TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut

RAADIOREKLAAMIDE TUVASTAMINE MASINÕPPE ALGORITMIDE ABIL

Bakalaureusetöö
ITI40LT

Üliõpilane: Olga Dalton
Üliõpilaskood: 104493
Juhendaja: Jüri Vain

Tallinn
2013

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Autor: Olga Dalton

15.05.2013

Annotatsioon

Käesoleva lõputöö põhieesmärgiks on leida meetod raadioreklaamide automaatseks tuvastamiseks masinõppe algoritmide abil. Lõputöö raames kirjutatud algoritm peab olema sobilik iOSi seadmetel töötamiseks.

Töö põhilisteks probleemideks on reklaami ja muusika selliste erinevuste leidmine, mida saaks tuvastada lisaks inimesele ka arvuti, ning nende erinevuste automaatne määramine. Seega on tähtsamateks lõputöö ülesanneteks analüüsida reklaami ja muusika erinevusi, kirjutada neid eristavaid parameetreid määrav algoritm, treenida parameetrite alusel masinõppe algoritmi ning valideerida saadud algoritmi täpsust.

Töö käigus jõutakse järeldusele, et põhiline erinevus reklaami ja muusika vahel seisneb tõsiasjas, et reklaam koosneb enamasti kõnest ja muusika mitte. Sellele põhinedes leitakse parameetrid, mis eristavad muusika ja reklaami spektreid ning ajaesituse graafikuid.

Töö tulemusena saadud algoritm on piisavalt täpne, et kasutada teda automaatseks reklaami tuvastamiseks. Ühtlasi sobib ta iOSi seadmetel jooksutamiseks. Lisaks pakutakse välja mitmeid edasiarenduse võimalusi, mille abil klassifitseerimise täpsust veelgi suurendada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 5 peatükki, 10 joonist, 4 tabelit.

Abstract

The main purpose of this thesis is to find a method to automatically distinguish between music and radio advertisements with machine learning algorithms. This method has to be lightweight enough to work on iOS devices.

The main problem to solve is to find differences between music and advertisements that can also be detected by computer. An algorithm extracting those features from an audio file was written using iOS specific frameworks. As a result, main tasks of this thesis were to analyze distinguishing features of music and advertisements, to write an algorithm to extract those features, to train the machine learning algorithm and to validate an accuracy of this algorithm.

The research showed that the main difference between music and advertisements is the fact that radio advertisement contains mostly speech, while music does not. Based on that some key features, differentiating music and advertisements frequency and time domains were found.

The result is accurate enough to be used for automatic advertisement detection. It is quite lightweight as well, so it can be used on iOS devices. Some improvement possibilities are also outlined so that classification accuracy can be slightly increased.

The thesis is in Estonian and contains 40 pages of text, 5 chapters, 10 figures, 4 tables.

Lühendite ja mõistete sõnastik

iOS Apple'i poolt arendatav operatsioonisüsteem mobiilsete seadmete jaoks

FFT *Fast Fourier transform,* kiire Fourier' teisendus

DJ *Disc jockey,* diskor, raadio saatejuht

IDE *Integrated development environment,* integreeritud arenduskeskkond

PCM *Pulse-code modulation,* analoogsignaalide digitaalne esitusviis

HPS *Harmonic product spectrum,* põhisageduse määramiseks kasutatav meetod

ARFF WEKA poolt kasutatav andmeformaat

WAV *Waveform Audio File Format,* tuntud audioformaat

Sisukord

Jooniste nimekiri

Tabelite nimekiri

1. Sissejuhatus

Bakalaureusetöö ülesande valikul on arvesse võetud tänapäeva inimeste eelistusi meedia tarbimisel, eelkõige raadio valdkonnas. Vastavalt 2011. aastal läbiviidud uuringule [1] on raadio tähtsus inimeste elus viimasel ajal aina kasvamas ja seda põhiliselt tänu infotehnoloogia arengule: internetiraadio suuremale levikule ja mobiilsete seadmete populaarsuse suurenemisele. Koos raadio kuulajaskonna laienemisega kasvab ka raadioreklaamide osatähtsus kogu reklaamiturul.

Raadio meelitab reklaamiandjaid eelkõige väiksema kulukusega ja suure kuulajaskonna lojaalsusega – tervelt 93% jätab sama raadiojaama ka reklaami ajaks [2]. Samas tekib küsimus, kas kõik need 93% tegelikult ka tahavad reklaami kuulata? Selles kaheldakse sügavalt. Raadio kuulamise puhul on tegemist eelkõige passiivse meedia tarbimisega – paljud inimesed kuulavad raadiot millegi muu tegemise kõrval (tööl, jõusaalis, autoga sõites) ning raadioreklaam võib sellisel juhul olla üsnagi häiriv, kanali vahetamine segaks aga esmast tegevust. Selliselt on seletatav suur raadiokuulajate lojaalsus. Lisaks on tõestatud, et reklaamil on otsene negatiivne mõju inimese tervisele [3].

Seega vaatamata faktile, et tegemist on suure ja õitseva äriga, oleks tavainimese jaoks üsnagi hea omada võimalust reklaami automaatseks tuvastamiseks ja ümberlülitamiseks.

Antud töö ülesandeks on kirjutada algoritm, mis oleks suuteline eristama reklaami muusikast. Sellist algoritmi saaks hiljem kasutada näiteks raadiorakenduse tegemisel, mis lülitaks reklaami tuvastamisel raadiokanali automaatselt ümber või paneks reklaami ajaks mängima kasutaja poolt määratud muusika.

Põhiküsimused, mis tuli lahendada enne algoritmi kirjutamist, olid:

- Mille põhjal eristab inimene reklaami muusikast?
- Kas samu meetodeid saab kasutada ka arvutipõhiseks reklaami tuvastamiseks?
- Mis oleksid need parameetrid, mille järgi saaks rakendus aru, et tegemist on reklaamiga?
- Mis oleks kõige mõistlikum ja realistlikum viis antud ülesande täitmiseks?

Vastavalt sellele oli töö käik jaotatud 5-ks suuremaks etapiks:

- Meetodi leidmine
- Testandmete kogumine
- Eristamisparameetrite määramine
- Algoritmi implementeerimine
- Valideerimine

Valdkonna ülevaate osas on välja toodud sarnase eesmärgiga tööde lähenemised ning iga pakutava lahenduse tugevad ja nõrgad küljed.

Ühtlasi kirjeldatakse algoritmi kirjutamiseks vajalikke teoreetilisi teadmisi nii sageduse määramise, FFT teisenduse kui ka masinõppe valdkonnast. Lisaks puudutatakse iOSi arendusvahendeid, kuna algoritm on kirjutatud iOSi platvormi jaoks.

Lahenduse osa algab reklaamiklipi analüüsiga ning parameetrite määramisega, mis võimaldaksid eristada reklaami ja muusikat. Osa valitud parameetritest on sageduspõhised, osa lähtub audiosignaali ajaesitusest. Sageduspõhiste parameetrite määramine tugineb omakorda FFT teisendusele, mille abil määratakse iga kindla ajaühiku tagant audiolõigu põhisagedus. Lahenduse osa lõpeb masinõppe algoritmide treenimisega. Parameetrite määramise algoritm on realiseeritud Objective-C Apple'i Accelerate ning Core Audio raamistike kasutades.

Programmeerimiskeele valik võib ehk esialgu põhjendamatuna tunduda, kuid Apple'i nutitelefonid on väga hea näide seadmetest, mida miljonid inimesed [4] kasutavad raadio kuulamiseks ja mille jaoks saaks kirjutada reklaami tuvastavaid rakendusi. Pealegi pakub iOSi platvorm arendajale palju valmislahendeid bakalaureusetöö raames vajaminevate ülesannete jaoks. Masinõppe algoritmide treenimiseks on kasutusel masinõppe tööriist – WEKA [5]. Tegemist on mugava viisiga proovida erinevaid masinõppe algoritme. Antud töörist sai valitud eelkõige selle lihtsuse ja tõhususe tõttu.

Valideerimise osas arutletakse algoritmi ja kogu meetodi efektiivsuse üle, võrreldakse seda teiste meetodite tulemustega ning antakse hinnang tulemuse adekvaatsusele. Kokkuvõttes on välja toodud töö tulemused, lahendamata probleemid ja edasiarendamise võimalused.

2. Valdkonna ülevaade

Antud osas käsitletakse vajalikke teoreetilisi aluseid ning sarnaste eesmärkidega töid ja nende tulemusi.

2.1. Sarnased tööd

Reklaami tuvastus ei ole kuigi hästi läbi uuritud valdkond, kuid mõningaid lähenemisi ja käsitlusi siiski leidub. Erinevad tööd keskenduvad kõnetuvastuse, metaandmete, reklaamikorduste ja audio omaduste põhisele tuvastamisele. Kõiki meetodeid ühendav põhimõte on lihtne: kindla pikkusega audioklipile tuleb määrata teda iseloomustavad parameetrid. Parameetrite valik ja tüüp sõltub valitud tuvastamise meetodist. Näiteks kui kasutatakse andmebaasi, siis parameetrite valik piirdub andmebaasis kasutatavate parameetritega. Audiolõiku iseloomustavate parameetrite kogumit nimetatakse sõrmejälgedeks (ingl. k fingerprints). Seejärel saab neid parameetreid kasutades kindlaks teha, kas tegemist on reklaami või muusikaga [6].

2.1.1 Kõnetuvastus

Kui võrrelda erinevate reklaamide sisu, siis on võimalik tähele panna pidevalt korduvaid märksõnu, mida tavasaadetes ja muusikas ei esine. Kasutades kõnetuvastust, saab nende märksõnade järgi teostada otsingut ja märksõnade arvu järgi tuvastada, kas tegemist on reklaamiga või mitte [7]. Sellise meetodi miinuseks on sõltuvus keelest: iga keele jaoks tuleb koguda oma andmebaas märksõnadest ja andmebaasi suurus teeb meetodi arvutuslikult keerukaks. Andmebaasi kogumine nõuab suurt ettevalmistust ja reklaamide sisu mõistmist. Vea tõenäosus on aga suur.

2.1.2 Striimide indekseerimine teadaolevate reklaamiklippide leidmiseks

Antud meetod kasutab suurt teadaolevate reklaamiklippide andmebaasi. Sageduse põhjal tehakse raadiostriimist iga kindla ajaühiku tagant sõrmejälgi ning seejärel tehakse päring reklaamide ja audioklippide andmebaasi, et määrata, kas tegemist on reklaami või muusikaga [8]. Ühes sellise lähenemisega töös kasutatakse andmebaasina YACASTi [9]. Tulemusena saadakse 95-

protsendiline täpsus. Kuigi tulemust võib pidada üsna heaks, on meetodi suureks miinuseks suure reklaamide andmebaasi vajadus. Sellised andmebaasid on sageli tasulised ja riigipõhised (salvestatakse ainult mingi kindla riigi raadioklippe). Uute reklaamiklippide tekkides ei pruugi need kohe andmebaasi jõuda, seega võib ka algoritm alguses uue klipi puhul eksida. Tegemist ei ole seega universaalse reklaamituvastuse meetodiga.

2.1.3 Audiopõhine tuvastus

Audiopõhine tuvastus töötab samal põhimõttel, millel ka inimene reklaami muusikast eristab. Seda eelkõige juhul, kui reklaam ei sisalda muusikat ega ole ise laulu kujul. Nimelt saab inimene reklaamist aimu eelkõige muusika katkemise ja inimkõne algamise tõttu. Inimkõne on tavaliselt üsna korrapärase struktuuri ja rütmiga erinevalt instrumentaalsest muusikast. Inimkõnes esineb vaiksemaid lõike, sõnadevahelisi pause ja eriliselt rõhutatud sõnu. Inimaju eristab muusika kõnest, toetudes eelkõige helilaine sagedusele ja energiale. Seetõttu suudab inimene eristada reklaami muusikast isegi võõrkeelse raadiojaama puhul.

Uurimustöödes on saadud sellise tuvastusmeetodi täpsuseks näiteks 95% [10], 85% [11] ja 92% [12]. Seega on potentsiaalne tulemus sama hea kui eelnevate lähenemiste puhul. Antud meetodi põhiliseks miinuseks on suutmatus eristada muusikalist reklaami muusikast ning DJ juttu reklaamist. See tuleneb faktist, et ka inimene eristab neid üksnes sisu põhjal, mitte audio erinevuste tõttu. Sellegipoolest on tegemist kõige universaalseima reklaamituvastuse viisiga, sest selle jaoks pole vaja suurt andmebaasi ja see töötab ükskõik millise keelega raadiojaama puhul. Antud töös lähtutakse samast meetodist.

2.2. iOS arendusvahendid

2007. aastal väljatulnud iPhone on muutnud täielikult meie arusaama mobiilirakendustest ja nende programmeerimisest mobiilsete seadmete jaoks. 2013. aasta aprilli seisuga on viimane IOSi versioon 6.1.3, mis pakub arendajatele ligi 50 raamistikku, hõlmates nendega kogu rakenduse jaoks vajalikke funktsionaalsusi, alustades andmete kuvamiseks kasutatavatest tabelitest ja lõpetades video töötlemise komponentidega. Lisaks pakub Apple oma arendajatele mugavat tasuta IDE tarkvara nimega Xcode. Käesoleval hetkel on maailmas üle 100 tuhande aktiivse arendaja, kes tegelevad iOS ja Mac OSX arendamisega [13].

2.2.1 Objective-C

iOSi rakendusi kirjutatakse Objective-C programmeerimiskeeles. “Objective-C on C-keele edasiarendus, kus viimasele on lisatud objektorienteeritus ja dünaamiline täitmine. Objective-C on pärinud C keelest süntaksi, primitiivsed andmetüübid ja juhtstruktuurid ning lisanud Smalltalki-laadse klasside ja meetodite defineerimise süntaksi” [14]. Tüüpiliselt on Objective-C klassil liidese fail .h laiendiga ja implementatsiooni fail .m laiendiga. Lisaks klassidele on võimalik defineerida protokolle, mis sisaldab kohustuslikke ja valikulisi meetodeid, mida protokolli täitev klass peaks implementeerima. Veidi omapärane mõiste on klassi kategooria, mis võimaldab lisada suvalisele enda või baasraamistiku klassile uusi meetodeid ja neid kasutada. See annab lihtsa võimaluse laiendada olemasolevat klassi ilma üleliigse alamklasside tegemiseta. Kõik C ja C++ programmid jooksevad ka Objective-C-s: C ja C++ klasse, funktsioone ja meetodeid saab kasutada Objective-C klassides ilma erilise vaevata.

```
// Liidese fail - Sample.h
#import <Foundation/Foundation.h>
@interface Sample : NSObject // Näidisklass
@property (nonatomic, strong) NSString *sampleName;
- (void) renameSample: (NSString *) newSampleName;
@end

// Implementatsiooni fail - Sample.m
#import "Sample.h"
@implementation Sample
@synthesize sampleName;
- (void) renameSample: (NSString *) newSampleName
{
    self.sampleName = newSampleName;
    NSLog(@"Class Sample got a new name - %@!",
self.sampleName); // Printimine konsooli
}
@end

// Initsialiseerimine ja meetodi väljakutse
[[[Sample alloc] init] renameSample: @"New name"];
```

Joonis 1. Objective-C näide

Joonisel 1 on esitatud Objective-C näidisklass. [[Sample alloc] init] tulemusena luuakse uus

Sample-klassi objekt, renameSample: meetodi väljakutse peale prinditakse konsoolile välja "Class Sample got a new name - New name!" ja klassi muutuja sampleName uueks väärtuseks saab "New name". Märk @ stringi ees tähendab, et tegemist on Objective-C objektiga (NSString), mitte C stringiga. Viimast saab samuti kasutada, kuid see pole enamasti otstarbekas.

2.2.2 Accelerate raamistik

Accelerate tarkvararaamistik on objektiklasside hulk, mis sisaldab suure jõudlusega vektorarvutuse komponente, mis töötavad nii OS X-is kui ka iOSis. Raamistik annab arendajale lihtsa abstraktsiooni vektoripõhiste operatsioonide kasutamiseks ilma vajaduseta teha madalatasemelisi arvutusi. Accelerate sisaldab komponente pilditöötluseks (vImage), signaalitöötluseks (vDSP) ja matemaatikateheteks (vMathLib ja vForce). Töös kasutatakse vDSP-d, mis sisaldab endas valmis funktsioone FFT teostamiseks [15].

2.2.3 Core Audio raamistik

Core Audio on digitaalse audio manipuleerimiseks, mängimiseks ja konverteerimiseks mõeldud raamistik iOS ja OS X jaoks. See sisaldab omakorda mitmeid alamraamistikke kõikvõimalike audioga seotud operatsioonide tegemiseks. Enamik Core Audio klasse manipuleerib audiofaile lineaarsel pakkimata kujul – seda formaati nimetatakse "linear PCM-iks". Digitaalne audiosalvestus loob PCM andmeid, mõõtes perioodiliselt analoogsignaali tugevust ning konverteerides iga mõõtetulemuse numbriliseks väärtuseks. iOSis konverteeritakse tulemused täisarvulisteks väärtusteks (Integer), et võimaldada kiiremaid arvutusi ning säästa akut. Core Audio pakub seejärel võimalust konverteerida audio andmed reaalarvulisteks väärtusteks. Core Audio poolt toetatavad formaadid rahuldavad parimal viisil raadio mängimise rakenduse vajadusi [16]. Toetatavad formaadid on esitatud tabelis 1.

Formaadi nimi	**Formaadi laiend**
AIFF	.aif, .aiff
CAF	.caf
MPEG-1	.mp3

MPEG-2/MPEG-4 ADTS	.aac
MPEG-4	.m4a, .mp4
WAV	.wav

Tabel 1. Core Audio poolt toetatavad audioformaadid [17]

2.3. FFT-põhine sageduse määramine

Helilaine kujutab endast õhurõhu muutumist ajas. Kõige lihtsam helilaine esitus on ajaesitus. Ajaesituses on x-teljeks aeg ja y-teljeks signaali intensiivsus. Helisignaalid – näiteks muusika on väga keeruka spektriga. Sellisel juhul on arvutuslikult üsna keerukas ajaesitusest määrata helisagedust - otstarbekam on minna üle helisignaali sagedusesitusele. Sagedusesituses on x-teljel sagedus ja y-teljel intensiivsus, millega vastav sagedus helilõigus esineb. Fourier' teisendus on matemaatiline protseduur, mis teisendab signaali ajaesituse sagedusesituseks [18].

2.3.1 Kiire Fourier teisendus

Fourier' teisendus on perioodilise signaali integraalne teisendus. Kiire Fourier' teisendus (FFT) on teisendus, mille käigus tehakse spetsiifilisi vaheteisendusi, vähendades seeläbi teisenduse teostamiseks vajalike arvutuste mahtu.

Fourier'i teisenduse puhul kasutatakse mõistet diskreetimissagedus. Diskreetimissagedus (ingl. k sampling rate) on mõõtarv, mis näitab kui tihti analoogsignaali digitaliseerimisel mõõdeti [18]. Sõltuvalt salvestatava heli sagedusest tuleb valida õige diskreetimissagedus. Diskreetimissagedus peab olema vähemalt kaks korda nii suur kui mõõdetava heli sagedus. Muusika puhul kasutatakse tüüpiliselt diskreetimissagedusena 44.1 kHz. Diskreetimissageduse vähendamine ja suurendamine Fourier' teisenduse puhul ei mõjuta sageduse mõõtmise täpsust, vaid muudab maksimaalse mõõdetava sageduse suurust.

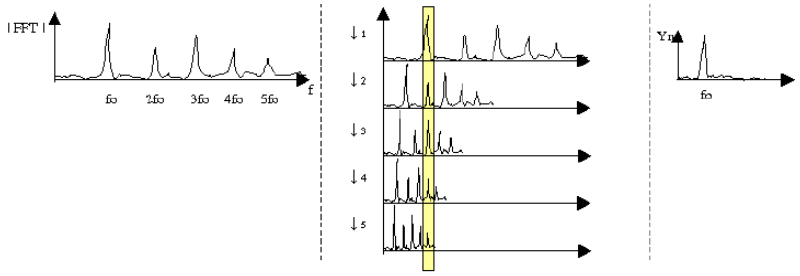
Fourier' teisenduse täpsusele on otsene mõju hoopis diskreetide (ingl. k samples) arvul. Mida suurem on diskreetide arv, seda täpsemalt võimaldab Fourier' teisendus helisagedust määrata. Diskreetide arv on tüüpiliselt kahe aste (näiteks 512, 1024, 2048, 4096 jne).

2.3.2 HPS

HPS ehk Harmonic Product Spectrum on üks lihtsamaid meetodeid helikõrguse määramiseks ning töötab enamike tingimuste puhul. Algoritm võtab sisendiks FFT teisenduse väljundi ja mõõdab ülemtoonide maksimaalset kokkusattumust.

Seejärel leitakse HPS väljundist maksimaalne väärtus, mille järgi arvutatakse välja audiolõigu põhisagedus. Põhiliseks probleemiks HPSi puhul on oktaavivead. See tähendab, et sageli on algoritmiga leitud sagedus tegelikust väärtusest oktaavi võrra kõrgem. Selle vea parandamiseks tuleb teha lisakontroll s.t. kui suuruselt teise haripunkti amplituud on ligikaudu ½ valitud sageduselt ja amplituudide suhe on teatud lävest suurem (näiteks 0.2 viie ülemtooni jaoks), siis tuleb valida tulemuseks suuruselt teine haripunkt [19].

Joonis 2 demonstreerib HPS algoritmi graafiliselt. Sagedusi, mis on väiksemad kui 50 Hz, ei peeta müra tõttu potentsiaalseks tulemuseks [19].



Joonis 2. HPS algoritm graafiliselt

2.4. Masinõpe

Masinõpe on tehisintellekti valdkond, mille eesmärk on treeningandmete põhjal otsuste ja ennustuste tegemine. Õppimine põhineb alati vaatlusandmetel, seega on masinõppe ülesanne teha otsuseid eelneva kogemuse põhjal. Probleemide valdkond, mida saab masinõppe algoritmidega lahendada on üsna lai. Sageli on tegemist ülesannetega, millel puudub üks kindel algoritmiline lahendus. Seega on probleemi lahendamiseks otstarbekas kasutada natuke teist lähenemist, milleks sobib hästi masinõpe [20]. Reklaamide tuvastamise puhul on tegemist ülesandega, mille jaoks on palju parem mõte kasuta masinõpet kui standardset lahendust.

2.4.1 WEKA

WEKA on kollektsioon masinõppe algoritmidest ja andmetöötluse tööriistadest, millel on olemas nii graafiline kasutajaliides, käsurea ligipääs kui ka võimalus WEKA klasse otse oma koodis kasutada. Tegemist on vabavaralise projektiga ja seetõttu on kogu masinõppe algoritmide kood kõigile huvilistele kättesaadav. See võimaldab portida Java koodi ka teistesse keeltesse, näiteks C-sse, et seda oleks võimalik Java-t mittetoetavatel platvormidel kasutada. Enamiku ülesannete puhul pole aga see vajalik, sest WEKA genereerib ise sisendandmete põhjal korrektse klassifitseerimismudeli.

WEKA võimaldab kiiresti erinevaid masinõppe algoritme treenida, kasutades standardset andmeformaati – arff. Arff andmefail koosneb päisest ja andmetest. Päises deklareeritakse atribuudid, mida iga andmekirje sisaldab, ning nende atribuutide andmetüübid. WEKA puhul on tähtis, et kõik sisendandmed oleksid ühes failis, sest relatsioonilised andmemudelid ei ole toetatud. Samas on olemas mitmeid tööriistu, mis võimaldavad teisendada mitu andmetabelit sisaldavat relatsioonilist andmebaasi üheks tabeliks, mis oleks sobilik WEKAs kasutamiseks [5].

Selle asemel, et implementeerida iga masinõppe algoritmi eraldi, on palju parem mõte kasutada standardset raamistikku valmis algoritmidega. Järgnevalt tutvustatakse mõningaid üsna tuntud ja levinud algoritme, mis leiavad kasutust käesolevas töös.

2.4.2 Naiivne Bayesi klassifitseerija

Naiivne Bayesi klassifitseerija on lihtne tõenäosuslik masinõppe algoritm. Klassifitseerija naiivsus seisneb parameetrite omavahelise sõltuvuse eitamises – eeldatakse, et ükski parameeter ei ole teistega korrelatsioonis. Naiivse Bayesi klassifitseerija puhul leitakse iga parameetri järgi tõenäosus, millisesse klassi andmed liigitada. Seejärel korrutatakse tõenäosused läbi ning andmed klassifitseeritakse suurima kogutõenäosusega klassi. Kuigi tegemist on üsna lihtsa lähenemisega, võib naiivne Bayesi klassifitseerija olla efektiivsem kui mitmed keerulised algoritmid. Üldjuhul, mida väiksem on parameetrite omavaheline korrelatsioon, seda parema tulemuse on võimalik naiivse Bayesi klassifikaatoriga saavutada.

2.4.3 Tehislikud närvivõrgud

Tehislik närvivõrk on süsteem, mis põhineb bioloogiliste närvivõrkude toimimise põhimõtetel – teisisõnu on tegemist katsega simuleerida aju töötamist, mis teeb otsuseid toetudes miljardite närvirakkude kollektiivsele otsusele. Bioloogiline neuron on lihtne andmeid töötlev element. Närvirakk võtab vastu sisendsignaale ning teisendab neid üksnes juhul, kui need on piisavalt tugevad. Neuronil võib olla mitu sisendit ja ainult üks väljund. Kõik bioloogilises närvivõrgus toimivad protsessid on keemilised [21].

Tehislik neuron on bioloogilise neuroni lihtsustatud mudel. Samamoodi nagu bioloogilisel närvirakul, saab tehislikul olla mitu sisendit ja ainult üks väljund. Sisendite kaudu võtab ta teistelt närvirakkudelt tuleva informatsiooni vastu, töötleb seda, kasutades etteantud funktsiooni ning väljastab tulemuse väljundi kaudu. Väljund võib olla omakorda seotud teise neuroni sisendiga. Iga sellist ühendust kutsutakse sünapsiks. Sünapsi iseloomustavaks suuruseks on kaal, millest tulenevalt võib ta informatsiooni kas tugevdada või nõrgendada. Seega sõltub iga neuroni lõpptulemus nii neuroni funktsioonist, mille järgi ta informatsiooni töötleb, kui ka sünapsi kaalust. Tänapäeval leiab kasutust mitu erinevat närvivõrkude implementatsiooni. Üks näide on mitmekihiline otsesuunatud (ingl.k feedforward) närvivõrk. Nimi tuleb närvivõrgu ühenduste organiseerimise viisist: iga kihi neuronid annavad oma väljundid üle järgmisele kihile. Sellises mudelis võivad olla nii nähtavad kui ka peidetud kihid [22].

Neuroni kui närvivõrgu komponenti saab lihtsalt kujutada klassina, mis on esitatud joonisel 3.

```
@interface SimpleNeuron : NSObject {
    int inputsNumber;
    NSMutableArray *inputWeights;
    float threshold;
}
- (id) initWithInputsNumber: (int) number;
@end
```

Joonis 3. Lihtsa närviraku klass

Muutuja *inputsNumber* näitab neuroni sisendite arvu, massiivis *inputWeights* hoitakse aga neuroniga ühendatud sünapsiste kaalusid. Muutuja *threshold* väljendab läve, mille puhul neuron signaali edasi saadab.

2.4.4 Otsustuste puu

Otsustuste puu puhul on tegemist ühe lihtsaima ja intuitiivseima masinõppe meetodiga, mille puhul kujutatakse lahenduskäiku järkjärguliste valikutega, iga alternatiivset varianti hinnatakse ning valitakse sobivaim ning selliselt jõutakse parima otsuseni. Masinõppes kasutatakse andmehulgast otsustuste puu genereerimiseks sageli ID3 algoritmi ja selle edasiarendusi (näiteks C4.5), mis on leiutatud Ross Quinlan poolt. ID3 alustab tööd kõigi treeningunäidetega puu juurtipus ning põhiidee seisneb andmete järkjärgulises jaotamises kahte ossa mingi kriteeriumi alusel, kuni igasse ossa jäävad vaid ühe klassi esindajad. Atribuudi headuse üle otsustatakse statistiliste andmete põhjal – põhiliselt selle järgi, kui hästi saab üksnes selle atribuudi järgi näiteid väljundklassidesse jaotada [23]. Töös kasutatakse otsustuste puu moodustamiseks ID3 algoritmi edasiarendust C4.5-e, mis on lihtsalt veidi optimeeritud variant ID3-st.

3. Raadioreklaamide tuvastusmeetod

Reklaami ja muusika erinevuste uurimiseks ning algoritmide väljaõpetamiseks on töö käigus kogutud 500 2-sekundilist reklaamiklippi ja sama palju 2-sekundilisi muusikaklippe. Klippide salvestamisel kasutati 4 erineva suunitlusega online raadiojaama:

- **977 Music Hitz Channel**, kus mängitakse lääne popmuusikat ja lastakse tüüpilist ameerikalikku reklaami. Raadio on ingliskeelne.
- **Star FMis** saab kuulata ka natuke vanemaid ja rahulikumaid lugusid, reklaamid on üsna sarnased teiste eesti raadiojaamadega. Raadio on eestikeelne.
- **Raadio Sky Plus** on suunatud veidi nooremale kuulajaskonnale ja seetõttu on nii muusika kui ka reklaamide valik energilisem ja nooruslikum. Raadio on eestikeelne.
- **Russkoje Radio** puhul on tegemist ühe eesti venelaste jaoks mõeldud raadiojaamaga, kus erinevalt eelnevatest jaamadest on esindatud ka vene popmuusika ja reklaamid on veidi teistsugused eelkõige keeleliste ja kultuuriliste erinevuste tõttu. Raadio on venekeelne.

Selline valik tagab nii muusika kui ka reklaamide mitmekesisuse, mis katab enamuse tuvastusalgoritmi õpetamiseks vajalikest tunnustest.

3.1. Audioanalüüs

Antud lõputöö raames kasutatakse reklaamide ja muusika eristamiseks nii audio aja- kui ka sagedusesitust ning nende põhjal määratavaid parameetreid. Kuna digitaalses audiofailis on helilaine juba iseenesest ajaesituses, siis ajaesituse parameetrite arvutamiseks pole vaja lisateisendusi teha. See-eest tuleb audio sageduspõhiste parameetrite kindlaks tegemiseks eelkõige sagedust määrata. Helilaine sageduse määramiseks on olemas palju erinevaid meetodeid ning sõltuvalt vajalikust kiirusest, täpsusest ning eesmärgist valitakse kõige sobilikum. Näiteks võib sageduse määramiseks kasutada helilaine nullist üleminekute loendamist, kuid antud töö jaoks pole see piisavalt täpne sageduse määramise viis. Seepärast kasutatakse sageduse

määramiseks audio teisendamist sagedusesitusele FFT abil ning seejärel HPSi meetodit põhisageduse leidmiseks.

3.1.1 Audiofail

Audiofaili mällu sisselugemiseks kasutatakse Core Audio tarkvararaamistikku. Kuna edaspidiseks FFT teisenduseks on vaja float-tüüpi andmeid, siis tuleb Core Audiole ette öelda, et ta teisendaks sisendandmed just float-kujule. Diskreetimissagedusena kasutatakse 44.1 kHz, mis on üsna levinud väärtus helitööstuses. Nagu joonisel 4 on näha, saab audio väljundformaati seadistada struktuuriga *AudioStreamBasicDescription*, millele tuleb vajalikud parameetrid ette öelda. Tulemuseks saadakse massiiv float tüüpi andmetega, mis esitab sisseloetud audioklippi.

```
AudioStreamBasicDescription audioFormat;
audioFormat.mSampleRate = 44100;
audioFormat.mFormatID = kAudioFormatLinearPCM;
audioFormat.mFormatFlags = kLinearPCMFormatFlagIsFloat;
```

Joonis 4. CoreAudio väljundformaadi seadistamine

3.1.2 Sageduse määramine FFT abil

Core Audio poolt väljastatud andmete massiivi saab seejärel kasutada FFT-põhiseks sageduse määramiseks. Kuna sellise meetodi puhul on teadaolevaks probleemiks madalsageduslikud mürad signaalis, siis eelkõige filtreeritakse massiivi kõrgpääsfiltriga. Kõrgpääsfiltriks kutsutakse filtrit, mis laseb läbi kõrgemad signaalid ning peatab madalamad. Antud juhul on filtri läveks valitud 50 Hz, kuna allpool seda läve on tõenäoliselt tegemist müraga. Andmete filtreerimise lihtsustamiseks kasutatakse Apple'i Accelerate raamistiku poolt pakutavat valmislahendust.

Filtreeritud andmed jaotatakse osadeks, igas osas 2048 diskreeti. Tegemist on üsna sobiva diskreetide arvuga antud ülesande jaoks, kuna see tagab piisava tulemuse täpsuse (±10 Hz) ja kiiruse. Spektri lekkimise vältimiseks, kus audio sagedusesituses tekivad energeetilised komponendid sinna, kus neid tekkida ei tohi, rakendatakse aknafunktsiooni [24]. FFT teisenduse tulemusena väljastab Accelerate raamistik kompleksarve, mis tuleb teisendada reaalarvudeks. Aknafunktsiooni rakendamise ja FFT teisenduse kood on esitatud joonisel 5.

```
// Aknafunktsioon
float *window = (float *)malloc(sizeof(float) *
numSamples);
vDSP_hann_window(window, numSamples, 0);
vDSP_vmul(samples, 1, window, 1, samples, 1, numSamples);
// FFT teisendus
vDSP_fft_zrip(fftSetup, &A, 1, log2n, FFT_FORWARD);
```

Joonis 5. FFT teisendus

Audio sagedusesitus on sisendiks HPSi algoritmile, mis leiab põhisageduse järjekorranumbri sagedusesituse massiivis. Kui on teada diskreetimissagedus ja diskreetide arv, saab selle järjekorranumbri järgi sageduse lihtsalt välja arvutada järgmise tehtega:

$$Sagedus(Hz) = järjekorraNumber * (\frac{diskreetimissagedus}{samplite\ arv})$$

3.2. Reklaami ja muusika võrdlev analüüs

Reklaami ja muusika eristamiseks tuli töö käigus leida sellised parameetrid, millele toetudes saaks audioklippi klassifitseerida kas muusikaks või reklaamiks. Osa leitud parameetritest on aja- ja osa sagedusesituse põhised. Parameetrite määramise algoritmi sisendiks on seega WAV formaadis audioklipp ja väljundiks joonisel 6 oleva struktuuriga klass.

```
@interface SegmentParam : NSObject
@property float standardDeviation;  // Sageduste
standardhälve
@property float averageValue; // Keskmine sagedus
// Keskmisest 3 korda väiksemate ja suuremate väärtuste arv
@property int numberOf3xSmallerValues,
numberOf3xBiggerValues;
@property int numberOfSilentValues; // Vaiksete väärtuste
arv
@property int numberOfDifferentValues; // Erinevate
väärtuste arv
@property float fileDeviation; // Ajaesituse standardhälve
// Nullist üleminekute määra standardhälve
@property float zeroCrossingDeviation;
@property float energy; // Koguenergia
@end
```

Joonis 6. Audioklippi iseloomustav klass

Seejuures on muutujat *averageValue* ehk keskmist klipi sagedust kasutatud üksnes teiste muutujate väärtuste arvutamisel. Kõik ülejäänud muutujad on tuletatud kas klipi aja- või sagedusesitusest ning need on kasutusel otseselt masinõppe algoritmi treenimisel.

Parameetrite määramise algoritmil on kaks otstarvet. Esiteks, seda kasutatakse masinõppe algoritmi treenimiseks ja valideerimiseks vajalike andmete kogumiseks. Teiseks on see vajalik, et määrata reaalset audioklippi iseloomustavad parameetrid, tuvastamaks kas tegemist on reklaami või muusikaga.

Järgnevalt põhjendatakse lühidalt iga muutuja valikut ning kirjeldatakse selle leidmise viisi.

3.2.1 Ajaesituse parameetrid

Ajaesituse parameetriteks on koguenergia, nullist üleminekute arvu standardhälve ja ajaesituse standardhälve.

Signaali x(n) koguenergia on signaali üks põhiparameetreid [25] ja seda on võimalik arvutada järgmise valemiga:

$$E = \sum_n x(n)^2$$

kus *n* on diskreetide arv.

On täheldatud, et reklaami koguenergia on võrreldes muusikaga enamasti väiksem. See tuleneb eelkõige faktist, et reklaami ajal on palju rohkem vaikust kui muusika ajal. Samuti on muusika spektris rohkem kõrgeid sagedusi, mis annavad olulise osa signaali koguenergiast. Joonisel 7 on näidatud tüüpiline raadiosalvestus, kus esineb nii muusika kui ka reklaam. Sellel on selgelt eristatav reklaamilõik, mis on oluliselt madalama energiaga kui ülejäänud salvestus.

Joonis 7. Raadiosalvestuse ajaesitus reklaami ja muusikaga

Nullist üleminekute määr (ingl.k zero crossing rate) näitab mitu korda ajaühikus ületab signaal nulli ehk muudab oma märgi. Tegemist on parameetriga, mis leiab kõnetuvastuses palju kasutust. Vaikuse puhul on nullist üleminekute määr palju suurem kui heli puhul ja seepärast kasutatakse väärtust vaiksete lõikude määramiseks kõnes. Kuid ka reklaami ja muusika eristamiseks on võimalik kasutada nullist üleminekute määra. Nimelt on muusika nullist üleminekute määr palju ühtlasem kui reklaami puhul, sest reklaamis esineb rohkem vaikseid hetki. Seega saab nullist üleminekute määra standardhälvet kasutada kui reklaami ja muusika eristamise parameetrit. Tegemist on John Saundersi poolt välja pakutud parameetriga [10].

Joonisel 8 on näidatud, kuidas leitakse audioklipi nullist üleminekute arvu. Selleks tuleb läbida andmemassiiv ning võrrelda järjestikuste väärtuste märki.

```
+ (int) calculateZeroCrossings: (float *) audioData
                      andDataLen: (int) numSamples {
    int numCrossing = 0;
    for (int p = 0; p < numSamples-1; p++) {
        if ((audioData[p] > 0 && audioData[p + 1] <= 0) ||
            (audioData[p] < 0 && audioData[p + 1] >= 0)){
            numCrossing++;
        }
    }
    return numCrossing;
}
```

Joonis 8. Nullist üleminekute määra leidmine

Joonisel 7 on selgelt eristatavad muusika ja reklaami koguenergia erinevused. Ühtlasi on võimalik tähele panna, et muusika energiatase on läbi aja palju ühtlasem kui reklaami oma. Reklaami puhul on palju rohkem energiahüppeid ja -langusi. Audio energia ühtluse iseloomustamiseks saab kasutada **ajaesituse standardhälvet**. Reklaami puhul peaks standardhälve olema suurem ja muusika puhul väiksem.

3.2.2 Sageduspõhised parameetrid

Lõputöö raames kasutatakse 5 erinevat sageduspõhist parameetrit, mille põhjal saab eristada muusikat ja reklaami. Nendest mõned on üksteisega korrelatsioonis - näiteks keskmisest 3 korda väiksemate ja suuremate väärtuste arv on sõltuvuses sageduste standardhälbest. Parameetrite sõltuvus üksteisest ei ole kuigi hea omadus, sest toob sisse tuvastuseks kasutatava informatsiooni liiasuse, kuid katse elimineerida kõik sõltuvad parameetrid ja jätta alles üksnes sõltumatud on viinud märkimisväärse klassifitseerimise täpsuse languseni. Seetõttu on korrelatsiooni peetud ebatähtsaks ning kõik parameetrid alles jäetud, kuna need annavad klassifitseerimisel olulist lisainfot ning parandavad algoritmi täpsust.

Sageduste standardhälve näitab, kui ühtlane on audioklipi sageduste jaotus. Nagu ka energia puhul on muusika sageduste jaotus palju ühtlasem. Reklaamis esineb aga vaiksemaid lõike, kus diktor teeb sõnade vahel pausi ja seetõttu on sagedus madalam või nullilähedane. Ühtlasi on

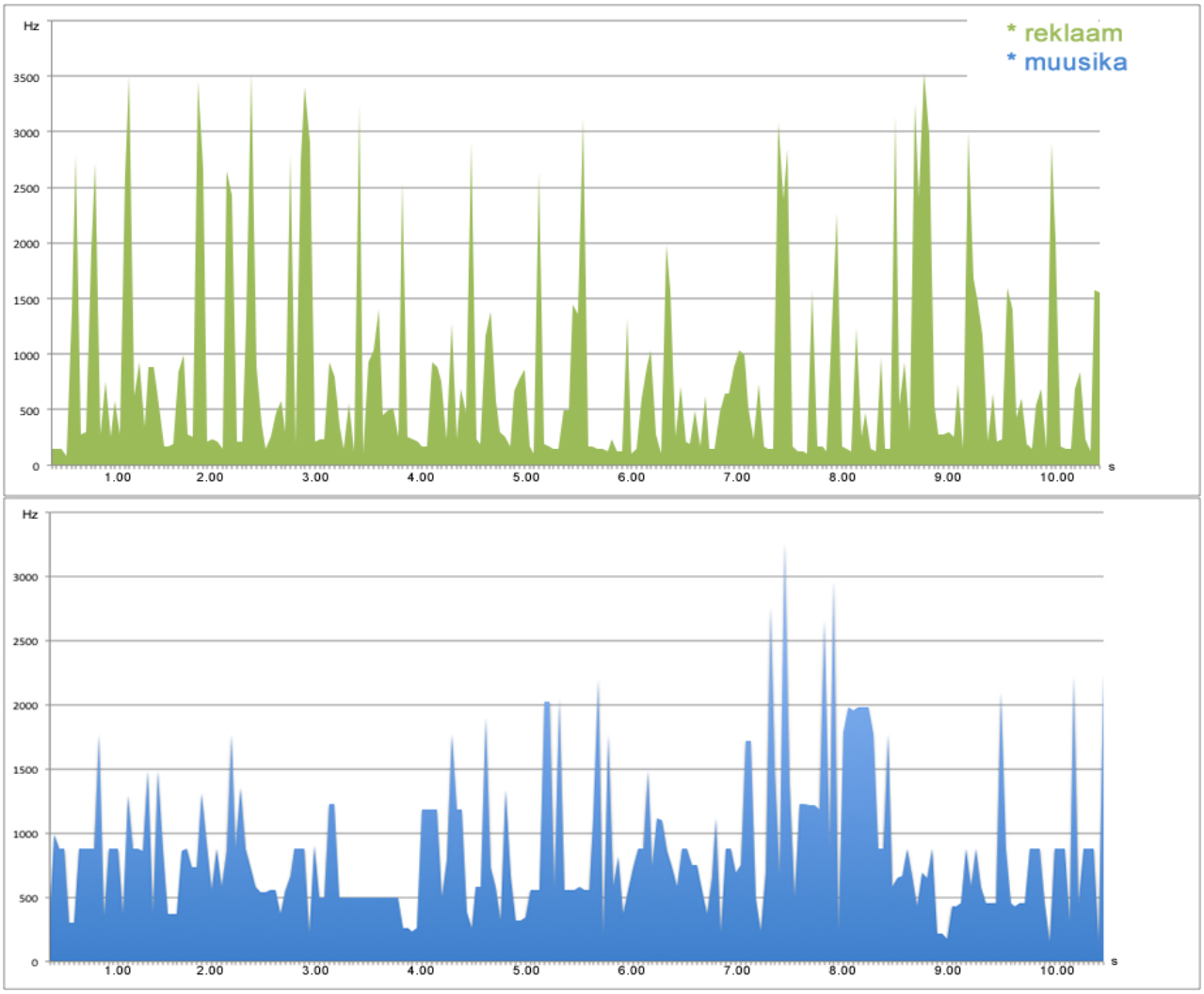
reklaamile omane kas reklaami märksõnade või põhisõnumi eriline rõhutamine põhitooni tõusuga, mistõttu esineb reklaamis keskmisest sagedusest mitu korda suuremaid sagedusi.

Keskmisest 3 korda väiksemate ja suuremate väärtuste arv iseloomustab sarnaselt standardhälbele sageduste jaotuse ühtlust, kuid annab parema ülevaate sageduste jaotuse maksimumite ja miinimumite kohta. Keskmisest sagedusest kolm korda suuremaid väärtusi on muusikal tavaliselt üsna vähe või ei ole üldse, samas kui reklaamil pole see harvaesinev nähtus. Reklaamilõigu keskmine väärtus on vaiksete osade tõttu tavaliselt sama valjusega muusikast väiksem, piirdudes üldjuhul 500-700 Hz-ga. Siin esineb muidugi ka erandeid. Kui keskmine sagedus on näiteks 500 Hz, siis tuleb nende parameetrite leidmiseks lugeda kokku mitu korda on sageduse väärtus olnud väiksem kui 166 Hz ja suurem kui 1500 Hz.

Vaiksete väärtuste arv näitab audioklipi "tühjade" sageduste arvu ning jällegi põhineb see faktil, et reklaami puhul on palju rohkem pause ja vaikust kui muusika puhul. Tühjaks peetakse antud juhul iga sagedust, mis on väiksem kui 15% keskmisest sagedusest. Selline kõrge lävi sai valitud eelkõige algoritmi müratundlikkuse vähendamiseks. Ühtlasi ignoreerib 15%-line lävi vaikset reklaami ajal mängivat taustamuusikat, mis on üsna levinud. See aitab parandada klassifitseerimise täpsust, kuna teeb algoritmi paindlikumaks erinevate reklaamitüüpide suhtes.

Erinevate väärtuste arvu leidmiseks tuleb iga sageduse väärtust ümardada lähima 10 Hz-ni, eemaldada tulemustest korduvad väärtused ning lugeda kokku järele jäänud väärtuste arvu. Suurem erinevate väärtuste arv iseloomustab reklaami ja väiksem muusikat, seda eelkõige reklaamile iseloomulike sagedushüpete tõttu. Muusikale omase ühtlasema sageduste jaotuse puhul on ka vähem sellise täpsusega erinevaid väärtusi. Samas, kui oleks võimalik määrata sagedust suurema täpsusega (±1 Hz), siis oleks muusikal just rohkem erinevaid väärtusi inimhääle monotoonsuse tõttu. Kuid antud juhul teeks nii kõrge täpsuse tagaajamine sageduse määramise äärmiselt aeglaseks ega lisaks olulist sisulist väärtust. 10 Hz täpsuse piirides erinevate väärtuste arv aitab see-eest reklaami ja muusikat eristada.

Joonisel 9 on esitatud kahe 10-sekundilise audioklipi sagedusjaotuse graafikud, rohelisena on märgitud reklaami graafik ja sinisena muusika oma.



Joonis 9. Reklaamile ja muusikale iseloomulikud sagedusjaotuse graafikud

Näite graafikute põhjal saab kinnitada, et eelpool kirjeldatud parameetrid peavad paika. Ka muusika sagedused ei ole kõik ühesugused, seal esineb nii sagedushüppeid kui ka langusi, kuid siiski on jaotus ühtlasem kui reklaami puhul ning vaikseid väärtusi on tunduvalt vähem.

Parameeter	**Muusika**	**Reklaam**
Sageduste standardhälve	552.37	928.14
Keskmine sagedus	863.85	834.31
Keskmisest 3 korda väiksemate väärtuste arv	15	91

Keskmisest 3 korda suuremate väärtuste arv	4	25
Vaiksete väärtuste arv	0	6
Erinevate väärtuste arv	55	82

Tabel 2. Reklaamile ja muusikale iseloomulikud parameetrid

Eelmises näites esitatud audioklippide kohta on arvutatud sageduspõhised parameetrid, mis on esitatud tabelis 2. Saadud tulemused vastavad täielikult ootustele: muusika puhul on standardhälve umbes samasuguse keskmise väärtuse korral tunduvalt väiksem. Väiksemad on ka teised parameetrid.

3.3. Masinõppe algoritmide treenimine

Erinevate masinõppe algoritmide treenimiseks ja testimiseks on käesolevas töös kasutusele võetud masinõppe meetodite kogu WEKA (Waikato Environment for Knowledge Analysis). WEKA võtab sisendiks arff formaadis andmefaili koos treenimise ja valideerimise andmetega. WEKA sisendandmete genereerimiseks on kirjutatud lisaklass, mis paneb kokku etteantud WAV faile iseloomustava arff faili, määrates iga WAV audioklipi kohta eelmises peatükis kirjeldatud parameetrid.

Leidmaks jõudluse ja tuvastustäpsuse mõttes parim signaali tunnuste kooslus katsetati käesolevas töös läbi mitu erinevat parameetrite kombinatsiooni. Parima tulemuse andsid kolm allpool kirjeldatud lähenemist, millega jätkasin täpsemate testide läbiviimist:

- **Ainult ajaesituse parameetrid**: koguenergia, nullist üleminekute määra standardhälve ja ajaesituse standardhälve. Sellise lähenemise suureks plussiks on parameetrite määramise lihtsus – tegemist on ajaesituse parameetritega, seega ei ole nende leidmiseks vaja teha arvutuslikult keerukat FFT teisendust. Lisaks pole parameetrid omavahel korrelatsioonis. Puuduseks on parameetrite ühekülgsus, mis võib viia klassifitseerimise täpsuse alla.

- **Ainult sageduspõhised parameetrid**: sageduste standardhälve, 3 korda väiksemate ja suuremate väärtuste arv, vaiksete väärtuste arv, erinevate väärtuste arv. Sageduspõhiste

parameetrite eeliseks on müratundlikkus ning reklaami taustamuusika ignoreerimine. Miinuseks on keeruka sageduse määramise vajadus ning parameetrite omavaheline korrelatsioon.

- **Ajaesituse ja sagedusesituse parameetrid koos.** Selle lähenemise puhul võetakse kasutusele eelneva kahe lähenemise efektiivsemad parameetrid. Parameetrite efektiivsuse hindamiseks kasutatakse WEKA "Select attributes" funktsionaalsust.

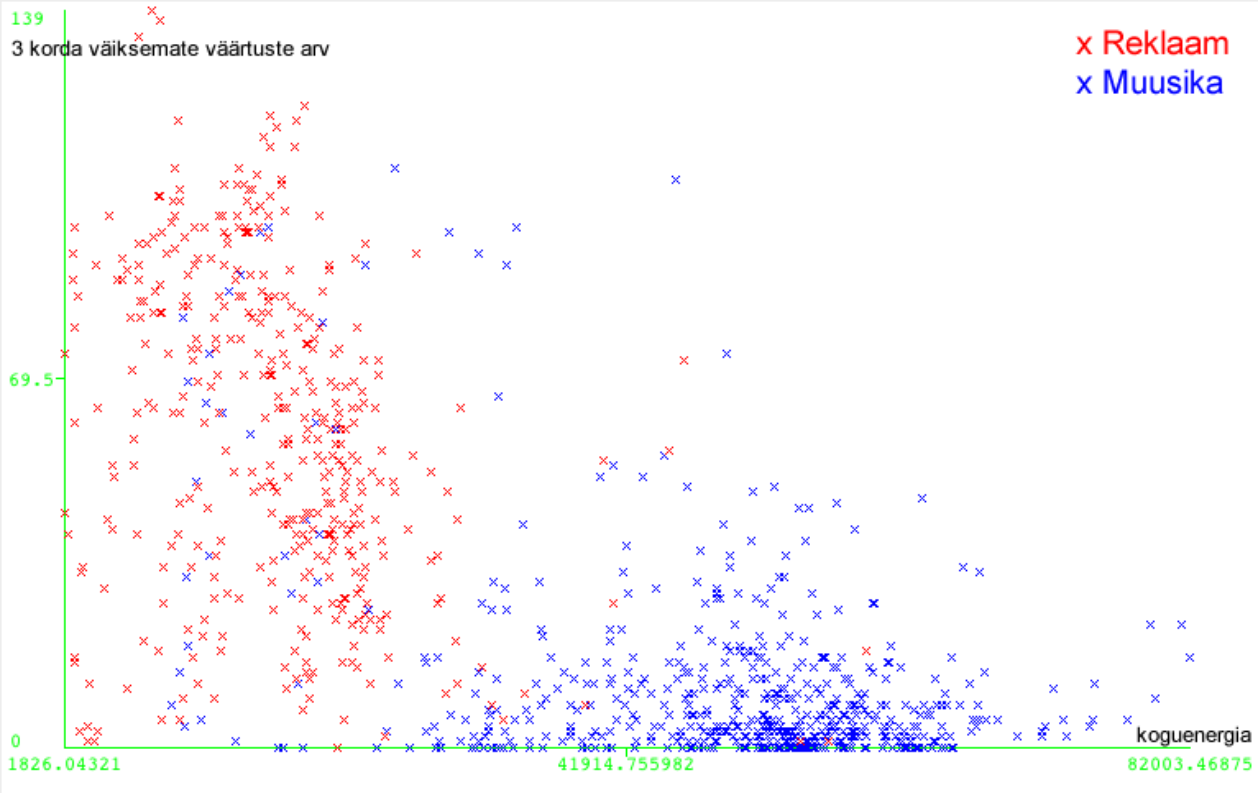
Ajaesituse puhul hindab WEKA atribuutide headuseks:

- Koguenergia – 64.1%
- Ajaesituse standardhälve – 63.9%
- Nullist üleminekute määra standardhälve – 21.7%

Sageduspõhiste parameetrite efektiivsus on järgmine:

- 3 korda väiksemate väärtuste arv – 47.1%
- Sageduste standardhälve – 42.7%
- 3 korda suuremate väärtuste arv – 37.7%
- Erinevate väärtuste arv – 26.2%
- Vaiksete väärtuste arv – 25.3%

Nendele tulemustele toetudes on otstarbekas loobuda kahest vähem olulisest parameetrist: nullist üleminekute määra standardhälbest ning vaiksete väärtuste arvust. Järelejäänud parameetritest võimaldab kahe efektiivseima parameetri (koguenergia ja 3 korda väiksemate väärtuste arv) kombinatsioon enamikul juhtudel audioklippi õigesti klassifitseerida. Ülejäänud parameetrid aitavad nende kahe parameetri tulemusi vähemal määral parandada. Joonisel 9 on kujutatud audioklippide jaotus kahe efektiivseima parameetri järgi, kus punasega on märgitud reklaam ja sinisega muusika.



Joonis 10. Audioklippide klassifitseerimine kahe parameetri järgi

Kombineeritud parameetrite kasutamise eeliseks on suurem paindlikkus ja täpsus. Miinuseks on parameetrite rohkus ja nende määramiseks vajalike operatsioonide suur arv, mis viib klassifitseerimise kiiruse alla.

4. Tulemuste valideerimine

Treeninguandmete kogumisele järgneb masinõppe algoritmide treenimine ja valideerimine. Treeningandmeteks on 500 2-sekundilist reklaamiklippi ja 500 2-sekundilist muusikaklippi, mis on pärit neljast erinevast raadiojaamast. Nende põhjal koostab WEKA sobiva matemaatilise mudeli andmete klassifitseerimiseks. Käesolevas peatükis kasutatakse kolme mudeli tüüpi: otsustuste puu, tehislik närvivõrk ning Bayesi klassifitseerija. Reklaamide praktiliseks tuvastuseks ja mudeli valideerimiseks on võimalik õpitud mudeli teisendada Objective-C koodiks. Mudel sisaldab otsustuste puud, mille saab esitada näiteks if-lausete jadana, kus if-tingimuse tõeseks osutumise korral kontrollitakse järgnevaid tingimusi uue if-lausega. Tehislike närvivõrkude puhul saab mudeli esitada neuroni klassi abil, mille muutujateks on neuroni sisendite arv, neuroniga ühendatud sünapsiste kaalud ja neuroni läve funktsioon ning tingimused.

Algoritmide tulemuste valideerimiseks kasutatakse ristkontrolli tehnikat. Ristkontrolli puhul hinnatakse algoritmi tulemusi sõltumatu valimiga, mis valitakse juhuslikult sisendandmete seast.

4.1. Algoritmide tulemused ja võrdlus

Algoritmide valideerimise tulemused on jagatud vastavalt parameetrite rühmitamisele. Iga lähenemise puhul on võrreldud masinõppe algoritmide tuvastusprotsenti ning audioklipi parameetrite määramiseks vajalikku aega. Kuna tegemist on iOSile suunatud algoritmiga, siis on väga tähtis parameetrite määramise algoritmi mälukasutuse säästlikkus ja jõudlus.

Tabelis 3 on välja toodud erinevaid parameetrite kombinatsioone kasutatavate algoritmide tulemused. Iga parameetrite kombinatsiooni puhul on katsetatud kõigi kolme mudeliga.

Tuvastusalgoritm	Otsustuste puu valideerimise tulemus	Tehisliku närvivõrgu valideerimise tulemus	Naiivse Bayesi klassifitseerija valideerimise tulemus
Ajaesituse parameetreid kasutav algoritm	90.31%	90.48%	89.45%
Sageduspõhiseid parameetreid kasutav algoritm	85.19%	84.68%	83.25%
Nii ajaesituse kui ka sageduspõhiseid parameetreid kasutav algoritm	92.63%	92.29%	91.11%

Tabel 3. Algoritmide valideerimise tulemused

Lisaks tuvastustäpsusele on võrreldud ka parameetrite määramiseks kuluvat aega ja mälukasutust. Kuna erinevad iPhone'i mudelid erinevad üksteisest jõudluse poolest, siis on parameetrite määramiseks kuluvat aega mõõdetud kõige vanema iPhone'i ja kõige uuema iPhone'i mudeliga ning väljendatud vahemikuna. Katsed on tehtud 2-sekundilise klipiga. Vastavad tulemused on esitatud tabelis 4.

Tuvastusalgoritm	Parameetrite määramiseks kuluv ajavahemik sekundites	Parameetrite määramiseks vajalik mälumaht MB-des
Ajaesituse parameetreid kasutav algoritm	0.09 – 0.38	2.35 MB
Sageduspõhiseid parameetreid kasutav algoritm	0.20 – 1.04	4.34 MB
Nii ajaesituse kui ka sageduspõhiseid parameetreid kasutav algoritm	0.38 – 1.39	6.85 MB

Tabel 4. Parameetrite määramiseks kuluv aeg ja mälumaht vastavalt tuvastusalgoritmile

4.2. Tulemuste analüüs

Parameetrite kombinatsioonidest näitab kõige paremat tulemust nii ajaesituse kui ka sageduspõhiste parameetrite segu. Parim tulemus on 92.63%. Kuid sellise parameetrite valikul põhineva tuvastusmeetodiga kaasneb suurem mälukasutus ning võrdlemisi pikk aeg parameetrite

määramiseks. Mälukasutus pole antud töö raames küll piisavalt optimeeritud, kuid kõige rohkem mälu tarbib FFT teisendus, mida saaks lihtsalt optimeerida üksnes läbi diskreetide arvu vähendamise. See tooks kaasa aga sageduse määramise täpsuse languse ning seeläbi ka audioklippide klassifitseerimise tulemuste halvenemise. Siiski on olemas ka täiendavaid meetodeid kuidas Accelerate raamistiku mälukasutust optimeerida, kuid tegemist pole lihtsate lahendustega ja need ei mahu käesoleva lõputöö raamidesse.

6.85 MB-ne mälukasutus pole ehk nii suureks probleemiks, aga kui on vaja korraga mitme audioklipi parameetreid määrata ning lisaks ka raadiot kasutajale mängida, siis võib sellest saada suurem mure. iOSi operatsioonisüsteemis pole kindlaks määratud kui palju võib üks rakendus korraga mälu kasutada, sest see sõltub teistest paralleelselt jooksvatest protsessidest ja seadme mälumahust. Üldjuhul paneb aga iOS rakenduse kinni siis, kui kasutatav mälumaht on umbes 30-40 MB.

Üksnes sageduspõhiseid parameetreid kasutavad algoritmid näitavad võrreldes teistega palju halvemaid tulemusi – keskeltläbi 85%-line täpsus. Mälukasutus ja ajakulukus on küll segalähenemisest väiksemad, kuid mitte niivõrd paremad, et ohverdada ligi 7% klassifitseerimise täpsusest.

Üllatavalt häid tulemusi näitavad ajaesituse parameetreid kasutavad algoritmid. Täpsus on nende puhul ligi 90%. Parameetrite määramiseks kuluvat aega võib nimetada olematuks (0.09 sekundit uusimal mudelil). Mälukasutus on samuti minimaalne ja tegemist on põhiliselt audiofaili mällu sisselugemiseks vajaliku mäluga, mida saab veelgi optimeerida.

Sõltumata parameetrite valikust on halvim tulemus Bayes'i klassifitseerijal. Tehislikel närvivõrkudel ja otsustuste puul on tulemus aga peaaegu ühesugune. Seega tasub ka algoritmi edasiarenduse puhul keskenduda kas otsustuste puule või tehislikele närvivõrkudele.

Mis aga puutub parameetrite valikusse, siis kui 90%-ne täpsus on rakenduse jaoks piisav, siis on ajaesituse parameetrite kasutamine parim valik antud ülesande jaoks. Ilmselt saab kas täiendava treenimise või uute parameetritega tõsta täpsust veelgi. Suurimaks probleemiks jääb klassifitseerimise müratundlikkus, sest ühegi parameetri määramisel müra ei filtreerita.

Edasiarenduseks pakub parimaid võimalusi segalähenemine, kus kasutatakse nii ajaesituse kui ka sageduspõhiseid parameetreid. Testides näitab see parimat klassifitseerimise täpsust. Ühtlasi on võimalik optimeerida nii mälukasutust kui ka parameetrite määramiseks kuluvat aega. Parandades kasutatavate parameetrite määramise täpsust, on võimalik klassifitseerimise täpsust märgatavalt tõsta. Lisaks on tegemist üsna müratundliku lahendusega, sest sageduse määramisel on kasutatud filtrit ja aknafunktsiooni.

Segalähenemise puhul on seega tuvastusvigade osatähtsus 7-8%. Mille poolest erinevad valesti klassifitseeritud audioklipid õigesti klassifitseeritutest? Tuleb välja, et enamik neist sisaldab muusikat või taustahelisid. Seejuures ei ole tegemist vaikse taustamuusikaga, mis saadab diktori häält. Tegemist on sellise reklaamiga, kus diktor räägib mõnda aega vaikse taustamuusika saatel, millele järgneb valjem muusikalõik ilma diktori saateta. Sellised reklaamid on üsna levinud Russkoje Radios, kus reklaamitakse palju kontserte, näidendeid ja muid üritusi. Sellise reklaami õigesti klassifitseerimiseks tuleks leida täiesti uus parameeter, mis eristaks seda muusikast.

Lisaks on algoritmi puhul probleemiks DJ klassifitseerimine reklaamiks. Kogutud 150-st audioklipist, mis sisaldavad DJ kõnet, klassifitseeritakse absoluutselt kõik reklaamiks. Katse tuua masinõppe algoritmisse uus klass nimega DJ ei anna positiivseid tulemusi, sest DJ ja reklaami ei suudeta omavahel eristada.

Eespool kirjeldatud probleemid on sellegipoolest ootuspärased. Töös parima tulemuse andnud algoritm põhineb eelkõige faktil, et reklaam sisaldab kõnet ja erandjuhul vaikset taustamuusikat. Töö käigus leitud muusikat ja reklaami eristavaid parameetreid võiks kasutada üsna edukalt ka üldisemalt muusika ja kõne eristamiseks. Algoritm ei tööta täpselt ainult juhtudel, kus reklaamis on esindatud vali muusika. DJ aga klassifitseerub sellise lähenemise puhul reklaamiks. Ka inimene suudab eristada DJ-d reklaamist eelkõige kõne sisu järgi, mitte aga heli karakteristikute põhjal.

5. Kokkuvõte

Käesolev töö näitab, et reklaami ja muusika automaatse eristamine on praktiliselt teostatav ülesanne. Saavutatud täpsust 92.63% võib pidada piisavalt heaks, arvestades et kasutatud parameetrid on kõik määratud üksnes audiostriimi põhjal ning peaaegu iga koodirea jaoks eksisteerib lihtsaid optimeerimise ja parandamise võimalusi. Ühtlasi võib tulemustest järeldada, et klassifitseerimise täpsus ei sõltu niivõrd valitud algoritmist, kuivõrd valitud parameetritest. Näiteks otsustuste puu, mida peetakse üheks lihtsaimaks masinõppe meetodiks, näitab sama häid tulemusi nagu tehislik närvivõrk.

Töö käigus valminud parameetrite määramise algoritmi ning masinõppe mudelit saab edukalt kasutada iOSi raadiorakenduse loomiseks, mis reklaami tuvastamise puhul keerab heli maha või lülitab raadiojaama ümber. Antud ülesanne pole ei mälukasutuse ega ajakulukuse poolest iOSi seadme jaoks ülejõukäiv.

Kuna valminud algoritmil on veel mitmeid puudujääke, saab seda edasi arendada. Edasiarendamise aluseks oleks mõistlik võtta nii ajaesituse kui ka sageduspõhiseid parameetreid kasutav otsustuste puu, kuna tegemist on kõige paremaid tulemusi näidanud meetodiga. Lisaks on otsustuste puud üsna lihtne implementeerida. Tulevasi töid saaks liigitada kolme rühma.

1. Mälukasutuse optimeerimine ja parameetrite määramise täpsuse tõstmine. Sageduspõhiste parameetrite määramine nõuab võrdlemisi palju mälu, mis on iOSi seadmetele arenduse tegemisel piirav faktor. Kui salvestada iga operatsiooni vahetulemus ning mitte hoida mälus kogu analüüsitavat audioklippi, on võimalik mälukasutust kahandada. Lisaks eksisteerib mitmeid võimalusi Accelerate raamistiku mälukasutuse optimeerimiseks. Parameetrite määramise täpsuse tõstmiseks oleks eelkõige vaja suurendada põhisageduse määramise täpsust. Seda saaks teha nii läbi FFT poolt töödeldavate diskreetide arvu suurendamise, HPS-ist efektiivsemate ja keerukamate põhisageduse määramise algoritmide kasutamise kui ka teistsuguste sageduse määramise meetodite kombineerimise FFT-ga.

2. Uute parameetrite ja efektiivsemate kombinatsioonide leidmine. Hetkel on mitmed sageduspõhised parameetrid korrelatsioonis üksteisega, mis kahandab märgatavalt iga parameetri

poolt pakutavat informatsiooni hulka. Seega tuleks praegu kasutatavast kuuest sageduspõhisest parameetrist ekstrahheerida kaks või kolm üksteisest sõltumatut parameetrit. See võimaldaks lihtsustada otsustuste puu või tehisliku närvivõrgu struktuuri, kahandada parameetrite määramiseks vajalikku aega ning tõsta klassifitseerimise täpsust. Lisaks tuleb leida täiendavad parameetrid, mis võimaldaksid eristada muusikat sisaldavat reklaami muusikast ning DJ kõnet reklaamist. Esimesel juhul on ilmselt võimalik tuvastada diktori kõne ja muusika vaheldust ning selle põhjal määrata, et tegemist on reklaamiga. Teisel juhul saab lähtuda DJ ja reklaami diktori intonatsioonide erinevusest, sest mõlemad järgivad üldjuhul kindlaid mustreid. DJ kõne on vaba ja rõõmsameelne, samas kui reklaamis püütakse inimest alateadvuslikult konkreetsete märksõnadega mõjutada. Seega peaks DJ-l olema vähem eriliselt rõhutatud sõnu ja reklaamis rohkem.

3. Kõnetuvastuse ja andmebaaside kasutamine. Kasutades kõnetuvastust koos antud töös kirjeldatud meetodiga on võimalik tõsta klassifitseerimise täpsust paari protsendi võrra. Seda eelkõige tänu reklaamile omaste märksõnade otsimisele. Põhiliseks probleemiks kõnetuvastuse puhul jääb siiski meetodi keerukus ja erinevate keelte rohkus. See-eest andmebaaside kasutamine võib olla täiesti reaalne viis algoritmi tulemusi parandada. Siin saaks kasutada kombineeritud lähenemist. Muusika identifitseerimiseks võib kasutada mõnda muusika tuvastamisplatvormi, nagu näiteks Echonest [26]. Kui rakendusel ei õnnestu lugu identifitseerida, siis on tõenäoliselt tegemist kas reklaami või DJ-ga. Ühtlasi on võimalik koos kasutajapoolse tagasisidega koostada lokaalset reklaamide andmebaasi: näiteks kui rakendus klassifitseerib reklaami muusikaks ja kasutaja annab sellest rakendusele teada, siis jäetakse rakenduses selle reklaami parameetrid meelde. See aitaks masinõppe mudeli järgi valesti klassifitseeruvat audioklippi korduse korral juba õigesti tuvastada.

Sõltumata valitud edasiarendamise suunast on selge, et kombineeritud meetodit kasutades on üsna tõenäoline saavutada 95-98%-line klassifitseerimise täpsus. See tähendab, et algoritm eksiks üksnes erandjuhtudel, väga loominguliste ja ebatavaliste reklaamide puhul.

Kasutatud kirjandus

[1] Andrew Green. Understanding Radio Audiences.. – *Warc Best Practice,* 2011, 8, 2-3. [*Online*] http://www.ipsos.com/mediact/sites/ipsos.com.mediact/files/pdf/Understanding_Radio_Audiences.pdf (02.04.2010)

[2] Philippe Generali, Warren Kurtzman, Bill Rose. What Happens When The Spots Come On? – *Radio Advertisement Bureau Report,* 2011, 1, 6-7. [*Online*] http://www.rab.com/public/reports/WhatHappens2011.pdf (05.04.2010)

[3] Lefa Teng, Michel Laroche, HiuHuang Zhu. The effects of multiple-ads and multiple- brands on consumer attitude and purchase behavior - Journal of Consumer Marketing 2007, 1, 27–35

[4] Laura Houston Santhaman, Amy Mitchell, Tom Rosenstiel. Audio: By the Numbers. *[WWW]* http://stateofthemedia.org/2012/audio-how-far-will-digital-go/audio-by-the-numbers/ (07.04.2013)

[5] WEKA Kodulehe URL. *[WWW]* http://www.cs.waikato.ac.nz/ml/weka/ (07.04.2013)

[6] Dalwon Jang, Seungjae Lee, Jun Seok Lee, Minho Jin, Jin S. Seo, Sunil Lee and Chang D. Yoo - *Automatic commercial monitoring for tv broadcasting using audio fingerprinting*, AES 29th International Conference, September 2006 *[Online]* http://mmp.kaist.ac.kr/paperdata/AES29th_Automatic_Commercial_Monitoring_for_TV_Broadcasting_Using_Audio_Fingerprinting.pdf (09.04.2013)

[7] Avery M. Abernethy, George R. Franke. The Information content of advertising: A meta-analysis – *Journal of Advertising,* 1996, 25, 1-18 *[Online]* http://www.jstor.org/discover/10.2307/4188999?uid=3737920&uid=2129&uid=2&uid=70&uid=4&sid=21102031598891 (14.04.2013)

[8] H. Khemiri. Automatic detection of known advertisements in radio broadcast with data-driven ALISP transcriptions - *Signal & Image Process. Dept., TELECOM ParisTech, Paris, France,* 223 – 228 *[Online]* IEEE Xplore Digital Library (10.04.2013)

[9] YACAST URL. *[WWW]* http://www.yacast.com (13.04.2013)

[10] John Saunders. Real-time discrimination of broadcast speech/music - *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* 1996 Mai, 993 – 996 *[Online]* IEEE Xplore Digital Library (13.04.2013)

[11] Courtenay Cotton. A Three-Feature Speech/Music Classification System : lõputöö. New York, Columbia Ülikool, 2006.

[12] Theodoros Theodorou, Iosif Mporas, Nikos Fakotakis. Automatic Sound Classification of Radio Broadcast News - *International Journal of Signal Processing, Image Processing and Pattern Recognition,* 2012, 5, Märts nr 1 *[Online]* http://www.sersc.org/journals/IJSIP/vol5_no1/4.pdf (13.04.2013)

[13] Estimating the Number of Apple Developers Coding Native Apps. *[WWW]* http://informationworkshop.org/2011/09/04/assessing-the-number-of-apple-developers-coding-native-apps/ (14.04.2013)

[14] About Objective-C. *[WWW]* http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html (14.04.2013)

[15] Taking Advantage of the Accelerate Framework. *[WWW]* https://developer.apple.com/performance/accelerateframework.html (14.04.2013)

[16] Core Audio Overview: What is Core Audio? *[WWW]* https://developer.apple.com/library/mac/#documentation/MusicAudio/Conceptual/CoreAudioOverview/WhatisCoreAudio/WhatisCoreAudio.html#//apple_ref/doc/uid/TP40003577-CH3-SW1 (14.04.2013)

[17] Core Audio Overview: Iphone Audio File Formats *[WWW]* https://developer.apple.com/library/mac/#documentation/MusicAudio/Conceptual/CoreAudioOverview/CoreAudioEssentials/CoreAudioEssentials.html#//apple_ref/doc/uid/TP40003577-CH10-SW47 (14.04.2013)

[18] Indrek Zolk. Helitöötlus arvutis. *[WWW]* http://home.tkug.tartu.ee/~zolki/konspektid/heli1.ps (20.04.2013)

[19] Patricio de la Cuadra, Aaron Master, Craig Sapp. Efficient Pitch Detection Techniques for Interactive Music - *Center for Computer Research in Music and Acoustics, Stanford University.*

[Online] https://ccrma.stanford.edu/~pdelac/research/MyPublishedPapers/icmc_2001-pitch_best.pdf (25.04.2013)

[20] Rob Schapire. COS 511: Theoretical Machine Learning – *Princeton University Lecture Notes [WWW]* http://www.cs.princeton.edu/courses/archive/spring13/cos511/scribe_notes/0205.pdf (01.05.2013)

[21] Eduard Petlenkov. Tehisnärvivõrgud ja nende rakendused – *Automaatikainstituud, Tallinna Tehnikaülikool. [Online]* http://www.dcc.ttu.ee/las/ISS0010/Tehisnarvivorgud-Eduard2004.pdf (10.05.2013)

[22] Neural Network Tutorial. *[WWW]* http://www.ai-junkie.com/ann/evolved/nnt1.html (11.05.2013)

[23] Quinlan, J. R. 1986. Induction of Decision Trees. *Mach. Learn.* 1, 1 (Mar. 1986), 81-106

[24] Toomas Ruuben. Digitaalne spektraalanalüüs - *TTÜ Raadio ja sidetehnika instituut. [Online]* http://www.lr.ttu.ee/spekter/2012sygis/DIGISPEKTER_6.pdf (12.05.2013)

[25] Proakis, J.G., D.G. Manolakis. 2000. Digital Signal Processing: Principles, Algorithms, and Applications.

[26] The Echo Nest *[WWW]* http://echonest.com (14.05.2013)

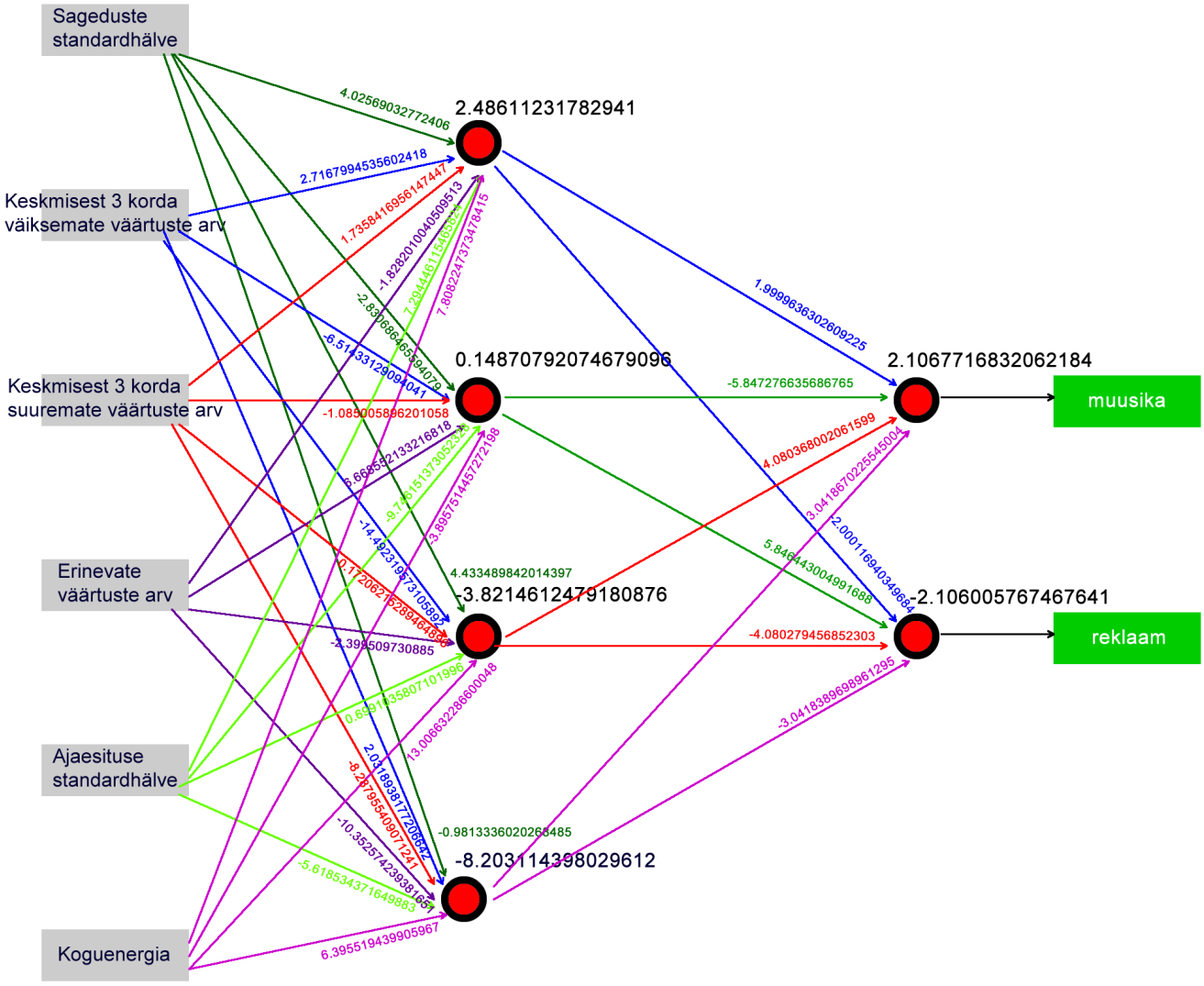
Lisa 1 – Otsustuste puu

Siin on esitatud WEKA poolt koostatud otsustuste puu, mis on teisendatud Objective-C koodiks. Otsustuste puu kasutab nii sageduspõhiseid kui ka ajaesituse parameetreid.

```
- (NSString *) classify: (SegmentParam *) clip
{
    if (clip.fileDeviation <= 0.261412) {
        if (clip.numberOf3xSmallerValues <= 8) {
            if (clip.energy <= 9138.537109) {
                return @"ad";
            } else {
                if (clip.numberOfDifferentValues <= 38) {
                    return @"music";
                } else {
                    if (clip.standardDeviation <= 418.493317) {
                        return @"music";
                    } else {
                        if (clip.numberOfDifferentValues <= 77.5) {
                            if (clip.numberOf3xBiggerValues <= 28.5) {
                                if (clip.standardDeviation <= 1030.356812) {
                                    if (clip.energy <= 48461.09375) {
                                        if (clip.numberOfDifferentValues <= 0.5) {
                                            return @"music";
                                        } else return @"ad";
                                    }  else return @"music";
                                } else return @"ad";
                            } else return @"ad";
                        } else return @"ad";
                    }
                }
            }
        } else return @"ad";
    } else {
        if (clip.energy <= 9471.505859) {
            return @"ad";
        } else return @"music";
    }
}
```

Lisa 3 – Tehislik närvivõrk

Siin on esitatud WEKA poolt treenitud tehislik närvivõrk. Iga noole juures on märgitud sünapsi kaal ja iga neuroni juures lävi, mille puhul ta informatsiooni edastab. Närvivõrk kasutab nii sageduspõhiseid kui ka ajaesituse parameetreid.



Lisa 3 – Sageduse määramine

```
- (float) runOnFftBuffer: (float *) samples andSize: (int) numSamples
{
    // Leia pikkus
    vDSP_Length log2n = log2f(numSamples);

    // Seadista FFT
    FFTSetup fftSetup = vDSP_create_fftsetup(log2n, FFT_RADIX2);

    // FFT jaoks peab numSamples olema alati kahe aste, seega on ta paarisarv
    int nOver2 = numSamples/2;

    // Aknafunktsioon
    float *window = (float *)malloc(sizeof(float) * numSamples);
    vDSP_hann_window(window, numSamples, 0);
    // Aknafunktsiooni rakendamin
    vDSP_vmul(samples, 1, window, 1, samples, 1, numSamples);

    COMPLEX_SPLIT A;
    A.realp = (float *) malloc(nOver2*sizeof(float));
    A.imagp = (float *) malloc(nOver2*sizeof(float));

    // Sisendandmed korrektsele kujule
    vDSP_ctoz((COMPLEX*)samples, 2, &A, 1, numSamples/2);

    // FFT teostamine, tulemused on A-s
    vDSP_fft_zrip(fftSetup, &A, 1, log2n, FFT_FORWARD);

    A.imagp[0] = 0;

    //Konverteeri kompleksarvud reaalarvudeks
    float amp[numSamples];
    amp[0] = A.realp[0]/(numSamples*2);
    for(int i=1; i<numSamples; i++) {

        if (i > 0)
        {
            amp[i]=A.realp[i]*A.realp[i]+A.imagp[i]*A.imagp[i];
        }

    }

    // Leia HPS-iga järjekorranumber
    int maxBin = [self doHPSOnFft: amp andSize: numSamples / 2 andHarmonics: 3];

    free(A.realp);
    free(A.imagp);

    // Leia põhisagedus
    float result = maxBin * (44100.0f/(float) numSamples);
```

```
    return result;
}

- (int) doHPSOnFft: (float *) spectrum
             andSize: (int) spectrumSize
        andHarmonics: (int) harmonics
{
    int i, j, maxSearchIndex, maxBin;

    maxSearchIndex = spectrumSize / harmonics;

    maxBin = 0;

    // Sisendandmete filtreerimine
    for (j = 1; j < maxSearchIndex; j++)
    {
        for (i = 1; i <= harmonics; i++)
        {
            spectrum[j] *= spectrum[j * i];
        }

        if (spectrum[j] > spectrum[maxBin])
        {
            maxBin = j;
        }
    }

    int correctMaxBin = 0;
    int maxsearch = maxBin * 3 / 4;

    for (i = 2; i < maxsearch; i++)
    {
        if (spectrum[i] > spectrum[correctMaxBin])
        {
            correctMaxBin = i;
        }
    }

    // HPSi oktaavivea parandamine
    if (abs(correctMaxBin * 2 - maxBin) < 4)
    {
        if (spectrum[correctMaxBin]/spectrum[maxBin] > 0.2)
        {
            maxBin = correctMaxBin;
        }
    }

    return maxBin;
}
```

Lisa 4 – Parameetrite määramine

```
+ (SegmentParam *) evaluateSegment: (NSArray *) segment
{
    SegmentParam *param = [[[SegmentParam alloc] init] autorelease];

    param.standardDeviation = [[self class] getStandardDeviation: segment];

    param.averageValue = [[self class] getSegmentAverage: segment];

    param.numberOf3xSmallerValues = [[self class] getXSmallValues: segment
                                                      withAverage: param.averageValue
                                                             andX: 3];

    param.numberOf3xBiggerValues = [[self class] getXBigValues: segment
                                                    withAverage: param.averageValue
                                                           andX: 3];

    param.numberOfSilentValues = [[self class] getSilentValues: segment
                                                    withAverage: param.averageValue
                                              andPercentThreshold: 15];

    param.numberOfDifferentValues = [[self class] getNumberOfDifferentValues:
segment];

    return param;
}

+ (float) getStandardDeviation: (NSArray *) array
{
    double sd;
    SDAccum *sdacc = [[[SDAccum alloc] init] autorelease];

    for(NSNumber *aNumber in array)
    {
        sd = [sdacc value: [aNumber doubleValue]];
    }

    return sd;
}

+ (float) getSegmentAverage: (NSArray *) array
{
    double sum = 0.0;
    int count = 0;

    for (NSNumber *num in array)
    {
        sum += [num doubleValue];
        count++;
    }
```

```
    return sum/count;
}

+ (int) getXSmallValues: (NSArray *) array
              withAverage: (float) average
                     andX: (int) x
{
    float threshold = average / x;
    int count = 0;

    for (NSNumber *num in array)
    {
        if ([num floatValue] <= threshold) count++;
    }

    return count;
}

+ (int) getXBigValues: (NSArray *) array
            withAverage: (float) average
                   andX: (int) x
{
    float threshold = average * x;
    int count = 0;

    for (NSNumber *num in array)
    {
        if ([num floatValue] >= threshold) count++;
    }

    return count;
}

+ (int) getSilentValues: (NSArray *) array
              withAverage: (float) average
      andPercentThreshold: (float) percent
{
    float threshold = percent/100.0f * average;

    int count = 0;

    for (NSNumber *num in array)
    {
        if ([num floatValue] <= threshold) count++;
    }

    return count;
}

+(int) getNumberOfDifferentValues: (NSArray *) array
{
    NSMutableArray *roundedNumbers = [NSMutableArray array];

    for (NSNumber *num in array)
```

```
    {
        int rounded = 10 * floor(([num floatValue]/10)+0.5);

        if (![roundedNumbers containsObject: @(rounded)])
        {
            [roundedNumbers addObject: @(rounded)];
        }
    }

    return [roundedNumbers count];
}
```