmények tovább javultak. Ha 30 résztartományt (131Hz-es résztartomány) vizsgáltam a beszédtartományon, akkor átlagosan a tanítópontok és tesztadatok 94%-át sorolta a megfelelő csoportba. Ez az eredmény a 15 résztartományhoz képest 1-2%-os javulást jelent, és ez már egy egészen kiváló osztályozást eredményez.

6.2.3 Elemzések az emberi alaphang tartományon

Az utolsó tesztem során az emberi alaphang tartományán próbáltam meg az adatokat elkülöníteni. Ezen a 60Hz és 300Hz közötti tartományon is az előzőekhez hasonlóan, itt is 15 illetve 30 résztartományt képeztem. Ez azt jelenti, hogy az első esetben 16Hz-es résztartományokat készítettem, míg a másodikban ennek a felét, vagyis 8Hz-es tartományokat.

Az 1 rejtett réteget tartalmazó neurális háló pontossága jelentősen csökkent a 60Hz és 4000Hz-es tartományon történő elemzésekhez képest. Ha 15 résztartományt vizsgáltam, az eredmény annyira leromlott, hogy a hallótartományon mért eredményeknél is rosszabb eredményeket ért el. 30 résztartomány esetén nem volt ennyire rossz a helyzet, de a 60Hz és 4000Hz közötti elemzés során kapott eredményektől elmaradt.

Az tartóvektor gép esetén azonban azt a meglepő eredményt kaptam, hogy 15 résztartományon a beszédtartományon elért eredményhez hasonlóan megközelítőleg 93%-os eredményt ért-el a teszt adatokon. A tanuló adatokon ráadásul még jobb eredményt is ért el. Tovább növelve a résztartományok számát 30-ra pedig még eredményesebb lett mind a tanuló mind a teszt eseteken.

Ahhoz, hogy ezt az érdekes eredményt jobban kivizsgáljam növeltem a résztartományok számát. Sajnos az 1 rejtett rétegű neurális hálónak túl sok egységre és ezzel együtt súlyra lenne szüksége ezért a ezeket a számításokat már csak a tartóvektor géppel végeztem el. A 20-20000 Hz közötti tartományt vizsgálva sokkal rosszabb eredményt kaptam. Viszont 240 résztartományra felosztva a 60 Hz és 300 Hz közötti illetve a 60 Hz és 4000 Hz-es tartományokat, a tanulás eredményesebb lett. Ami azonban még fontosabb, hogy tesztadatok eredményei is javultak valamelyest. Az, hogy kevesebb résztartomány esetén jobb eredmény született az alaphangok frekvenciatartományán történő vizsgálódás során, leginkább annak

köszönhető, hogy a résztartományok sokkal kisebb frekvenciatartományt jelentenek. Ha vesszük a frekvenciatartományokat akkor azt kapjuk, hogy 300-60 = 240 és ezt elosztjuk 30 részre akkor azt kapjuk, hogy 8Hz-es tartományokat átlagol az előfeldolgozó rendszer. Ha azonban a hallás frekvencia tartományát nézzük, akkor 4000-60 = 3940 akkor közel 131Hz-es frekvencia tartományokat átlagol az előfeldolgozó algoritmus. Ez jóval kisebb tartományok mint amit eddig használtam. Ha pedig azt nézzük, hogy növelve 240-re a résztartományok számát, hogy az első esetben 1Hz-es tartományok keletkeztek a második esetben pedig nagyjából 16Hz-es tartományokat jelent, akkor láthatjuk hogy ez jelentősen kisebb részt átlagolnak mint az előbb. Ebből következtethetünk, hogy ha kisebb de több résztartományt veszünk akkor javíthatjuk az eredményt.

6.2.4 Összegzés

Összegezve az eredményeket a leghatékonyabb módszernek a 60Hz és 4000Hz közötti tartomány vizsgálata tűnt. Az azonban rögtön kiderült, hogy az tartóvektor gép elég hatékonyan tudta osztályozni minden esetben az adatokat. Az 1 rejtett réteget tartalmazó neurális háló is viszonylag pontosan osztályozta a pontokat. Azonban egyik esetben sem sikerült tökéletesen szétválogatni a mintákat. Bár tökéletes osztályozást nem sikerült meg valósítani, az eredmények így is nagyon bíztatóak. Észrevehető továbbá az a tény, beszéd azonosítás sokkal eredményesebb volt akkor, amikor az egyes résztartományok kisebb frekvenciatartományt átlagoltak.

Amit nem sikerült megvalósítani:

Az elemzés során felmerült bennem, hogy milyen eredményekkel szolgálnának ezen eljárások olyan adatokra, amelyek kedvezőbb körülmények között lettek volna rögzítve, mint ahogy azt az én szűkös kereteim megengedték. Sajnos a mintákat tekintve szinte mindegyik csupán egy-egy szót tartalmazott, amit valaki felolvasott, ezek pedig nem tekinthetőek teljes mértékben „életszerűnek”. Ez a tény persze nem változtat azon, hogy emberi beszéd lett rögzítve, inkább csak a minták minőségét befolyásolja. Továbbá ha figyelembe vesszük, hogy a minta három személytől származott, akkor nem nevezhetjük az eredményeket reprezentatívnak. Ezen felül a minták elemszáma is korlátolt volt. A tesztek során a három személlyel 50-50 olyan felvételt készítetem amelyen beszéd szerepel. A beszédet nem

tartalmazó felvételek száma pedig 120 volt. Továbbá érdekes lett volna kipróbálni az algoritmusokat ennél sokkal nagyobb mintahalmazra, valamint bonyolultabb szövegekre is, azonban jelen keretek között erre nem adatott lehetőség. Valamint a rengeteg teszt során felmerült bennem az a lehetőség, hogy az előfeldolgozási fázisban más módszereket is kipróbáljak és ne csak a „nyers” frekvenciatérbeli felbontással dolgozzak. Különböző szűrések és egyéb technikák, amelyek implementálása javíthatta volna a tanuló algoritmusok hatékonyságát, amelyek által az elemzés során még jobb eredményeket érhettem volna el.

7. fejezet

Összefoglalás

A technológia nagy ütemű fejlődése egyre inkább lehetővé teszi, hogy a számítógépes alkalmazások képesek legyenek egyre bonyolultabb feladatokat megvalósítani. Továbbá az egyre komplexebb igények következtében mindinkább előtérbe kerülnek az olyan alkalmazások, amelyek valamilyen módon emberi viselkedési mintát követve működnek. A mesterséges intelligencia, még ha csupán az elmúlt fél évszázadot is vesszük figyelembe, amióta az elnevezése megszületett, a mai napig folyamatosan bővülő terület. Segítségével a tudomány számos ágán belül rengeteg új lehetőség kínálkozik különböző problémák újszerű megközelítésére és azok megoldására. A dolgozatomban egy ilyen emberi viselkedést, azaz problémamegoldást próbál modellezni, mégpedig az emberi hang azonosítását egy általam írt program segítségével.

A dolgozat elején az a célt tűztem ki magam elé, hogy számítógépes alkalmazások segítségével azonosítsam az emberi beszédet. A probléma megoldása során, olyan matematikai és informatikai eszközöket alkalmaztam, amelyek egyszerűen implementálhatóak. A rendelkezésemre álló szakirodalom mind a neurális hálók, mind a Fourier transzformálás tekintetében igen széleskörű. A diplomamunkám során megpróbáltam ezeket érthetően bemutatni és a leghasznosabb információkat összegyűjteni, az adott probléma szempontjából, majd ezeket a módszereket alkalmazva egy szoftvert elkészíteni és a kapott eredményeket elemezni.

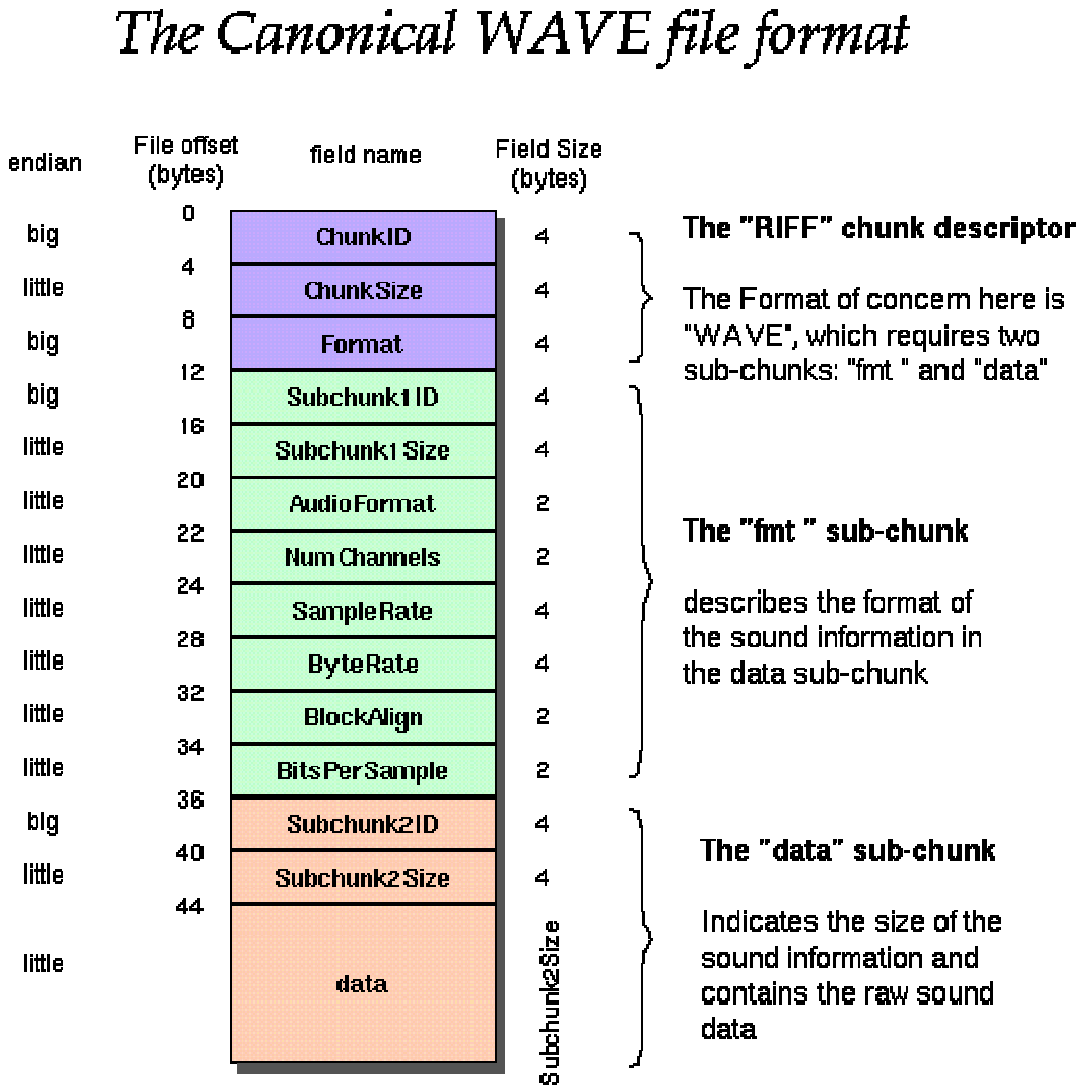
A program a beolvasott hanganyagoknak képzi a Fourier transzformáltját, majd az így kapott eredmények segítségével tanítottam és teszteltem a tanuló algoritmusokat. Az elemzések során pedig figyelembe vettem a hallástartományt, a beszéd frekvenciatartományát, illetve az emberi hang alaphangjának frekvenciatartományát. A tesztelések során a program átlagosan 90%-os biztonsággal azonosította az emberi beszédet.

Annak ellenére, hogy tökéletes megoldást nem sikerült produkálni, az elkészített programok nagy hatékonysággal osztályozták az általam összeállított mintákat. A program legkritikusabb része a neurális hálózat bemeneti értékeinek megfelelő megválasztása. Ezen értékek az

elkészített programban viszonylag egyszerűen módosíthatóak. Mivel a tanuló algoritmusok tesztelését R környezetben végeztem, így a Java-ban írt program eredményei más célokra is könnyen felhasználhatóak.

Végezetül az általam bemutatott és elemzett tanuló algoritmusok eredményei az mutatják, hogy a módszer hatékony, továbbá olyan területeken, ahol elfogadható bizonyos hibaszázalék hatásosan alkalmazható. Sok egyéb, jelen dolgozatban nem tárgyalt módszerrel és további fejlesztésekkel ráadásul ez a hibaszázalék csökkenthető.

Függelék:


The Canonical WAVE file format
endian
File offset (bytes)
field name
Field Size (bytes)
big
little
big
big
little
little
little
little
little
little
big
little
little
0
4
8
12
16
20
22
24
28
32
34
36
40
44
ChunkID
ChunkSize
Format
Subchunk1ID
Subchunk1Size
AudioFormat
NumChannels
SampleRate
ByteRate
BlockAlign
BitsPerSample
Subchunk2ID
Subchunk2Size
data
4
4
4
4
4
2
2
4
4
2
2
4
4
Subchunk2Size
The "RIFF" chunk descriptor
The Format of concern here is "WAVE", which requires two sub-chunks: "fmt " and "data"
The "fmt " sub-chunk
describes the format of the sound information in the data sub-chunk
The "data" sub-chunk
Indicates the size of the sound information and contains the raw sound data

Irodalomjegyzék:

Csákány Antal, Bagoly Zsolt – Jelfeldolgozás:
http://itl7.elte.hu/html/jelfel/index.htm (egyetemi jegyzet)

Dr. med. habil. Vass Zoltán – Fül:
http://www.fulspecialista.hu/index.php?page=content&method=static&id=58

Mark Handley – Fourier Transforms:
http://www.cs.columbia.edu/~hgs/teaching/ais/slides/03-fourier.pdf (dia vetítés)

Stuart Russel, Peter Norvig – Mesterséges Intelligencia Modern megközelítésben – Második átdolgozott kiadás, Panem könyvkiadó, 2005

Tornai Róbert – Hangtechnika jegyzet:
http://cgip.inf.unideb.hu/rtornai/ (elektronikus jegyzetek)

További segédanyagok:

WAVE formátum, „Perceptual Audio Coding Projects”:
http://ccrma.stanford.edu/courses/422/projects/WaveFormat/ (elektronikus jegyzet)

Audacity, program:
http://audacity.sourceforge.net/

R project - R (programozási nyelv):
http://www.r-project.org/