VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Václav Martinek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ROZPOZNÁVÁNÍ HUDEBNÍCH COVERVERZÍ POMOCÍ TECHNIK MUSIC INFORMATION RETRIEVAL

RECOGNITION OF MUSIC COVER VERSIONS USING MUSIC INFORMATION RETRIEVAL TECHNIQUES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Václav Martinek

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Tomáš Kiska

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Bc. Václav Martinek ***ID:*** 171620
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Rozpoznávání hudebních coververzí pomocí technik Music Information Retrieval

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je shrnout poznatky v oblasti tzv. Music Information Retrieval za účelem rozpoznávání hudebních coververzí za pomocí strojového učení. Bude sestavena databáze hudebních skladeb a jejich coververzí. Následně pak budou tyto skladby analyzovány z hlediska barvy zvuku, rytmiky a dynamiky. Cílem diplomové práce bude vytvoření databáze nahrávek různých žánrů a jejich coververzí, a dále návrh vyhodnocovacího systému na rozpoznání coververzí za pomoci strojového učení. Následně bude pak bude tento systém implementován a otestován.

DOPORUČENÁ LITERATURA:

[1] Music similarity and retrieval: an introduction to audio- and web-based strategies. New York, NY: Springer Berlin Heidelberg, 2016. ISBN 9783662497203.

[2] MÜLLER, M. Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications [online]. Springer International Publishing Switzerland, 2015, 483 s. ISBN 978-3-319-21945-5.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Tomáš Kiska

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato diplomová práce se zabývá návrhem a realizací systému rozpoznávajícího hudební coververze. Úvodní část je věnována výpočtu parametrů z audio signálu pomocí technik Music Information Retrieval. Následně jsou definovány různé podoby coververzí a hudební aspekty, které coververze sdílí. V práci je rovněž podrobně popsána tvorba a rozdělení databáze coververzí. Dále jsou zde uvedeny metody a techniky pro porovnání a zpracování vypočítaných parametrů. Pozornost je pak věnována metodě OTI, výpočet CSM a metodám, které se zabývají selekcí parametrů. Další část se věnuje návrhu systémů na rozpoznávání coververzí. V práci jsou pak srovnány již navržené systémy na rozpoznávání coververzí. Následně jsou popsány techniky strojového učení a evaluační metody pro vyhodnocení klasifikace. Větší část je věnována umělým neuronovým sítím. Poslední kapitola se zabývá implementací dvou systémů v prostředí MATLAB a Python. Tyto systémy jsou následně otestovány na vytvořené databázi coververzí. V závěru je diskutována úspěšnost těchto systémů a případné možnosti pro zlepšení.

KLÍČOVÁ SLOVA

audio coververze, binární, CSM, klasifikace, konvoluční neuronové sítě, MIR, mRMR, neuronové sítě, OTI, strojové učení

ABSTRACT

This master's thesis deals with designs and implementation of systems for music cover recognition. The introduction part is devoted to the calculation parameters from audio signal using Music Information Retrieval techniques. Subsequently, various forms of cover versions and musical aspects that cover versions share are defined. The thesis also deals in detail with the creation and distribution of a database of cover versions. Furthermore, the work presents methods and techniques for comparing and processing the calculated parameters. Attention is then paid to the OTI method, CSM calculation and methods dealing with parameter selection. The next part of the thesis is devoted to the design of systems for recognizing cover versions. Then there are compared systems already designed for recognizing cover versions. Furthermore, the thesis describes machine learning techniques and evaluation methods for evaluating the classification with a special emphasis on artificial neural networks. The last part of the thesis deals with the implementation of two systems in MATLAB and Python. These systems are then tested on the created database of cover versions.

KEYWORDS

audio cover song, binary, CSM, classification, convolution neural network, MIR, mRMR, neural network, OTI, machine learning

MARTINEK, Václav. *Rozpoznávání hudebních coververzí pomocí technik Music Information Retrieval*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 199 s. Diplomová práce. Vedoucí práce: Ing. Tomáš Kiska

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Václav Martinek
VUT ID autora:	171620
Typ práce:	Diplomová práce
Akademický rok:	2020/21
Téma závěrečné práce:	Rozpoznávání hudebních coververzí pomocí technik Music Information Retrieval

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Tomáši Kiskovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych chtěl poděkovat panu Ing. Štěpánovi Miklánkovi a Ing. Adamovi Ligocki za praktické rady při programování v prostředí Python a učení neuronových sítí. V neposlední řadě patří poděkování mé rodině za podporu.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato diplomová práce pojednává o rozpoznávání hudebních coververzí pomocí technik Music Information Retrieval (MIR). Techniky MIR můžeme volně přeložit jako získávání informace z hudby. Pomocí MIR jsme schopni parametrizovat hudební nahrávky a ze získaných parametrů pak vytvářet různé systémy. Za zmínku stojí například systémy na doporučování hudby, rozpoznávání hudebního žánru, rozpoznávání jednotlivých nástrojů atd. Hudební coververze je alternativní ztvárnění dříve či později zaznamenané a vydané skladby. Většinou jsou hudební coververze vytvářeny pouze pro radost z interpretování známé a oblíbené skladby, dříve byly vytvářeny za účelem zisku.

Hudební coververze podléhají autorskému právu. Z tohoto důvodu jsou systémy na rozpoznávání coververzí a identifikaci (hudby, obrazu, textu...) velice atraktivním tématem. Většina internetových hudebních přehrávačů (např. You Tube) a streamovacích služeb má systém na rozpoznávání hudební coververze a identifikaci skladeb. Z důvodu široké variace hudebních aspektů, které coververze mezi sebou mohou sdílet, jsou tyto systémy často velice složité. V důsledku rozšíření internetu je k dispozici velké množství dat hudebních coververzí. V roce 2018 byla koupena mobilní a počítačová aplikace *Shazam* na identifikaci hudby společností *Apple*.

Myšlenky algoritmů strojového učení byly definovány už v průběhu 50. let minulého století. V této době však nebylo možno používat strojové učení, protože nebyly vykonné počítače ani databáze obsahující velké množství dat. Dnes jsou algoritmy strojového učení využívány pro širokou škálu problémů. Tyto algoritmy jsou jednou z podoblastí umělé inteligence a zahrnují počítání s vektory a maticemi.

Cílem této práce je vytvořit databázi hudebních coververzí různých žánrů a jejich originálních verzí. Práce bude obsahovat návrh systému pro rozpoznávání coververzí, který bude využívat techniky MIR a algoritmy strojového učení. Hlavním výstupem bude selekce parametrů, které jsou nejvhodnější pro klasifikaci coververzí. Druhým a neméně důležitým výstupem pak bude samotný systém na klasifikaci coververzí, který bude otestován na vytvořené databázi. Následně bude vyhodnocena úspěšnost klasifikace.

1 Music Information Retrieval

Music Information Retrieval (MIR) byl vyvinutý v 90. letech a využívá se k získávání informací a parametrizaci hudby. V českém překladu znamená MIR doslova získání hudebních informací. Informace mohou být získávány přímo z hudební nahrávky nebo z kontextových zdrojů. Kontextové zdroje jsou například: notový zápis, webová stránka nebo MIDI. Prvotní myšlenka MIR bylo získání parametrů k vytváření systémů na doporučování a vyhledávání hudby. Mezinárodní nezisková organizace, která pořádá každoroční konference, zkoumá inteligentní metody MIR a vyhlašuje soutěže využívající MIR, se nazývá ISMIR (International Society for Music Information Retrieval).

V dnešní době se zvyšuje výpočetní výkon počítačů a roste i využití systémů využívajících MIR. Mezi problémy řešené pomocí MIR řadíme například: vyhledávání a doporučování hudby, rozpoznávání hudebního žánru, synchronizace hudební nahrávky s notovým zápisem, zjišťování původu interpretace atd. Pro komunitu ISMIR se identifikace coververze stala aktivní studijní oblastí již v roce 2006. Vzniklo tak několik studií zabývajících se touto problematikou. Existuje několik publikovaných metod, různé metody využívají různé testovací databáze. Některé studie například jen upravují některé parametry již uvedených metod a dosahují tak lepších výsledků [1, 2, 3].

Získávat informace z hudby nebo z kontextových zdrojů teoreticky můžeme pomocí libovolného programovacího prostředí nebo programovacího jazyka. V současné době je vytvořeno a stále vyvíjeno speciální MIR rozšíření pro prostředí MATLAB. Toto rozšíření nese název MIRtoolbox a je vyvíjeno na finské universitě v Jyväskylä [4]. Získané parametry pomocí MIRtoolboxu dělíme do tří základních skupin a to na: parametry popisující barvu zvuku, parametry popisující tempo a parametry popisující dynamiku. Pro úplné pochopení jednotlivých MIR parametrů je dobré uvést základní hudební parametry, které s MIR parametry souvisí.

1.1 Základní vlastnosti zvuku a tónů

Vše, co slyšíme, považujeme za zvuk ve slyšitelném spektru. Zvuk vzniká chvěním hmoty (chvěním různých těles, kapalin a plynů). K šíření zvuku je třeba prostředí, které zvuk přenáší, proto nelze zvuk přenášet ve vakuu. V každém prostředí se šíří zvuk jinou rychlostí. Nejčastěji se zvuk šíří vzduchem, ten pak rozechvívá sluchové ústrojí a vzniká tak sluchový vjem. Zvuky můžeme rozdělit na tóny a hluk. Tóny vznikají pravidelným chvěním hmoty a mají svojí specifickou výšku, barvu, sílu a délku. Tyto aspekty závisí na instrumentaci. Hluky jsou všechny ostatní zvuky kromě tónů a vznikají nepravidelným chvěním hmoty. Nelze u nich určit jednoznačně

výšku. Jsou to například údery do činelů. Základním rozdílem mezi hlukem a tónem je, že tón si můžeme zazpívat (jednoznačně definovaná výška), hluk ne [5]. Spojením dvou tónů vzniká souzvuk. Souzvuk alespoň tří tónů pak nazýváme akord.

1.1.1 Výška

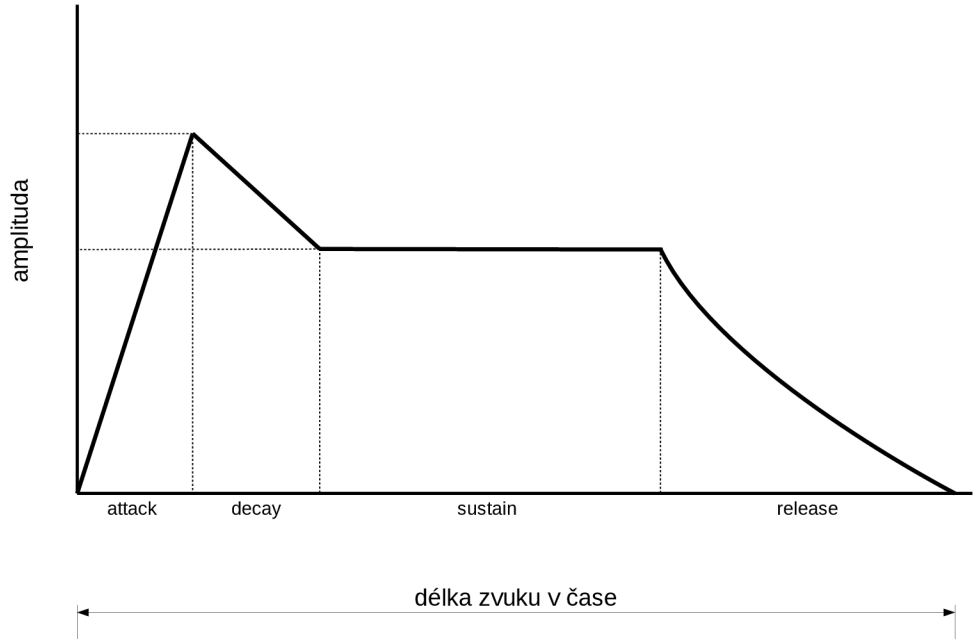
Podle výšky rozlišujeme tóny na hluboké a vysoké. Výška tónu zavisí na počtu kmitů kmitajícího tělesa za jednu vteřinu. Technicky řečeno na frekvenci kmitajícího tělesa. Frekvenci značíme f a její jednotkou je hertz (Hz). Jeden hertz pak znamená, že jeden kmit trval právě jednu vteřinu. Jako slyšitelné zvuky označujeme ty, které mají frekvenci základního fundamentu od 20Hz (hluboké tóny) do 20 000Hz (vysoké tóny). Sluch každého jedince se liší. Mění se i se stářím nebo při dlouhodobém vystavení vysokému akustickému tlaku. Výškový rozdíl mezi tóny se nazývá hudební interval. V notové osnově je výška tónu nahrazována značkou na příslušné lince nebo mezeře a zvoleném klíči.

1.1.2 Barva

Hrají-li dva rozdílné hudební nástroje stejný tón o stejné výšce (o stejné frekvenci fundamentálního tónu), tak dokážeme rozlišit barvu těchto tónů. Barva tónu, jinak nazývaná „*témbr*", závisí na počtu harmonických složek hudebního signálu a jejich poměrech. Pokud je některá část frekvenčního spektra zvýrazněna pro všechny tóny, tak dochází k rezonanci. Tuto rezonanci nazýváme jako „*formant*". Každý hudební nástroj nebo lidský hlas může mít několik formantových částí frekvenčního spektra. Označení barva má své opodstatnění. Při popisu barvy používáme výrazy stejné jako pro popis barvy vnímané zrakem popřípadě hmatem. Popis barvy bývá subjektivní. Každý si však umí představit tón, který má barvu „*temnou a hrubou*" nebo „*světlou a jemnou*". Na výslednou barvu má vliv typ nástroje a způsob instrumentace (typ trsátka, použití dusítka, typ smyčce, dynamika hry, preparování klavíru ...).

1.1.3 Síla

Podle zvukové intenzity tónu (síly) rozlišujeme subjektivně tóny na: „*slabé, silné, slabší, velmi silné atd.*" Základní značení intenzity hudebního přednesu a její změny se nazývá dynamika. Dynamika se v notové osnově značí zkratkami italského názvosloví (*p, mf, f, atd.*). Tóny se mohou v čase zesilovat nebo zeslabovat. Zesilování a zeslabování se v notové osnově značí symboly *crescendo* a *decrescendo*. Kromě popisu zvukové intenzity může síla tónu znamenat změnu frázování, tzn. změnu přízvuku, akcentu atd.



Obr. 1.1: ADSR obálka hudebního signálu zobrazující délku zvuku

1.1.4 Délka

Délku tónu chápeme jako časový úsek od vzniku tónu po jeho doznění útlumem nebo vzniku dalšího tónu. Pokud si představíme ADSR obálku (podle obrázku 1.1), tak se jedná o časovou vzdálenost od počáteční fáze vzniků tónu *attack* po fázi dozní-vání *release*. Dobu trvání je možno měřit objektivně v běžných časových jednotkách. V notové osnově se délka tónu značí podle příslušného tvaru zapsané noty nebo po-mlky. S délkou tónu přímo souvisí i metrum, frázování, tempo a rytmus. Dříve se k popisu tempa skladby používala opět italská názvosloví (*andante, allegro, presto, atd.*). Toto značení je velice relativní a nepřesné. Od 19. století se používá zařízení s názvem metronom, pomocí kterého jsme schopni přesně určit délku tónu v přísluš-ném tempu. Rychlost tempa udává počet úderů (většinou čtvrťových not) za jednu minutu. Jednotka tempa se značí jako BPM (beat per minute).

2 Vybrané parametry obsažené v MIRtoolboxu

Pomocí MIRtoolboxu můžeme z hudebního signálu vypočítat mnoho parametrů, pomocí nichž lze realizovat nespočet metod. Parametry se mohou vztahovat k základním vlastnostem zvuku a tónů nebo mohou být spíše technického rázu. Získané parametry můžeme rozdělit do tří základních úrovní. Úrovně se odlišují podle míry abstrakce parametrů. Vysokoúrovňové parametry popisují hudbu z pohledu běžného posluchače a nesou velké množství informací. Jejich nevýhodou je složitější výpočet, časová náročnost výpočtu a chybovost. Mezi vysokoúrovňové parametry patří například parametry popisující tóninu, akordy, hlavní melodii atd. Naopak nízkoúrovňové parametry zjistíme jednoduše přímo ze zkoumaného signálu rychle a přesně. Tyto parametry ale nenesou velké množství informací.

Další možností klasifikace parametrů obsažených v MIRtoolboxu je rozdělení parametrů podle základních vlastností zvuků a tónů.

2.1 Parametry popisující výšku

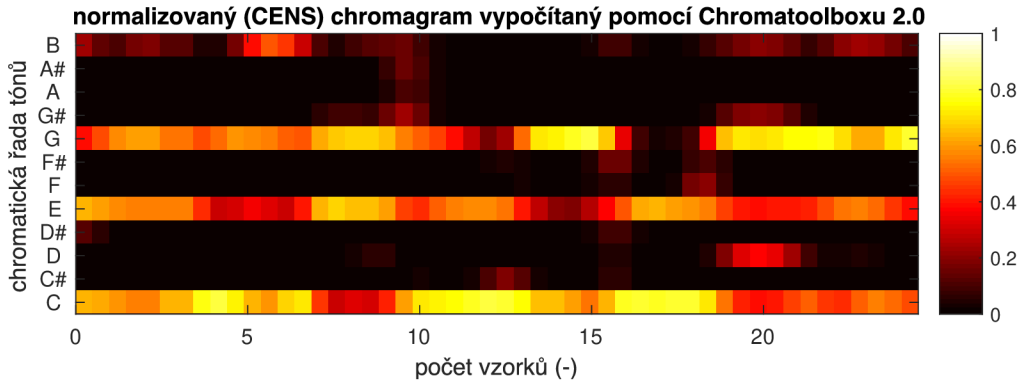
2.1.1 Mód

Tento parametr popisuje, zda-li je nahrávka v *moll* nebo *dur*. MIRtoolbox vrací číselnou hodnotu mezi <-1,+1>. Čím blíže je hodnota k +1, tím více předpokládáme durovou tóninu. Záporné hodnoty pak ukazují na tóninu mollovou.

2.1.2 Chromagram

Chromagram je velice obsáhlý parametr a bývá využívaný k identifikaci coververze. Mezi výhody chromagramu patří odolnost vůči změně tempa nebo dynamiky. Podob chromagramu může být několik, neexistuje žádná univerzální podoba, která je nejlepší pro všechny aplikace. Výsledkem chromagramu je matice nebo vektor o dvanácti řádcích. Počet sloupců matice chromagramu **nezávisí** na vzorkovací frekvenci zkoumaného signálu. Každých 100 vzorků matice chromagramu vypočítaného pomocí MIRtoolboxu charakterizuje $1s$ signálu. Chromagram má právě 12 sloupců, protože v západní hudbě rozlišujeme 12 půltónů na jednu oktávu. (C , $C^{\#}$, D, $D^{\#}$, E, F, $F^{\#}$, G, $G^{\#}$, A, $A^{\#}$, H(B)). [1] Celkem rozlišujeme 9 oktáv. Matice chromagramu bývá vypočítána pomocí použití vhodné banky filtrů, kde střední kmitočet každého

[1] tón $C^{\#}$ a D^{b} je v temperovaném ladění stejný tón – enharmonická záměna

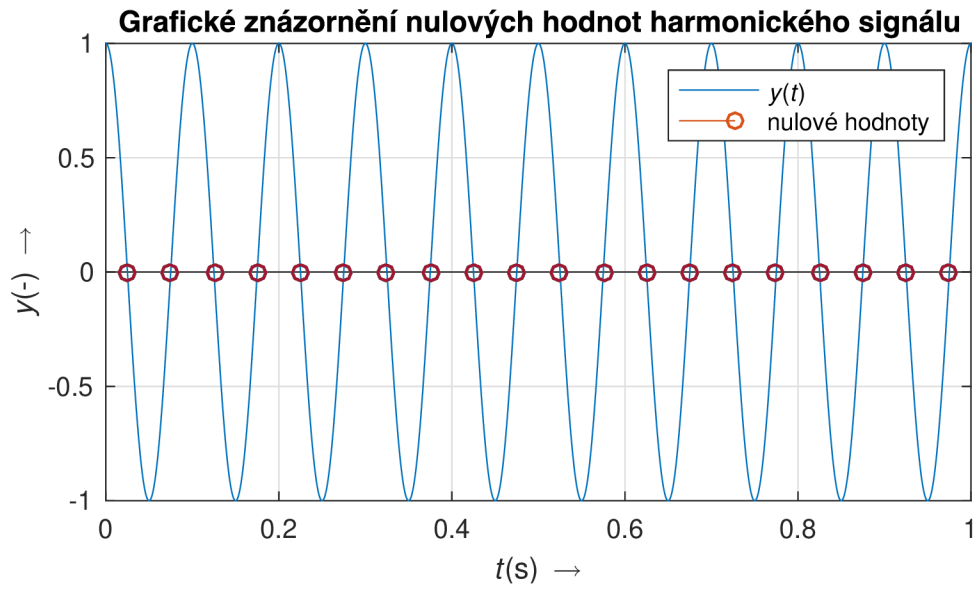


Obr. 2.1: Zobrazení normalizovaného CENS (Chroma Energy Normalized Statistics) chromagramu vytvořeného pomocí Chromatoolboxu 2.0 [6]

filtru odpovídá příslušnému tónu. Následně je v každém pásmu vypočítána krátkodobá efektivní hodnota. Tím získáme tzv. PCP (pitch class profile). K reprezentaci dvanácti půltónů pak přejdeme jednoduše součtem zjištěných hodnot pro stejný tón ve všech analyzovaných oktávách. Dále mohou být zjištěné hodnoty lineárně normalizovány od 0 do 1. Pro zohlednění lidského vnímání zvuku se často využívá logaritmická komprese [6]. Pro identifikaci popových coververzí slouží lépe chromagram založený na tzv. CPCP (cochlear pitch class profile). Pro výpočet chromagramu se používají funkce odvozené z přenosové funkce lidského ucha. Tento chromagram dokáže lépe sledovat hlavní melodii (většinou zpívanou) nahrávky [3].

2.1.3 Tónina

Bereme-li v potaz paralelní nebo modální stupnice, tak určení tóniny není vůbec jednoduchý úkol ani v reálném světě. Pomocí MIRtoolboxu můžeme zjistit přibližně tóninu nahrávky. Parametr s názvem *síla tóniny* (anglicky *key strength*) vrací číselnou hodnotu mezi <-1,+1>. Tyto hodnoty charakterizují korelační koeficienty korelace mezi chromagramem a speciálními objekty, které charakterizují všechny existující tóniny [4]. Dalším rozšířením parametru *síla tóniny* je parametr s názvem *klíč* (anglicky *key*). Tento parametr detekuje tóninu a skládá se ze špiček křivky vypočítané *sílou tóniny*. Jedná se o odhad, proto se nemůžeme spoléhat na přesnost. Tento parametr tedy vrací číselnou hodnotu v intervalu <1,12>.



Obr. 2.2: Grafické znázornění nulových hodnot harmonického signálu

2.2 Parametry popisující barvu

2.2.1 Počet průchodů nulou

V angličtině *zero-crossing rate* je nízkoúrovňový parametr s nízkou chybovostí, který nám říká, kolikrát protne hudební signál osu x. Číselná hodnota je extrahována přímo ze samotného signálu. Pomocí *počtu průchodů nulou* můžeme zjistit, jestli v nahrávce převažují „*rychlé*" perkusivní zvuky či „*pomalé*" zvuky například smyčců. Výpočet počtu průchodů nulou je řešení rovnice 2.1, grafické znázornění je na obrázku 2.2. [1].

$$ZRC_t = \frac{1}{2} \cdot \sum_{k=t \cdot K}^{(t+1) \cdot K - 1} |\mathrm{sgn}(s(k)) - \mathrm{sgn}(s(k+1))| \tag{2.1}$$

2.2.2 Melovské kepstrální koeficienty

Melovské kepstrální koeficienty mají svůj původ ve zpracování řeči, ale své využití nachází i v jiných aplikacích. Jedná se o parametr, který dobře charakterizuje hudební zabarvení, protože zahrnuje nelinearity lidského ucha. Nejprve je signál segmentován. Výpočet probíhá ve frekvenční oblasti, nejčastěji se volí Hammingovo okno s délkou 10 až 30 *ms*. Převod z časové do frekvenční oblasti je pomocí rychlé Fourierovy transformace (FFT). Následuje filtrace trojúhelníkovými filtry, které jsou rozloženy na melovské frekvenční stupnici. Lineární frekvenční rozložení v hertzích je převedeno na melovské logaritmické rozložení frekvencí v Melech. Přepočet frekvence na

Melovskou stupnici je definován takto: [7]

$$m = 2595 \cdot \log_{10}(1 + \frac{f}{700}), \tag{2.2}$$

m je vypočítaná frekvence v logaritmické melovské škále o jednotce [Mel], f je zadaná frekvence v hetrzích. Nakonec dochází k výpočtu výkonu spektra rámce, vynásobení výkonového spektra jednotlivými trojúhelníkovými filtry a zpětná diskrétní kosinova transformace (DCT) [8, 9, 1].

2.2.3 Bělost spektra

V angličtině se parametr *bělost spektra* nazývá *brightness*. Někdy se tento parametr překládá jako *jasnost*. *Bělost spektra* nám udává množství energie nad zvoleným mezním kmitočtem. Většinou se volí mezní kmitočet 1000, 1500 nebo 3000Hz. Vyjádření *bělosti spektra* je procentuální část energie nad mezním kmitočtem [4].

2.2.4 Parametry určené z obálky ADSR

Jedná se o parametry určené z amplitudové obálky ADSR zobrazené na obrázku 1.1. Můžeme tak určit dobu: náběhu – *attack*, poklesu – *decay*, držení – *sustain* a poklesu – *release*.

2.2.5 Spektrální centroid

Udává oblast frekvenčního pásma, ve kterém se soustředí nejvíce energie. Parametr souvisí s *„jasností"* zvuku. Spektrální rozložení lze popsat statistickými momenty. Jedná se geometrický střed pravděpodobnostního rozdělení spektra [9].

$$SC_t = \frac{\sum_{n=1}^{N} m_t(n) \cdot n}{\sum_{n=1}^{N} m_t(n)} \tag{2.3}$$

2.3 Parametry popisující dynamiku

2.3.1 Efektivní hodnota

Pro výpočet energie a intenzity signálu je jedním ze základních parametrů *efektivní hodnota*. Anglicky je nazývána RMS (*root mean square* – kořenový průměr čtverce

amplitudy). Jedná se o nízkoúrovňový parametr, který řadíme mezi parametry popisující dynamiku. Hlasitější a méně dynamické skladby budou mít *efektivní hodnoty* vyšší. Efektivní hodnota je definována jako: [4]

$$\mathbf{x}_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + ... + x_n^2}{n}}. \tag{2.4}$$

2.3.2 Výpočet nízké energie

Tento parametr je vhodný pro zjištění tichých pasáží nahrávky. Udává procento segmentů, jejichž energie je menší než průměrná energie nahrávky [4].

2.4 Parametry popisující tempo

2.4.1 Fluktuace

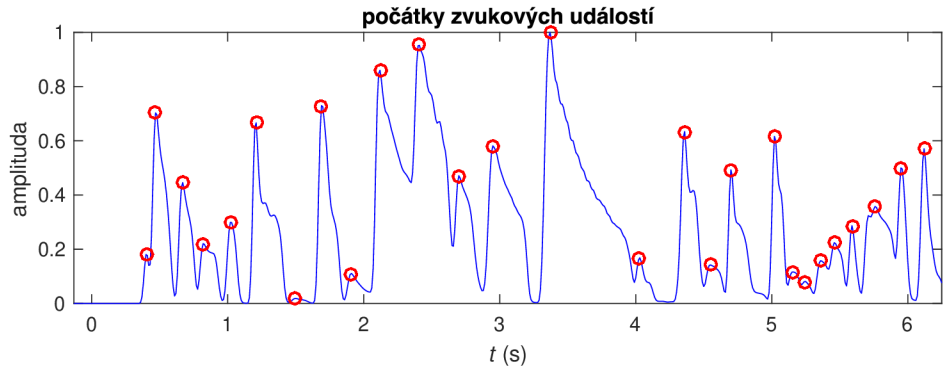
Jedním z možných odhadů rytmické periodicity je *fluktuace*. Je založena na výpočtu spektrogramu zahrnutím sluchového modelu vnějšího ucha a odhadu spektra v každém frekvenčním pásmu. Rozdělení do frekvenčních pásem je v Barcích nebo Melích. *Fluktuace* nám určuje periodicitu v každém frekvenčním pásmu.

2.4.2 Počátky zvukových událostí

Tento parametr je vypočítán odhadem detekční křivky, na které je provedena detekce zvukové události pomocí výběru vrcholů (*peak detection*). Jinými slovy nám parametr *počátky zvukových událostí* zobrazuje odhadované polohy událostí jako jsou jednotlivé tóny, akordy, atd. [4]. Grafické zobrazení počátků zvukových událostí lze pozorovat na obrázku 2.3.

2.4.3 Hustota zvukových událostí

Hustota zvukových událostí je vypočítávána z parametru *počátky zvukových událostí*. Jedná se o frekvenci *počátků zvukových událostí* za jednu sekundu.



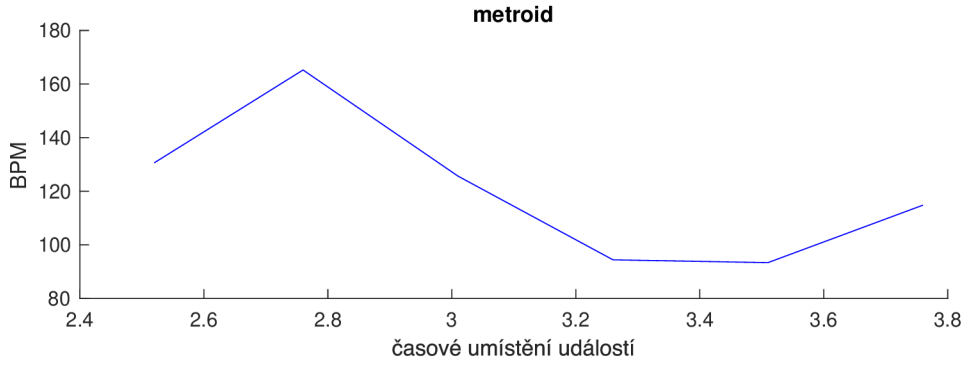
Obr. 2.3: Grafické zobrazení parametru počátky zvukových událostí

2.4.4 Tempo

Jedná se o velmi *„silný"* parametr, který určuje hodnotu tempa v BPM, viz 1.1.4. Výpočet je prováděn pomocí periodicity z křivky parametru *počátky zvukových událostí*.

2.4.5 Metroid

Parametr metroid je udáván v BPM a popisuje rytmické dělení. Vysoké hodnoty BPM parametru *metroid* popisují *„rychlé"* dělení (například v šestnáctinových notách). Naopak nízké hodnoty pak značí dlouhé tóny, které jsou běžně zapsané čtvrťovou notou. *Metroid* tedy popisuje křivka. Na obrázku 2.4 můžeme pozorovat příklad křivky *metroidu*.



Obr. 2.4: Grafické zobrazení parametru metroid

2.5 Metriky porovnávající parametry

Většina parametrů získaných pomocí MIRtoolboxu je rovnou zobrazována graficky v prostředí MATLAB. Chceme-li tyto parametry uložit do proměnných, které jsou typické pro MATLAB, musíme volat funkci MIRtoolboxu *mirgetdata()* [4]. Tímto postupem získáváme parametry jako matice, vektory nebo skaláry. Abychom mohli porovnávat jednotlivé parametry, tak je třeba zmínit různé porovnávací a vyhodnocovací metody.

2.5.1 Euklidovská metrika

Jedním ze základních parametrů pro porovnání vektorů a matic je výpočet velikosti. Existuje několik různých norem pro určování velikosti vektorů a matic. Nejběžnější a nejpoužívanější normou je euklidovská norma, která je nazývána také jako „*dvojková*". Pro vektor $\mathbf{x}$ o počtu vzorků N je tato norma definována jako:

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{N} |x_i|^2}. \tag{2.5}$$

Pro matice je možné použít Frobeinovu normu. Tato norma je definována jako: [10]

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M} |x_{ij}|^2}, \tag{2.6}$$

kde N a M je počet řádků / sloupců matice $\mathbf{X}$. Existují také řádkové nebo sloupcové normy matic.

Pokud chceme zjistit vzdálenost dvou vektorů $\mathbf{x}$ a $\mathbf{y}$, tak pomocí euklidovské metriky zjistíme vzdálenost jako:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}. \tag{2.7}$$

Vektory parametrů coververze a originální verze by měli mít v prostoru co nejmenší vzdálenost. Tato vzdálenost by měla být menší než vzdálenost mezi parametry zcela odlišných skladeb.

Prostředí MATLAB má implementovány výpočty několika různých norem. Příkaz *norm()* má několik různých parametrů, které ovlivňují typ normy. Standartně příkaz *norm()* počítá „*dvojkovou*" normu, parametrem *'fro'* pak volíme u matic normu Frobeinovu.

2.5.2 Kosinova metrika

Podobně jako u euklidovské metriky můžeme vypočítat vzájemnou vzdálenost vektorů pomocí kosinové metriky. Kosinova metrika je definována jako podíl skalárního

součinu dvou vektorů a součinem jejich euklidovských norem. Výsledek kosinovy metriky je kosinus úhlu, který tyto dva vektory svírají.

$$d(\mathbf{x}, \mathbf{y}) = \cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}. \tag{2.8}$$

Kosinova metrika se často používá ve vysokodimenzionálních prostorech, například v algoritmech porovnávajících podobnost textů, kde každému pojmu je přiřazena jiná dimenze a celý textový dokument je charakterizován vektorem. Velikost v každé dimenzi udává kolikrát se daný pojem v dokumentu opakuje. Pomocí kosinovy vzdálenosti pak měříme podobnost těchto dokumentů [11].

2.5.3 Dynamické borcení časové osy

Dynamické borcení časové osy je algoritmus, který se využívá v odvětví zpracování signálů a to nejen hudebních. Tento algoritmus je rozšířeně používaný již od 70. let 20. století [12]. Původně však byl vyvinut pro rozpoznávání řeči. Pokud získáme parametry různých zvukových coververzí, tak ve většině případů očekáváme podobnost těchto parametrů. Parametry se však budou měnit v čase nahrávky (vektory nebo matice parametrů). Jinými slovy nám budou popisovat, jak se nahrávka měnila v čase. Je potřeba srovnávat parametry tak, aby charakterizovaly stejný časový úsek skladby. Pro časovou synchronizaci dvou skladeb (získaných parametrů) nám slouží technika dynamického borcení časové osy (DTW – Dynamic Time Warping). Výsledkem algoritmu DTW je tzv. optimální cesta (warping path). Optimální cesta popisuje, jaký vzorek parametru $\mathbf{x} = [x_1, x_2, x_3, ..., x_i, ..., x_n]$ má být srovnáván s daným vzorkem parametru $\mathbf{y} = [y_1, y_2, y_3, ..., y_i, ..., y_m]$. Způsob výpočtu distanční matice $\mathbf{D}$ o velikosti $n \cdot m$ definujeme jako:

$$\mathbf{D}(n, m) = |\mathbf{x}(n) - \mathbf{y}(m)| + \min \begin{cases} D(n-1, m) \\ D(n, m-1) \\ D(n-1, m-1), \end{cases} \tag{2.9}$$

pomocí matice $\mathbf{D}$ pak vypočítáme optimální cestu [13, 14, 15].

Pro lepší pochopení algoritmu byl vytvořen jednoduchý obrázek 2.5, který zobrazuje jednotlivé kroky. Nejdříve definujeme vektory $\mathbf{x}$ a $\mathbf{y}$ jako $\mathbf{x} = [1, 2, 4, 3, 5, 3, 2, 3, 2, 5]$ a $\mathbf{y} = [2, 3, 2, 3, 5, 3, 4, 2, 1, 1]$. Vektory jsou zobrazeny na okrajích matice a jsou zvýrazněny červeně a modře. Podle rovnice 2.9 pak plníme jednotlivé pozice matice $\mathbf{D}$, začínáme od nejmenších indexů n a m. Po naplnění hledáme optimální cestu. Optimální cesta je hledána od pozice distanční matice s největšími indexy n a m. Každá pozice v distanční matici (kromě $n, m = 0$) má alespoň jednoho souseda s menším n nebo m. Tito sousedé jsou zobrazeni jako žlutá políčka v třetí části obrázku 2.5, nejdůležitější je soused s minimální hodnotou (tedy 0), který určuje

směr a další pozici optimální cesty. Stejným postupem pak hledáme další sousedy a nacházíme optimální cestu, která je zobrazena ve čtvrté části obrázku 2.5. Pokud bychom srovnávali dva zcela stejné vektory, tak optimální cesta bude na hlavní diagonále distanční matice.

Prostředí MATLAB má již tento algoritmus implementovaný. Příkaz *dtw()* slouží pro výpočet optimální cesty dvou vektorů nebo matic. Návratové hodnoty jsou pak vektory indexů (vztahujících se k vstupním vektorům) popisující optimální cestu. Poslední návratovou hodnotou je vzdálenost dvou porovnávaných vektorů (matic). Vzdálenost je vypočítána jako součet prvků vektoru optimální cesty. V našem případě na obrázku 2.5 je vzdálenost rovna 3 [13].

Y_{10}	2	16									
Y_9	3	15									
Y_8	2	13									
Y_7	3	12									
Y_6	5	10									
Y_5	3	6	$= \lvert x_1 - y_5 \rvert + D(x_1,y_4) = \lvert 1-3 \rvert + 4 = 6$								
Y_4	4	4									
Y_3	2	1									
Y_2	1	0									
Y_1	1	0									
		1	2	4	3	5	3	2	3	2	5
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}

Y_{10}	2	16	8	8	4	6	2	1	1	0	3
Y_9	3	15	8	6	3	5	1	1	0	1	3
Y_8	2	13	7	5	3	5	1	0	1	1	4
Y_7	3	12	7	3	2	2	0	1	1	2	4
Y_6	5	10	6	2	2	0	2	5	5	6	4
Y_5	3	6	3	1	0	2	2	3	3	4	6
Y_4	4	4	2	0	1	2	3	5	6	8	9
Y_3	2	1	0	2	3	6	7	7	8	8	11
Y_2	1	0	1	4	6	10	12	13	15	16	20
Y_1	1	0	1	4	6	10	12	13	15	16	20
		1	2	4	3	5	3	2	3	2	5
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}

Y_{10}	2	16	8	8	4	6	2	1	1	0	3
Y_9	3	15	8	6	3	5	1	1	0	1	3
Y_8	2	13	7	5	3	5	1	0	1	1	4
Y_7	3	12	7	3	2	2	0	1	1	2	4
Y_6	5	10	6	2	2	0	2	5	5	6	4
Y_5	3	6	3	1	0	2	2	3	3	4	6
Y_4	4	4	2	0	1	2	3	5	6	8	9
Y_3	2	1	0	2	3	6	7	7	8	8	11
Y_2	1	0	1	4	6	10	12	13	15	16	20
Y_1	1	0	1	4	6	10	12	13	15	16	20
		1	2	4	3	5	3	2	3	2	5
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}

Y_{10}	2	16	8	8	4	6	2	1	1	0	3
Y_9	3	15	8	6	3	5	1	1	0	1	3
Y_8	2	13	7	5	3	5	1	0	1	1	4
Y_7	3	12	7	3	2	2	0	1	1	2	4
Y_6	5	10	6	2	2	0	2	5	5	6	4
Y_5	3	6	3	1	0	2	2	3	3	4	6
Y_4	4	4	2	0	1	2	3	5	6	8	9
Y_3	2	1	0	2	3	6	7	7	8	8	11
Y_2	1	0	1	4	6	10	12	13	15	16	20
Y_1	1	0	1	4	6	10	12	13	15	16	20
		1	2	4	3	5	3	2	3	2	5
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}

Obr. 2.5: Grafické znázornění výpočtu distanční matice **D** a následné zvýraznění optimální cesty

3 Hudební coververze

Na hudební coververze můžeme nahlížet z několika úhlů a tato kapitola by nám měla odpovědět na otázku, co je to vlastně coververze a jak ji můžeme chápat. Hudební coververzí většinou označujeme alternativní ztvárnění dříve či později nahrané a vydané skladby. Pojem coververze je úzce spojen s autorským právem a duševním vlastnictvím. Tento fakt má za následek špatnou dostupnost hudebních databází. Hledání coververzí je velice atraktivní z hlediska komerčního využití. Ať už se jedná o systémy na identifikaci písně jako je například Shazam nebo systémy hlídající autorské právo. Například systém Content ID na serveru youtube chrání majitele písní a filmů. Využívá-li někdo ve svém videu píseň, kterou nevlastní a která není volně šiřitelná, tak mu nepřísluší zisk za shlédnutí videa. Prvotním účelem coververzí bylo dosažení zisku z hitů, které se držely na předních příčkách hitparád. Tak tomu bylo i v socialistickém Československu. Díky tomu známe mnoho zahraničních písní, které jsou přezpívané v českém jazyce. Problematiku autorského práva a duševního vlastnictví řeší právně například všeobecná úmluva o autorském právu. Jedná se o mezinárodní úmluvu sjednanou a podepsanou v Ženevě dne 6. září 1952 [16]. Někdy se coververze vytváří pro přizpůsobení vkusu, jazyku a kultury jiné země. Spousta coververzí vzniká jako vyjádření pocty autorovi nebo jen pro potěšení z hraní oblíbené písně. Různé coververze můžeme běžně najít například na serverech youtube[1], coverinfo[2], secondhandsong[3], coverville[4] atd. [17]. Tyto internetové stránky tak vytváří sociální komunitu uživatelů, kteří mezi sebou sdílí, objevují a nahrávají různé coververze. Coververze se liší od originální předlohy v několika aspektech jako je tempo, tónina, hudební struktura, nástrojové obsazení, barva zvuku atd.

3.1 Klasifikace coververzí

Hudební coververze je široký pojem, který je možno klasifikovat na několik jednotlivých typů coververzí. U každého z jednotlivých typů coververze očekáváme jiné aspekty, kterými se odlišuje od verze originální. Jako coververzi můžeme například chápat:

- klavírní přepis – často u klasické hudby, přepis orchestrálního provedení pro klavír
- instrumentální verze – verze, ve které se nezpívá, melodii hlasu hraje hudební nástroj

[1] http://www.youtube.com
[2] http://www.coverinfo.de
[3] http://www.secondhandsongs.com
[4] http://www.coverville.com

- živé vystoupení – záznam živého vystoupení
- remaster – na sklabě je proveden nový proces masteringu
- demo nahrávka – nahrávka zachycující prvotní nápad autora
- akustická verze – verze na akustické nástroje
- duet – hlavní nosnou melodii hraje či zpívá více umělců
- medley – jedná se o sestřih a propojení více skladeb v jednu
- remix – nově smíchaná skladba, typická pro DJ a elekronickou hudbu
- a cappella – vokální hudba bez hudebního dobrovodu
- odlišná instrumentace – často u klasické hudby, například danou skladbu hraje jiný orchestr

Každá z výše zmíněných coververzí se bude od originální předlohy odlišovat v různých hudebních aspektech. Při srovnání originální verze s odlišnou instrumentací v klasické hudbě budou jemné změny tempa, dynamiky, barvy zvuku či artikulace. Naopak srovnání akustické verze elektronické hudby může vést k velkým odlišnostem například v barvě. Největší odlišnosti pozorujeme na coververzích, které jsou v jiném hudebním stylu než originální předloha. Často nacházíme například coververze populárních písní předělané do jazzu či metalu. Díky této skutečnosti se identifikace coververze stává složitým úkolem. Jednotlivé změny hudebních parametrů definujeme jako změny: [17]

- barvy – barvu zvuku ovlivňuje daná instrumentace a způsob záznamu coververze
- tempa – tempo se často mění u živých vystoupení
- frázování – frázování je velice jemný hudební parametr, souvisí s tempem
- struktury písně – píseň můžeme rozepsat jako několik opakujících se bloků (předehra, sloka, refrén....) coververze, některé tyto bloky prodlužují, zaměňují nebo úplně vynechávají
- tóniny – tónina bývá zaměněna například kvůli možnostem interpreta (neschopnost zazpívat melodii v originální tónině)
- harmonizace – změnu harmonizace pozorujeme například v coververzích v jiném hudebním stylu než je předloha jako jsou jazzové coververze, kde se jedná o průchodové akordy, změnu basové melodie atd.
- slov nebo jazyka textu písně – záměnu slov a jazyka textu písně nacházíme u přetextovaných nebo přeložených písní
- šumu a hluku – živá vystoupení nebo coververze zaznamenané v odlišném prostředí budou vykazovat odlišný okolní hluk či šum

Pro lepší přehlednost rozdílných hudebních parametrů, které můžeme očekávat u různých typů coververzí byla vytvořena tabulka 3.1 shrnující tyto poznatky [17].

typ coververze	barva	tempo	frázování	struktura	tónina	harmonizace	změna slov	hluk a šum
klavírní přepis	*		*			*		
instrumentální verze	*						*	*
živé vystoupení	*	*	*					*
remaster	*							
demo nahrávka	*	*	*	*	*	*	*	*
akustická verze	*	*	*		*	*		*
duet	*	*	*	*	*	*	*	*
medley	*	*	*	*	*			*
remix	*	*		*	*	*	*	*
A capella	*	*	*	*	*	*		*
odlišná instrumentace	*	*	*					*

Tab. 3.1: Tabulka srovnání změny hudebních parametrů u několika typů coververze [17]

4 Databáze hudebních coververzí

V předešlé kapitole bylo vysvětleno, co je to coververze. K detekci coververzí jsou potřeba databáze, na kterých je možné tyto systémy testovat. Kvůli autorskému právu existuje velmi málo volně šiřitelných databází. Pro potřeby MIR však lze nalézt různé databáze skladeb. Bohužel je velmi málo těch, které obsahují coververze. Databáze představuje sadu dat, na kterých je daný systém testován, proto se setkáváme s názvem dataset.

4.1 Volně dohledatelné databáze

Volně šiřitelných databází je velmi málo. Myšlenka hledání coververzí pomocí MIR je velice atraktivní. V roce 2006 byla vyhlášena soutěž komunitou Music Information Retrieval Evaluation eXchange (MIREX) na hledání coververzí. První publikované práce zabývající se touto problematikou nacházíme už od roku 2006 [18]. Všechny dohledané práce musely použít nějaký dataset. Není však jednoduché tyto datasety získat. Většinou nelze získat dataset jako soubor zvukových stop, ale jen jako sadu parametrů získaných z těchto zvukových stop. Jedním z datasetů, který je možné volně získat, je *cover80*[1].

4.1.1 Cover80 dataset

Na tomto datasetu byla testována práce *A Tempo-Insensitive Distance Measure For Cover Song Identification Based On Chroma Features* [18]. Dataset obsahuje celkem 160 skladeb, 80 skladeb originálních a ke každé originální skladbě jednu coververzi. Většinou se jedná o záznam z živého vystoupení a originální studiovou nahrávku. Všechny nahrávky jsou monofonní a jsou ve formátu MPEG, 16kHz, 32kbps. Dataset není žánrově rozdělen a většinou se jedná a písně rockové nebo popové. Součástí datasetu jsou i funkce a skripty pro identifikaci coververze psané v prostředí MATLAB.

4.1.2 Million Song Dataset

Jedná se o internetový server[2], který zahrnuje volně dostupné funkce a metadata milionu současných zvukových skladeb. Účelem tohoto serveru je podporovat výzkumy algoritmů, pomoc vědcům MIR atd. Celková velikost metadat je 280GB. Tento dataset tedy neobsahuje skladby, které bychom si například mohli jednoduše

[1] https://labrosa.ee.columbia.edu/projects/coversongs/covers80/
[2] http://millionsongdataset.com/

poslechnout, ale obsahuje pouze funkce a metadata milionu skladeb, které poskytuje *The Echo Nest*[3]. To si můžeme představit například jako vypočítané chromagramy těchto skladeb. Million Song Dataset zahrnuje také databázi *Second Hand Songs*[4], která sdružuje coververze. S využitím metadat tohoto datasetu pracovala studie *Large-Scale Cover Song Recognition Using The 2D Fourier Transform Magnitude* [19].

4.2 Rozdělení vzorků databáze

Pro algoritmy strojového učení je třeba vhodně rozdělit databázi na několik podmnožin. Běžně se databáze rozdělují na dvě až tři podmnožiny. V praxi se může stát, že není databáze dostatečně objemná. V takovém případě se uplatňují techniky, kdy vytvoříme jednu z podmnožin z celé množiny dat a ostatní podmnožiny vytvoříme uměle. Celé rozdělení je znázorněno na obrázku 4.1.

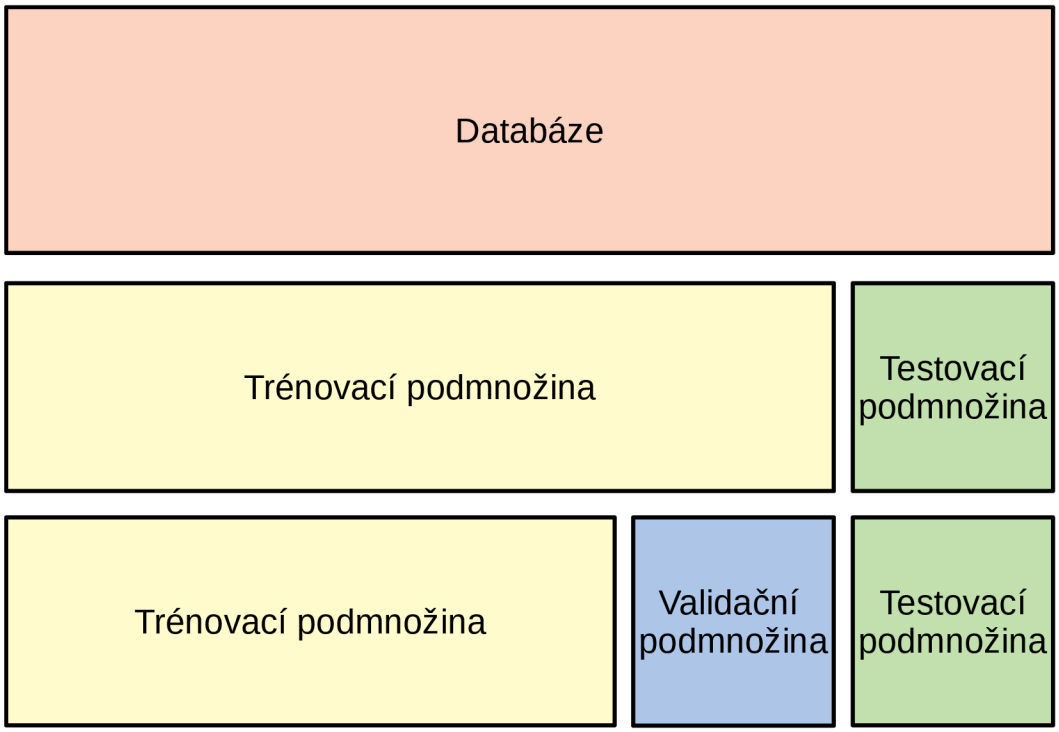
4.2.1 Rozdělení na dvě podmnožiny

Základní rozdělení vzorků databáze je na dvě podmnožiny: trénovací a testovací. Trénovací podmnožina slouží pro natrénování (naučení) algoritmu strojového učení. Abychom mohli vyhodnotit, jak je takový algoritmus naučený, musíme algoritmu předložit testovací podmnožinu. To jsou data, která algoritmu nebyla nikdy předložena. Pokud bychom vyhodnotili úspěšnost naučeného algoritmu po předložení trénovací podmnožiny, tak by byla úspěšnost klasifikace přesně taková, jaká byla při trénování. V praxi je trénovací podmnožina vždy několikanásobně větší než podmnožina testovací.

4.2.2 Rozdělení na tři podmnožiny

Druhým možným rozdělením vzorků databáze je dělení na tři podmnožiny. V tomto rozdělení je navíc podmnožina validačních dat, která je vytvořena z části trénovací podmožiny. Validační data slouží k monitorování změn v učení. Pomocí validačních dat můžeme monitorovat přeučení. Dále nám validační data zaručují, aby v procesu učení nebyly nikdy algoritmu předloženy data testovací.

[3] http://the.echonest.com/
[4] https://secondhandsongs.com/



Obr. 4.1: Jednoduché zobrazení rozdělení vzorků databáze

4.3 Vytváření vlastní databáze

Nemožnost získat databázi coververzí byla důvodem pro vytvoření vlastního datasetu. Při vytváření datasetu se potýkáme s několika problémy. Kvalitní dataset by měl obsahovat co nejvíce coververzí a originálních písní. Základním řazením písní ve vlastní databázi bylo rozdělení databáze na hudební žánry, ke každému žánru stejný počet skladeb a ke každé skladbě stejný počet coververzí. Rozdělení datasetu na hudební žánry nese spoustu výhod při vyhodnocování použitých systémů na detekci coververze, protože je možno srovnat, v jakém žánru systém vykazuje nejlepších výsledků nebo jaké parametry zvolit při detekci daného stylu. Čím více obsahuje dataset skladeb, tím více je náročný na paměť. Například dataset ve formátu .wav, $192kHz$, 32 bit-float bude téměř nemožné vytvořit a bude potřeba velkých datových úložišť pro manipulaci a uchování takového datasetu. Zároveň by všechny skladby v datasetu měly mít stejné nebo podobné technické parametry. Je velmi těžké najít coververze, které jsou ve stejném hudebním formátu, proto je nutno tyto vzorky předzpracovat 5.1.

Asi největším a nejlépe dostupným serverem obsahujícím originální písně a coververze všech žánrů je server youtube. Lidé z celého světa zde sdílí své coververze oblíbených písní. To má však za následek rozdílné parametry nahrávek. Všechny

zvukové stopy jsou součástí videa. Kvalitu audiostopy na youtube ovliňují tři zásadní skutečnosti:

1. technické vybavení při vzniku nahrávky (diktafon, nahrávací studio...),
2. programová hlasitost nahrávky (LUFS), formát zvuku a zvolený kodek při vytváření hudebního videa, [20]
3. zvolená kvalita přehrávaného nebo stahovaného videa [21].

Z tohoto důvodu je nemožné zajistit stejnou kvalitu všech získaných vzorků databáze. Získané nahrávky podléhají autorskému právu, v tomto případě však není porušeno, protože se jedná o školní práci [22].

4.3.1 Získávání vzorků

Existuje několik aplikací a služeb, které umožňují získat video nebo zvukovou stopu z videí na youtube. Pro získávání byla zvolena aplikace *youtube-DL*[5], která umožňuje získat několik vzorků pro jeden vyhledávací dotaz klíčového slova (search query). Tato skutečnost značně urychlí získávání vzorků, bohužel to má i své nevýhody, protože vždy **nezískáme** jen ty vzorky, na které jsme se dotazovali. Aplikace je kompatibilní pro platformy Windows i Mac a spouští se přes příkazový řádek (Windows) nebo terminál (MAC). Kód obsahuje několik parametrů, kterými lze volit výstupní parametry získaných vzorků. Aplikace byla použita na platformě MAC, kód vkládaný do terminálu pak vypadá takto:

Výpis 4.1: Příklad syntaxe spouštění aplikace youtube-DL v terminálu

```
youtube-dl "ytsearch15:'Hledaná skladba '"
-i --no-check-certifi cate --max-filesize 20m
--title --write-thumbnail --add-metadata
--metadata-from-title "%(artist)s - %(title)s"
--extract-audio --audio-format mp3 --audio
-quality 0 --postprocessor-args "-ar 44100"
```

Důležitou částí kódu je *ytsearch15* (číslo patnáct udává, kolik souborů má aplikace stáhnout), *audio-format*, *audio-quality* nebo *postprocessor-args*. Poslední tři parametry ovlivňují kvalitu a velikost výsledných souborů. *Quality 0* znamená nejlepší dostupná zvuková kvalita (nejvyšší dostupný datový tok). Další z parametrů určují například maximální velikost jednoho souboru, pojmenování skladby nebo získávání metadat. Více o volbě parametrů lze nastudovat přímo z dokumentace o aplikaci.

[5] https://youtube-dl.org/

Tímto způsobem bylo možné získat velké množství vzorků v relativně krátkém čase. Tato databáze obsahuje stejný počet vzorků pro každý hudební styl. Vzorky byly vybírány podle počtu existujících coververzí, které byly typické pro daný hudební styl (nejlépe tak, že tento daný styl definovaly). Po získání vzorků následoval proces jejich kontroly. Bylo nutné vyřadit ty, které nepatřily mezi hledané coververze nebo kvalitativně vyčnívaly z ostatních vzorků. Tato ruční korekce byla časově náročná. Databáze obsahuje celkem 1200 skladeb, 10 hudebních žánrů, pro každý žánr 20 originálních skladeb a pro každou skladbu 5 coververzí. Velikost databáze je 12,5GB a všechny skladby jsou ve formátu .mp3. Z tohoto důvodu není možné přiložit databázi jako součást práce, vybrané písně je však možno najít online v tabulce, na kterou je odkazováno v příloze.

5 Metody zpracovávající parametry

V této kapitole je definováno několik metod, které zpracovávají extrahované parametry tak, aby bylo možno tyto parametry použít k rozpoznávání coververzí. Jsou zde vybrány pouze takové metody, které budou následně použity při řešení.

5.1 Předzpracování vzorků

Jak již bylo zmíněno v subkapitole 4.3 (pojednávající o tvorbě vlastní databáze coververzí), je třeba vzorky nejdříve předzpracovat. Předzpracování vzorků popisuje obrázek 5.1, podle něhož byla vytvořena funkce v prostředí MATLAB s názvem `predzprac.m`. Tím zajistíme co nejvíce stejné podmínky pro porovnávání parametrů. Funkce předzpracování má několik volitelných parametrů, které jsou voleny v závislosti na používaném vyhodnocovacím systému.

5.1.1 Převod stereo signálu na mono signál

Všechny nahrávky jsou uloženy jako signál pro pravý a levý reproduktor (stereo signál). Některé již vznikly například snímáním jedním mikrofonem, tzn. signál pro pravý a levý reproduktor je stejný. Proto je dobré převést všechny nahrávky na mono signál reprezentovaný jedním sloupcovým vektorem.

5.1.2 Sjednocení časového úseku

Některé systémy využívají k vyhodnocování coververzí prvních n sekund nahrávky. Tím se zajistí stejný rozměr všech parametrů, které jsou extrahovány v čase. Většinou jsme schopni u popových skladeb identifikovat coververzi pomocí refrénu. Z tohoto důvodu je vhodné volit takový čas, aby skladba obsahovala alespoň jeden refrén. Výběr je pak jednoduchý, prvních n sekund nahrávky je charakterizováno jako všechny vzorky vektoru nahrávky od první pozice po pozici označenou číslem $n \times$ *vzorkovací kmitočet*. Pokud je daná nahrávka kratší jak n sekund, tak zbylé hodnoty ponecháme rovny nule.

5.1.3 Vzorkovací frekvence, datový tok a podvzorkování

Všechny získané vzorky mají stejnou vzorkovací frekvenci (44 100Hz), tento fakt je zajištěn pomocí aplikace *youtube-DL*. Nejvíce se od sebe liší v počtu bitů za jednu sekundu tedy v datovém toku. *Youtube-DL* stahuje nejlepší možnou kvalitu, tzn. nejvyšší možný datový tok dané nahrávky. Přesto se nahrávky liší, této skutečnosti však

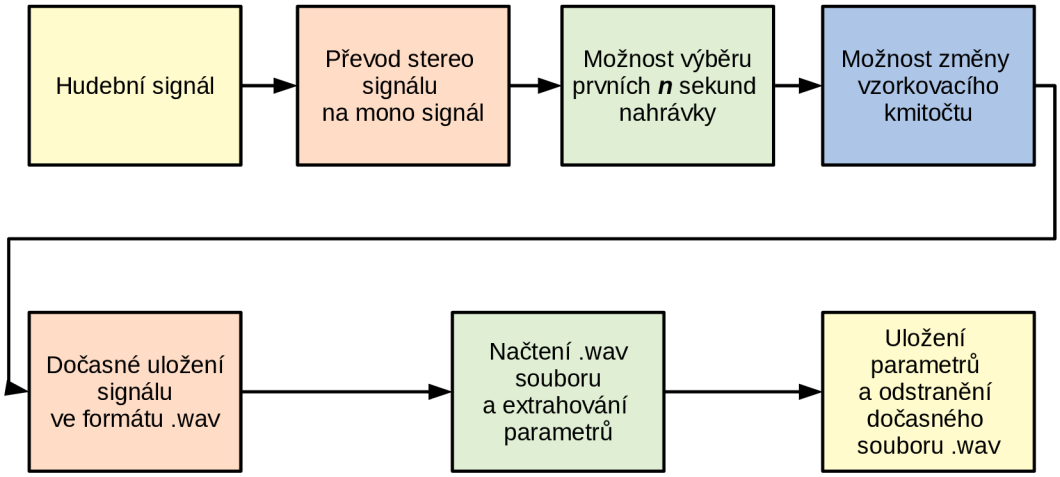
nelze nijak předejít. Hudební soubory jsou na youtube uloženy ve formátu .mp3, tzn. získávat soubory .wav nemá smysl a jen to zvyšuje objem databáze.

Pro menší výpočetní náročnost a tedy rychlejší výpočty, je vhodné vzorky podvzorkovat. Podvzorkováním rozumíme snížení vzorkovací frekvence. Podle Shannon - Kotělnikova teorému musí být vzorkovací frekvence alespoň dvakrát vyšší než nejvyšší frekvence vzorkovaného signálu, aby nedošlo k překrývání spekter a tím ke zkreslení (*aliasingu*). Z těchto důvodů se při vzorkování používá tzv. antialiasingový filtr. Antialiasingový filtr je analogový filtr typu dolní propusti, jehož mezní kmitočet a strmost jsou voleny tak, aby před vzorkováním utlumil nežádoucí vysoké kmitočty, které by způsobovaly aliasing. V této části bude signál podvzorkován, z tohoto důvodu je třeba použít antialiasingový filtr (digitální). Samotné podvzorkování je v prostředí MATLAB zajištěno funkcí s názvem *resample()*. Parametry funkce resample jsou: signál, jehož vzorkovací frekvence má být změněna, koeficient nadvzorkování, koeficient podvzorkování a řád strmosti antialiasingového FIR filtru dolní propusti. Změna vzorkovací frekvence musí být prováděna v celočíselném poměru. Chceme-li podvzorkovat signál, tak musí být poměr mezi novou a původní vzorkovací frekvencí celé číslo. Z tohoto důvodu je třeba někdy signál nejprve nadvzorkovat na nejmenší společný násobek stávající vzorkovací frekvence a požadované vzorkovací frekvence. Z nadvzorkovaného signálu je potom signál podvzorkován s celočíselným poměrem na požadovaný kmitočet. Řád strmosti FIR filtru je vypočítáván jako $2 \times n \times \max(p, q)$, kde n je námi zvolená konstanta, která přímo ovlivňuje řád strmosti, p a q jsou poměry nadvzorkování a podvzorkování [23].

Pro zpracování řeči je typický vzorkovací kmitočet 8kHz, nejmenší běžně používaný vzorkovací kmitočet pro hudbu je 44,1kHz (CD standard). Při velkém podvzorkování (nízká vzorkovací frekvence) dochází v audio signálu k úbytku vysokých frekvencí. Tyto frekvence však nemusí obsahovat informace pro identifikaci coververze. Pokud bychom stále zmenšovali vzorkovací kmitočet, tak by coververze a verze originální nemusela být již rozeznatelná. Nastává otázka, do jakého zmenšování vzorkovacího kmitočtu je stroj (počítač) schopen tyto verze rozpoznat a do jakého zmenšování vzorkovacího kmitočtu člověk.

5.1.4 Převod audio formátu

Některé toolboxy v programu MATLAB (například Chroma Toolbox, SM Toolbox) dokáží pracovat pouze s formáty .wav. Z tohoto důvodu bude před extrakcí parametrů převedena nahrávka dočasně do formátu .wav, poté budou extrahovány všechny potřebné parametry. Před uložením předzpracovaného signálu do formátu .wav je signál normalizován mezi hodnoty $< -1, 1 >$. Normalizace nám zajišťuje, aby nebyl audio signál zkreslený. Algoritmy pro detekování coververzí budou pracovat pouze



Obr. 5.1: Jednoduché zobrazení systému pro předzpracování vzorků

se získanými parametry. Po získání parametrů bude soubor .wav odstraněn. Tímto způsobem budou zajištěny nejvíce rovnocenné podmínky pro každou skladbu.

5.2 Zmenšování dimenze vektorového prostoru

Extrahované parametry mohou být skaláry, vektory nebo matice. Skalár (číselná hodnota) vzniká například při extrahování parametru *key strength*. Pokud zavedeme segmentaci, tj. pravidelné extrahování parametru z krátkých úseků skladby, tak výstupem je řádkový vektor. Pokud je parametr získáván v jednotlivých frekvenčních pásmech, tak bude výstupem skalár v každém frekvenčním pásmu, tzn. výstupem bude sloupcový vektor. Jestliže se jedná o parametr, který je získáván v jednotlivých frekvenčních pásmech a zároveň je segmentován (*melovské kepstrální koeficienty*), tak je výstupem matice o rozměrech $m \times n$, kde m je počet pásem, ve kterých je parametr extrahován a n je závislé na délce skladby a délce segmentu. Při dodržení stejně dlouhých délek segmentů a stejné vzorkovací frekvence je rozměr (dimenze) vektoru a matic závislý na délce skladby.

Abychom mohli parametry dobře zpracovávat, porovnávat a zároveň zmenšili nároky na paměť, tak je dobré zmenšit jejich dimenzi. Z matic vytvořit vektory a z vektorů skaláry. Většina algoritmů strojového učení vyžaduje na vstupu vektor příznaků. To je takový vektor, jehož hodnoty charakterizují daný vzorek pro danou klasifikační úlohu a neobsahují nadbytečná data. Jednou z možností, jak zmenšit dimenzi parametrů z MIRtoolboxu a co nejvíce zachovat vypovídající hodnotu, je použít statistické metody. Vzniknou tak nové parametry, staré nebudou zacho-

vány. Pro každý vzorek (každou skladbu) vznikne vektor s jednotnými rozměry. Základní statistické metody budou popsány v následujících subkapitolách. Prostředí MATLAB má tyto metody implementováno jako funkce, které pracují s vektory i maticemi.

5.2.1 Aritmetický průměr

Aritmetický průměr je vypočítán jako součet všech hodnot vydělen jejich počtem. Aritmetický průměr vektoru $\mathbf{x}$ o n prvcích můžeme matematicky zapsat jako:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{5.1}$$

5.2.2 Medián

Pokud seřadíme vzestupně n prvků vektoru $\mathbf{x}$, tak medián je definován jako:

$$\mathrm{Me}(x) = \begin{cases} \frac{x_{(n+1)}}{2} & \text{pro lichá } n, \\ \frac{x_{(n/2)} + x_{(n/2)+1}}{2} & \text{pro sudá } n. \end{cases} \tag{5.2}$$

5.2.3 Rozptyl

Rozptyl (*variance*) vektoru $\mathbf{x}$ značíme jako $\mathrm{Var}(\mathbf{x})$. Rozptyl vektoru $\mathbf{x}$ je definován jako průměr druhých mocnin vzdáleností od průměru vektoru $\mathbf{x}$. Matematicky je definován jako:

$$\mathrm{Var}(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2. \tag{5.3}$$

5.2.4 Směrodatná odchylka

Podobně jako rozptyl nám určuje směrodatná odchylka, jak moc jsou hodnoty rozptýleny od průměru. Směrodatná odchylka se značí řeckým písmenem σ a je vypočítána jako druhá odmocnina rozptylu.

$$\sigma = \sqrt{\mathrm{Var}(x)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{5.4}$$

5.2.5 Kvartil

Kvartil jsou skalární hodnoty, které rozdělují statistický soubor na čtvrtiny. První kvartil Q_1 je hodnota, která oddělí nejmenších 25 % prvků od nejvyšších 75 %. Druhý kvartil Q_2 rozdělí setříděné prvky na poloviny, proto je roven mediánu. Třetí kvartil Q_3 odděluje 75 % nejmenších prvků od zbylých 25 % prvků. Pro výpočet kvartilů je několik metod, jejichž výsledky se mohou lehce lišit.

5.2.6 Mezikvartilové rozpětí

Mezikvartilové rozpětí (*Interquartile range*) je hodnota rozdílu mezi třetím a prvním kvartilem.

$$IQR = Q_3 - Q_1 \tag{5.5}$$

5.2.7 Percentil

Podobně jako kvartil rozděluje percentil statistický soubor. Kvartil rozděluje na čtvrtiny a percentil na setiny. První percentil $1p$ je hodnota, pod kterou leží 1 % hodnot. První kvartil je dvacátý páty percentil. V praxi se často počítá 1. a 99. precentil.

5.2.8 Mezipercentilové rozpětí

Mezipercentilové rozpětí (IR) je hodnota rozdílu mezi prvním a devadesátým devátým percentilem.

$$IR = 99p - 1p \tag{5.6}$$

5.3 Redukce příznakového prostoru

Příznakovým prostorem se rozumí vektor parametrů (příznaků), který charakterizuje daný vzor. Čím více hodnot takový vektor obsahuje, tím více bude náročnější klasifikovat jednotlivé vzory. Z těchto důvodů byly vyvinuty metody, které jsou schopné selektovat pouze takové příznaky, které jsou po danou klasifikaci co nejvíce významné. Volba správných příznaků má vliv na přesnost klasifikace. Existuje několik metod a přístupů, jak redukovat příznakový prostor, rozlišujeme mezi transformací a selekcí. Jako transformaci můžeme chápat vytváření nového parametru

pomocí matematických operací ze stávajících parametrů. V takovém případě nevíme, co nový parametr charakterizuje a jakou má vypovídající schopnost. Selekce znamená výběr parametrů z jejich celkového množství.

5.3.1 Algoritmus mRMR

Jedním ze zástupců selektivních metod je algoritmus s názvem minimum Redundancy Maximum Relevance (mRMR). Minimální redundance znamená minimální nadbytečnost. Maximální relevance označuje maximální významnost (důležitost). Maximální relevance hledá vhodné funkce z předpisu 5.7, které aproximují $D(S, c)$ ze vztahu 5.8, kde S je sada funkcí a m množství hledaných příznaků x_i. Hlavním účelem je najít takové funkce S se střední hodnotou všech vzájemných informací mezi třídou c a příznaky x_i.

$$\max \; D(S,c), \quad D = \frac{1}{|S|} \sum_{x_i \in S} I(x_i; c) \tag{5.7}$$

$$\max \; D(S,c), \quad D = I(\{x_i, i = 1, ..., m\}; c) \tag{5.8}$$

S velkou pravděpodobností budou funkce vybrané maximální relevancí mít velkou redundanci. Jinými slovy budou obsahovat mnoho nadbytečných (závislých) funkcí. Pokud jednu ze dvou závislých funkcí odstraníme, příslušná třídně diskriminační síla se moc nezmění. Z tohoto důvodu je zavedena podmínka minimální redundance, která je definována vztahem 5.9.

$$\min \; R(S), \quad R = \frac{1}{|S|} \sum_{x_i, x_j \in S} I(x_i; x_j) \tag{5.9}$$

Minimální redundance a maximální relevance je kombinací vztahů 5.7 a 5.9, proto je zaveden operátor Φ, který oba vztahy kombinuje [24].

$$\max \; \Phi(D, R) \quad \Phi = D - R \tag{5.10}$$

5.3.2 Algoritmus dopředného a zpětného výběru

Algoritmy dopředného (**SFFS** - *Sequential Forward Floating Selection*) a zpětného (**SFBS** - *Sequential Backward Floating Selection*) plovoucího výběru jsou označované za hladové algoritmy. *SFFS* a *SFBS* jsou rozšířené (plovoucí) varianty algoritmů *SFS* a *SBS*. Anglicky se tyto algoritmy nazývají slovem *greedy*, které překládáme jako hladový, nenasytný, chamtivý. Jedná se o iterační algoritmy, které v každém kroku zvolí jeden parametr. Tento parametr se pak podílí i na dalších iteracích a tak ovliňuje celé řešení a tím nemusí dosáhnout požadovaného optima. Tyto algoritmy vybírají p parametrů z celkové množiny parametrů, kde p je menší než

celkový počet parametrů vektoru příznaků X. Základní myšlenka je taková, že algoritmus iterativně přidává (dopředný) nebo odebírá (zpětný) jeden parametr (y^+, y^-) na základě úspěšnosti klasifikace, dokud není selektováno k parametrů. Z těchto kritérií vyplývá, že k dopřednému a zpětnému výběru je potřeba klasifikátor, na jehož úspěšnosti klasifikace závisí selektování parametrů. Algoritmus *SFS* lze definovat v několika krocích.

- **vstup** – vektor příznaků - $X = (x_1, x_2, x_3, ..., x_d)$
- **výstup** – redukovaný vektor příznaků - $Y_k = (y_j | j = 1, 2, 3, ..., k; y_j \in X)$ kde $k = (0, 1, 2, 3, ..., d)$
- **inicializace** – $Y_0 = \varnothing, k = 0$
- **krok 1.** – nalezení a přidání nového příznaku podle:
  $y^+ = \arg\max J(Y_k + y)$ kde $y \in X - Y_k$
  $Y_{k+1} = Y_k + y^+$ a $k = k+1$, y^+ maximalizuje funkci kritéria spojenou s nejmenší chybou klasifikace
- **krok 2.** – opakování prvního kroku dokud není splněno $k = p$

Algoritmus zpětného výběru (*SBS*) pracuje na obdobném principu. V inicializační fázi je vektor Y_0 roven vektoru příznaků X a $k = d$. V každém kroku pak hledá a odebírá taková y^-, která zlepšují klasifikaci a zmenšuje k. Plovoucí verze algoritmů přidává jeden iterační krok. Po prvním kroku je při každé iteraci k zkoumán dosavadní výběr parametrů (obsah vektoru Y_k), kdy jsou přidávány nebo odebírány příznaky z vektoru Y_k za účelem zlepšení přesnosti klasifikace [25, 26].

5.4 Transpoziční metoda OTI

Některé systémy využívají k identifikaci coververze parametry založené na tzv. *pitch class profile*. Většinou se jedná o parametr chromagram. Coververze se často liší v tónině. Pro srovnání dvou chromagramů je výhodné sjednotit jejich tóninu. Systémy, které využívaly sjednocení tóniny vykazovaly až o 17 % větší přesnost klasifikace než systémy, které tóninu nesjednocovaly [27].

Jednou z metod, jak sjednotit tóninu dvou chromagramů, je metoda cirkulačního posunu chromagramu. Chromagram obvykle obsahuje 12 řádků (C, C$^\#$, D, D$^\#$, E, F, F$^\#$, G, G$^\#$, A, A$^\#$, H(B)). Cirkulačním posunem rozumíme posouvání jednotlivých řádků matice chromagramu. Posunutí o jeden index znamená posunout celý řádek s indexem 1 na řádek s indexem 2, poslední řádek je pak posunut na první pozici. Metody optimálního transpozičního indexu řeší, jak najít index posunutí, který zajistí sjednocení tóniny dvou chromagramů. Jednou z možností je odhad tóniny pomocí parametru obsaženém v MIRtoolboxu s názvem *key*. Tento odhad tóniny

je nepřesný a není vhodný pro algoritmus sjednocení tóniny. Mezi hlavní dvě metody patří metoda globálních chromagramů a metoda hrubé síly. Metoda globálních chromagramů měla průměrnou přesnost (*MAP - Mean Average Precision*) podle [27] 0,698. U metody hrubé síly byla průměrná přesnost 0,729 [27].

5.4.1 Metoda založená na globálním chromagramu

Matici chromagramu originální písně označíme jako **A**, matici chromagramu coververze jako **B**. Cílem metody je najít optimální index cirkulačního posunutí matice **B** tak, aby tóniny obou chromagramů (písní) byly stejné. Nejprve je z každé matice chromagramu vypočítán tzv. vektor globálního chromagramu **gA**, který je definovaný jako:

$$gA = \frac{\sum_{i=1}^{N} h_{A,i}}{\max\left(\sum_{i=1}^{N} h_{A,i}\right)}, \tag{5.11}$$

kde $h_{A,i}$ značí i-tý řádek matice chromagramu **A**. Výsledkem je normalizovaný vektor **gA**, jehož rozměr odpovídá počtu řádků matice chromagramu. Tento postup je aplikován i pro chromagram **B**. Pomocí globáních vektorů **gA** a **gB** můžeme určit optimální transpoziční index, který je definován podle vztahu:

$$\text{OTI}\ (gA, gB) = \underset{j}{\text{argmax}}\ (gA \cdot \text{Circshift}(gB, j)), \tag{5.12}$$

kde operátor $\cdot$ značí skalární součin vektorů. Pro chromagramy, které mají 12 řádků, volíme j od 1 do 12. Funkce Circshift() pak značí cirkulační posunutí matice nebo vektoru. V prostředí MATLAB je pro cirkulační posun implementovaná funkce *circshift()*. Výsledkem této metody je pak optimální index pro cirkulaci chromagramu **B**.

5.4.2 Metoda založená na hrubé síle

Robustnější a výpočetně náročnější metodou je metoda hrubé síly (*brute force*). Tato metoda nejprve spočítá skalární součiny mezi **gA** a mezi všemi možnými transpozicemi **gB**. Výsledkem je vektor **R**, který obsahuje M hodnot, které odpovídají podobnosti mezi chromagramem originální verze **A** a j-tou transpozicí chromagramu **B**. Tato část algoritmu je definována jako:

$$R_j(gA, gB) = gA \cdot \text{Circshift}(gB, j). \tag{5.13}$$

Nalezení optimálního transpozičního indexu je pak jednoduché. Pokud si zapamatujeme indexy všech transpozic, z nichž jsou vypočítány jednotlivé skalární součiny, tak ze seřazeného vektoru **R** můžeme vytvořit vektor transpozičních indexů. Tento

vektor pak nazveme $\mathbf{OTI}_R$, jehož hodnoty odpovídají transpozičním indexům seřazených od nejlepšího po nejhorší. Na první pozici vektoru $\mathbf{OTI}_{R(1)}$ pak bude stejná hodnota jako při výpočtu metody založené na globálním chromagramu.

V praxi se může stát, že nalezený optimální transpoziční index $\mathbf{OTI}_{R(1)}$ není správný. Tento fakt byl vyzkoumán například u paralelních (C dur - a moll) nebo modálních ($f^{\#}$ moll - h moll dórská) stupnic. Pro konečné určení optimálního indexu pomocí metody hrubé síly je třeba vypočítat Frobeinovy normy mezi maticemi $\mathbf{A}$ a $\mathbf{B}_{OTI_{R(j)}}$, kde $\mathbf{B}_{OTI_{R(j)}} = \text{Circshift}(\mathbf{B}, \mathbf{OTI}_{R(j)})$. Optimální index transpozice je pak definován jako:

$$\text{OTI}\ (A, B_{OTI_{R(j)}}) = \underset{j}{\text{argmin}}\ \|A - B_{OTI_{R(j)}}\|_F. \tag{5.14}$$

Optimální transpoziční index se většinou nachází na n prvních pozicích vektoru $\mathbf{B}_{OTI_{R(j)}}$. Proto není třeba počítat všechny Frobeinovy normy [27].

5.5 Matice křížové podobnosti

Jednou z možností, jak porovnat dvě skladby, je matice křížové podobnosti. V anglicky psané literatuře se používá název *Cross Similiarity Matrix* a zkratka *CSM*. Tyto matice většinou vznikají porovnáváním parametrů dvou skladeb. Mezi nejčastější parametry pro vznik matic křížových podobností patří chromagram nebo melovské kepstrální koeficienty. Před samotným výpočtem matice křížové podobnosti mohou být použity metody na sjednocení tóniny (*OTI*) nebo tempa (*Beat tracking, Dynamic Programing*) [27, 28]. Způsobů výpočtu matic křížových podobností je několik. Obecný přístup je takový, že matice křížové podobnosti porovnává každý segment daného parametru skladby $\mathbf{A}$ s každým segmentem daného parametru skladby $\mathbf{B}$. Způsob porovnání pak závisí na dané metrice. V praxi se často používá Euklidovská vzdálenost nebo skalární součin [29, 30]. Matici křížové podobnosti s využitím euklidovské metriky definujeme jako:

$$\text{CSM}(a, b) = \|(A(:, a) - B(:, b)\|_2, \tag{5.15}$$

kde a a b jsou indexy, které nabývají hodnot od prvního segmentu až po poslední. Syntaxe $A(:, a)$ (stejná jako v prostředí MATLAB) značí celý a-tý segment. U chromagramu, který má 12 řádků, to představuje sloupcový vektor obsahující 12 hodnot.

Výsledkem je pak matice křížové podobnosti, která má největší vypovídající hodnotu při grafickém zobrazení. Pokud jsou dvě skladby zcela totožné, tak je matice křížové podobnosti zrcadlená podle hlavní diagonály. Matice křížových podobností obsahují mnoho redundantních hodnot, protože porovnáváme například první segment z $\mathbf{A}$ s posledním segmentem z $\mathbf{B}$. Z těchto důvodů pak nefungují na matice křížových podobností metriky jako například normy, DTW atd.

Za zmínku stojí *SM Toolbox* od autorů [31] pro prostředí MATLAB. Tento toolbox je nádstavbou *Chroma Toolboxu* od stejných autorů a umožňuje extrahování matic křížových podobností a další zpracování jako například prahování (tresholding), vyhlazování atd. Na extrahovanou matici křížové podobnosti bývá většinou použita nějaká forma normalizace.

Na obrázku 5.2 je grafické zobrazení čtyř matic křížových podobností. Všechny matice vznikly porovnáváním chromagramů odpovídajících prvním 180 sekundám příslušné nahrávky. Chromagramy byly extrahovány pomocí MIRtoolboxu s oknem o délce 1 sekunda (počet vzorků jednoho okna odpovídá f_{vz}) s nulovým překryvem. Před výpočtem matic křížových podobností byla využita metoda sjednocení tóniny OTI. Matice křížových podobností byly vypočítány podle rovnice 5.15. Nakonec byla každá z matic normalizována podle vztahu [30]:

$$\mathrm{CSM} = \frac{\max(\mathrm{CSM}) \text{ - } \mathrm{CSM}}{\max(\mathrm{CSM})}, \tag{5.16}$$

kde funkce max(CSM) vrací největší hodnotu z celé matice CSM. Od této hodnoty je celá matice CSM odečtena a následně vydělena. Pro zobrazení pak byla otočena osa y tak, aby byl počátek souřadnic $[0, 0]$ v levém dolním rohu. Toto otočení pak změní směr hlavní diagonály matice, to znamená z horního levého rohu do pravého spodního rohu na orientaci ze spodního levého rohu do horního pravého rohu.

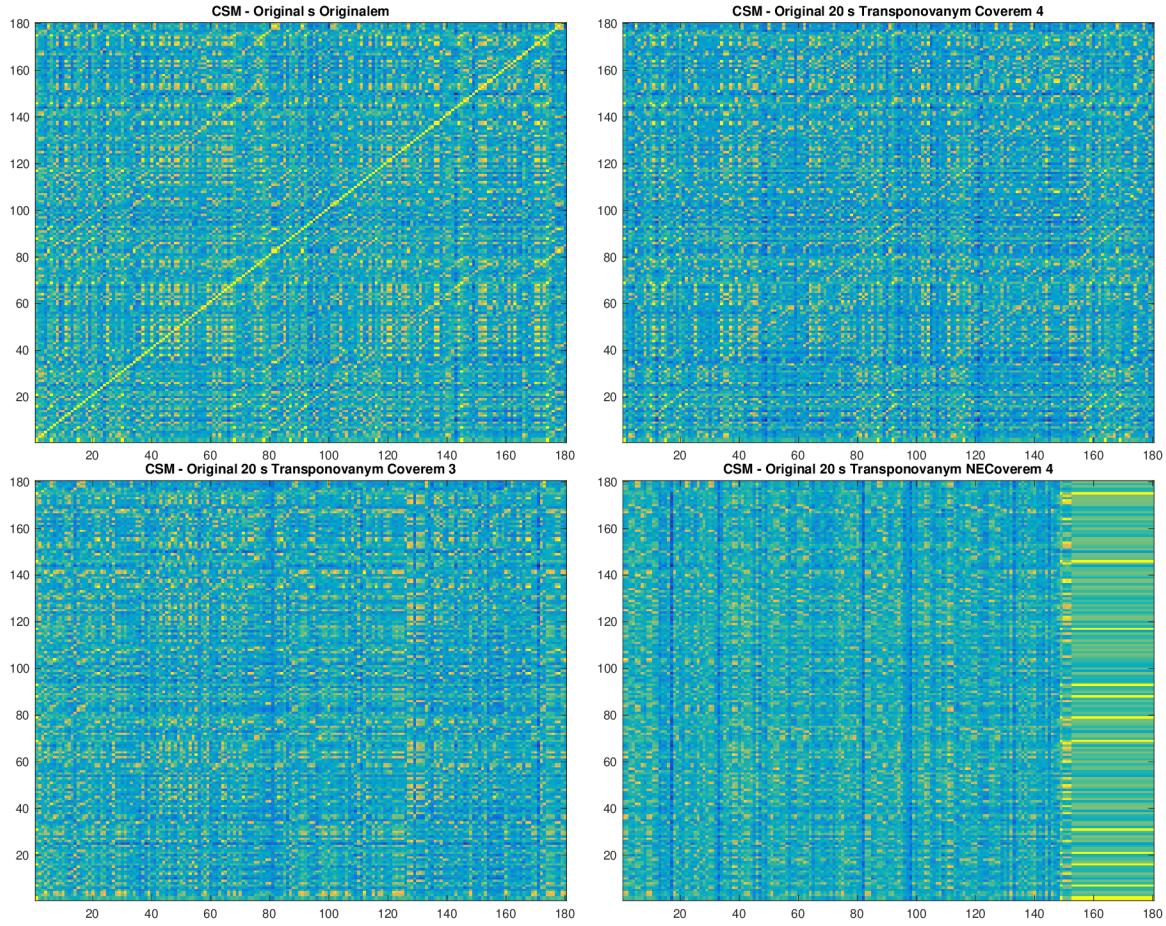
První matice (levá horní) na obrázku 5.2 byla vytvořena srovnáním dvou stejných originálních verzí písně od *Justina Biebera – Love Yourself*. Na této matici pozorujeme silnou hlavní diagonálu. Kolem diagonály si pak můžeme všimnout čtvercových bloků, které charakterizují jednotlivé bloky písně (předehra, sloka, refrén atd.).

Druhá matice (pravá horní) byla vytvořena srovnáním originální verze stejné písně jako v první matici a coververze[1], která je ve stejné tónině. Na této matici můžeme pozorovat diagonální pruhy.

Třetí matice (levá spodní) srovnává výše zmíněnou originální verzi s coververzí[2], která původně nebyla ve stejné tónině jako verze originální. Chromagram této coververze byl však transponován pomocí metody OTI. Na tomto obrázku můžeme stále pozorovat diagonální pruhy.

Čtvrtá matice (pravá spodní) vznikla srovnáním písně *Justina Biebera – Love Yourself* s coververzí, písně *Daft Punk – Get Lucky*[3]. Délka této coververze je 152 sekund. Zbylé segmenty chromagramu (153 - 180 sekund) pak byly nulové. Na čtvrté matici nepozorujeme žádné diagonální pruhy ani souměrné čtvercové bloky. Naopak zde můžeme pozorovat spíše horizontální a vertikální strukturu. Posledních 27

[1] https://www.youtube.com/watch?v=YnOd-bF5O2oab_channel=KysonFacer
[2] https://www.youtube.com/watch?v=NbB5F4U3eiYab_channel=LeroySanchez
[3] https://www.youtube.com/watch?v=E4mDz3a0aIoab_channel=MisterKanish



Obr. 5.2: Grafické zobrazení matic křížových podobností

sloupců této matice vykazuje ojedinělou strukturu. Tato struktura přímo souvisí s kratší coververzí a doplněním o nulové hodnoty.

6 Návrh vyhodnocovacího systému

Tato kapitola srovnává dosavadní metody zabývající se rozpoznáváním coververzí a identifikací skladeb. Jsou zde uvedeny jednotlivé publikace a jejich vzájemné porovnání úspěšnosti. Nakonec jsou navrhnuty dva různé systémy. Návrhy obsahují bloková schémata a popis jednotlivých bloků.

6.1 Dosavadní práce zabývající se rozpoznáváním coververzí

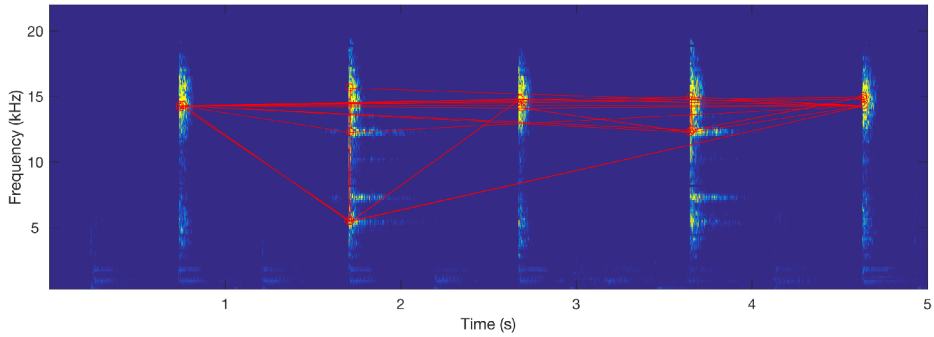
V reálném světě existuje několik systémů na detekci hudby. Jedním z nejznámějších je *Shazam*[1], tato aplikace funguje již od roku 1999. V roce 2018 byla společnost *Shazam Encore* koupena společností *Apple*. Tato aplikace funguje pomocí porovnávání tzv. zvukových otisků prstů hledané části skladby s databází skladeb [32, 33, 34]. Podobně jako u lidských otisků prstů má každá skladba svůj jedinečný otisk.

Zvukové otisky prstu musí mít vysokou specifičnost, většinou tedy stačí pár vteřin pro úspěšnou detekci skladby. Uvažujeme-li skutečnost, že zvukový otisk je extrahován z části skladby, která je například získávána pomocí chytrého telefonu v hlučném prostředí obchodního domu, tak tento systém nesmí být náchylný na vnější vlivy. Zvukový otisk by měl být tedy časově invariatní, protože bývá extrahován z krátkého fragmentu skladby (jen několik sekund). Všechny tyto podmínky splňuje koncept spektrálních vrcholů. Spektrální vrcholy jsou vypočítávány pomocí STFT (krátkodobá Fourierova transformace) a jsou založeny na vlastnostech spektrogramu. Systém, který využívá *Shazam*, detekuje nejsilnější vrcholy spektrogramu. Tyto vrcholy spojí přímkami a vytvoří tak síť připomínající pavučinu nad spektrogramem. Tato síť je mnohem řidší než spektrogram a je odolná vůči bílému šumu. Získávat otisky můžeme i jinými metodami, například společnost *Phillips* také definovala způsob získávání otisků, který je jiný, než jaký uvádí *Shazam*. Zvukové otisky nebyly definovány v kapitole 2, protože nejsou v MIRtoolboxu definované [34, 35].

Pomocí zvukových otisků prstů tedy můžeme velice dobře identifikovat danou skladbu, to ale není úkolem této práce. Primárně coververze nesdílí stejné tempo, barvu a tóninu. Z těchto důvodů jsou zvukové otisky pro rozpoznávání coververze nevyhovující [36]. Grafické zobrazení zvukového otisku prstu můžeme pozorovat na obrázku 6.1.

Od roku 2006 až do současnosti (2020) jsou vyhlašovány soutěže v rámci komunity MIREX. Jednou z disciplín je rozpoznávání coververze. Každý rok se soutěže

[1] https://www.shazam.com/



Obr. 6.1: Grafické zobrazení zvukového otisku prstů pro prvních 5 sekund skladby *Back To Black* od *Amy Winehouse* [35]

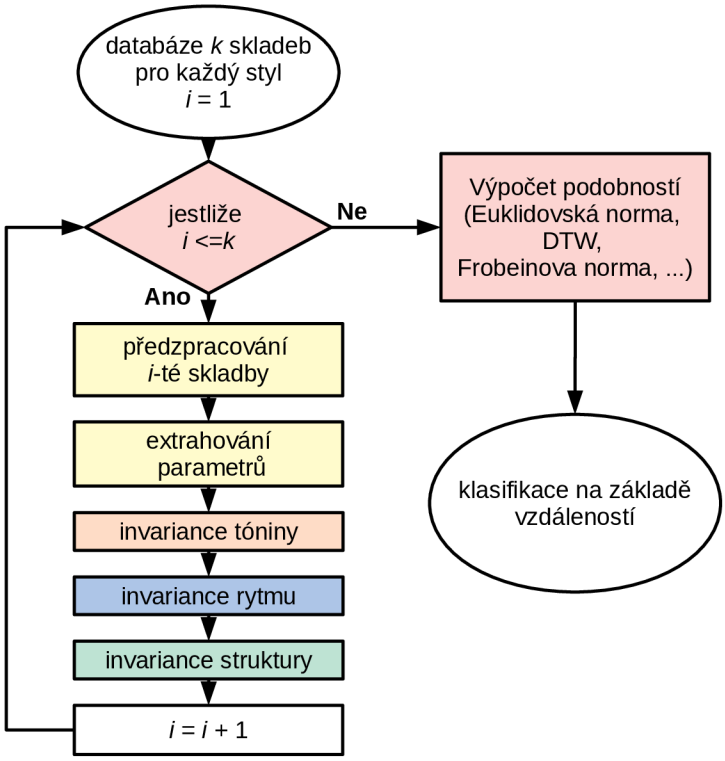
zúčastní několik soutěžících. Výsledky jsou pak publikovány na internetové stránce[2] pod záložkami *results by year*. V rámci soutěže bylo od roku 2006 publikováno mnoho publikací s rozdílnými přístupy, výsledky a datasety. Nejčastěji byly využity datasety: *Cover80*, *Mixed Collection*, *Sapp's Mazurka Collection*[3]. Současně v těchto letech vznikaly vědecké studie, vysokoškolské práce a články na téma rozpoznávání coververzí. Srovnávání výsledků jednotlivých publikací není zcela objektivní. Mnoho publikací se liší v použité evaluační metrice. Soutěž v rámci komunity MIREX sice ustanovila evaluační metriku, ne všechny datasety však byly vhodné pro tuto metriku. Některé publikace dosahují velice dobrých výsledků. Při důkladném zkoumání však zjistíme, že byl použit například dataset obsahující pouze různé přednesy skladeb z oblasti vážné hudby nebo dataset obsahoval MIDI přepis skladeb. U mnoha publikací obsahoval dataset málo vzorků. Tyto podmínky můžeme považovat za laboratorní. To znamená, že není zcela jasné, zda by systémy dosahovaly takových dobrých výsledků i v reálném provozu [28, 37, 38].

Počáteční publikace mezi lety 2006 - 2010 využívají většinou parametry založené na principu klasifikace výšky tónu (*PCP – Pitch Class Profile*) a zřídkakdy melovské kepstrální koefeicienty. *PCP* parametry jsou založené na spektrogramu a obsahují například všechny typy chromagramů, sledování hlavní melodie, akordů atd. Jako nejdůležitější se při identifikaci coververze uvažují hudební aspekty, které mezi sebou coververze sdílí. Mezi takový aspekt, který je nezávislý na hudebním stylu, je hlavní melodie písně.[4] *PCP* parametry jsou následně upravovány a zpracovávány různými metodami tak, aby byly **invariantní** (neměnné) vzhledem ke změnám tóniny, tempa a struktury písně. Tyto metody můžeme rozdělit na 3 kategorie:

[2] https://www.music-ir.org/mirex/wiki/MIREX_HOME

[3] http://www.mazurka.org.uk/

[4] I lidský mozek primárně rozezná coververzi podle hlavní melodie nebo textu (u zpívaných písní), méně důležité aspekty jsou pak pro mozek rytmus, barva atd.



Obr. 6.2: Návrh systému na rozpoznávání coververzí podle [17, 39]

1. **Metody invariance tóniny** – OTI, 2D výkonové spektrum, 2D autokorelace, metody odhadu tóniny, atd.
2. **Metody invariance tempa** – beat tracking, 2D výkonové spektrum, dynamické programování, komprese a expanze tempa atd.
3. **Metody invariance struktury písně** – dynamické programování, sekvenční okno, Smith-Watermanův algoritmus atd.

Po aplikaci výše zmíněných metod byly následně použity různé typy vzdálenostních metrik, jako například: DTW, křížová korelace, Frobeinova norma, euklidovská vzdálenost, skalární součin atd. Pomocí těchto metrik pak byly coververze rozpoznávány. Na obrázku 6.2 je znázorněn jednoduchý blokový diagram na rozpoznávání coververze za pomocí výše zmíněných technik a metrik [17, 39].

S vývojem počítačů a rozvojem umělé inteligence byly pro identifikaci coververzí v posledních letech použity modernější metody. Několik publikací pak používá křížové matice podobnosti a konvoluční neuronové sítě pro identifikaci coververze [40, 30, 41]. Rozpoznávání zvukové nahrávky se mění na rozpoznávání obrazu. Tento přístup identifikace coververzí dosahuje mnohem lepších výsledků a to i při použití velkých a rozmanitých datasetů. Jednou z hlavních výhod těchto systémů je binární

klasifikace. Více budou tyto systémy popsány v subkapitole 6.2.2.

Tabulka 6.1 porovnává jednotlivé publikace mezi sebou. Nezachycuje všechny dohledatelné publikace, ale pouze ty nejvýznamnější. Jako významnou publikaci považuji takovou, která uvádí evaluační metriku dosahující alespoň 40% úspěšnosti. Publikace jsou seřazeny podle roku vydání. Dále jsou zde zmíněny využité parametry a metody k získání invariantnosti, metriky a algoritmy strojového učení pro rozpoznávání parametrů coververzí. Následuje dataset v syntaktické podobě $o/c/a$, kde o znamená počet originálních verzí v datasetu, c je počet coververzí ke každé originální skladbě a a je počet všech skladeb datasetu. Z tabulky je zřejmé, že pokud $o + (c \cdot o) \neq a$, tak dataset obsahoval i písně, které nepatřily do žádné skupiny coververzí nebo originálních verzí. Poslední dva sloupce tabulky 6.1 uvádí úspěšnost jednotlivých publikací. Je zde vždy uvedena evaluační metrika. Ne všechny publikace uvádí všechny parametry a metody uváděné v tabulce.

Autor	param.	metoda sjednocení tóniny	metoda sjednocení tempa	metoda sjednocení struktury	vzdálenost / stroj. uč.	dataset	metrika úspěš.	MIREX MAP
Ellis, Daniel, Cotton (2007)[28]	PCP	všechny transpozice	beat	-	kříž. korelace	80/1/160	P@1 0,675	-
Gómez, Ong Herrera (2006)[37]	PCP	odhad tóniny	DP	RP	DTW	30/2/90	Fmeas 0,41	-
Kurth, Müller (2008)[38]	PCP	všechny transpozice	komp / exp	sekv. okno	skalární součin	-/-/1167	R@1 0,97	-
Traile, Bendich (2015)[42]	MFCC	-	beat	-	SWLA KNN	80/1/160	42/80 52,5%	-
Heo, Kim, Kim, Lee (2017)[43]	PCP CENS	OTI	-	-	SiMPle	30/11/ 1000	-	0,81
Chang, Lee, Choe, Lee (2017)[30]	PCP	OTI	-	-	CSM CNN	30/11/ 1000	-	0,84
Lee, Chang, Choe, Lee (2018)[41]	PCP	OTI	-	-	CSM CNN	1175/-/ 1000	-	0,93
Silva, Zhu, Yeh, Batista (2018)[44]	PCP	OTI	-	-	SiMPle	50/7/350/ 49/<41,95>/ 2919	P@10 0,14 0,95	0,59 0,88
Yu, Xiaoou, Xu, Yang (2019)[45]	CQT	-	-	-	kosinova metr. CNN	80/1/160 8858/-/>10^5	P@10 0,09 0,47	0,84 0,66

Tab. 6.1: Srovnání postupů a výsledků vybraných publikací [17, 39]

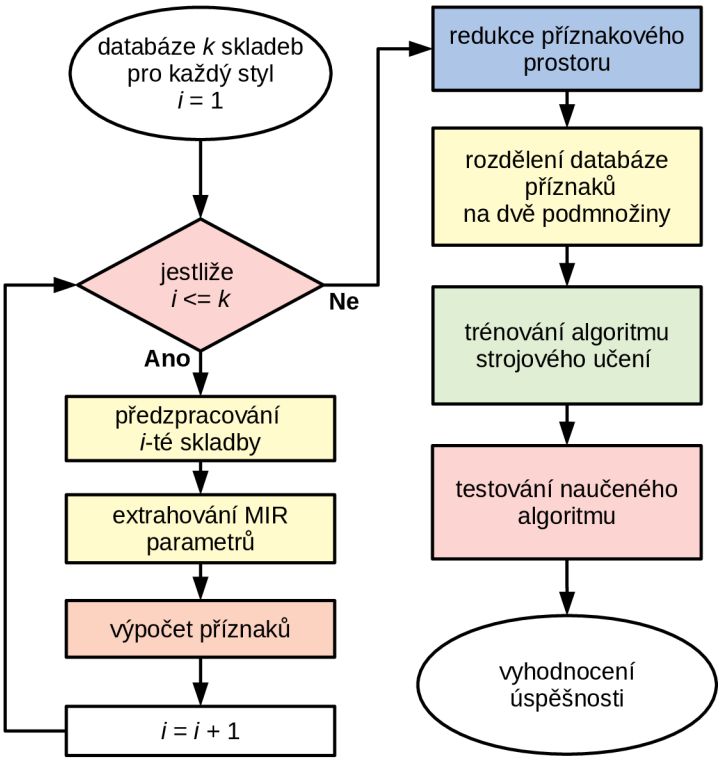
6.2 Návrh vlastních systémů

6.2.1 Systém založený na statistických parametrech

Na obrázku 6.3 je zobrazen návrh systému založený na všech parametrech MIRtoolboxu, který bude využívat tato práce. Nejdříve budou jednotlivé skladby postupně předzpracovány (podle 5.1) a následně budou extrahovány všechny parametry, které MIRtoolbox poskytuje. Z těchto parametrů budou pomocí statistických funkcí vypočítávány vektory příznaků. Každá skladba bude charakterizována vysoce redundantním vektorem příznaků. Tyto příznaky budou následně selektovány a redukovány pomocí algoritmu *mRMR*. Redukce příznaků bude vždy vypočítána pro každý hudební styl databáze. Z tohoto důvodu se mohou lišit selektované příznaky pro každý styl. Následně bude databáze rozdělena na tři podmnožiny. Celkem je k dispozici šest skladeb pro každou klasifikační třídu. Trénovací podmožina bude obsahovat verzi originální a první tři coververze. Jako validační podmnožina bude použita čtvrtá coververze. K testování bude sloužit poslední coververze. Každý z hudebních stylů obsahuje dvacet klasifikačních tříd. Na těchto datech bude natrénován algoritmus strojového učení a následně bude otestován. Nakonec bude vyhodnocena úspěšnost.

6.2.2 Systém založený na rozpoznávání podobnostních matic

Tento typ systému je inspirovaný publikacemi [30, 40, 41]. Jedná se o systém, který nejdříve vybere prvních 180 sekund ze zvukového signálu. Tím sjednotí délku všech vstupujících signálů a velikost všech podobnostních matic. Následně budou ze všech signálů extrahovány chromagramy. Chromagramy budou extrahovány pomocí MIRtoolboxu s délkou okna 1 sekunda bez překryvu. U skladeb kratších než 180 sekund budou zbylé vzorky doplněny nulovými hodnotami. Z extrahovaných chromagramů budou vypočítány matice křížových podobností podle stejného postupu, který byl uveden v subkapitole 5.15 (OTI transpozice a normalizace). Tyto matice budou rozděleny na trénovací a validační. Následně budou předány konvoluční neuronové síti *CovNet-1*. V porovnání s předchozími systémy dochází v této fázi ke dvěma zásadním změnám. Z klasifikace do n tříd, kdy $n > 2$ a $n =$ *počet originálních verzí v datasetu*, se stává klasifikace binární (coververze a ne-coververze). Tento způsob klasifikace nezkoumá, ke které originální písni patří dotazovaná coververze. Řeší pouze, jak moc je pravděpodobné, že dotazovaná matice podobnosti zobrazuje coververzi. Druhá změna nastává v tom, že z klasifikace audio signálu (vektoru příznaků) se stává klasifikace obrazu (matice). Na obrázku 6.4 je jednoduché blokové schéma tohoto systému.



Obr. 6.3: Návrh systému založeného na statistických parametrech

Na obrázku 6.5 je uvedena topologie konvoluční neuronové sítě *CovNet-1*, která byla využita v několika předchozích publikacích [30, 40, 41]. Obrázek popisuje jednotlivé bloky této sítě. Například *Conv2D* $(32 \times 5 \times 5)$, *ReLu* znamená 2D konvoluce s 32 jádry o velikosti 5×5. Výstup konvoluce je následně předán jako vstup funkce ReLu. Názvy jednotlivých bloků a parametry odpovídají syntaxi objektů knihovny *Keras* pro prostředí Python, ve kterém bude tato síť implementována. V pravé části obrázku jsou pak uvedeny rozměry, které odpovídají výstupním rozměrům obrazu po průchodu daným blokem. Síť *CovNet-1* obsahuje celkem $0,58 \times 10^6$ parametrů, které jsou během procesu učení aktualizovány - učeny. Na výstupu sítě jsou dva neurony s výstupní funkcí SoftMax. Funkce SoftMax vrací hodnotu v intervalu $< 0, 1 >$. V procesu trénování je očekávaný výstup [1,0] pro obrazy, které charakterizují coververze a [0,1] pro obrazy, které charakterizují ne-coververze. Principy a základní bloky konvolučních neuronových sítí budou vysvětleny v subkapitole 7.3.4.

Po natrénování sítě je poslední fází testování a vyhodnocení úspěšnosti. Výstup sítě si můžeme představit jako $[p, 1-p]$, kde p je pravděpodobnost, že vstupní obraz charakterizuje coververzi. Síť tedy není schopna rozeznat, se kterou skladbou testovací databáze byla dotazovaná podobnostní matice vytvořena. Pro klasifikaci dotazované podobnostní matice coververzí byl zaveden výstupní systém vyhodnocování.

Pro otestování bude využito N skladeb. Pokud provedeme výpočet křížových matic podobností pro každý pár z testovací množiny, tak počet testovacích matic vzroste na N^2. Předložíme-li tyto podobnostní matice natrénované síti, pak bude výsledkem matice pravděpobností $\mathbf{P}$(i,j) o velikosti N^2. Každý řádek matice $\mathbf{P}$ odpovídá vektoru pravděpodobnosti mezi i-tou a všemi j-tými skladbami, kde $j \in < 1, N >$. Jednotlivé řádky matice $\mathbf{P}$ odpovídají vztahovému vektoru $R_i = [p_{i1}, p_{i,2}, ..., p_{i,N}]$. Vzdálenost mezi dvěma vztahovými vektory R_1 a R_2 je definována různými metrikami. Nejjednodušší je seřadit sestupně jednotlivé pravděpodobnosti (vektory R_i). Pravděpodobnosti hledaných skladeb se pak nachází na prvních pozicích seřazených vektorů.

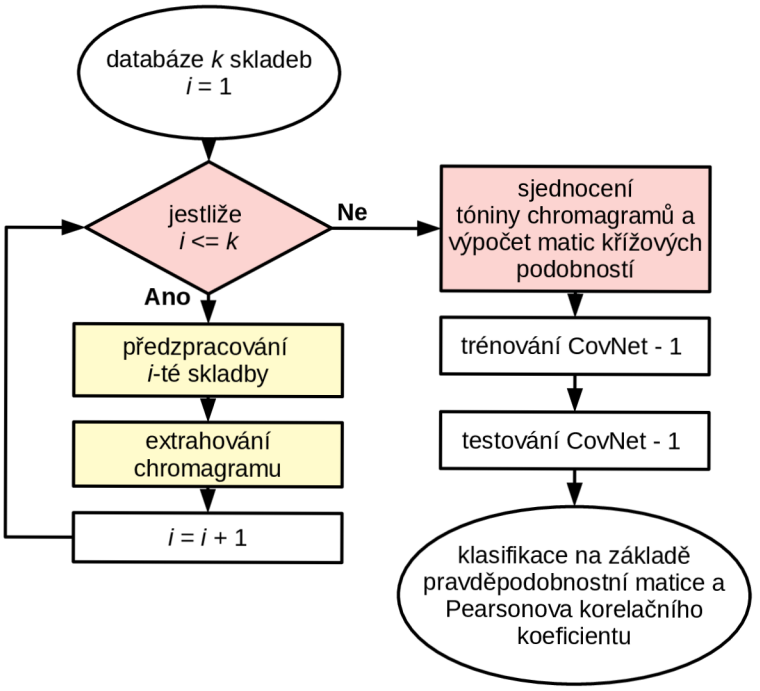
Podle [41] byl většinou nejúspěšnější výpočet vzdálenosti mezi dvěma vztahovými vektory vypočítán na základě **Pearsonova korelačního koeficientu**, který definujeme pro dva vztahové vektory R_1 a R_2 jako:

$$\mathrm{dist}(R_1, R_2) = 1 - \frac{(R_1 - \overline{R}_1) \cdot (R_2 - \overline{R}_2)}{\|R_1 - \overline{R}_1\|_2 \|R_2 - \overline{R}_2\|_2}, \tag{6.1}$$

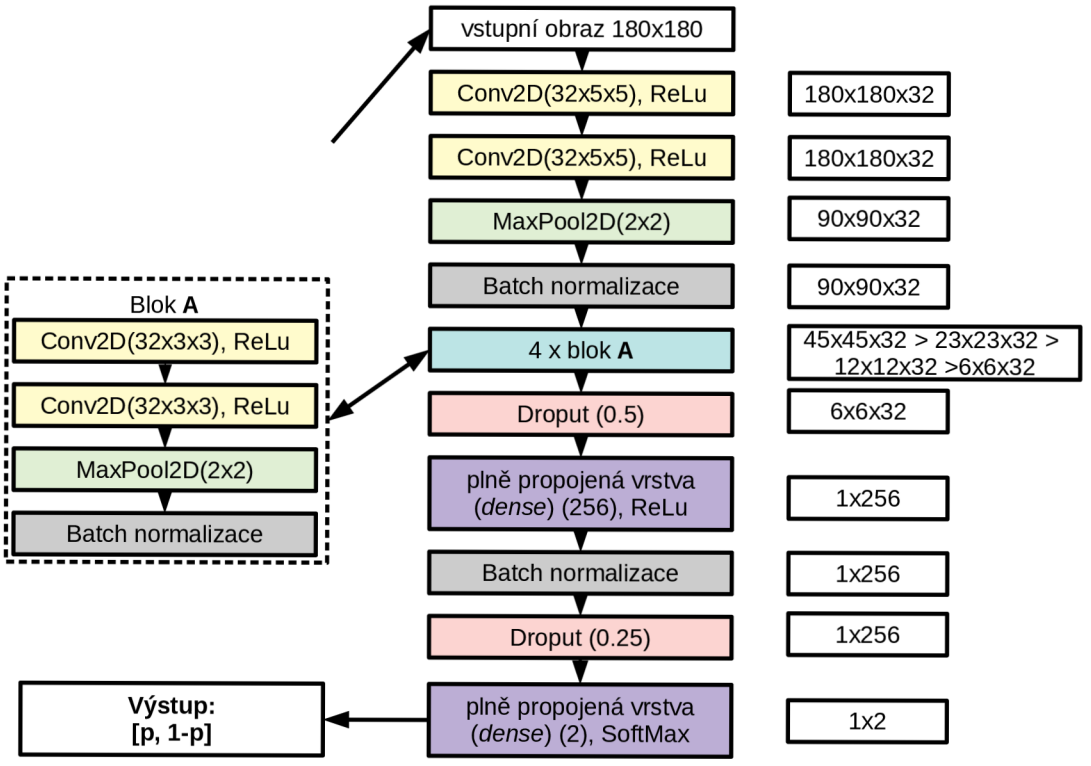
kde $\overline{R}_1$ značí průměrnou hodnotu vektoru R_1 a $\cdot$ značí skalární součin. Pokud pro všechny R_i provedeme výpočet vzdálenosti se všemi R_i (i se sebou samým), tak coververze odhadneme vzestupným seřazením těchto vzdáleností [30, 40, 41].

$$\mathrm{Rank}_i = \mathrm{argsort}(\mathrm{dist}(R_i, R_1), ..., \mathrm{dist}(R_i, R_N)) \tag{6.2}$$

Kosinova vzdálenost je další použitou metrikou pro výpočet vzdálenosti mezi dvěma vztahovými vektory. Tato vzdálenost byla definována v subkapitole 2.5.2. Abychom mohli pro konečné vyhodnocení využít rovnici 6.2, tak při vypočítané vzdálenosti podle rovnice 2.8 je nutné vypočítanou vzdálenost odečíst od čísla 1 (stejně jako v případě Pearsonova korelačního koeficientu v rovnici 6.1).



Obr. 6.4: Návrh systému založeného na rozpoznávání podobnostních matic [30, 40, 41]



Obr. 6.5: Topologie konvoluční neuronové sítě *CovNet-1* [30, 40, 41]

7 Strojové učení

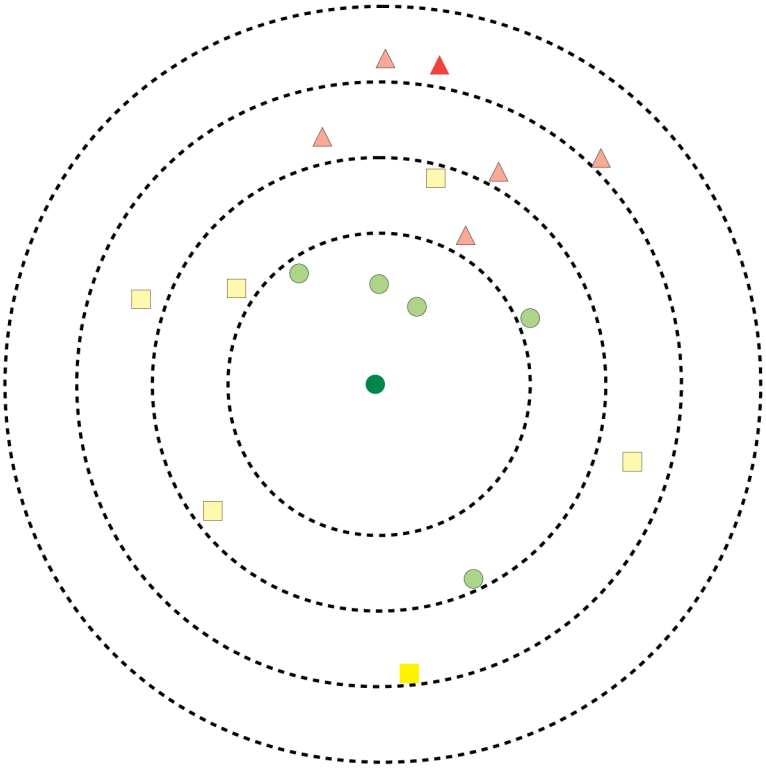
Strojové učení (v angličtině nazýváno jako *machine learning*) je podoblast umělé inteligence. Strojovým učením rozumíme schopnost počítače se „učit" z dat, přijímat rozhodnutí s minimálním zásahem člověka. Učením v daném kontextu rozumíme takovou změnu vnitřního stavu systému, která zefektivní schopnost přizpůsobení se změnám okolního prostředí. Základem strojového učení je práce s vektory a maticemi. S vývojem počítačů, rostoucím objemem a rozmanitostí dat se stává strojové učení čím dál více atraktivní. Prvotně bylo popsáno již v 50. letech [46]. Jedná se tedy o vědní obor, který není nový, ale získal novou dynamiku. Využití strojového učení dnes nalezneme v mnoha odvětvích, jako jsou například: systémy pro doporučování (elektronické obchody, sociální sítě, hudební přehrávače...), systémy v biomedicínské informatice, dopočet již pořízené fotografie (mobilní telefony) [47]. Rozlišujeme tři základní typy:

1. **učení bez učitele** – učící systém má k dispozici pouze vstupní data
2. **učení s učitelem** – učící systém má k dispozici vstupní data a výstupy, které od něj očekáváme
3. **zpětnovazebné učení** – systém má k dispozici vstupní data a během výpočtu dostává odezvu určující úspěšnost, podle odezvy mění metody výpočtu

Jedním ze základních problémů, který lze řešit pomocí strojového učení, je klasifikace. Klasifikací rozumíme přiřazování označení digitálním objektům, že patří do nějaké skupiny. Druhým problémem je shlukování (cluster). Shlukováním rozumíme porovnávání digitálních objektů na základě podobnosti, kdy nás nezajímá obsah těchto digitálních médií [48]. Úloha hledání coververze zahrnuje oba problémy. V následujících subkapitolách budou popsány jednotlivé algoritmy strojového učení.

7.1 K-nejbližších sousedů

Jedním ze základních algoritmů je algoritmus s názvem *k-nejbližších sousedů*. Tento algoritmus poskytuje vynikající výkon při použití dobrých měřících metod vzdáleností. V testovací fázi bychom pak uložili množinu vektorů charakterizujících parametry jednotlivých nahrávek. Tuto množinu dat nazýváme jako slovník. Při testování získáme parametry testovací nahrávky a porovnáváme ji s parametry nahrávek uložených ve slovníku. Pro porovnání se nejčastěji používá euklidovská vzdálenost 2.5.1. Jako originální verzi nebo coververzi pak označujeme tu, která má k hledanému vektoru parametrů nejkratší vzdálenost nebo k nejbližších vzdáleností (pokud hledáme právě k nahrávek). Na obrázku 7.1 je jednoduché zobrazení *k-nejbližších*



Obr. 7.1: Jednoduché zobrazení algoritmu *k-nejbližších sousedů*

sousedů, kde každý ze symbolů symbolizuje pět coververzí a jednu verzi originální (zelený kruh uprostřed). Přerušované kružnice pak značí oblasti stejných vzdáleností od středu (od testovací nahrávky).

7.2 *K*-mean

Jedním z algoritmů, který je vhodný pro shlukování, je algoritmus k-mean. Tento algoritmus třídí data do k shluků na základě jejich vlastností. Vstupem algoritmu je množina dat $X = \{x_1, x_2, ..., x_n\}$. Číslo k nám udává počet vektorů μ_j (shluků). Počet shluků k musí být menší než počet objektů vstupní množiny dat. Algoritmus je iterativní a pracuje ve dvou hlavních krocích. V inicializační fázi jsou vytvořeny vektory μ_j (okolo nich se budou shlukovat vstupní data), kde $j = 1, ..., k$. Hodnoty vektorů μ_j jsou zvoleny náhodně nebo pokud známe problematiku úlohy, tak volíme hodnoty podle intuice. V úloze rozpoznávání coververze nás budou zajímat shluky vektorů v okolí originální verze. Tyto shluky pak můžeme považovat za coververze. Dvěma hlavními kroky jsou:

1. **klasifikace** – pro všechna vstupní data x_i je vypočítána vzdálenost (nejčastěji

euklidovská) k μ_j a podle vypočítané vzdálenosti jsou přiřazeny do příslušného nejbližšího shluku.

$$y_i = \underset{j=1,\ldots,k}{\operatorname{argmax}} \|x_i - \mu_j\| \tag{7.1}$$

2. **učení** – pomocí aritmetických průměrů všech bodů daného shluku se nově vypočítají hodnoty vektoru μ_j. Takto jsou znova vypočítány všechny vektory μ_j z každého shluku.

$$\mu_j = \frac{1}{n_j} \sum_{i \in \{i:y_i=j\}} x_i, \tag{7.2}$$

kde n_j je počet vzorků x_i, které spadají do příslušného μ_j, jak bylo zjištěno v prvním kroku algoritmu.

Kroky 1 a 2 jsou opakovány, dokud se alespoň jeden z vektorů x_i nepřidá do jiného shluku μ_j, než byl klasifikován v kroku 1 [49, 50, 51].

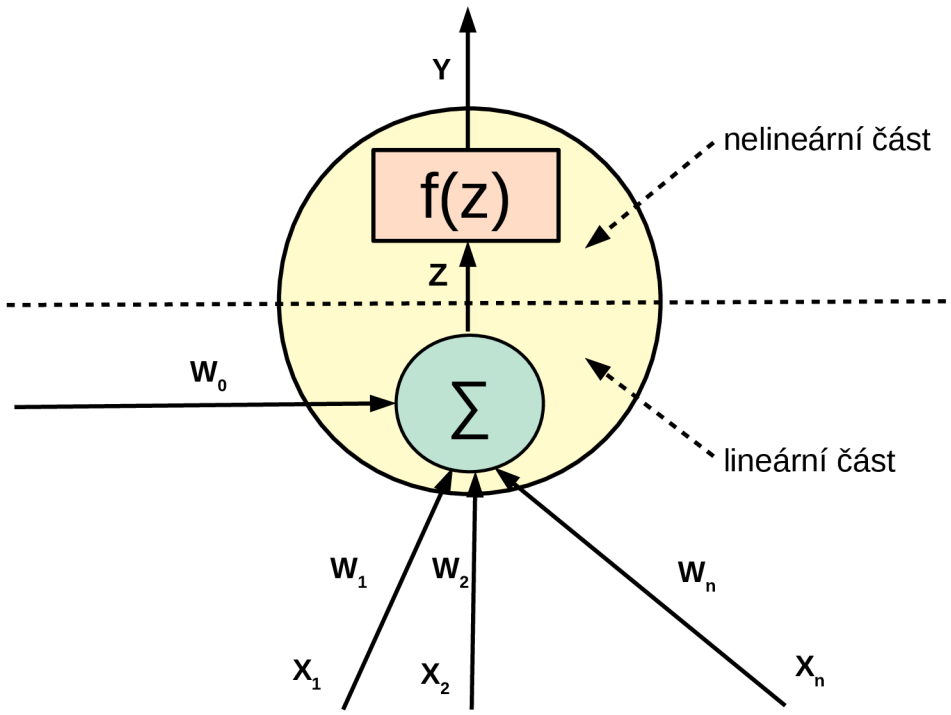
7.3 Umělé neuronové sítě

Nejmodernější a stále se vyvíjející strojové učení jsou umělé neuronové sítě. Obecná umělá neuronová síť je systém složený z umělých neuronů. Umělým neuronem rozumíme výpočetní jednotku, která má několik vstupů x_n, ale jen jeden výstup y. Každý vstup má odpovídající váhu w_n. Váhu můžeme chápat jako sílu propojení mezi neurony. Váha w_0 je označována jako práh neuronu (bias). Obrázek 7.2 popisuje model umělého neuronu. Lineární část neuronu řeší sumaci součinů vstupů neuronu s příslušnou váhou vstupu a následný součet s prahem neuronu. Lineární část popisujeme jako:

$$z = w_0 + \sum_{i=1}^{n} x_i \cdot w_i. \tag{7.3}$$

Do nelineární části vstupuje výstup části lineární, tzn. číslo z. Podle příslušné funkce je pak vypočítán výstup y celého neuronu. Tato funkce může být nelineární a díky nelineární části jsou neuronové sítě schopny modelovat nelineární křivky.

Neuronová síť je rozdělená do několika vrstev. Vnější vrstva je pak označována jako výstupní. Naopak nejvnitřnější vrstva je označována jako vstupní. Neuronové sítě, které mají více jak dvě skryté vrstvy, nazýváme jako hluboké neuronové sítě. Učením neuronové sítě rozumíme odladění váhovacích koeficientů. Existuje mnoho typů neuronových sítí. Jednotlivé typy neuronových sítí se od sebe liší v topologii (uspořádání a propojení neuronů), způsobu učení (algoritmus měnící váhy), přenosové funkci neuronů a ve využití dané neuronové sítě v praxi. Výpočty neuronových



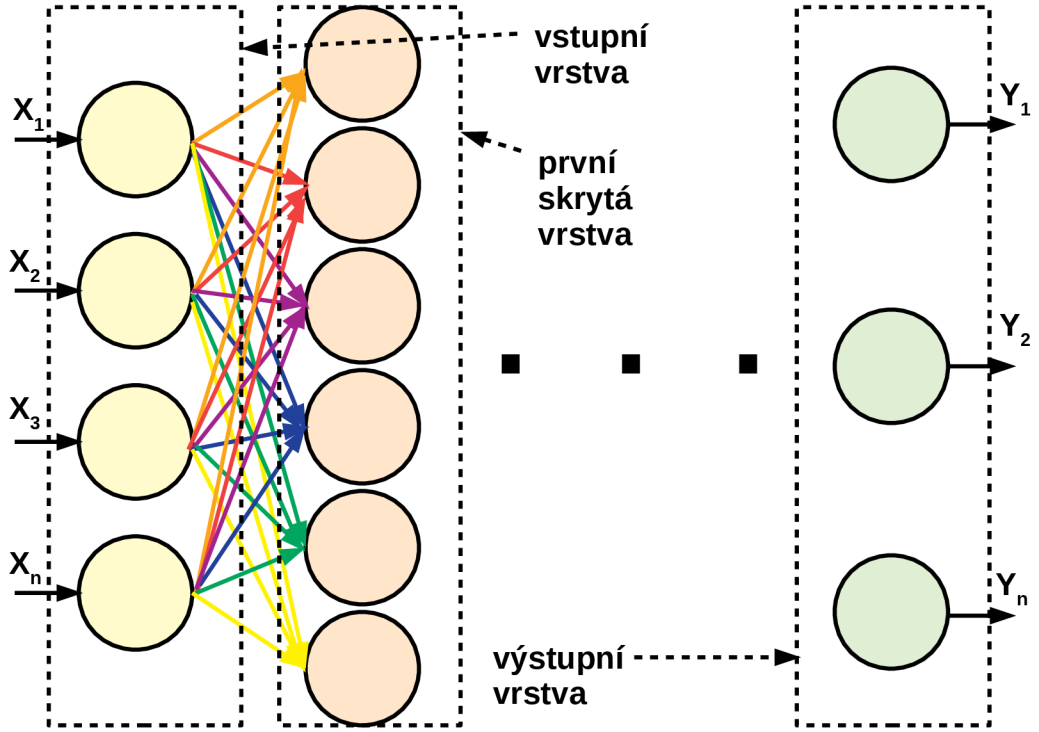
Obr. 7.2: Model umělého neuronu [52, 53]

sítí jsou náročné a s rostoucím počtem neuronů roste exponenciálně požadovaný výpočetní výkon. Vyšší vrstvy neuronů musí čekat na výsledek nižších vrstev. Z toho důvodu jsou výpočty většinou prováděny na vysoce výkonných grafických kartách (GPU), které jsou schopny zpracovávat informace paralelně. Výhoda neuronové sítě je například odolnost proti šumu [52, 53].

7.3.1 Vícevrstvá neuronová síť se zpětnou propagací chyby

Jedná se o učení s učitelem. Tato síť je vhodná pro spoustu klasifikačních úloh. Obsahuje tři a více vrstev (vstupní, skryté, výstupní), mezi jednotlivými vrstvami je úplné propojení (každý neuron nižší vrstvy je propojen se všemi neurony vyšší vrstvy). Na obrázku 7.3 je znázorněna topologie vícevrstvé neuronové sítě.

Vstupní vrstva slouží pouze k distribuci signálu. Počet neuronů ve vstupní vrstvě je stejný jako počet příznaků daného vzoru. Počet výstupních neuronů je stejný jako počet tříd, do kterých chceme daný vzor klasifikovat. Počet neuronů ve skrytých vrstvách a počet skrytých vrstev je volitelný. Čím více neuronů skrytá vrstva obsahuje, tím více je schopna neuronová síť modelovat nelineární křivky. S rostoucím množstvím neuronů roste výpočetní náročnost. Zatím neexistuje žádné pravidlo, které by



Obr. 7.3: Topologie vícevrstvé neuronové sítě se zpětnou propagací chyby

bylo schopno určit optimální počet neuronů. Existují jistá doporučení nebo algoritmy, které přidávají nebo ubírají neurony v závislosti na výstupní chybě sítě.

Přenosová funkce neuronů ve vstupní vrstvě bývá lineární ($f(z) = z$). Přenosové funkce neuronů ve skrytých vrstvách jsou nelineární. Nejčastěji to bývá sigmoida 7.7 s výstupy od 0 do 1, hyperbolická tangenta 7.5 s výstupy od -1 do +1, ReLu s výstupy $< 0, \infty >$ 7.6 (rectified linear unit) a případné obměny jako Leaky ReLu.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{7.4}$$

$$f(z) = \frac{1 - e^{-z}}{1 + e^{-z}} \tag{7.5}$$

$$f(z) = \begin{cases} z & \text{pro } z > 0 \\ 0 & \text{pro } z \leq 0 \end{cases} \tag{7.6}$$

Pro výstupní vrstvu se pak hojně využívá funkce SoftMax, která normalizuje výstupní vektor (K je počet výstupních neuronů), tato funkce je definována jako:

$$f(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_k}}. \tag{7.7}$$

Funkce musí být spojitá, diferencovatelná a monotónně neklesající. Inicializace vah bývá náhodná. Algoritmus učení je iterativní gradientní, který minimalizuje čtverce chybové funkce **E**. Chyba je počítána postupně mezi jednotlivými vrstvami. Výpočet probíhá od výstupní vrstvy až po vstupní vrstvu. Chybou se rozumí rozdíl aktuálního a požadovaného výstupu sítě nebo vrstvy. Výstupní chyba sítě ${}^{s}\mathbf{E}$ při klasifikaci s-tého vzoru je vypočítávána jako:

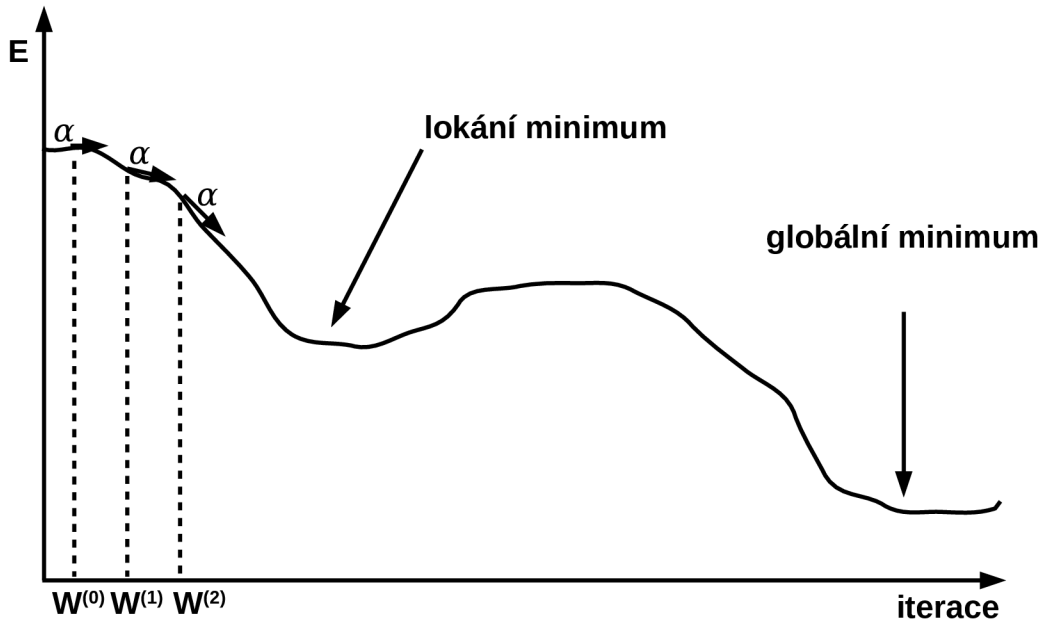
$$ {}^{s}E = \frac{1}{2}\sum_{i=1}^{N}({}^{s}y_i - {}^{s}\,d_i)^2, \tag{7.8} $$

kde N je počet neuronů (počet klasifikačních tříd) ve výstupní vrstvě a s značí s-tý trénovací vzor, ${}^{s}y_i$ značí reálný výstup sítě i-tého neuronu při předložení s-tého trénovacího vzoru, ${}^{s}d_i$ je očekávaný výstup (učení s učitelem). Po předložení celé trénovací množiny vypočítáváme celkovou chybu sítě jedné iterace. Tato chyba je vypočítána jako součet výstupních chyb sítě po každém vzoru podle:

$$ E = \sum_{s=1}^{P} {}^{s}E. \tag{7.9} $$

Podobně jako výstupní chybu sítě počítáme postupně od výstupních vrstev po vstupní vrstvy chybu mezi jednotlivými vrstvami. Jako proces učení označujeme aktualizaci vah, která má za cíl zmenšení chyb. Celková chyba záleží na nastavení vah mezi všemi vrstvami sítě. Z tohoto důvodu je tento proces složitý a ve vícedimenzionálním prostoru téměř nepředstavitelný. K minimalizaci chyby je využívána gradientní metoda, která vyžaduje diferencovatelnost chybové funkce. Pro lepší pochopení je průběh chybové funkce E znázorněn na obrázku 7.4. V bodě $W^{(0)}$ je neuronová síť ve stavu inicializace, kdy jsou váhy nastaveny jako malá náhodná čísla okolo nuly. K adaptaci vah slouží tečný vektor, který je vypočítán jako: $-\alpha \cdot \frac{\partial \mathbf{E}}{\partial \mathbf{W}}(W^{(0)})$, tento tečný vektor se nazývá gradient. Posunutím se ve směru gradientu o koeficient α získáme novou konfiguraci $W^{(1)} = W^{(0)} + \Delta W^{(1)}$. Tento postup opakujeme, až se limitně dostaneme do minima.

Nevýhodou gradientní metody je uvíznutí v lokálním minimu. To je situace, kdy se postup adaptace zastaví, protože gradient je nulový. Z těchto důvodů je velmi důležité, jaký bude zvolený koeficient α. Tento koeficient je nazýván jako učící koeficient. Správnou volbou učícího koeficientu dokážeme překonat lokální minimum a dostat se do minima globálního. Nastavení a nalezení vhodného koeficientu učení se liší pro různé konfigurace sítí, tréninkové množiny atd. To znamená, že nelze dopředu určit vhodnou velikost. Ideální nastavení tohoto koeficientu je takové, kdy chybová funkce klesá v co nejmenším počtu iterací a dosahuje globálního minima. Při špatném nastavení může dojít k rozkmitání chybové funkce, uvíznutí v lokálním minimu nebo nalezení globálního minima za cenu mnoha iterací. Z těchto důvodů existuje algoritmus s proměnným koeficientem učení. [53, 54].



Obr. 7.4: Průběh chybové funkce a gradientní metoda [53]

7.3.2 Rekurentní neuronová síť

Topologie rekurentních (zpětnovazebních) neuronových sítí (RNN) je stejná jako klasických neuronových sítí (ANN), ale v každém kroku si ukládá předchozí výstup. Obecně vzato je topologie se zpětnou vazbou vhodnější na zpracovávání sekvenčních vstupů. Jedná se o aplikace, kde je aktuální výstup závislý na předchozích výstupech. Jako sekvenční vstup označujeme například text, hudbu atd. Mezi zástupce zpětnovazebních neuronových sítí patří například *Hoepfieldova* síť.

7.3.3 LSTM neuronová síť

Dlouhodobě krátkodobá neuronová síť (*Long Short-Term memory*) je speciální typ zpětnovazebních neuronových sítí, které byly vyvinuty v roce 1997. Topologie této sítě je zcela odlišná od předchozích sítí. Základní stavební jednotkou této sítě je tzv. buňka. Celá síť se skládá z několika buněk, každá buňka má svůj vnitřní stav. Uvnitř buňky je několik bran (gate). Základní sít má tři brány: vstupní, výstupní a zapomínající. Tento mechanismus bran umožní síti zahazovat data, která již nejsou potřebná. Jakmile jsou nadbytečná data zapomenuta, tak jsou přidána nová. Výsledkem je možnost sítě uchovávat a reagovat na velice staré výstupy. Síť je tedy schopná

se naučit dlouhodobým závislostem. LSTM sítě našly uplatnění v predikci časových řad, rozpoznávání řeči a ručně psaných textů, skládání hudby, psaní scénářů atd. [55, 56]. Popis LSTM sítě je složitý a není náplní této práce.

7.3.4 Konvoluční neuronová síť

Konvoluční neuronové sítě se nejčastěji používají v oblasti počítačového vidění. Nejvíce je popularizoval Yann LeCun v 90. letech, který je nyní ředitelem centra pro umělou inteligenci společnosti Facebook. Obecně vzato se konvoluční neuronové sítě hodí na typ dat, které mají hodnoty uspořádány v mřížce, to je například pixelová reprezentace obrazu v počítači. Tyto sítě jsou schopné se naučit, co je obsahem obrazu a tím ho klasifikovat. Pro většinu neuronových sítí je vstupním objektem vektor příznaků. U konvolučních neuronových sítí je vstupním objektem obraz. Pro zpracování obrazu se používá konvoluční jádro, které je nazýváno jako konvoluční filtr. Pokud bychom neuronové síti předali celý obraz např. $50 \times 50 \times 3$ (šířka $\times$ výška $\times$ tři vrstvy – RGB), tak by každý neuron musel mít při plném propojení $50 \times 50 \times 3$ (7500) vah, aby zpracoval tento obraz. Při použití konvolučního filtru o velikosti $5 \times 5 \times 3$ (3 = hloubka jádra) nám stačí pouze 75 vah. Konvoluční filtr nám umožní redukci dat. Tuto operaci popisujeme matematicky jako 2-D konvoluci, která je definována jako:

$$g(x,y) = f(x,y) * h(x,y) = \sum_{i=\frac{-S}{2}}^{\frac{S}{2}} \sum_{j=\frac{-R}{2}}^{\frac{R}{2}} f(x-i, y-j) \cdot h(i,j), \tag{7.10}$$

kde $g(x,y)$ značí výstupní obraz, $f(x,y)$ vstupní obraz a $h(x,y)$ konvoluční filtr o rozměrech $R \times S$. Pro lepší pochopení je vhodné vysvětlit 2-D konvoluci graficky. Na obrázku 7.5 je znázorněna 2-D konvoluce na binárním obrazu s rozměry 4×4. Jedná se tedy o prostorové posouvání konvolučního filtru nad obrázkem a stanovení odezvy. Pro každou vzájemnou polohu obrazu a filtru je vypočítán součet hodnot pixelů obrazu vážených příslušnými koeficienty filtru, přičemž tento součet určuje výstupní hodnotu obrazu v daném bodě. Výstupní obraz se nazývá aktivační mapa. Konvoluční filtr se nemusí posouvat v jednom kroku a volba velikosti tohoto filtru souvisí s množstvím parametrů konvoluční neuronové sítě. Filtr by měl být vždy menší než vstupní obraz. Velikost výstupního obrazu se vypočítá podle vztahu 7.11, kde rozměry vstupního obrazu $\mathbf{N}$, rozměry filtru $\mathbf{F}$ a krok posuvu filtru odpovídají příkladu na obrázku 7.5.

$$\frac{\mathbf{N} - \mathbf{F}}{krok} + 1 = \frac{4-3}{1} + 1 = 1 + 1 = 2 \tag{7.11}$$

Pokud bude $krok > 1$, nemusí filtr obsáhnout všechny body obrazu. Z tohoto důvodu se zavádí tzv. *zero-padding*. To znamená, že se celý obraz rozšíří o daný počet

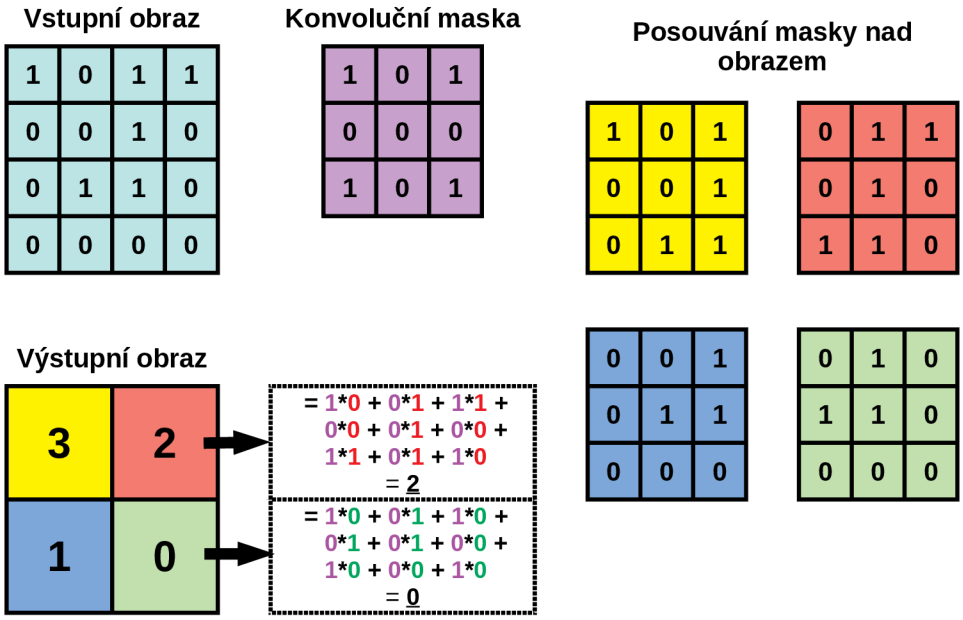
nulových pixelů. Při použití rozšíření obrazu o nulové hodnoty je vztah 7.11 upraven jako:

$$\frac{\mathbf{N} - \mathbf{F} + 2\mathbf{P}}{krok} + 1, \tag{7.12}$$

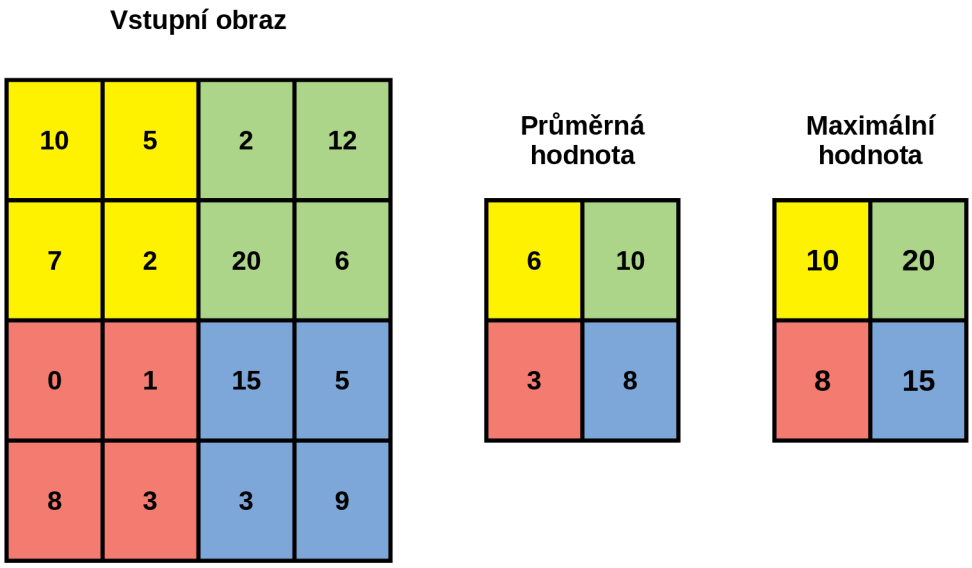
kde **P** značí počet nulových vektorů, o které byl obraz rozšířen z jedné strany obrazu. Aby mohla být síť nelineární, tak v rámci aktivační mapy je zahrnuta aktivační funkce. Existuje několik funkcí, které je možno využít, v praxi se však často využívá funkce ReLU, která je definována jako $f(x) = \max(0, x)$.

Další často používanou technikou je sdružování tzv. *pooling*. Sdružování má za úkol razantně zmenšit velikost vstupního obrazu a tím redukovat množství parametrů. K tomu slouží sdružovací maska a funkce, která je pro sdružování použita. Sdružování nám poskytuje invariantnost v tom, že nezáleží na jakém místě se hledaný objekt v obrazu nachází. Na obrázku 7.6 je graficky znázorněno sdružování s funkcí průměrování a výběru maximální hodnoty. Výsledkem 2-D konvoluce a sdružování je na výstupu sítě vektor, který je následován řadou plně propojených vrstev, které se starají o samotnou klasifikaci. Obecně platí, že čím více konvolučních kroků máme, tím více složité funkce bude konvoluční síť schopna modelovat.

Učení konvoluční sítě spočívá v minimalizaci vnitřních stavů sítě. Stejně jako v algoritmu se zpětnou propagací chyby je srovnáván výstup konvoluční neuronové sítě s očekávaným výstupem. Následuje zpětná propagace chyby a následná aktualizace vah, konvolučních filtrů a všech ostatních parametrů [57, 58, 59, 60].



Obr. 7.5: Jednoduché grafické znázonění 2-D konvoluce binárního obrazu



Obr. 7.6: Jednoduché grafické znázonění techniky sdružování

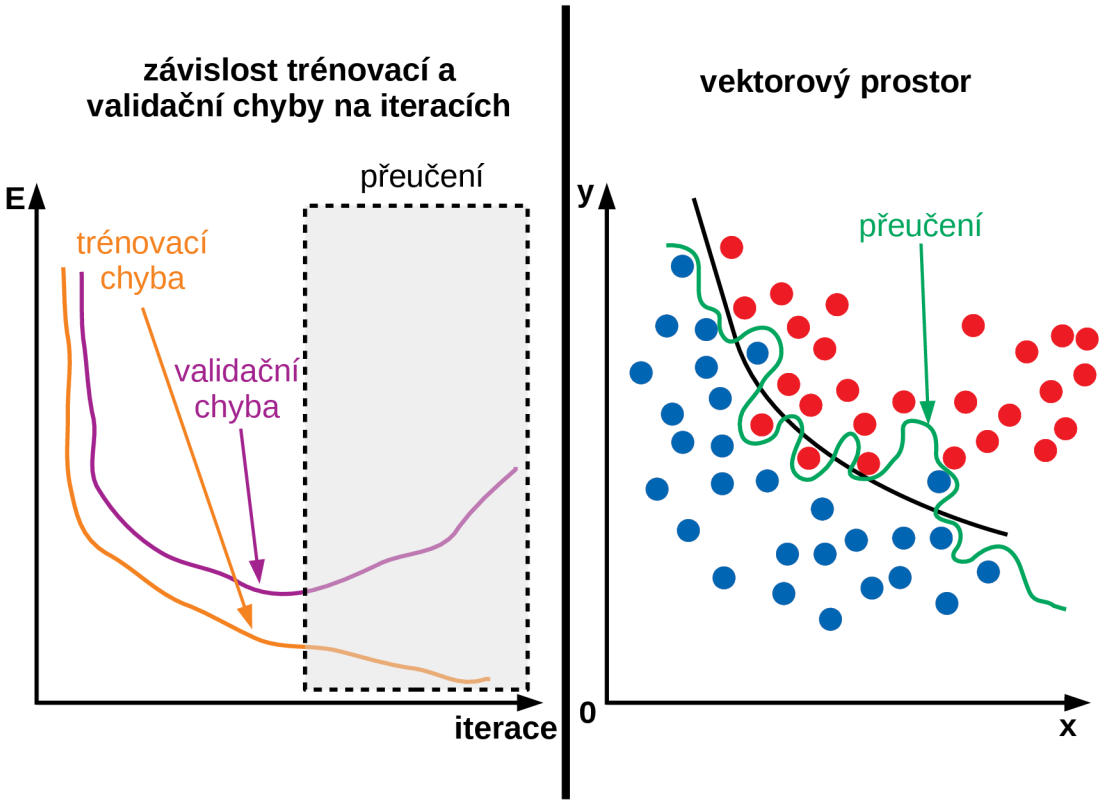
7.4 Doporučení spojená s učením algoritmů strojového učení

V praxi se vyskytuje několik překážek, které zabraňují úspěšnému naučení nebo testování algoritmů strojového učení. Část těchto překážek souvisí jen s neuronovými sítěmi, část pak můžeme vztáhnout na celou množinu algoritmů strojového učení. V celém procesu rozpoznávání je velké množství parametrů (tvorba a redukce příznakového prostoru, strojové učení), které ovlivňují celkový výsledek klasifikace. Jedná se o optimalizační problém, který lze řešit za pomocí optimalizačních algoritmů nebo při menším množství parametrů manuálně. Řada doporučení, která zvyšují úspěšnost algoritmů strojového učení, byla objevena heuristicky a osvědčila se v mnoha aplikacích. Většinou nejsou matematicky podložená. Většina uvedených technik byla publikována na přednáškách předmětu umělá inteligence.

7.4.1 Přeučení

Přeučení se nazývá v anglické literatuře *overfitting*. Jako přeučení označujeme takový stav, kdy výstupní chyba klasifikace trénovacích dat konverguje k určitému malému číslu a zároveň roste chyba klasifikace validačních dat. Jinými slovy se algoritmus naučí perfektně klasifikovat trénovací data, avšak data, která nebyla nikdy předložena v procesu učení (validační), klasifikuje s chybou, která by byla menší, kdyby byla větší chyba trénování. Z tohoto důvodu je třeba využít naučený model s nejmenší validační chybou. Jako výstup klasifikačních algoritmů si můžeme představit křivku (ve vícedimenzionálním prostoru rovinu atd.), která je schopna rozdělit prostor na daný počet klasifikačních tříd tak, aby křivka oddělovala jednotlivé třídy (trénovací data z jednotlivých tříd). Jaká je optimální křivka pro rozdělení vstupních dat? Na tuto otázku nelze odpovědět předem. Důležité je, aby byl systém schopný generalizovat. Pro lepší pochopení bylo vytvořeno grafické zobrazení. Na obrázku 7.7 je grafické zobrazení přeučení. Levá část obrázku znázorňuje závislost trénovací a validační chyby na iteracích. Pravá část obrázku zobrazuje trénovací data ve vektorovém prostoru, kde zelená křivka znázorňuje přeučení. Tato křivka je schopna dokonale klasifikovat trénovací data, ale není schopna generalizovat.

Existuje několik možností, jak předcházet přeučení. Přeučení často vzniká právě tehdy, když máme k dispozici malé množství trénovacích vzorů. Pokud není možné získat další trénovací vzory, je možné si tyto vzory vyrobit uměle. Techniky pro rozšíření trénovacích vzorů se nazývají augmentace. V praxi to znamená, že například ke stávajícím vzorům přidáváme šum, zkreslení atd.



Obr. 7.7: Grafické zobrazení přeučení

7.4.2 Zamíchání trénovacích vzorů

V trénovací množině by měly být zastoupeny všechny klasifikační třídy rovnoměrně. V praxi bylo zjištěno, že promíchání vzorů před každým učícím cyklem zmenšuje šanci na přeučení.

7.4.3 Aktualizování vah

V procesu učení neuronových sítí dochází k aktualizaci váhových koeficientů, které vedou ke zmenšení výstupní chyby. V praxi je několik možností, kdy mohou být váhy aktualizovány vzhledem k předloženým vzorům trénovací množiny. Parametr, který ovlivňuje tyto možnosti, se u většiny sítí nazývá anglicky *Batch*. Tento parametr můžeme překládat jako *dávkování*. Všechny možnosti v závislosti na dávkování můžeme rozdělit do tří hlavních skupin:

1. **Batch Gradient Descent** – k aktualizaci vah dochází až po předložení celé trénovací množiny (*Batch = počet trénovacích vzorů*)
2. **Stochastic Gradient Descent** – k aktualizaci vah dochází po předložení jednoho trénovacího vzoru. V jednom cyklu, kdy byly předloženy všechny tré-

novací vzory, došlo právě k tolika aktualizacím jako je počet vzorů trénovací množiny ($Batch = 1$)

3. **Mini-Batch Gradient Descent** – aktualizace vah je závislá na zvoleném parametru ($Batch$), kdy ($1 < Batch <$ *počet trénovacích vzorů* $= 1$)

Pokud počet trénovacích vzorů není dělitelný parametrem *Batch*, tak poslední skupina trénovacích vzorů je menší než *Batch*.

7.4.4 Prořezávání

Jednou z technik, která byla objevena heuristicky, je prořezávání, v anglické literatuře se tato technika nazývá *dropout*. Prořezávání spočívá v tom, že v procesu učení odstraní náhodně zvolené neurony a jejich spojení. Parametrem pro prořezávání je většinou poměr, kolik neuronů z celkového počtu neuronů se nemá podílet v dané iteraci učení. Tato technika zlepšuje schopnost generalizace sítě a zabraňuje přeučení. Při testovací fázi se dropout nepoužívá.

7.4.5 Batch normalizace

Z anglického jazyka je *batch normalization* těžce přeložitelné (dávková normalizace). Proto tuto normalizaci budeme nazývat v následujícím odstavci jen jako normalizace. Tuto techniku publikovali v roce 2015 zaměstnanci firmy *Google* Sergey Ioffe a Christian Szegedy [61].

Distribuce každé vstupní vrstvy se během trénování mění, protože se mění parametry předchozích vrstev. To zpomaluje učení a vyžaduje menší koeficient učení α. Tento jev se nazývá jako kovariantní posun [61]. Pokud při trénování aktualizujeme váhy po určitém množství vzorů, tak dochází k tomu, že některé vstupy více ovlivňují neuronovou síť. Z těchto důvodů je třeba tyto vstupy normalizovat. V některých případech normalizace eliminuje potřebu prořezávání. Normalizace také zmenšuje validační a testovací chybu [61].

7.5 Evaluace systému

Evaluací systémů rozumíme vyhodnocení úspěšnosti systémů. Pro vyhodnocování je dobré znát dopředu dataset, na kterém je systém testován. Na úkol rozpoznávání hudebních coververzí můžeme nahlížet několika způsoby. Jedním z nich je, kdy hledáme k originální verzi skladby co nejvíce coververzí, které obsahuje daný dataset. Druhým způsobem může být hledání originální verze ke coververzi. Další způsob je binární klasifikace, to znamená, jestli dvojice skladeb je nebo není coververzí. Ať už

nahlížíme na úkol rozpoznávání coververzí z kteréhokoliv úhlu pohledu, tak vždy může dojít k několika situacím. Pokud známe dataset, předpovídáme, jestli se hledaná verze v datasetu nachází nebo ne.

1. Označená skladba je skutečně hledanou skladbou (pravdivá pozitivní – **TP**).
2. Neoznačená skladba není hledanou skladbou (pravdivá negativní – **TN**).
3. Označená skladba není hledanou skladbou (nepravdivá pozitivní – **FP**).
4. Neoznačená skladba je hledanou skladbou(nepravdivá negativní –**FN**).

Z výčtu situací pak můžeme určit úspěšnost vyhodnocování (pouze první dvě situace jsou správné řešení hledání coververze). Celkovou prediktivní schopnost pak značíme jako **A** – přesnost – *Accuracy* a vypočítáme ji jako poměr mezi úspěšným určením a všemi možnými určeními. Pro určení v procentech pak vynásobíme **A** ·100 %. Přesnot systému je definována jako:

$$A = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7.13}$$

Tato evaluační metrika není vhodná pro datasety, ve kterých nejsou všechny klasifikační třídy zastoupeny rovnoměrně. V takovém případě by mohla být dosažena vysoká úspěšnost klasifikací do majoritní třídy. Z tohoto důvodu jsou zavedeny jiné evaluační metriky.

Další evaluační metrikou je preciznost – **P** – *Precision*. Preciznost nám říká, jaký podíl pozitivních identifikací bylo ve skutečnosti správných. Jinými slovy nám preciznost říká, jak moc můžeme věřit, že systém odhalil pozitivní výsledek. Preciznost je vypočtena jako poměr pravdivých pozitivních a součtu pravdivých pozitivních s nepravdivými pozitivními. Pro určení v procentech pak vynásobíme **P** ·100 %. Preciznost systému je definována jako:

$$P = \frac{TP}{TP + FP}. \tag{7.14}$$

Poslední evaluační metrikou je senzitivita – **R** – *Recall*. Senzitivita udává, jaký je podíl skutečných pozitiv, které systém správně odhalil. Znamená to tedy, jaký podíl všech *„ano"* odhalíme. Senzitivita je vypočtena jako poměr pravdivých pozitivních a součtu pravdivých pozitivních s nepravdivými negativními. Pro určení v procentech pak vynásobíme **R** ·100 %. Senzitivita systému je definována jako [39, 62, 63]:

$$R = \frac{TP}{TP + FN}. \tag{7.15}$$

7.5.1 Vyhodnocovací metriky soutěže MIREX

Většina stávajících systémů na rozpoznávání coververzí využívá metriky uvedené v soutěži MIREX[1] . Tyto metriky jsou definovány již od roku 2006 [64]. Základní metrikou je výpočet průměrné preciznosti – **AP** – *average precision*. Toto evaluační vyhodnocení bylo pro nejlepších 10 výsledků. Testovací databáze tedy měla alespoň 10 coververzí pro jeden dotaz. Průměrná preciznost je definována jako [39]:

$$AP = \frac{\sum_{i=1}^{n}(P(k) \times rel(k))}{C}, \tag{7.16}$$

kde n je počet získaných výsledků, k je pořadí prvku v seznamu výsledků, C je celkový počet coververzí použitých pro detekci. Funkce *rel(k)* vrací hodnotu 1, pokud je k-tá položka seznamu coververze. Pokud se nejedná o coververzi, tak návratová hodnota funkce *rel(k)* je 0. Průměrná preciznost je pak používána pro určení střední průměrné preciznosti – **MAP** – *mean average precision*. Střední průměrnou preciznost systému vypočítáme jako:

$$MAP = \frac{\sum_{i=1}^{N} AP(i)}{N}, \tag{7.17}$$

kde i je pořadí dotazu a N je celkový počet provedených dotazů [39, 64].

Další zavedenou metrikou soutěže MIREX je metrika **MNIT10** – *Mean number of covers identified in top 10*. Tato metrika udává průměrný počet správně identifikovaných skladeb pro prvních 10 predikcí. Z toho vyplývá, že testovací databáze měla alespoň 10 coververzí pro jeden dotaz. Pokud naše databáze obsahuje n coververzí, tak zavádíme **MNIT**n. Čím více se hodnota **MNIT**n blíží číslu n, tím více byl systém schopný rozpoznat coververze (*čím větší, tím lepší*).

Poslední často využívanou metrikou je **MR1** – *Mean rank of first correctly identified cover*. Výhodou **MR1** není závislost na počtu coververzí k dotazované skladbě testovací databáze. Tato metrika udává průměrnou hodnotu první predikce, která správně identifikovala coververzi. Pokud je hodnota **MR1** = 1, tak byl náš systém schopný identifikovat coververzi již v první predikci (*čím menší, tím lepší*).

[1] https://www.music-ir.org/mirex/wiki/2019:Audio_Cover_Song_Identification

8 Praktická část a vlastní řešení

V této kapitole budou uvedeny všechny postupy a úvahy při implementaci použitých algoritmů. Budou zde otestovány a vyhodnoceny oba navrhnuté systémy na rozpoznávání coververzí.

8.1 Vyhodnocení systému založeného na statistických parametrech

Tento systém využívá všechny běžné parametry, které je MIRtoolbox schopen extrahovat z audio signálu. Blokové schéma tohoto systému popisuje obrázek 6.3. Kromě rozpoznávání coververzí je hlavní přínos tohoto systému v hledání relevantních parametrů pro rozpoznávání coververze pro každý styl databáze. Dataset obsahuje 10 různých stylů. V následujících subkapitolách bude popsána praktická implementace v prostředí MATLAB. Všechny potřebné funkce a skripty obsahuje složka přílohy s názvem `První Metoda`.

8.1.1 Výpočet všech parametrů pomocí MIRtoolboxu

Nejprve byly vytvořeny v programu MATLAB dvě funkce, které postupně načtou a předzpracují jednotlivé zvukové vzorky daného stylu databáze. První funkce s názvem `nacitani.m` postupně načítá z dané složky v počítači jednotlivé skladby zvoleného stylu databáze. Vstupní proměnné této funkce jsou: řetězec s názvem složky stylu databáze, vzorkovací frekvence a počet sekund. Pro správné spuštění této funkce je nutné uložit cestu k celé databázi skladeb do proměnné *myFolder*. Skladby jsou postupně načítány a předávány funkci s názvem `predzprac.m`. Tato funkce postupně předzpracuje každou skladbu. Volitelnými parametry jsou změna vzorkovací frekvence a výběr prvních n sekund dané skladby. Oba tyto parametry byly předány funkci `predzprac.m` z funkce `nacitani.m`.

Po předzpracování každé skladby dochází k samotnému extrahování všech parametrů. Pro extrahování všech základních parametrů MIRtoolboxu byla vytvořena funkce s názvem `extrahovani_parametru.m`. Tato funkce je volána funkcí `predzprac.m`. Vstupní proměnnou je pouze název předzpracované skladby, která je po dobu výpočtu parametrů dočasně uložena ve formátu .wav a nachází se ve složce daného žánru databáze. Funkce MIRtoolboxu jsou volány tak, aby extrahované parametry měly co největší vypovídající hodnotu. Tím je myšleno, že například před výpočtem parametrů jako je chromagram nebo MFCC je skladba nejdřív segmentována. Segmentace je volána jako vstupní argument příslušné funkce MIRtoolboxu pomocí příkazu *'Frame'*. Délka segmentu a překryv je pak nastaven s vý-

chozí hodnotou dané funkce MIRtoolboxu. Tím mají jednotlivé parametry rozdílné rozměry (skaláry, vektory, matice). Z tohoto důvodu je návratovou hodnotou této funkce struktura (*struct*), která je schopna pojmout rozdílné datové typy. Nakonec jsou do této struktury funkcí `predzprac.m` přidány textové řetězce obsahující název skladby a cestu k souboru skladby. Tato struktura je návratovou hodnotou funkce `predzprac.m`. Struktury jednotlivých skladeb jsou ve funkci `nacitani.m` uloženy do nadřazené struktury, která je výstupní proměnnou funkce `nacitani.m`. Tato nadřazená struktura obsahuje všechny vypočítané parametry všech skladeb daného stylu databáze. Výše zmíněné funkce jsou součástí přílohy a jsou příslušně okomentovány.

Bylo zjištěno, že některé parametry neobsahují jen číselné hodnoty, ale i hodnoty *NaN – Not a Number*. Hodnota *NaN* vzniká například při dělení nulou. Tyto hodnoty se vyskytovaly ojediněle. Pokud byla použita segmentace, tak se zřídkakdy nacházely na konci nebo na začátku matic a vektorů parametrů. Občas se tyto hodnoty nacházely i tam, kde nebyla segmentace použita. Jedná se nejspíš o problém MIRtoolboxu (verze 1.7.2). V prostředí MATLAB jsou hodnoty *NaN* problematické v tom, že při další matematické operaci je výsledek opět hodnota *NaN* (např. $1 + NaN = NaN$). Tyto hodnoty se vyskytovaly ojediněle, proto byla vytvořena funkce `NaNfilter.m`, která postupně prochází každý vypočítaný parametr. Pokud se v parametru vyskytuje hodnota *NaN*, tak ji nahradí hodnotou 0. Tato úprava pak může zkreslovat další zpracování parametrů (např. výsledky statistických parametrů). Bylo však zjištěno, že výskyt hodnot *NaN* je ojedinělý a zkreslení způsobené nahrazením nulovými hodnotami je pak irelevantní.

Extrahování všech parametrů pro jeden zvolený styl (120 skladeb) bylo provedeno na počítači MAC Pro s procesorem Intel Xeon X5690 (12 jader; 3,46GHz) a operační pamětí 64GB. Časová náročnost tohoto výpočtu závisí na zvolené vzorkovací frekvenci skladeb a volbě časového úseku skladeb. Pokud byla zachována výchozí vzorkovací frekvence (44 100Hz) a délka písní nebyla nějak zkracována, tak výpočet trval 10 až 12 hodin. Při zmenšování vzorkovacího kmitočtu a sjednocení délky všech skladeb klesal i čas výpočtu.

8.1.2 Výpočet statistických parametrů

V předchozí subkapitole je popsáno, jak byly vypočítány všechny běžné parametry, které MIRtoolbox obsahoval. Následně byly tyto parametry ošetřeny tak, aby mohly být dále zpracovávány. Výstupem vypočítaných parametrů je struktura parametrů. Každý řádek této struktury charakterizuje jednu skladbu. Celkem má tato struktura 120 řádků - skladeb. V jednotlivých sloupcích se pak nachází vypočítané parametry. Cílem zpracování parametrů je výpočet vektoru příznaků pomocí statistických metod. Pro výpočet vektoru příznaků byly využity všechny statistické metody de-

finované v kapitole 5.2. Při využití všech definovaných statistických metod budou příznakové vektory vysoce redundantní. Redundance bude následně redukována.

Vypočítané parametry jsou matice, vektory a skaláry. Skalární hodnoty budou tvořit jednu hodnotu příznakového vektoru. Z vektorových parametrů budou použity všechny statistické metody. U matic je několik možností, jak tyto statistické metody použít. Jednou z možností je vytvořit z matice vektor pomocí řádkového rozkladu a následný výpočet statistické metody. Druhou možností, která byla použita, je pak použití statistických metod na jednotlivé řádky matice. To znamená, z matice chromagramu, která má rozměr 12 × počet segmentů, bude pro každý z dvanácti řádků vypočítán průměr, medián, rozptyl atd.

Pro vypočítání vektorů příznaků ze struktury vypočítaných parametrů byla vytvořena funkce s názvem `parametr_na_skalar.m`. Vstupní proměnnou této funkce je struktura parametrů, která byla vytvořena pomocí funkce `nacitani.m`. Funkce `parametr_na_skalar.m` postupně vypočítá všechny statistické parametry. Funkce byla naprogramována pro parametry extrahované ze skladeb s vzorkovací frekvencí 44 100Hz (při menší f_{vz} mají některé parametry menší rozměry). Vektor příznaků pro každou skladbu pak obsahuje přesně 824 hodnot. Návratovou hodnotou této funkce je buňka (*cell*) obsahující v prvním sloupci název příslušného statistického parametru. První dva řádky této buňky obsahují název příslušné skladby a absolutní cestu ke skladbě. Vektory příznaků k dané skladbě jsou pak obsaženy v jednotlivých sloupcích této buňky. Tyto vypočítané buňky byly převedeny a uloženy do formátu .csv. S tímto formátem je schopna pracovat většina programovacích jazyků. Zobrazení a editaci formátu .csv umožňují i tabulkové editory, jako jsou například *Microsoft Excel*, *LibreOffice Calc* atd. Uložení příznakových vektorů ve formátu .csv je oproti nativnímu formátu prostředí MATLAB .mat výhodnější. Hlavní výhoda spočívá v možnosti zobrazit parametry bez nutnosti prostředí MATLAB. Vypočítané vektory příznaků pro všechny styly databáze jsou součástí přílohy (ve formátu .csv i .mat).

8.1.3 Redukce příznakového prostoru metodou mRMR

Vypočítané vektory příznaků jsou vysoce redundantní. Tento fakt je způsoben například i kvůli tomu, že jsou vypočítány téměř všechny definované statistické parametry. Pro redukci příznakového prostoru byla použita metoda mRMR. Metoda mRMR (*fscmrmr*) je součástí prostředí MATLAB od verze R2019b. Tato práce využívá MATLAB verze R2016a. Z tohoto důvodu byla použita již předprogramovaná metoda mRMR od autorů [65], která byla vytvořena již v roce 2007 a používá soubory typu .mex. To jsou soubory, které zprostředkovávají rozhraní mezi prostředím MATLAB a programovacím jazykem C nebo C++. Tyto soubory musí

být nejdříve zkompilovány pro příslušný operační systém. Pro zkompilování souborů .mex slouží funkce `makeosmex.m`, která je součástí balíčku s předprogramovanou metodou mRMR. Pro zkompilování souborů .mex na počítačích MAC je vyžadována příslušná verze vývojářského software *Xcode*. Verze *Xcode* musí být podporována použitou verzí MATLAB. Pro spuštění mRMR na novějších verzích prostředí MATLAB je třeba upravit řádky 56 a 65 funkcí `estjointentropy.cpp` a `estcondentropy.cpp`. Úprava kódu spočívá ve změně příkazů $log(2)$ na $log(2.0)$. Poslední úpravou je pak změna importovaných knihoven ve funkci `estjointentropy.cpp` z *#include <math.h>* na *#include <cmath>*. Tyto informace nejsou důležité pro řešení této práce, ale usnadní spuštění předprogramované funkce mRMR v prostředí MATLAB pro budoucí uživatele.

Před samotnou redukcí příznakového prostoru pomocí metody mRMR jsou parametry předzpracovány v několika krocích. Tyto metody jsou volány funkcí `mRMR.m`, která zajišťuje předzpracování. Součástí této funkce je volání funkcí pro redukci příznakového prostoru `mrmr_mid_d.m` a `mrmr_miq_d.m`. Tyto funkce pak zajišťují redukci příznakového prostoru. Předzpracování vektoru příznaků spočívá ve třech krocích. První krok spočívá v možnosti normalizace jednotlivých příznakových vektorů mezi hodnoty -1 až +1. Druhý krok umožňuje odstranění hodnot, které jsou neměnné ve všech příznakových vektorech. Tyto hodnoty zjistíme tak, že vypočítáme směrodatnou odchylku daného statistického parametru skrze všechny příznakové vektory. Pokud je směrodatná odchylka rovna nule, tak tyto parametry nejsou podstatné, protože jsou stejné pro všechny příznakové vektory. Tyto parametry pak byly odstraněny. Příznakové vektory se zmenšily až o 200 příznaků. Předprogramovaná metoda mRMR vyžaduje kategorická data. Kategorizace je pak třetím krokem předzpracování. Do funkcí pro redukci příznakového prostoru nevstupují hodnoty příznakových vektorů, ale kategorizovaná data do předem zvolených hodnot. Tyto předem zvolené hodnoty mohou být například tři: -1, 0 a 1. Autoři [65] uvádí způsob výpočtu diskretizace vektoru příznaků x na tři hodnoty jako:

$$t_{1,2} = \overline{x} \pm k \cdot \sigma(x), \tag{8.1}$$

kde $t_{1,2}$ značí dvě úrovně (*treshold*). Všechny hodnoty nad úroveň t_1 jsou nahrazeny hodnotou 1. Hodnoty mezi úrovní t_1 a t_2 ponecháme rovny 0. Zbytek příznakového prostoru je pak roven hodnotě -1. Konstanty k pak ovlivňují výpočet úrovní $t_{1,2}$ a nabývají hodnot od 0 do 1. Řecké písmeno σ značí směrodatnou odchylku. Pokud by nebyla data kategorizována, tak u objemnějších příznakových vektorů funkce `mrmr_mid_d.m` a `mrmr_miq_d.m`. nefungují a výpočet v prostředí MATLAB končí chybovou hláškou. Empiricky bylo zjištěno, že kategorizace dat vede k lepším výsledkům než použití samotných hodnot příznakových vektorů (kontinuálních dat) [66]. Předpokládáme, že větší počet kategorizačních hodnot bude vést k lepším

výsledkům selekce příznakového prostoru. Z tohoto důvodu byla naprogramována funkce `kategorizace.m`. Tato funkce je schopna kategorizovat data do n hodnot, kde n je liché číslo. Důležitým parametrem jsou pak konstanty úrovní k. V praxi se osvědčilo kategorizovat data do 15 hodnot (-7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7). Celkem pak bylo třeba 14 konstant k. Při navyšování n docházelo k úspěšnější klasifikaci. Vhodnou volbou konstant úrovní a počtu kategorizačních hodnot lze pozorovat změnu úspěšnosti klasifikace a změnu zobrazení parametrů pomocí metody PCA. Metoda PCA je volána funkcí `VypocetVzdalenost.m`. Tato metoda zobrazuje vybrané parametry (skladby) ve vektorovém prostoru tak, aby docházelo k co nejmenšímu zkreslení. Pokud vybrané parametry vytváří shluky, tak očekáváme vysokou úspěšnost klasifikace. Hlavním parametrem, u kterého je očekávána největší závislost na úspěšnosti klasifikace, je počet selektovaných parametrů. Z tohoto důvodu bude úspěšnost algoritmů strojového učení testována v závislosti na počtu selektovaných parametrů metody mRMR.

Funkce `mRMR.m` je naprogramována tak, aby bylo možné ovlivnit jednotlivé kroky zpracování příznakových vektorů. Pro úspěšnou redukci příznaků je potřeba definovat vektor klasifikačních tříd. To je vektor, který algoritmu mRMR označuje, do jaké klasifikační třídy jednotlivé vektory příznaků patří. Tento vektor je tvořen funkcí `mRMR.m`. Všechny vstupní a výstupní parametry a proměnné jsou okomentovány v popisu funkce. Tabulka 8.1 porovnává prvních 10 nejvýznamnějších statistických parametrů pro daný styl databáze. Selektováno bylo 25, 50, 75, 100, 125, 150 a 200 parametrů. Prvních x parametrů je pro daný styl vždy stejný nezávisle na počtu selektovaných parametrů. Konstanty kategorizací dat k byly (1; 0,75; 0,5; 0,25; 0,1; 0,05; 0,025 -0,025; -0,05; -0,1; -0,25; -0,5; -0,75; -1). V tabulce 8.1 je vždy u maticových parametrů uvedena statistická metoda, v závorce je uveden parametr a číslo příslušného řádku matice. U vektorových parametrů není uvedeno číslo řádku. U skalárních parametrů není uváděna statistická metoda.

Mezi prvními 10 nejvýznamnějšími parametry se často objevuje parametr *Tempo*, statistické metody několika řádků parametru *Keystrength* a *Tonalcentroid*. Ze statistických parametrů je nejvíce používán průměr, medián, percentil a kvartil. Součástí přílohy je soubour `Nejvýznamnější parametry.xls`, který obsahuje 200 nejvýznamnějších parametrů pro daný styl.

pořadí	**pop**	**rock**	**disco**	**blues**	**folk a country**
1.	Tempo	Tempo	Tempo	Tempo	99p(Fluktuace,7)
2.	mean(Keystrength,3)	mean(Eevents)	Me(Fluktuace,11)	σ(Beatspectrum)	Q_1(Metroid)
3.	99p(Pulse)	mean(Chromagram,5)	99p(Keystrength,10)	Me(MFCC,13)	mean(Tonalcentroid,2)
4.	Q_1(Fluktuace,15)	mean(Hcdf)	σ(Key)	mean(Keystrength,10)	Var(Key)
5.	Q_3(RMS)	99p(Keystrength,12)	Me(Metroid)	IR(MFCC,8)	mean(Events)
6.	99p(Tonalcentroid,2)	Q_3(Keystrength,9)	mean(Keystrength,11)	99p(Keystrength,9)	Q_3(Keystrength,3)
7.	mean(Hcdf)	mean(Key)	Q_3(Keystrength,8)	mean(Hcdf)	Me(Regularity)
8.	Q_1(Pulse)	IR(Fluktuace,3)	σ(Events)	mean(Tonalcentroid,3)	1p(Metroid)
9.	99p(Fluktuace,2)	Q_3(Keystrength,12)	IR(MFCC,2)	Q_3(Keystrength,3)	Me(Tonalcentroid,3)
10.	Me(Keystrength,1)	Me(Events)	mean(Keystrength,11)	IR(Fluktuace,2)	Me(Keystrength,1)

pořadí	**hip hop a rap**	**jazz**	**metal**	**vážná hudba**	**reggae**
1.	mean(Keystrength,1)	IR(Events)	Var(Events)	99p(Keystrength,2)	mean(Keystrength,6)
2.	Tempo	Me(Tonalcentroid,1)	mean(Keystrength,5)	IR(Events)	Q_3(Events)
3.	mean(Keystrength,5)	IR(MFCC,12)	Tempo	99p(MFCC,10)	Q_3(Keystrength,4)
4.	Q_3(Keystrength,9)	mean(Chromagram,1)	IR(Tonalcentroid,2)	Me(Keystrength,10)	Tempo
5.	σ(Events)	IR(Fluktuace,1)	Me(Events)	mean(Tonalcentroid,3)	99p(Keystrength,2)
6.	Q_3(Tonalcentroid,4)	mean(Keystrength,6)	mean(Key)	IR(Tonalcentroid,6)	99p(MFCC,4)
7.	σ(MFCC,1)	Me(Events)	Me(Tonalcentroid,4)	Q_3(ZRC)	Rolloff
8.	99p(Keystrength,12)	IR(Tonalcentroid,2)	Var(Events,F)	mean(Chromagram,1)	mean(Tonalcentroid,4)
9.	99p(Keystrength,4)	σ(Eevents)	Q_3(Metroid)	IR(Tonalcentroid,1)	mean(Keystrength,8)
10.	99p(Keystrength,8)	mean(Keystrength,10)	Me(Keystrength,1)	mean(Keystrength,3)	IR(Keystrength,7)

Tab. 8.1: 10 nejvýznamnějších parametrů selektovaných metodou mRMR pro každý hudební styl

8.1.4 Výběr algoritmu strojového učení

Prostředí MATLAB obsahuje aplikaci *Classificiation Learner*. Tato aplikace umožňuje trénovat modely strojového učení s učitelem na námi zvolené množině dat. Rozdělení datasetu probíhá pomocí křížové validace. Z algoritmů strojového učení obsahuje *Classificiation Learner* například rozhodovací stromy, SVM, KNN, diskriminační analýzu atd. Výsledkem *Classificiation Learner* je pak výběr algoritmu strojového učení, u kterého je předpovídána největší úspěšnost klasifikace. *Classificiation Learner* ve verzi MATLABu R2016a neobsahuje neuronové sítě. Pro vyhodnocení úspěšnosti selektovaných příznakových vektorů bude využita aplikace *Classificiation Learner* a jednoduchá plně propojená neuronová síť. V tabulce 8.2 jsou uvedeny předpokládané úspěšnosti algoritmů strojového učení pro každý hudební styl databáze při selektování 100 parametrů mRMR. Předpokládaná úspěšnost algoritmů strojového učení je vždy srovnána s úspěšností neuronové sítě, která je popsána v následující subkapitole. Neuronová síť byla vždy testována dvakrát pro každý styl. V prvním případě testování byl využit model neuronové sítě, který měl nejmenší validační chybu. V druhém případě pak byl použit model neuronové sítě s nejmenší trénovací chybou.

Trénování a testování bylo provedeno pro 25, 50, 75, 100, 125, 150 a 200 selektovaných parametrů pro každý styl databáze. Následně byly porovnány úspěšnosti neuronové sítě a úspěšnosti aplikace *Classificiation Learner*. Toto srovnání není zcela objektivní, protože aplikace *Classificiation Learner* používá pro rozdělení dat k-násobnou křížovou validaci. Zatímco námi zvolené rozdělení dat je originální skladba a tři první coververze jako trénovací data, čtvrtá coververze je použita k validaci a pátá k testování. Toto srovnání je však vhodné pro ověřování správnosti selektovaných dat a pozorování chování algoritmů strojového učení nad selektovanými parametry.

Z tabulky 8.2 je zřejmé, že úspěšnost rozpoznávání coververze zavisí na hudebním stylu. Pouze u stylů pop, metal a vážná hudba byla neuronová síť úspěšnější. Úspěšnost neuronové sítě se pak liší při testování modelu s nejmenší validační chybou a nejmenší trénovací chybou.

8.1.5 Neuronová síť

Pro klasifikaci byla naprogramována plně propojená neuronová síť s učícím algoritmem zpětného šíření chyby. Tato síť byla vytvořena v prostředí MATLAB. Síť obsahuje vstupní vrstvu, dvě skryté vrstvy a výstupní vrstvu. Počet neuronů vstupní vrstvy je roven délce příznakového vektoru. Počet neuronů ve skrytých vrstvách je volitelným parametrem. Počet neuronů v první skryté vrstvě byl nastaven na dvojnásobek počtu selektovaných parametrů. Počet neuronů v druhé skryté vrstvě

styl	metoda	**úspěšnost** (Accuracy)	**neuron. síť** min. valid. E (Accuracy)	**neuron. síť** min. trén. E (Accuracy)
pop	Linear disc.	52,5%	45%	**55%**
rock	Ens. sub. disc.	**70%**	65%	65%
disco	Ens. sub. KNN	**55,5%**	50%	40%
blues	Ens. sub. disc.	**51,7%**	30%	50%
folk a country	Ens. sub. disc.	**60,8%**	45%	34%
hip hop a rap	KNN	**69,2%**	40%	55%
jazz	KNN	**79,2%**	60%	65%
metal	Ens. sub. disc.	**85%**	**85%**	**85%**
vážná hudba	Ens. sub. disc.	94,8%	**95%**	**95%**
reggeae	Ens. sub. disc.	**58,5%**	50%	45%

Tab. 8.2: Předpokládaná úspěšnost algoritmů strojového učení a reálná úspěšnost neuronové sítě pro 100 selektovaných parametrů

pak odpovídal počtu selektovaných parametrů. Výstupní vrstva obsahuje právě tolik neuronů, kolik je klasifikačních tříd (20), které určují velikost matice očekávaných výstupů sítě při trénování. Přenosové funkce neuronů ve všech skrytých vrstvách a ve vrstvě výstupní je hyperbolický tangens. Výstup sítě je v intervalu (-1, 1). Pro klasifikaci se pak využívá tzv. *One Hot End Encoding*, kde aktivací příslušného výstupního neuronu (číslo blízké hodnotě +1) je klasifikován vstupní vzor. Neurony ostatních klasifikačních tříd jsou aktivovány na hodnoty blízké -1. Jinými slovy výstupní neuron aktivovaný největším číslem ze všech výstupních neuronů klasifikuje daný vzor do dané klasifikační třídy. Parametry pro nastavení optimálního trénování sítě jsou koeficient učení α, momentum M, počet trénovacích iterací a chyba, na kterou má být síť natrénována. Tuto neuronovu síť je možno spustit pomocí funkce `BPNN.m`, vstupní a výstupní proměnné a parametry jsou okomentovány v popisu funkce. Výhodou této implementace v prostředí MATLAB je využití maticových operací, na které je MATLAB optimalizován.

Trénování proběhlo pro každý hudební styl databáze. Dataset byl rozdělen pomocí funkce `rozdeleni.m`. Před trénováním byly váhové koeficienty a váhy w_0 nastaveny na náhodná čísla blízká kolem hodnoty 0. Trénovací vzory byly náhodně promíchány před každou trénovací iterací. Zamíchání trénovacích vzorů zajišťuje funkce `zamichani.m`, která využívá náhodné permutace počtu trénovacích vzorů. Aktuali-

zace váhových koeficientů byla provedena po každém trénovacím vzoru ($batch = 1$). Tento způsob aktualizace zvyšuje pokles trénovací chyby. Nevýhoda je možnost přeučení. Protože je pro daný styl k dispozici malé množství dat, tak byla vytvořena funkce `add_noise.m`, která přidá uměle vytvořené příznakové vektory do trénovací množiny. Tato funkce přidá šum, který je normalizován na volitelný rozsah. Nejlepších hodnot bylo dosaženo přidáním 100 zašuměných vzorů k jednomu vzoru s normalizovaným šumem do intervalu $< -0,06;\ 0,06 >$. Těchto 100 vzorů bylo přidáno ke každé skladbě. Celkem tak bylo 8 080 trénovacích vzorů pro jeden styl databáze. Na konci každé trénovací iterace (předložení všech trénovacích vzorů) byl proveden test na validační množině. Po natrénování byly uloženy matice váhových koeficientů s nejmenší validační chybou a s nejmenší trénovací chybou. Pro testování byla vytvořena funkce `Testovani.m`. Natrénovaným sítím s nejmenší validační a trénovací chybou byla předložena testovací množina obsahující 20 skladeb, kde každá skladba odpovídala jedné klasifikační třídě. Následně byla vyhodnocena úspěšnost testování, která udává procentuální počet správně klasifikovaných skladeb z celkového množství testovacích skladeb. Tento způsob vyhodnocení odpovídá metrice přesnosti *Accuarcy*.

8.1.6 Vyhodnocení

Pro každý styl databáze bylo natrénováno a otestováno 7 neuronových sítí. Neuronové sítě byly trénovány a testovány v závislosti na počtu selektovaných parametrů mRMR. Selektováno bylo 25, 50, 75, 100, 125, 150 a 200 parametrů. Celkem bylo natrénováno 70 neuronových sítí. Nejrychleji se sítě byly schopny natrénovat s koeficientem učení 0,002 a parametrem momentum nastaveném na hodnotě 0,8. Všechny sítě byly natrénovány na výstupní chybu 10^{-3}. Maximální počet iterací trénování byl nastaven na 200. Časová náročnost výpočtů jedné iterace se pak zvyšovala s rostoucím počtem neuronů (selektovaných parametrů). Grafické průběhy trénovacích a validačních chyb jsou obsaženy v příloze spolu se zobrazením parametrů (skladeb) ve vektorovém prostoru pomocí metody PCA. Dále jsou zde obsažené tabulky porovnávající jednotlivé úspěšnosti.

Z výsledného srovnání úspěšnosti klasifikace je zřejmé, že tato metoda není zcela vhodná pro rozpoznávání coververzí a je vysoce závislá na datasetu (hudebním žánru). U žánrů jako je například vážná hudba, jazz nebo metal jsou úspěšnosti algoritmů uspokojivé. Tento fakt je přisuzován tomu, že každá skladba je z hlediska harmonie jedinečná. U vážné hudby většinou coververze sdílí i stejný notový zápis. Tím je z velké části zajištěna invariance struktury písně, tóniny a částečně i tempa v rámci jednotek BPM. Studie zabývající se rozpoznáváním coververzí dosahují většinou uspokojivých výsledků při použití datasetu s vážnou hudbou. Naopak

u žánrů, kde jednotlivé písně mohou sdílet harmonické postupy (blues, pop atd.) se tato metoda neosvědčila. Při porovnání úspěšnosti s prvními studiemi zabývajícími se rozpoznáváním coververzí je tato metoda na podobné úrovni. Většina prvních studií používala různé metody sjednocení tóniny, tempa nebo struktury a metriky po výpočet vzdáleností. Prostor pro zlepšení této metody by mohl spočívat v použití metod pro sjednocení tóniny, tempa před selekcí parametrů nebo v úpravě již vypočítaných parametrů.

Z testované úspěšnosti algoritmů strojového učení pomocí aplikace *Classificiation Learner* byl nejčasteji volen algoritmus *Ensemble Subspace Discriminant*, *Ensemble Subspace KNN* nebo *KNN* využívající různé metriky (kosinova, euklidovská atd.). Při srovnávání úspěšností s neuronovou sítí je třeba brát v potaz, že *Classificiation Learner* vybírá asi ze 17 typů a variant algoritmů strojového učení. Nejlepších úspěšností dosahovaly algoritmy strojového učení pro 75, 100 a 125 selektovaných parametrů.

8.2 Vyhodnocení systému založeného na rozpoznávání podobnostních matic

Tento systém byl definován na obrázcích 6.4 a 6.5, je u něj očekávána vyšší úspěšnost něž u systému založeného na statistických parametrech. V následujících subkapitolách bude popsána implementace tohoto systému v prostředí MATLAB a v programovacím jazyce Python. V prostředí MATLAB byly naprogramovány funkce na extrahování chromagramů, sjednocení tóniny dvou chromagramů a výpočet matic křížových podobností. Následně zde byly vytvořeny funkce pro tvorbu trénovací, validační a testovací množiny. Celý proces v prostředí MATLAB lze spustit pomocí skriptu `demo.m`. V programovacím jazyce Python byla implementována konvoluční neuronová síť pomocí knihovny *Keras*. Tato síť byla následně natrénována a otestována. Všechny potřebné funkce a vypočítané parametry jsou součástí přílohy ve složce `Druhá metoda`.

8.2.1 Výpočet chromagramů

Všechny skladby zvoleného stylu byly nejdříve načítány a předzpracovány pomocí funkcí `nacitani.m` a `predzprac.m`. Tím byly sjednoceny na délku 180 sekund. Následně byly ke každé skladbě vypočítány chromagramy pomocí MIRtoolboxu. Pro segmentaci musel být nejdříve vytvořen objekt MIRtoolboxu příkazem *mirframe()*. Z těchto objektů byly následně vypočítány chromagramy. Skladby daného žánru databáze byly načítány postupně a uloženy do souboru .mat, který je obsahem přílohy.

	1	2	3	4	5	6	7	8	9	10
cover 1	11	6	4	2	8	12	9	6	2	5
cover 2	11	7	2	8	10	4	3	12	12	12
cover 3	12	7	12	12	1	2	6	3	7	11
cover 4	2	5	3	10	4	10	6	12	3	1
cover 5	12	12	1	8	9	2	5	4	2	12

	11	12	13	14	15	16	17	18	19	20
cover 1	12	1	8	3	8	12	12	12	10	12
cover 2	12	11	12	2	8	12	12	11	12	10
cover 3	12	12	4	12	9	7	4	10	12	10
cover 4	12	10	3	3	12	12	1	12	12	12
cover 5	12	12	5	3	10	2	11	7	12	12

Tab. 8.3: Grafické vyhodnocení úspěšnosti metody hrubé síly OTI

8.2.2 Vyhodnocení metody OTI

Pro sjednocení tóniny dvou chromagramů byla vytvořena funkce `transpozice.m`. V této funkci byly implementovány obě metody podle 5.4. Metoda hrubé síly byla testována na popové části databáze. Celkem zde bylo testováno sjednocení tóniny 100 chromagramů. Vždy mezi originálním chromagramem a pěti chromagramy coververzí. Vyhodnocení popisuje tabulka 8.3, kde jsou uvedeny odhadnuté transpoziční indexy mezi danými skladbami. Indexy uvedené na červeném pozadí jsou odhadnuty špatně. Pokud je index 12, tak to znamená, že tónina dvou skladeb je stejná. Z tabulky 8.3 můžeme pozorovat, že transpoziční index byl odhadnut 27× špatně. To znamená, že pro popovou část databáze bylo úspěšně odhadnuto 73 % indexů. Při zmenšování vzorkovacího kmitočtu se zmenšovala úspěšnost metody OTI. Z tohoto důvodu byla vzorkovací frekvence ponechána na výchozí hodnotě 44 100Hz.

8.2.3 Tvorba trénovací a validační množiny

Pro trénování byla vytvořena množina obsahující 2 100 matic podobností zobrazujících coververze a 2 100 matic zobrazujících ne-coververze. Validační množina obsahovala 300 matic podobností zobrazujících coververze a 300 matic podobností zobrazujících ne-coververze. Toto poměrově stejné zastoupení (50:50) bylo zvoleno pro lepší monitorování úspěšnosti trénování. Pro tvorbu trénovací množiny bylo využito 7 různých žánrů databáze. Pro validační množinu pak byl využit 1 žánr databáze. Každý žánr obsahuje 20 originálních verzí a ke každé originální verzi je

5 coververzí. Jako matici, která zobrazuje coververzi, označujeme i takovou matici, která vznikne srovnáním dvou různých coververzí. Pokud při výpočtu podobnostní matice nezáleží na pořadí chromagramů, tak počet podobnostních matic pro originální verzi a 5 coververzí je vypočítán jako kombinace výběru 2 (k) prvků z 6 (n). Tato kombinace je definována jako:

$$\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!} = \frac{6!}{(6-2)! \cdot 2!} = \frac{6 \cdot 5 \cdot \cancel{4!}}{\cancel{4!} \cdot 2!} = \frac{30}{2} = \underline{\mathbf{15}}. \tag{8.2}$$

Pro každý žánr databáze tak můžeme vypočítat 300 ($15 \cdot 20$) křížových matic podobností porovnávající dvě coververze. Pro 7 hudebních žánrů to pak odpovídá 2 100 vzorkům. Pro tvorbu trénovacích matic byla vytvořena funkce s názvem `rozdeleni_tren_chroma.m`. Vstupem do této funkce byla struktura obsahující chromagramy jednoho žánru databáze (120 chromagramů). Tyto chromagramy byly následně uloženy do buňky Y o rozměrech 6×20. Každý sloupec této buňky odpovídal šestici coververzí. Každá šestice byla pak vstupem funkce *nchoosek()*. Tato funkce vytvořila přesně 15 kombinací dvojic chromagramů z každého sloupce buňky Y. Tyto dvojice chromagramů pak byly uloženy do buňky C s rozměry 300×2. Podobným způsobem byly vytvořeny i křížové matice podobností ne-coververzí. Z buňky Y byly vybírány kombinace po řádcích. Tím byl zajištěn výběr ne-coververzí. Z vybraných dvojic byly následně vypočítány matice křížových podobností. Nejdříve byla sjednocena tónina těchto chromagramů pomocí funkce `transpozice.m` (OTI). Pro výpočet podobnostních matic byla vytvořena funkce `CSM.m`. Výstupem funkce `rozdeleni_tren_chroma.m` byly pak dvě buňky obsahující 300 matic coververzí a 300 matic ne-coververzí. Tímto způsobem byla vytvořena trénovací a validační množina.

Trénovací a validační buňky matic křížových podobností byly sjednoceny do nadřazených buněk (*Tren_Data* a *Valid_Data*) a uloženy jako .mat soubory. Zároveň byly vytvořeny matice obsahující očekávaný výstup konvoluční neuronové sítě. Tyto matice byly také uloženy jako soubory .mat (*Tren_Label* a *Valid_Label*).

8.2.4 Tvorba testovací množiny

Testovací množina byla vytvořena ze zbývajících dvou žánrů databáze. Teoretický způsob testování a vyhodnocení popisuje subkapitola 6.2.2. Celkem tvoří testovací dataset 240 skladeb. Celkem tedy bylo vytvořeno 240^2 (tj. 57 600) podobnostních matic pro testování. Pro tvorbu testovací množiny byla naprogramována funkce s názvem `rozdeleni_test_chroma.m`. Tvorba testovacích matic byla obdobná jako trénovacích (OTI a výpočet CSM). Testovací matice byly uloženy do výstupní buňky o rozměrech 240×240. Dále byla vytvořena matice o velikosti 240×240, která

obsahuje hodnoty 0 a 1. Hodnota 1 se nachází na pozicích, kde očekáváme pravděpodobnosti konvoluční neuronové sítě blízké hodnotě 1 (coververze). Výsledná buňka testovacích matic byla objemná (12,7GB). Pro uložení souboru .mat, který přesahuje 2GB, je třeba použít argument *'-v7.3'* ve funkci *save()*. Tvorba trénovacích, validačních a testovacích množin byla provedena na počítači s operačním systémem macOS. Konvoluční neuronová síť pak byla trénována a testována na počítači s operačním systémem Windows. Pro přenos dat mezi těmito operačnímy systémy je nutný naformátovaný pevný disk tak, aby bylo možné z macOS na disk soubory zapisovat a z Windows tyto soubory číst. Takový disk však umožní přenášet data pouze do určitého objemu (většinou 4GB). Z tohoto důvodu byla testovací množina rozdělena na 4 podmnožiny (*part 1* až *part 4*).

8.2.5 Konvoluční neuronová síť CovNet-1

Programovací jazyk Python je v současné době jeden z nejvhodnějších programovacích jazyků pro implementaci algoritmů umělých neuronových sítí. Tento fakt je zajištěn díky knihovnám *TensorFlow*, *PyTorch* a *Keras* od firem *Google* a *Facebook*. Pomocí knihovny *Keras* je implementace relativně snadná. Konvoluční neuronová síť byla implementována v prostředí *PyCharm*, k implemetaci byly využity objekty knihovny *Keras* (verze 2.3.1). Jako interpret byl používán Python 3.6. Všechny výpočty byly provedeny na procesoru Intel i3 8100 (4 jádra, 4GHz). Knihovna *Keras* vyžaduje instrukce procesoru *AVX* nebo *AVX2* (procesor Xeon X5690 neobsahuje tyto instrukce).

Konvoluční neuronová síť byla implementována přesně podle obrázku 6.5. Tato síť je součástí přílohy pod názvem `model.py`. Pro trénování této sítě bylo třeba načíst jednotlivé .mat soubory, které jsou načteny hlavním skriptem `main.py`. Po načtení jsou tato pole upravena tak, aby odpovídala rozměrům a datovému formátu vstupu sítě. Tato úprava je volána funkcí *format()*, která je obsažena ve skriptu `prevod.py`. Výsledkem je pole knihovny *numpy* s rozměry (n, 180, 180), kde n značí počet matic křížových podobností. Knihovna *Keras* používá datový formát *float32*. Z tohoto důvodu byly všechny vstupní matice a očekávané výstupy přetypovány na tento datový typ. Podle [30] byly vstupní matice ještě normalizovány jako:

$$Y = \frac{X - \overline{X}}{\sigma(X)}, \tag{8.3}$$

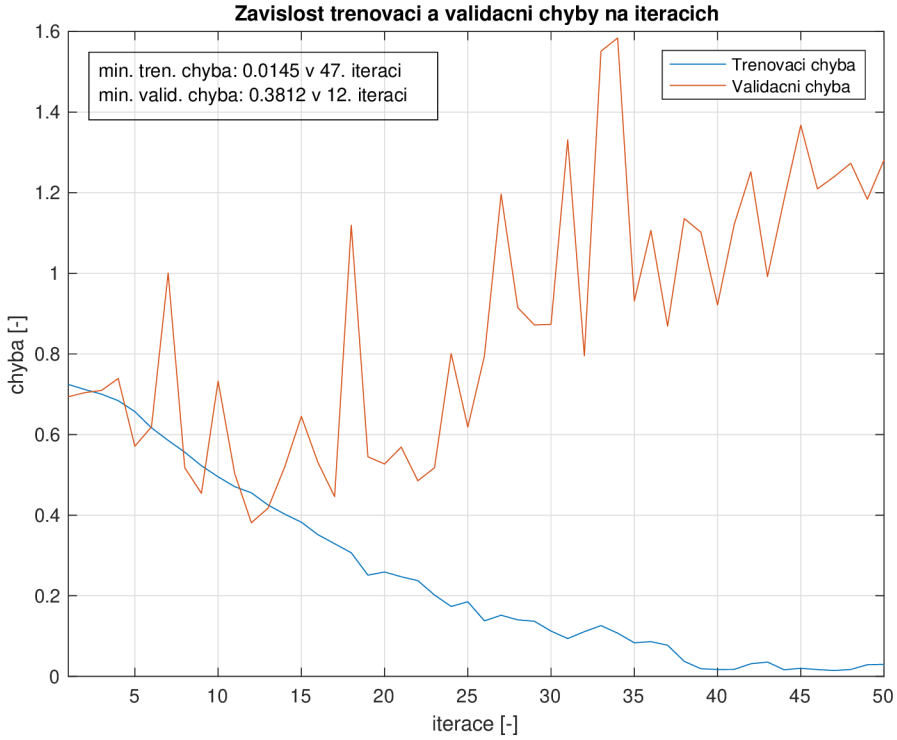
kde X značí matici křížové podobnosti, $\overline{X}$ průměrnou hodnotu a $\sigma(X)$ směrodatnou odchylku. Tato normalizace je provedena funkcí *normalizace()*, kterou obsahuje skript `Zmeannorm.py`. Všechny potřebné funkce byly volány v hlavním skriptu `main.py`.

Úspěšnost trénování závisí na volbě optimálních parametrů. Jako optimalizační algoritmus byl použit *Adam*. Volba učícího koeficentu α je možná v objektu *opt* argumentem *learning_rate* ve skriptu `model.py`. Dalšími parametry byly například: počet vzorů, po kterých mají být aktualizovány váhy (*batch*), počet iterací (*epoch*) sítě, zamíchání vstupních dat před každou iterací, volba poměru prořezávání mezi plně propojenými vrstvami atd. Všechny tyto parametry ovlivňují schopnost sítě se učit. Jako ztrátová funkce byla použita funkce binární křížové entropie (*binary crossentropy*), pozorovanou vyhodnocovací metrikou pak byla použita binární přesnost (*binary accuracy*). Pro vyhodnocení byla trénovací chyba a úspěšnost spolu s validační chybou a úspěšností zapisovány v každé učící iteraci do .csv souboru. Pomocí objektů třídy *callbacks* je možné ovlivnit další parametry trénování jako je například změna učícího koeficientu v průběhu učení v závislosti na poklesu validační chyby. Učící koeficient byl ponechán na výchozí hodnotě (0,001). V průběhu učení byl uložen model CNN s nejmenší validační chybou. Na konci učení pak byl uložen aktuální (poslední) model. Časová náročnost výpočtu záležela na zvoleném parametru *batch*. Pro *batch* = 50 byla délka jedné učící iterace (bez validačního testování) přibližně 240s. Čím větší *batch* je zvolen, tím rychleji je vypočítána jedna iterace (menší počet aktualizací vah). Horní hranice parametru *batch* je dána velikostí operační paměti počítače nebo grafické karty (při učení na GPU). CNN byla trénována na 50 iterací. Celkem bylo trénováno a testováno 5 konvolučních neuronových sítí. Výběr žánrů pro validaci a testování byl rozdělen podle tabulky 8.4. Zbylé žánry pak byly použity pro trénování CNN. Na obrázcích 8.1 a 8.2 jsou uvedeny

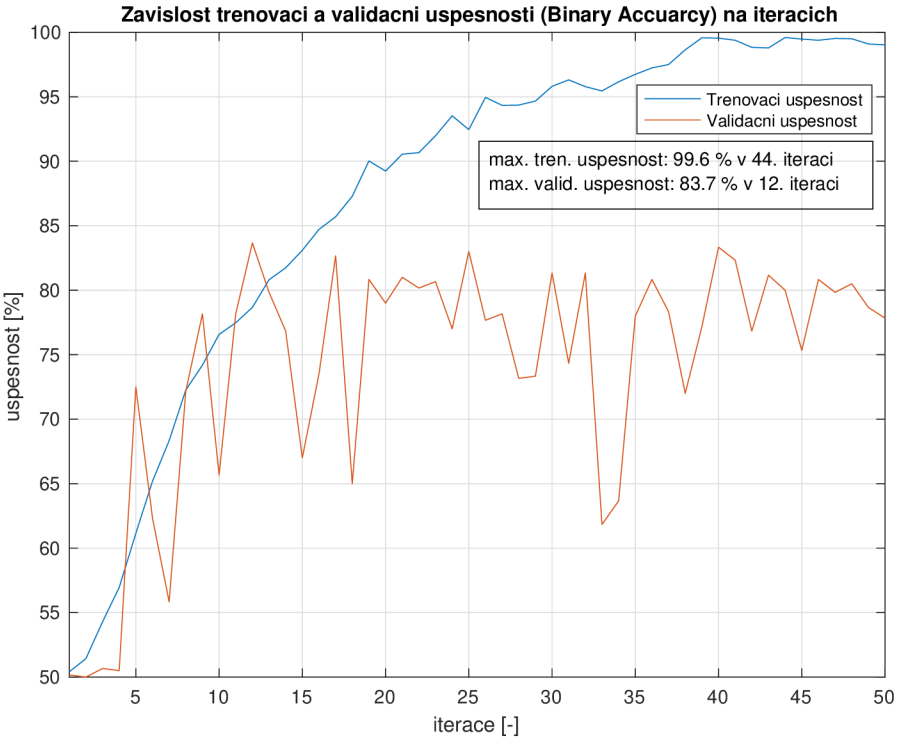
pořadí	**validační žánr**	**testovací žánry**	
1.	metal	vážná hudba	jazz
2.	disco	pop	rock
3.	folk a country	hip hop a rap	blues
4.	pop	metal	reggae
5.	blues	folk a country	disco

Tab. 8.4: Rozdělení žánrů pro trénování a testování 5 CNN

průběhy chyb a úspěšností při trénování a validaci. Data pro trénování a validaci odpovídají prvnímu rozdělení podle tabulky 8.4. Z obrázku 8.1 je zřejmé, že od určité iterace dochází k přeučení. Změna parametrů *batch* a poměru prořezávání nevykazovala velký vliv na přeučení. Trénovací data byla před každou iterací promíchána. Grafické zobrazení 2. – 5. pořadí podle tabulky 8.4 je uvedeno v příloze C.



Obr. 8.1: Grafické zobrazení trénovací a validační chyby



Obr. 8.2: Grafické zobrazení trénovací a validační úspěšnosti

8.2.6 Testování a vyhodnocení úspěšnosti

Pro otestování byl vytvořen skript `Testovani.py`. Tento skript nejdříve načítá jednotlivé části testovací množiny. Pro soubory .mat uložené ve formátu 7.3 je třeba použít knihovnu *H5py*. Pro převod načtených souborů pro vstupní formát sítě byla vytvořena funkce *format_H5PY()*, která je obsažena ve skriptu `prevod.py`. Následně byla pole upravena jako při trénování (datový typ a normalizace). Po této úpravě proběhlo samotné testování, které bylo provedeno na modelu s nejmenší validační chybou a na modelu s vahami po poslední trénovací iteraci. Výstupy těchto modelů na vstupní vzory testovací množiny pak byly uloženy do příslušných souborů .mat. Tyto soubory byly následně načteny v prostředí MATLAB, kde bylo provedeno konečné vyhodnocení. Vyhodnocení v prostředí MATLAB je zajištěno skriptem `Testovani.m` pomocí vyhodnocovacího systému definovaného v kapitole 6.2.2. Pro zařazení vzorků do jednotlivých tříd byl na každý otestovaný model CNN použit vyhodnocovací systém podle největší pravděpodobnosti, minimální kosinovy vzdálenosti a minimálního Pearsonova korelačního koeficientu. Pro každý z těchto vyhodnocovacích systémů byly vypočítány převzaté evaluační metriky soutěže MIREX definované v kapitole 7.5.1. Pro správnou klasifikaci všech dotazovaných skladeb bylo třeba správně klasifikovat 1440 skladeb.

Všechny výsledky jsou uvedeny v tabulce C.1, která je součástí přílohy. Z tabulky je zřejmé, že pro modely CNN s nejmenší validační chybou byl ve většině případů nejúspěšnější vyhodnocovací systém Pearsonova korelačního koeficientu. Naopak pro přeučené CNN byl nejúspěšnější samotný výstup pravděpodobností sítě. Vyhodnocování pomocí kosinovy vzdálenosti se neosvědčilo. Nejlepších výsledků dosáhl model s metalovou validační množinou. Tyto výsledky jsou srovnatelné se studií [41]. Pro otestování tohoto modelu byl použitý žánr vážné hudby a jazzu. Oba tyto žánry dosáhly vysoké úspěšnosti již při testování první metody založené na statistických parametrech. Naopak nejhorších výsledků bylo dosaženo s bluesovou validační množinou. Bluesový žánr nevykazuje časté změny v harmonii jednotlivých skladeb. Mezi hudebníky je harmonický postup (tzv. „bluesová dvanáctka“) označován za základ většiny bluesových skladeb. Z tohoto důvodu je možné, že diagonální struktura byla přítomna i na CSM dvou ne-coververzí a tento žánr pak není vhodný pro validaci. Obecně je tento způsob rozpoznávání méně závislý na hudebním žánru než systém založený na statistických parametrech.

Prostor pro zlepšení této metody by mohl spočívat v několika krocích. Chromagramů existuje několik typů (CENS, CPCP, ...), hlavním parametrem při výpočtu chromagramu je délka segmentu a překryvu segmentů při výpočtu chromagramu z audio signálu. Kratší segment ovlivňuje rozlišení chromagramu. Další zpracování chromagramů, vylepšení metody invariantnosti tóniny a případné použití metody

pro invariantnost tempa by mohlo zajistit jasnější struktury CSM. Hlavním problémem bylo přeučení. Při využití většího množství trénovacích dat dosahovala síť *CovNet-1* lepších úspěšností podle [41]. Trénovací množina by však nebyla zastoupena poměrově stejně. Větší množství trénovacích dat nebo malá změna topologie *CovNet-1* a případné odladění parametrů prořezávání by mohlo ovlivnit přeučení. Více trénovacích dat a možnosti natrénovat více modelů v závislosti na všech možných rozděleních databáze podle žánrů vyžaduje výkonnější hardware.

9 Závěr

Tato diplomová práce se zabývá rozpoznáváním hudebních coververzí za pomocí technik Music Information Retrieval (MIR).

První kapitola krátce zmiňuje oblast MIR, způsob extrahování parametrů popisujících hudební signál a MIRtoolbox pro MATLAB. Je zde vysvětlen rozdíl mezi hluky a zvuky, základní vlastnosti zvuku jako je barva, výška, síla a délka. Následující kapitola pak pojednává o vybraných parametrech, které můžeme extrahovat pomocí MIRtoolboxu. Tyto parametry jsou rozděleny podle základních zvukových vlastností uvedených v první kapitole. Získané parametry můžeme porovnávat pomocí několika různých metrik, proto je konec kapitoly věnován dvěma základním metrikám a algoritmu dynamického borcení časové osy.

Třetí kapitola se zabývá definicí coververze. Jsou zde rozebrány všechny možné typy coververzí a jejich vzájemné odlišnosti. Čtvrtá kapitola hovoří o databázích hudebních coververzí. Obsah hudebních databází bývá chráněn autorským právem a z tohoto důvodu nejsou snadno šiřitelné. Dostupné jsou jen ty, které nejsou dostatečně objemné. Z tohoto důvodu je na konci kapitoly uveden systém, který byl využit pro tvorbu vlastní databáze hudebních coververzí.

V páté kapitole jsou pak vysvětleny vybrané metody a algoritmy pro předzpracování audio signálů a zpracování vypočítaných parametrů. Téměř všechny tyto metody jsou pak použity v implementaci systému pro rozpoznávání coververzí. První část se zabývá předzpracováním audio signálu. V druhé části jsou definovány výpočty statistických parametrů. Třetí část páté kapitoly se zabývá selekcí a výběrem parametrů pomocí metod mRMR a SFFS. V poslední části je definována metoda pro sjednocení tóniny dvou chromagramů a výpočet CSM.

Šestá kapitola pak shrnuje poznatky z nastudovaných systémů zabývajících se identifikací skladeb a rozpoznáváním hudebních coververzí. Je zde vysvětlen princip identifikace skladeb, který využívá firma *Shazam* a popsána úvaha rozpoznávání coververzí pomocí zvukových otisků prstů. Následuje rešerše, která srovnává dosavadní studie zabývající se rozpoznáváním coververzí. Dále jsou zde uvedeny dva systémy na rozpoznávání coververzí. První systém je založen na principu selekce vhodných statistických parametrů a následné klasifikace. Druhý systém je založen na výpočtu CSM a následné klasifikace.

V sedmé kapitole je vysvětlen pojem strojové učení. Je zde uvedeno několik základních algoritmů strojového učení. Větší pozornost je věnována umělým neuronovým sítím a jejich variantám. Na konci této kapitoly jsou uvedena praktická doporučení pro trénování algoritmů strojového učení a následné evaluační metriky pro vyhodnocení úspěšnosti těchto algoritmů.

Poslední osmá kapitola je praktickou částí této diplomové práce. Zabývá se kom-

pletní implementací navržených systémů v prostředí MATLAB a Python. Jsou zde uvedeny jednotlivé kroky a všechny překážky, které bylo třeba vyřešit při programování těchto systémů.

Návrh prvního systému je obecně analogicky podobný jako systémy, které jsou využívány při klasifikaci a parametrizaci hudebních nebo řečových signálů. Z výsledků testování tohoto systému na vytvořeném datasetu je zřejmé, že je tento systém vysoce závislý na hudebním žánru. Při úvaze, které hudební aspekty a parametry nejlépe charakterizují coververze, není snadné tyto parametry předem odhadnout. Nejúspěšnější klasifikace ($Accuracy > 70$ %) byla dosažena při použití 75 až 125 parametrů s žánry vážná hudba, jazz a metal. Předpokládaný prostor pro zlepšení tohoto systému je ve využití metod pro invariantnost tempa a tóniny klasifikovaných skladeb.

Druhý systém pak využívá binární klasifikaci založenou na rozpoznávání struktury obrazů pomocí konvolučních neuronových sítí. Tento systém byl následně vyhodnocován tak, aby mohly být testovací vzory klasifikovány do jednotlivých klasifikačních tříd. Celkově byl tento systém úspěšnější než systém první. Mezi přednosti patří využití metody pro sjednocení tóniny a nízké nároky na výpočetní výkon při tvorbě matic křížových podobností. Tento systém se ukázal méně závislý na hudebním žánru než systém první. Z podstaty výpočtu CSM byla nezávislost na hudebním žánru předpovídána. Výsledky testování tohoto systému byly podobné jako uvedené výsledky v předchozí studii. Prostor pro zlepšení by pak mohl spočívat v délce časového okna při výpočtu chromagramů a případně samotného typu chromagramu (CENS, CPCP...). Druhou možností pro zlepšení je úprava topologie CNN, změna velikosti nebo počtu konvolučních jader. Hlavním zlepšením by mohlo být použití většího počtu trénovacích dat. Tato vylepšení by mohla být předmětem dalších studií.

Na závěr je nutno podotknout, že to, co se může zdát jednoduché „na papíře“, je často mnohem složitější při implementaci v praxi a naopak.

Literatura

[1] KNEES, P., SCHEDL M. *Music similarity and retrieval: an introduction to audio and web-based strategies.* 1. New York, NY: Springer Berlin Heidelberg, 2016. ISBN 9783662497203.

[2] ISMIR [online], 2000. [cit. 2019-11-18]. Dostupné z: <`https://www.ismir.net/`>

[3] CHEN, Ning, J. Stephen DOWNIE, Hai-dong XIAO a Yu ZHU, 2015. *Applied Acoustics: Cochlear pitch class profile for cover song identification.* Vol 99. Elsevier. 0003-682X. Dostupné také z: <`https://www.sciencedirect.com/science/article/abs/pii/S0003682X15001681`>

[4] LARTILLOT, Olivier, *MIRtoolbox 1.7* [online]. In: . June 3rd 2017 [cit. 2019-11-18]. Dostupné z: <`https://www.jyu.fi/hytk/fi/laitokset/mutku/en/research/materials/mirtoolbox/manual1-7.pdf`>

[5] ZENKL, Luděk, 2012. *ABC hudební nauky.* 2. vydání, 2012. Praha: Bärenreiter Praha. ISBN 978-80-86385-21-1.

[6] MÜLLER, Meinard a Sebastian EWERT, *Chroma Toolbox: Matlab Implementations for Extracting Variants of Chroma-Based Audio Features.* [online]. In: . 215 - 220 [cit. 2019-11-21]. Dostupné z: <`http://resources.mpi-inf.mpg.de/MIR/chromatoolbox`>

[7] Wikipedia contributors. (2019, September 26). *Mel scale. In Wikipedia, The Free Encyclopedia.* Retrieved 13:06, November 22, 2019, Dostupné z: <`https://en.wikipedia.org/w/index.php?title=Mel_scale&oldid=917928223`>

[8] MIKLÁNEK, Štěpán, 2019. *Určení místa původu hudebních interpretací české komorní a orchestrální hudby za pomoci technik Music Information Retrieval.* Brno. Dostupné také z: <`https://www.vutbr.cz/studenti/zav-prace/detail/118138?zp_id=118138`>. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Tomáš Kiska.

[9] SEMELA, René, 2018. *Doporučování hudebního obsahu založené na technikách získávání hudební informace.* Brno. Dostupné také z: <`https://www.vutbr.cz/studenti/zav-prace/detail/110154?zp_id=110154`>. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Tomáš Kiska.

[10] Přispěvatelé Wikipedie, Norma matice [online], Wikipedie: Otevřená encyklopedie, c2019, Datum poslední revize 23. 04. 2019, 14:18 UTC, [citováno 29. 11. 2019] <https://cs.wikipedia.org/w/index.php?title=Norma_matice&oldid=17170552>.

[11] Wikipedia contributors. (2019, August 11). *Cosine similarity.* In Wikipedia, The Free Encyclopedia. Retrieved 12:26, December 17, 2019, Dostupné z: <https://en.wikipedia.org/w/index.php?title=Cosine_similarity&oldid=910391235>

[12] PORTILLA, Ricardo, Brenner HEINTZ a Denny LEE, 2019. *Understanding Dynamic Time Warping: Part 1 of our Using Dynamic Time Warping and MLflow to Detect Sales Trends Series* [online]. Company blog [cit. 2019-12-02]. Dostupné z: <https://databricks.com/blog/2019/04/30/understanding-dynamic-time-warping.html>.

[13] DI ROSSO, Simone, 2018. *Dynanmic Time Warping.* In: You Tube [online]. 29. 4. 2018 [cit. 2019-12-07]. Dostupné z: <https://www.youtube.com/watch?v=tfOevFKQIjQ>.

[14] SEHN KÖRTING, Thales, 2017. *How DTW (Dynamic Time Warping) algorithm works.*In: You Tube [online]. [cit. 2019-12-07]. Dostupné z: <https://www.youtube.com/watch?v=_K1OsqCicBY&t=92s>.

[15] SAKOE, Hiroaki a Seibi CHIBA, 1978. *Dynamic programming algorithm optimization for spoken word recognition.* In: IEEE Transactions on Acoustics, Speech, and Signal Processing [online]. IEEE, Feb 1978, 43 - 49 [cit. 2019-12-07]. DOI: 10.1109/TASSP.1978.1163055. ISSN 0096-3518. Dostupné z: <https://ieeexplore.ieee.org/document/1163055/authors#authors>

[16] Přispěvatelé Wikipedie, *Všeobecná úmluva o autorském právu* [online], Wikipedie: Otevřená encyklopedie, c2019, Datum poslední revize 28. 02. 2019, 11:37 UTC, [citováno 16. 11. 2019]<https://cs.wikipedia.org/w/index.php?title=V%C5%A1eobecn%C3%A1_%C3%BAmluva_o_autorsk%C3%A9m_pr%C3%A1vu&oldid=16999846>

[17] SERRÀ, Joan, Emilia GÓMEZ a Perfecto HERRERA, 2010. *Audio cover song identification and similarity: background, approaches, evaluation, and beyond. Studies in Computational Intelligence* [online]. 2010, 2010(274), 307 - 332 [cit. 2019-11-16]. DOI: 10.1007/978-3-642-11674-2_14. Dostupné z: <https://www.researchgate.net/publication/225612184_Audio_

Cover_Song_Identification_and_Similarity_Background_Approaches_Evaluation_and_Beyond>

[18] JESPER, Jensen, Christensen MADS, Jensen a Ellis DANIEL, 2008. A tempo-insensitive distance measure for cover song identification based on chroma features. *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on* [online]. 1.3.2008, 2008, 2209 - 2212 [cit. 2019-11-17]. DOI: 10.1109/ICASSP.2008.4518083. Dostupné z: <https://www.researchgate.net/publication/220731844_A_tempo-insensitive_distance_measure_for_cover_song_identification_based_on_chroma_features/citation/download>

[19] THIERRY, Bertin-Mahieux a Ellis DANIEL, 2012. Large-scale cover song recognition using the 2D Fourier transform magnitude. *13th International Society for Music Information Retrieval Conference ISMIR 2012* [online]. 1.1.2012, 2012, 241 - 246 [cit. 2019-11-17]. Dostupné z: <https://ismir2012.ismir.net/event/papers/241_ISMIR_2012.pdf>

[20] Video and audio formatting specifications, Google [online]. [cit. 2019-11-17]. Dostupné z: <https://support.google.com/youtube/answer/4603579?hl=en>

[21] VOGT, Nick, 2015. YouTube Audio Quality Bitrate Used For 360p, 480p, 720p, 1080p, 1440p, 2160p. *H3xed* [online]. 2015 [cit. 2019-11-17]. Dostupné z: <https://www.h3xed.com/web-and-internet/youtube-audio-quality-bitrate-240p-360p-480p-720p-1080p>

[22] FENCL, Ivan, 2019. Limity užití autorského díla ve školách. In: *Advokátní kancelář Kropáček LEGAL* [online]. 15.05.2019 [cit. 2019-11-17]. Dostupné z: <https://pravopropodnikatele.cz/limity-uziti-autorskeho-dila-ve-skolach/>.

[23] Webová stránka firmy MathWorks: dokumentace k funkci resample. MathWorks: resample [online]. [cit. 2020-12-17]. Dostupné z: <https://www.mathworks.com/help/signal/ref/resample.html>

[24] HANCHUAN, Peng, Fuhui LONG a Chris DING, August 2005. *Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy.* IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. 27(8). Dostupné také z: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1453511>.

[25] RASCHKA, Sebastian. Sequential Feature Selector [online]. [cit. 2021-02-08]. Dostupné z: <http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/>.

[26] SOMOL, Petr, Jana NOVOVICOVA a Pavel PUDIL, 2010. *Efficient Feature Subset Selection and Subset Size Optimization.* Pattern Recognition Recent Advances. InTech, 2010-02-01. ISBN 978-953-7619-90-9. Dostupné z: doi:10.5772/9356

[27] SERRÁ, Joan, Emilia GÓMEZ a Perfecto HERRERA, 2008. *Transposing Chroma Representations to a Common Key.* IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects. 2008, 45-48. 88-7595-010-5. Dostupné také z: <http://mtg.upf.edu/files/publications/jserra08TransposigChroma2commonKey.pdf>.

[28] P.W. ELLIS, Daniel a Courtenay V. COTTON, 2007. *THE 2007 LABROSA COVER SONG DETECTION SYSTEM.* Music Information Retrieval Evaluation eXchange. 2007. Dostupné také z: <https://www.researchgate.net/publication/228926795_The_2007_LabROSA_cover_song_detection_system>.

[29] FOOTE, Jonathan, November 1999. *Visualizing music and audio using self-similarity.* MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia. Orlando, Florida, 1999, 77-80. Dostupné z: <doi:10.1145/319463.319472>

[30] CHANG, Sungkyunv, Juheon LEE, Sang Keun CHOE a Kyogu LEE, 1 Dec 2017n. 1 *Audio Cover Song Identification using Convolutional Neural Network.* Center for Superintelligence - Seoul National University, 2017. Dostupné také z: <https://www.researchgate.net/publication/321487749_Audio_Cover_Song_Identification_using_Convolutional_Neural_Network>

[31] Meinard Müller, Nanzhu Jiang, and Harald G. Grohganz *SM Toolbox: MATLAB Implementations for Computing and Enhancing Similarity Matrices* In Proceedings of 53rd Audio Engineering Society (AES), 2014 Dostupné z: <https://www.audiolabs-erlangen.de/resources/MIR/SMtoolbox/>

[32] Přispěvatelé Wikipedie, *Shazam (software)* [online], Wikipedie: Otevřená encyklopedie, c2018, Datum poslední revize 20. 12. 2018, 20:00 UTC, [citováno 10. 12. 2019] Dostupné z: <https://cs.wikipedia.org/w/index.php?title=Shazam_(software)&oldid=16794916>.

[33] Shazam [online], [cit. 2019-12-10]. Dostupné z: <https://www.shazam.com/cs>.

[34] MÜLLER, Meinard, 2015. *Fundamentals of music processing.* New York, NY: Springer Berlin Heidelberg. ISBN 978-3-319-21944-8.

[35] SCHALKWIJK, Jerome, *A Fingerprint for Audio.* Intrasonics [online]. [cit. 2019-12-10]. Dostupné z: <https://medium.com/intrasonics/a-fingerprint-for-audio-3b337551a671>

[36] TRALIE, Chris, 2018. *Audio Cover Song Identification: Beyond The Notes.* Dostupné také z: <http://nemisig2019.nemisig.org/images/tralieSlides/slides.pdf>

[37] GÓMEZ, Emilia, Beesuan ONG a Perfecto HERRERA, 2006 October 5–8. *Automatic tonal analysis from music summaries for version identification: AES 121th Convention.* Dostupné také z: <files/publications/e8cb50-AES121-GomezOngHerrera.pdf>

[38] KURTH, F. a M. MULLER. *Efficient Index-Based Audio Matching.* IEEE Transactions on Audio, Speech, and Language Processing. 2008(16), 382-395. Dostupné z: <doi:10.1109/TASL.2007.911552>

[39] RODRIGUES DUARTE, Carlos Manuel, 2015. *Audio Cover Song Identification.* Lisabon. Dostupné také z: <https://fenix.tecnico.ulisboa.pt/downloadFile/563345090413860/MEIC-79263-Carlos-Duarte.pdf.>. Diplomová. Instituto Superior Técnico. Vedoucí práce Doctor David Manuel Martins de Matos.

[40] LEE, Juheon, Sungkyun CHANG, Donmoon LEE a Kyogu LEE, 2018. *COVERNET: COVER SONG IDENTIFICATION USING CROSS-SIMILARITY MATRIX WITH CONVOLUTIONAL NEURAL NETWORK.* Seoul National University, Korea, (2018). Dostupné také z: <https://www.music-ir.org/mirex/abstracts/2018/LCLL1.pdf>

[41] LEE, Juheon, Sungkyun CHANG, Sang Keun CHOE a Kyogu LEE, 2018. *Cover Song Identification Using Song-to-Song Cross-Similarity Matrix with Convolutional Neural Network.* 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Center for Superintelligence, Seoul National University, 08826, Korea, 2018, 396 - 400. 2379-190X. Dostupné z: <doi:10.1109/ICASSP.2018.8461395>

[42] TRALIE, Christopher J. a Paul BENDICH, July 21, 2015. *Cover Song Identification with Timbral Shape Sequences.* 1-12. arXiv1507.05143. Dostupné také z: <https://arxiv.org/abs/1507.05143v1>

[43] HEO, Hunn, Hyunwoo J. KIM, Wansoo KIM a Kyogu LEE, January 2007. *Cover Song Identification with Metric Learning Using Distance as a Feature.* The International Society of Music Information Retrieval (ISMIR). 2007. Dostupné také z: <https://www.researchgate.net/publication/317951528_Cover_Song_Identification_with_Metric_Learning_Using_Distance_as_a_Feature>

[44] SILVA, Diego F., Chin-Chia M. YEH, Yan ZHU, Gustavo E. A. P. A. BATISTA a Eamonn KEOGH, 2019. *Fast Similarity Matrix Profile for Music Analysis and Exploration.* IEEE Transactions on Multimedia. 21(1), 29-38. ISSN 1520-9210. Dostupné z: <doi:10.1109/TMM.2018.2849563

[45] YU, Zhesong, Chen XIAOOU, Xiaoshuo XU a Deshun YANG, November 2019. *Learning a Representation for Cover Song Identification Using Convolutional Neural Network.* Wangxuan Institute of Computer Technology, Peking University. netarXiv: 1911.00334v1. Dostupné také z: <https://www.researchgate.net/publication/337005573_Learning_a_Representation_for_Cover_Song_Identification_Using_Convolutional_Neural_Network

[46] HALL, Patrick, Wen PHAN a Katie WHITSON, 2016. *The Evolution of Analytics: Opportunities and Challenges for Machine Learning in Business* [online]. 1005 Gravenstein Highway North, Sebastopol: O'Reilly Media [cit. 2019-12-12]. ISBN 978-1-491-95471-3. Dostupné z: <https://www.sas.com/en_us/whitepapers/evolution-of-analytics-108240/download.html#formsuccess>.

[47] Wikipedie: Otevřená encyklopedie: *Strojové učení* [online]. c2019 [citováno 12. 12. 2019]. Dostupný z: <https://cs.wikipedia.org/w/index.php?title=Strojov%C3%A9_u%C4%8Den%C3%AD&oldid=17660652>.

[48] KURFÜRSTOVÁ, Jana, *Strojové učení kouzla zbavené.* EDTECH KISK: Educational Technology, KISK Masaryk University [online]. Apr 16, 2018 [cit. 2019-12-12]. Dostupné z: <https://medium.com/edtech-kisk/strojov%C3%A9-u%C4%8Den%C3%AD-kouzla-zbaven%C3%A9-e066d79ebe51>.

[49] KUČERA, Jiří, *Shluková analýza: Algoritmus k-means* [online]. [cit. 2019-12-15]. Dostupné z: <https://is.muni.cz/th/172767/fi_b/5739129/web/web/kmeans.html>.

[50] ŠOCHMAN, Jan, *Cvičení z RPZ - Shlukování k-means* [online]. 7. prosince 2005 [cit. 2019-12-15]. Dostupné z: <http://cmp.felk.cvut.cz/cmp/courses/recognition/Labs/kmeans/kmeans.pdf>.

[51] STARMER, Josh, *StatQuest: K-means clustering. In: YouTube* [online]. 23. 5. 2018 [cit. 2019-12-15]. Dostupné z: <https://www.youtube.com/watch?v=4b5d3muPQmA>.

[52] VOGLER, B.; OTHMAN, A.: *Music Genre Recognition.* [online]. Bauhaus-Universität Weimar, Weimar, 2016. [citováno 15. 12. 2019]. Dostupné z: <https://benediktsvogler.com/downloads/DocumentationAudiotechnikMusicGenreRecognition.pdf>.

[53] VOLNÁ, Eva, 2008. *NEURONOVÉ SÍTĚ 1.* Ostava. Dostupné také z: <https://web.osu.cz/~Volna/Neuronove_site_skripta.pdf.> Studijní materiály pro distanční kurz: Neuronové sítě 1. Ostravská universita v Ostravě.

[54] ŠNOREK M.,JIŘINA M. *Neuronové sítě a neuropočítače*, ČVUT, Praha 1996. ISBN 80-01-01455-X

[55] HOCHREITER, Sepp a Jürgen SCHMIDHUBER, 1997. *Long Short-Term Memory. Neural Computation.* MIT Press, 9(8), 1735-1780. ISSN 0899-7667. Dostupné z: doi:10.1162/neco.1997.9.8.1735

[56] *Understanding LSTM Networks.* Git Hub [online]. August 27, 2015 [cit. 2020-12-17]. Dostupné z: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

[57] LECUN, Yann, Patrick HAFFNER, Léon BOTTOU a Yoshua BENGIO, 22 October 1999n. l. *Object Recognition with Gradient-Based Learning* [online]. Berlin, Heidelberg: Springer [cit. 2021-02-22]. ISBN 978-3-540-46805-9. Dostupné z: <doi:https://doi.org/10.1007/3-540-46805-6_19>

[58] ZACHA, Jiří, Květen 2019. *Konvoluční neuronové sítě pro klasifikaci objektů z LiDARových dat.* Praha. Dostupné také z: <https://dspace.cvut.cz/bitstream/handle/10467/82351/F3-BP-2019-Zacha-Jiri-Konvolucni_neuronove_site_pro_klasifikaci_objektu_z_LiDARovych_dat.pdf.> Bakalářská práce. České vysoké učení technické v Praze. Vedoucí práce Ing. Patrik Vacek.

[59] LEE, James, Apr 19, 2018. *Convolutional Neural Networks: The Biologically-Inspired Model.* Codementor comunity [online]. [cit. 2021-02-22]. Dostupné z: <https://www.codementor.io/@james_aka_yale/

convolutional-neural-networks-the-biologically-inspired-model-iq6\
s48zms>

[60] MITRENGA, M. *Konvoluční neuronová síť pro segmentaci obrazu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 55 s. Vedoucí bakalářské práce doc. Ing. Václav Jirsík, CSc.

[61] IOFFE, Sergey a Christian SZEGEDY, 2015. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. arXiv1502.03167. Dostupné také z: <https://arxiv.org/abs/1502.03167

[62] *Machine Learning*, Google.com [online]. [cit. 2019-12-15]. Dostupné z: <https://developers.google.com/machine-learning/crash-course>.

[63] Přispěvatelé WikiSkript, *Požadavky na vyšetřovací techniky* [online], , c2018, Datum poslední revize 8. 02. 2018, 12:44 UTC, [citováno 15. 12. 2019] Dostupné z: <https://www.wikiskripta.eu/index.php?title=Po%C5%BEadavky_na_vy%C5%A1et%C5%99ovac%C3%AD_techniky&oldid=398041>.

[64] *2006:Audio Cover Song: Task Description*, MIREX [online]. 13 May 2010 [cit. 2019-12-15]. Dostupné z: <https://www.music-ir.org/mirex/wiki/2006:Audio_Cover_Song>.

[65] Hanchuan Peng (2021). *mRMR Feature Selection (using mutual information computation).* Dostupné z: <https://www.mathworks.com/matlabcentral/fileexchange/14608-mrmr-feature-selection-using-mutual-information-computation>, MATLAB Central File Exchange. Retrieved April 2, 2021.

[66] PENG, Hanchuan. *MRMR (minimum Redundancy Maximum Relevance Feature Selection)* [online]. [cit. 2021-04-02]. Dostupné z: <http://home.penglab.com/proj/mRMR/>

Seznam symbolů a zkratek

ADSR	amplitudová obálka signálu – Attack, Decay, Sustain, Release
BPM	počet úderů za minutu – Beat per Minute
CNN	konvoluční neuronová síť – Convolutional Neural Network
CSM	matice křížové podobnosti – Cross Similarity Matrix
DCT	diskrétní kosinova transformace – Discrete Cosine Transform
DTW	dynamické borcení časové osy – Dynamic Time Warping
FFT	rychlá Fourierova transformace – Fast Fourier Transform
ISMIR	mezinárodní společnost pro získávání hudebních informací – International Society for Music Information Retrieval
LUFS	jednotka pro měření programové hlasitosti – Loudness Unit Full Scale
MFCC	Melovské kepstrální koeficinety – Mel-frequency cepstral coeficient
MIDI	standart pro digitální komunikaci mezi hudebními nástroji a zařízením – Musical Instrument Digital Interfacel
MIR	získání hudebních informací – Music Information Retrieval
MIREX	komunita pořádající každoroční konference ISMIR – Music Information Retrieval Evaluation eXchange
mRMR	metoda selekce parametrů – minimum Redundancy Maximum Relevance
OTI	metoda sjednocení tóniny – Optimal Transposition Index
RMS	efektivní hodnota signálu – Root Mean Square

Seznam příloh

A Databáze coververzí a přiložené CD

Databáze coververzí nemůže být přiložena na CD ani v informačním systému. Pro získání databáze kontaktujte studenta na školní e-mail: xmarti73@stud.feec.vutbr.cz. Pro nahlédnutí na obsah databáze slouží online tabulka dostupná z:<`https://bit.ly/3e4yfCn`>.

Přiložené CD obsahuje v kořenovém adresáři samotný text práce ve formátu .pdf. Kořenový adresář pak obsahuje i složky `MATLAB` a `Python`, kde jsou skripty a funkce vytvořené v prostředí MATLAB R2016a a Python 3.6.

`MATLAB` obsahuje dvě základní podsložky `První Metoda` a `Druhá Metoda`. Každá z těchto podsložek obsahuje skripty s názvem `demo.m`, které slouží pro spuštění metody a základní orientaci v hierarchii skriptů a funkcí. Dále jsou v těchto podsložkách obsaženy předprogramované metody, které tato práce využívá (mRMR, cell2csv, Chroma Toolbox atd.). `První Metoda` pak obsahuje výčet 200 nejvýznamnějších parametrů selektovaných metodou mRMR. Tento výčet byl vytvořen pro každý styl databáze a je uložen v souboru `Nejvýznamnější parametry.xls`. Dále jsou zde obsaženy již vypočítané parametry pro každý styl databáze ve formátech .mat a .csv. `Druhá Metoda` obsahuje tři hlavní podsložky, které obsahují implementaci metod pro výpočet CSM pomocí MIRtoolboxu, Chroma Toolboxu a následného testování.

`Python` obsahuje pět základních funkcí pro načtení a následnou úpravu dat z formátu .mat a trénování konvoluční neuronové sítě. Načtení, úprava dat a trénování je spouštěno skriptem `main.py`. Testování je spouštěno skriptem `Testovani.py`.

- Václav Martinek - Diplomová práce - 171620.pdf
- MATLAB
  - První Metoda
    - Nejvýznamnější parametry.xls
    - cell2csv
    - pca
    - Extrahované parametry
      - Statistika
      - CSV
    - mrmr_d_matlab_src
    - mi
    - demo.mat
    - ostatní funkce a skripty (.m)
  - Druhá Metoda
    - Pomoci MIRtool
      - demo.m
      - MAT Chromagramy
      - ostatní funkce a skripty (.m)
    - Pomoci Chromatool
      - demo.m
      - ostatní funkce a skripty (.m)
      - MAT Chromagramy
    - Testovani
      - Testovani.m
      - funkce a skripty pro testování (.m)
- Python
  - Natrenované sítě
  - main.py
  - model.py
  - prevod.py
  - Zmeannorm.py
  - Testovani.py

B Úspěšnost metody založené na statistických parametrech

V následujících tabulkách je uvedeno procentuální srovnání úspěšnosti (*Accuracy*) při testování natrénovaných neuronových sítích s předpokládanou úspěšností vybraných algoritmů strojového učení aplikací *Classification Learner* v závislosti na počtu selektovaných parametrů metodou mRMR.

Dále jsou zde obsaženy grafy trénování jednotlivých neuronových sítí a zobrazení selektovaných parametrů ve vektorovém prostoru pro každý styl databáze.

Pop	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	20	35	45	40	45	30	40
Neuronová síť min. trén. E (Accuracy)	45	**55**	45	45	35	45	30
předpokládaná úspěšnost podle *Classification Learner*	**47**	48	**53**	**53**	**52**	**47**	**50**
Rock	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	45	55	55	65	60	60	60
Neuronová síť min. trén. E (Accuracy)	45	65	**70**	**70**	60	60	65
předpokládaná úspěšnost podle *Classification Learner*	**66**	**68**	68	68	**65**	**66**	**67**

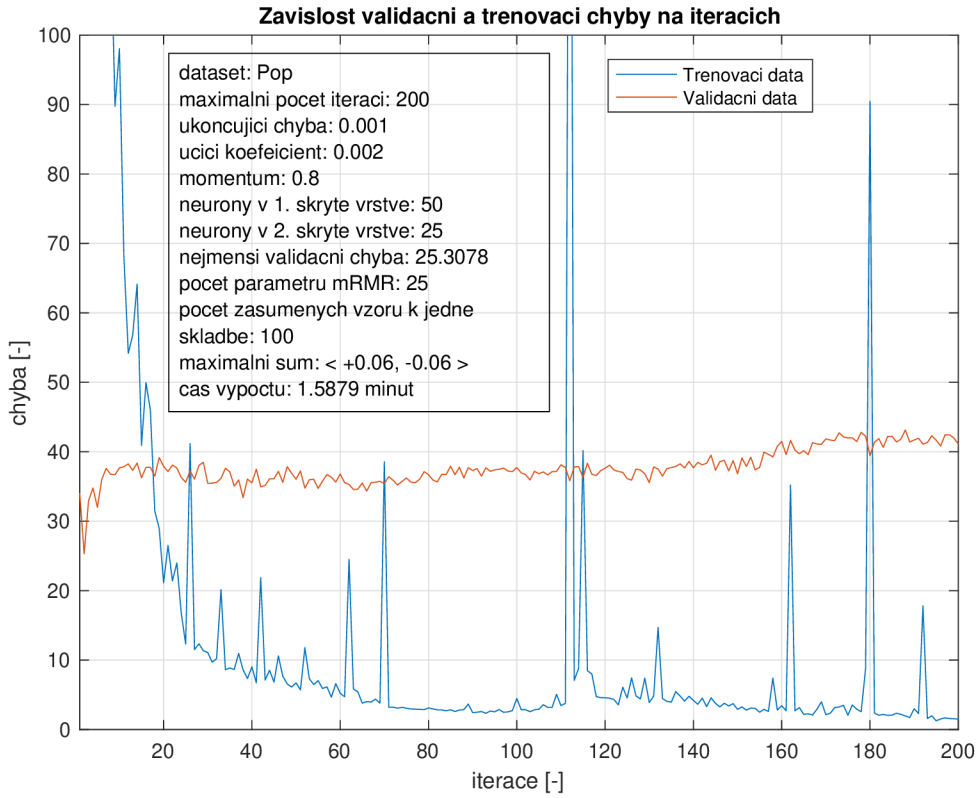
Disco	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	30	40	40	45	35	30	40
Neuronová síť min. trén. E (Accuracy)	45	45	50	55	50	**50**	40
předpokládaná úspěšnost podle *Classification Learner*	**53**	**52**	**56**	**57**	**52**	**50**	**52**

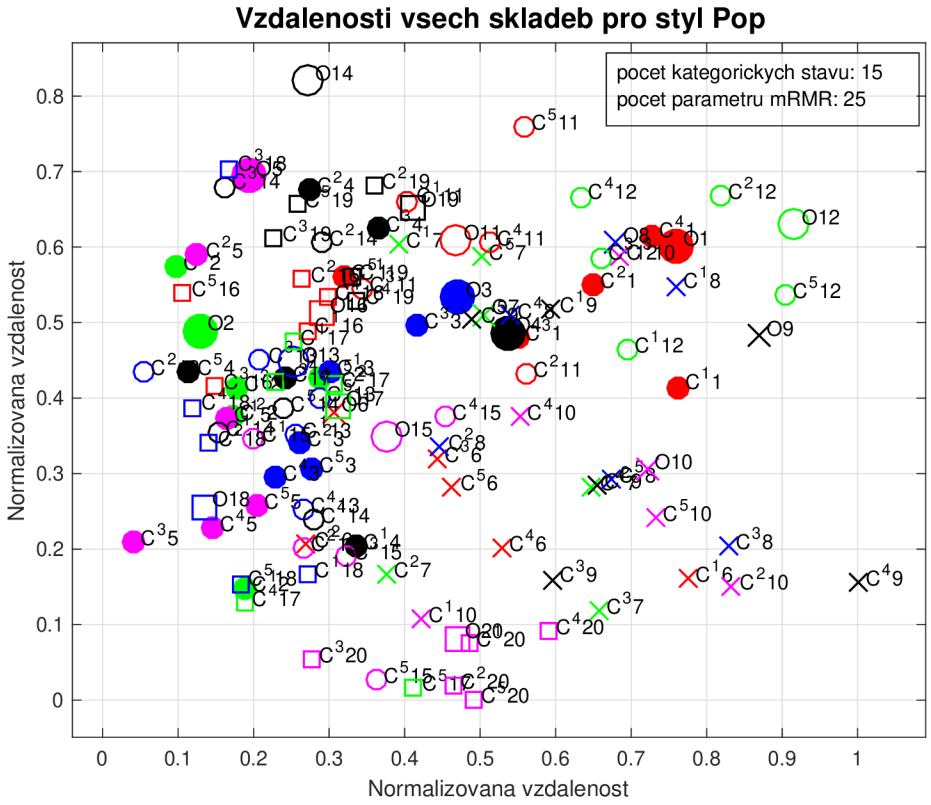
Blues	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	25	45	30	50	45	45	35
Neuronová síť min. trén. E (Accuracy)	40	40	50	35	40	45	35
předpokládaná úspěšnost podle *Classification Learner*	**49**	**46**	**54**	**52**	**50**	**46**	**44**

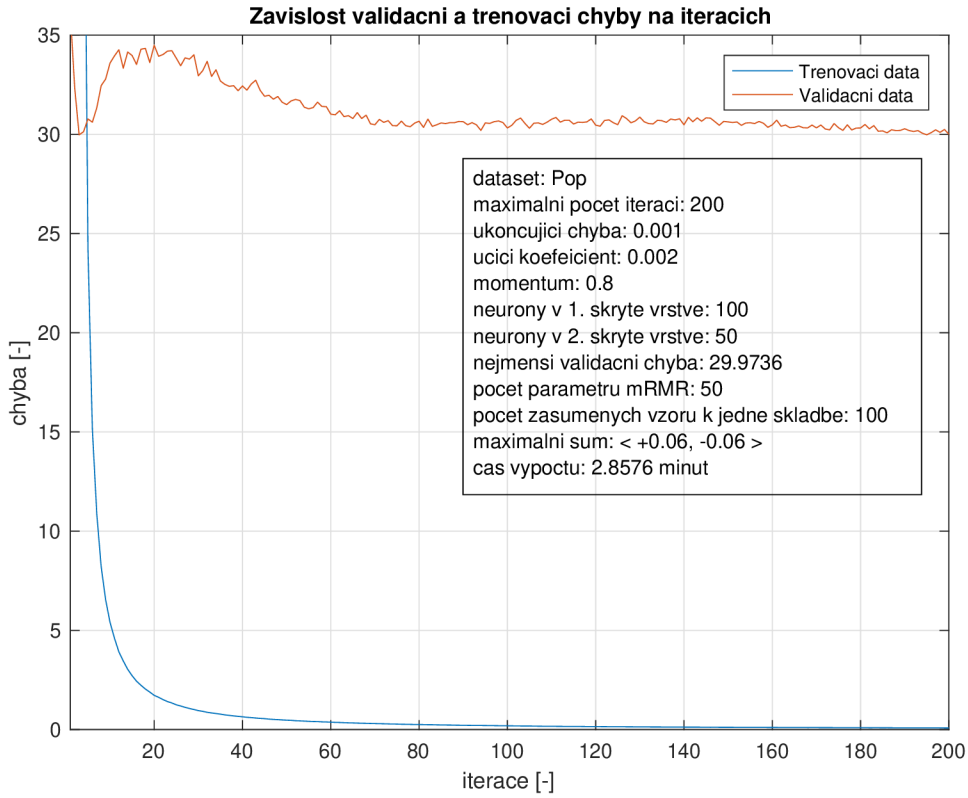
Folk a Country	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	20	20	35	35	45	30	40
Neuronová síť min. trén. E (Accuracy)	20	30	50	40	40	35	50
předpokládaná úspěšnost podle *Classification Learner*	**56**	**58**	**60**	**65**	**60**	**62**	**62**

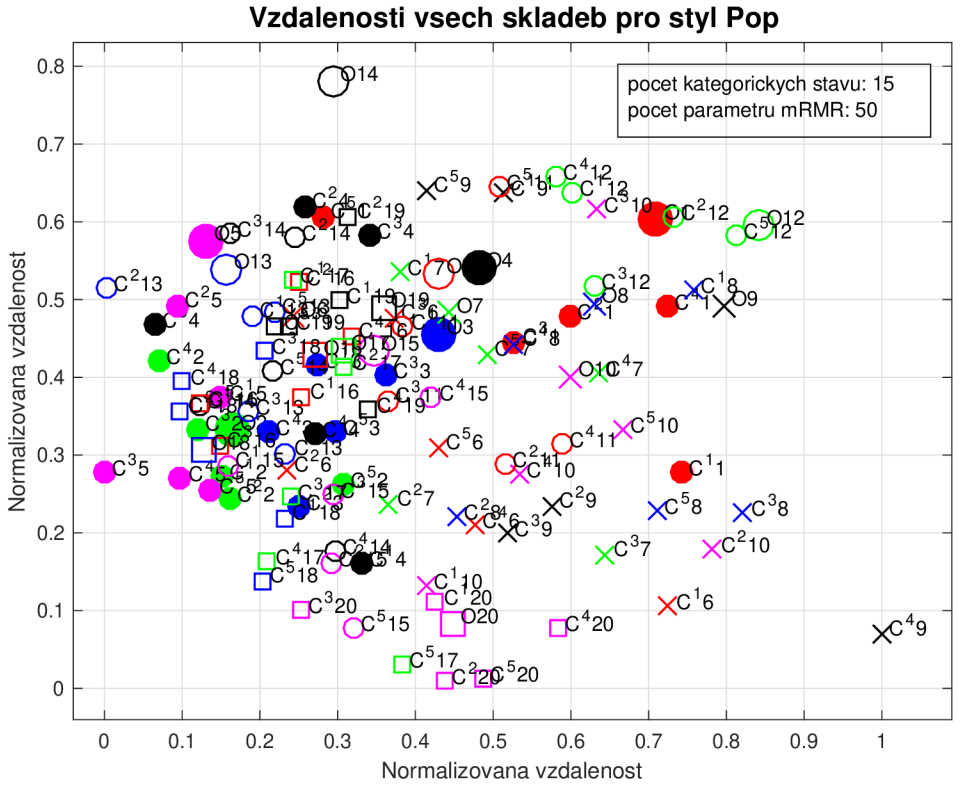
Hip Hop a Rap	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	40	60	50	55	55	55	**65**
Neuronová síť min. trén. E (Accuracy)	45	50	45	60	**75**	**70**	60
předpokládaná úspěšnost podle *Classification Learner*	**65**	**67**	**68**	**69**	70	69	62

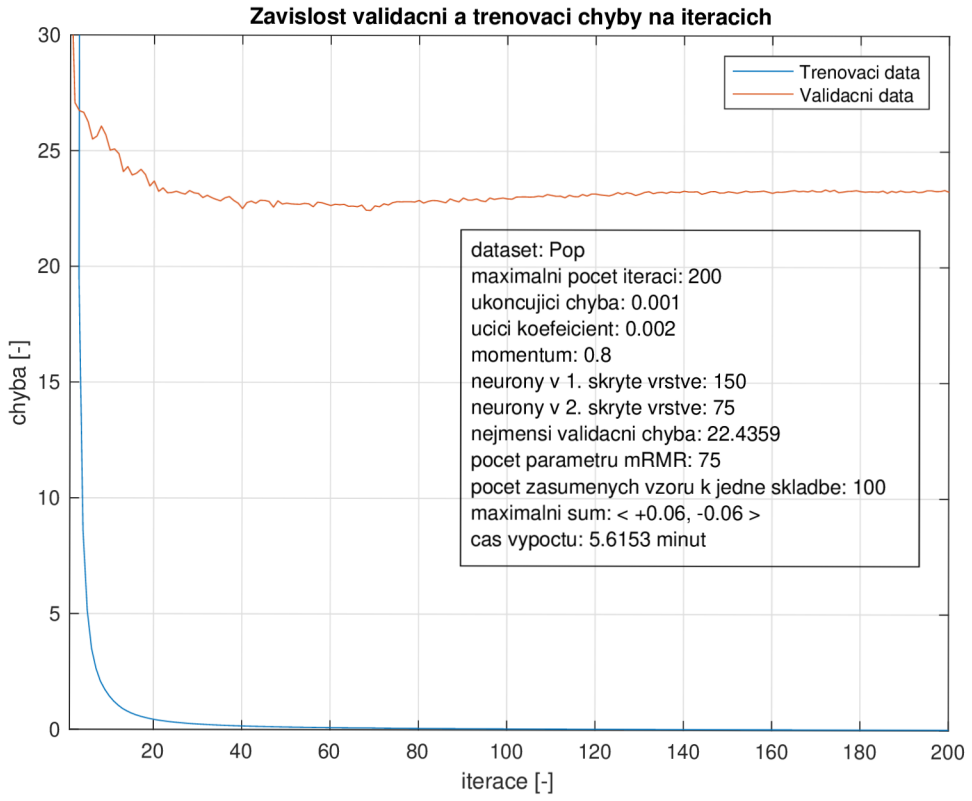
Jazz	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	50	65	65	55	75	65	65
Neuronová síť min. trén. E (Accuracy)	50	60	65	70	65	60	60
předpokládaná úspěšnost podle *Classification Learner*	**76**	**77**	**75**	**78**	**77**	**78**	**79**
Metal	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	55	80	75	80	**90**	75	80
Neuronová síť min. trén. E (Accuracy)	75	**85**	**90**	**85**	**90**	75	80
předpokládaná úspěšnost podle *Classification Learner*	**86**	83	84	83	85	**83**	**85**
Vážná hudba	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	70	85	**95**	**95**	**95**	**95**	**95**
Neuronová síť min. trén. E (Accuracy)	**95**	90	**95**	**95**	**95**	**95**	**95**
předpokládaná úspěšnost podle *Classification Learner*	93	**93**	93	93	92	94	93
Reggae	25	50	75	100	125	150	200
Neuronová síť min. valid. E (Accuracy)	20	40	45	40	35	35	50
Neuronová síť min. trén. E (Accuracy)	40	40	55	45	40	45	50
předpokládaná úspěšnost podle *Classification Learner*	**61**	**58**	**62**	**59**	**58**	**55**	**57**

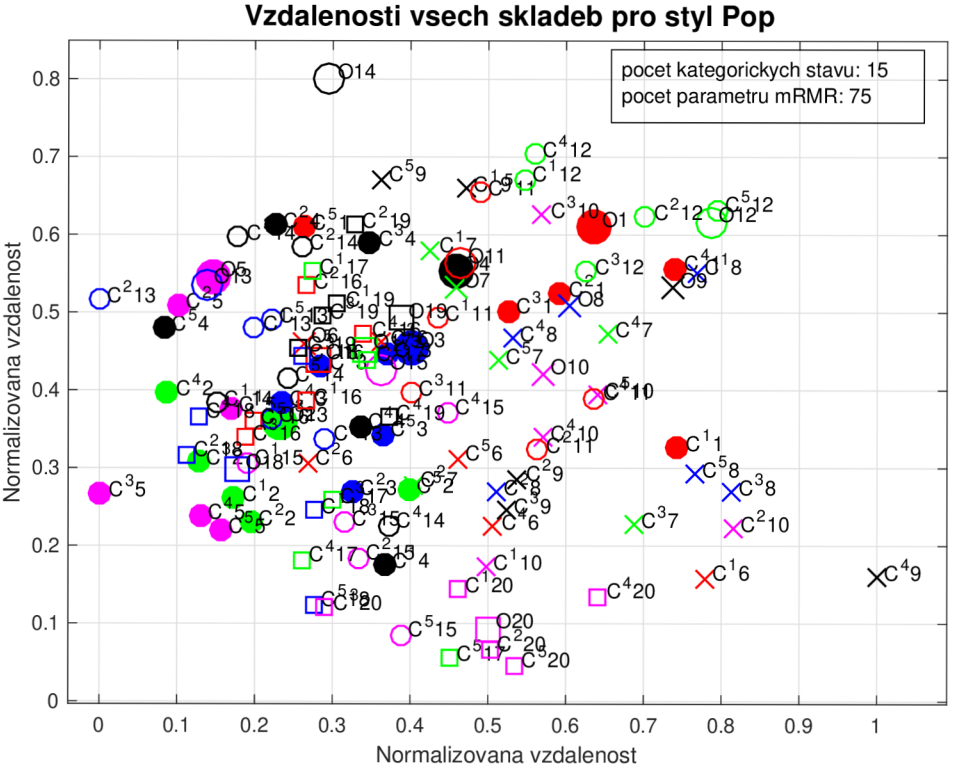

Zavislost validacni a trenovaci chyby na iteracich
dataset: Pop
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 25.3078
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne
skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 1.5879 minut
Trenovaci data
Validacni data
chyba [-]
iterace [-]

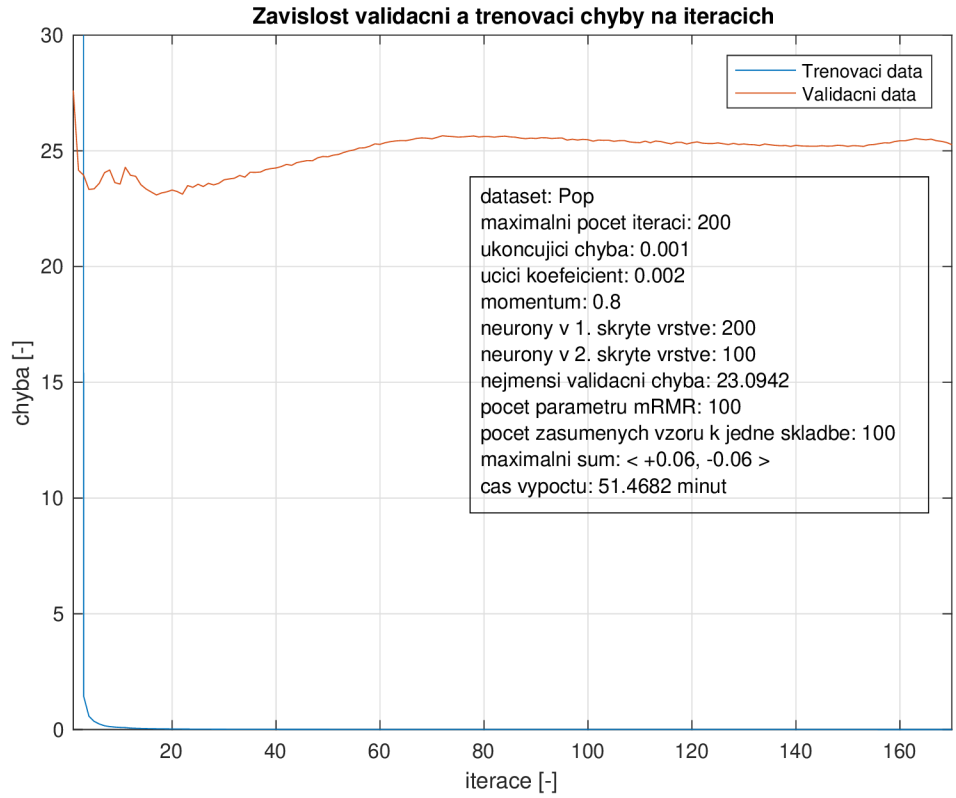

Vzdalenosti vsech skladeb pro styl Pop
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

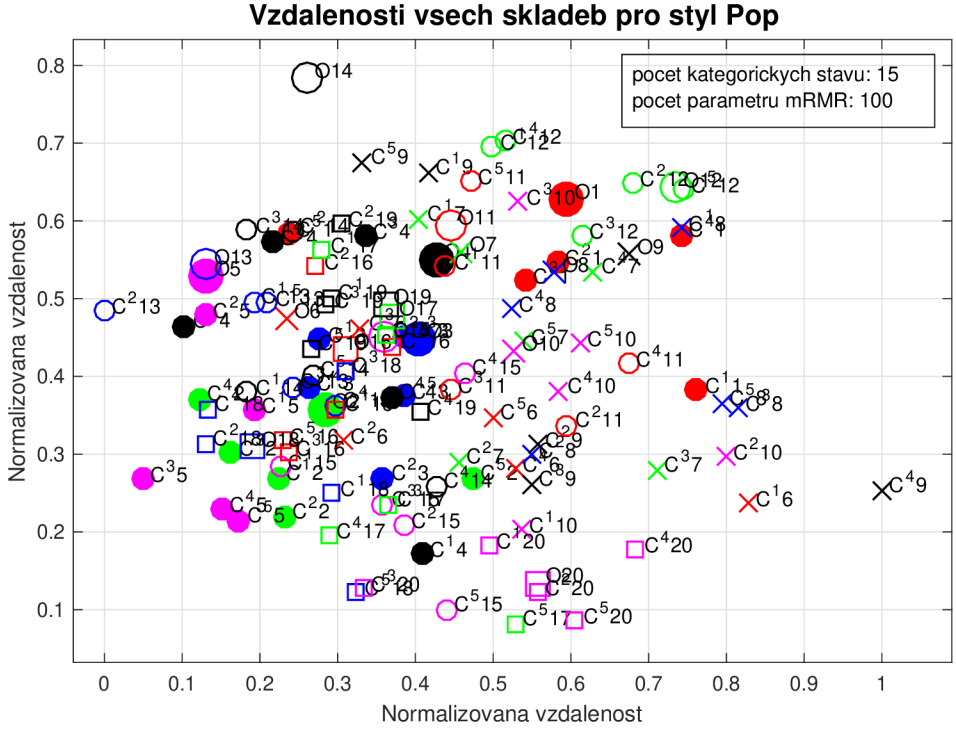

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Pop
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 29.9736
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 2.8576 minut
chyba [-]
iterace [-]

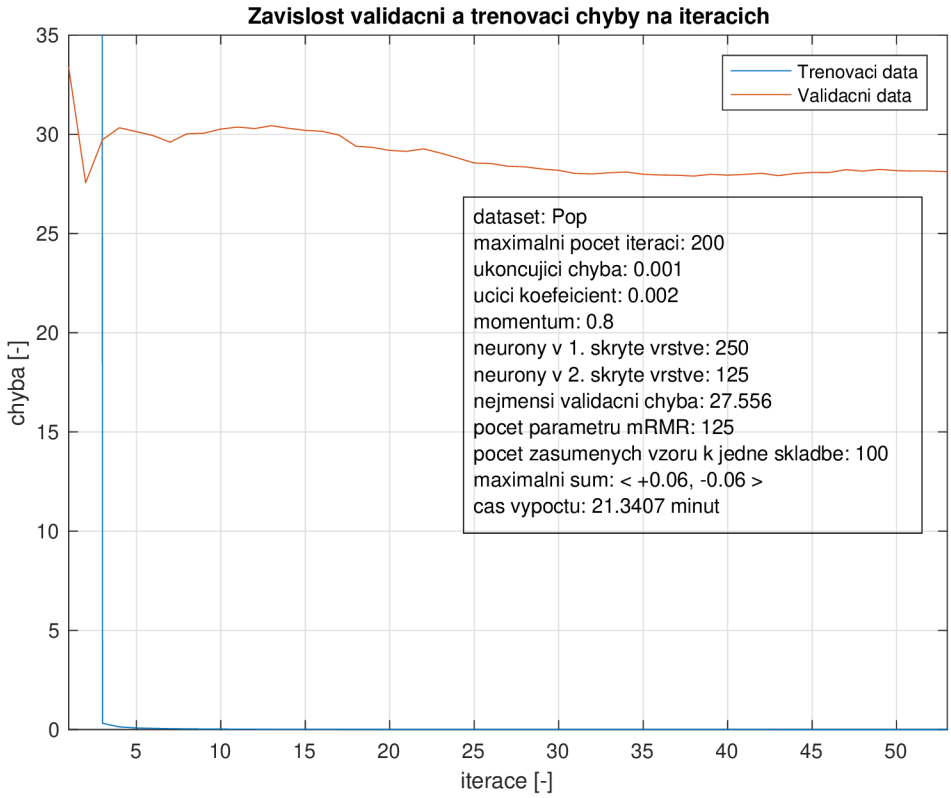

Vzdalenosti vsech skladeb pro styl Pop
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

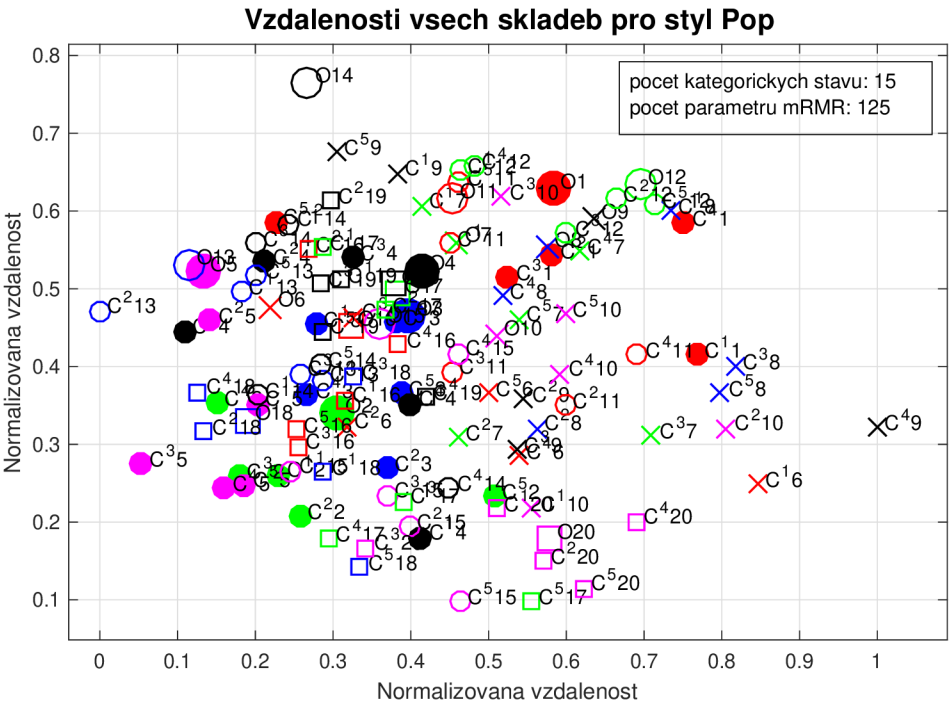

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Pop
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 22.4359
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 5.6153 minut
chyba [-]
iterace [-]

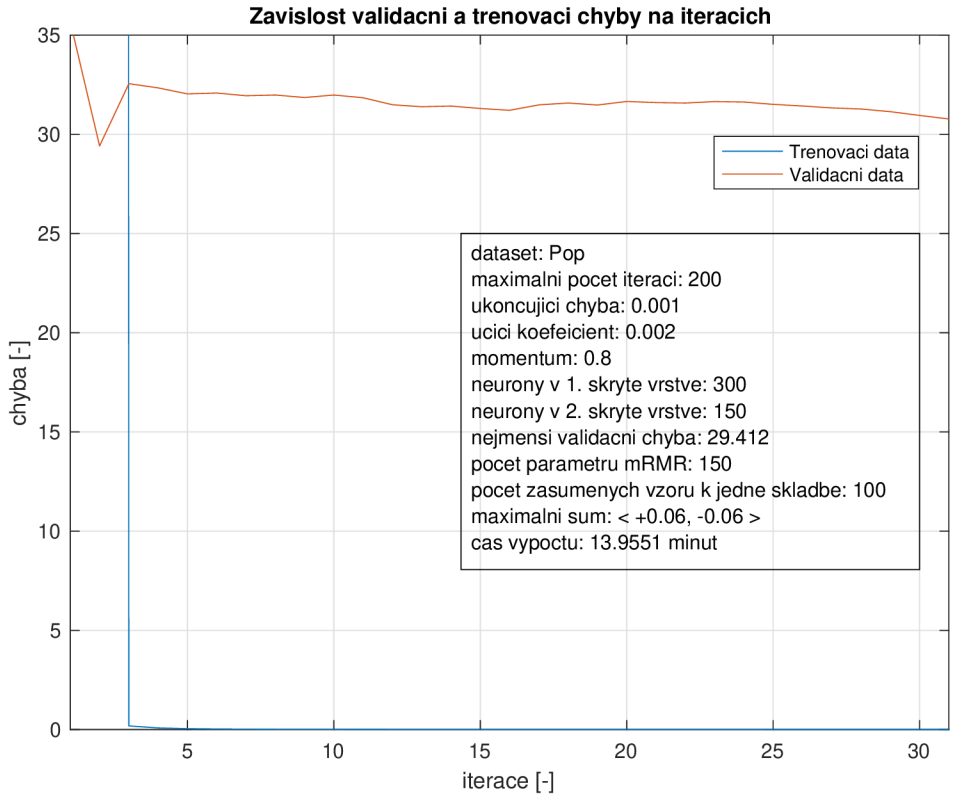

Vzdalenosti vsech skladeb pro styl Pop
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost

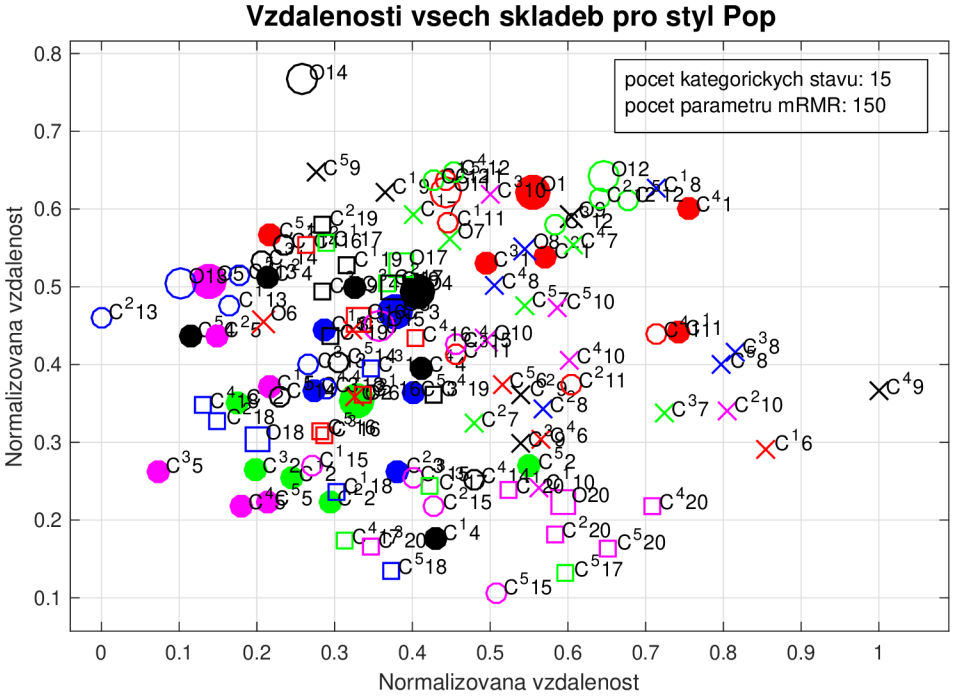

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Pop
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 23.0942
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 51.4682 minut
chyba [-]
iterace [-]

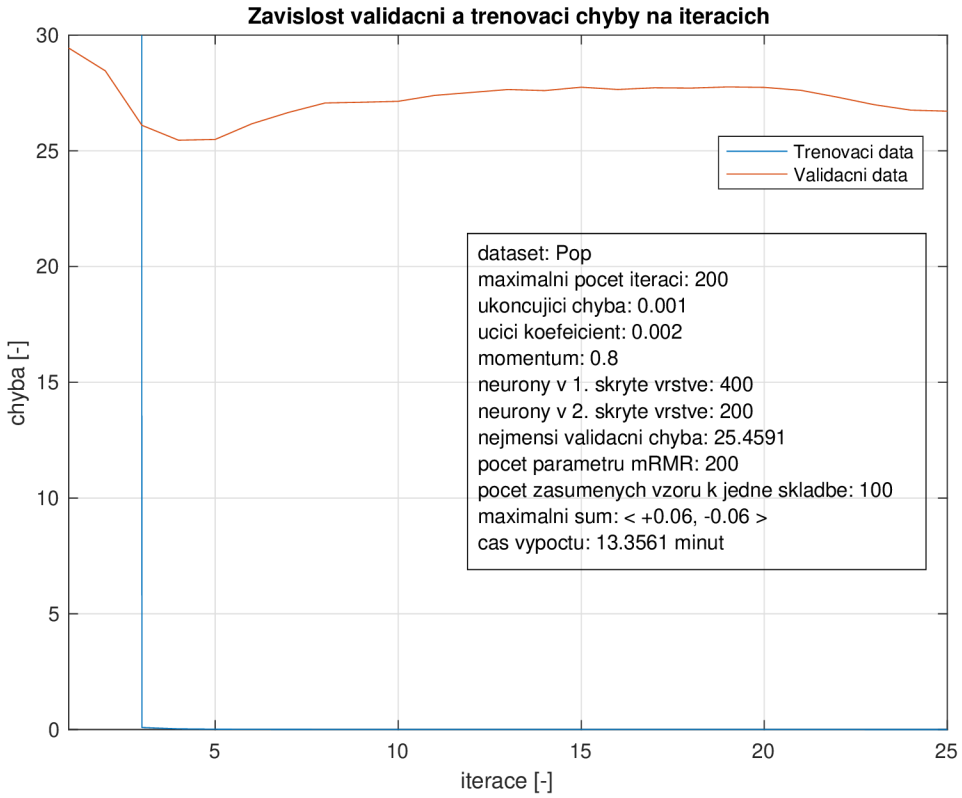

Vzdalenosti vsech skladeb pro styl Pop
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost

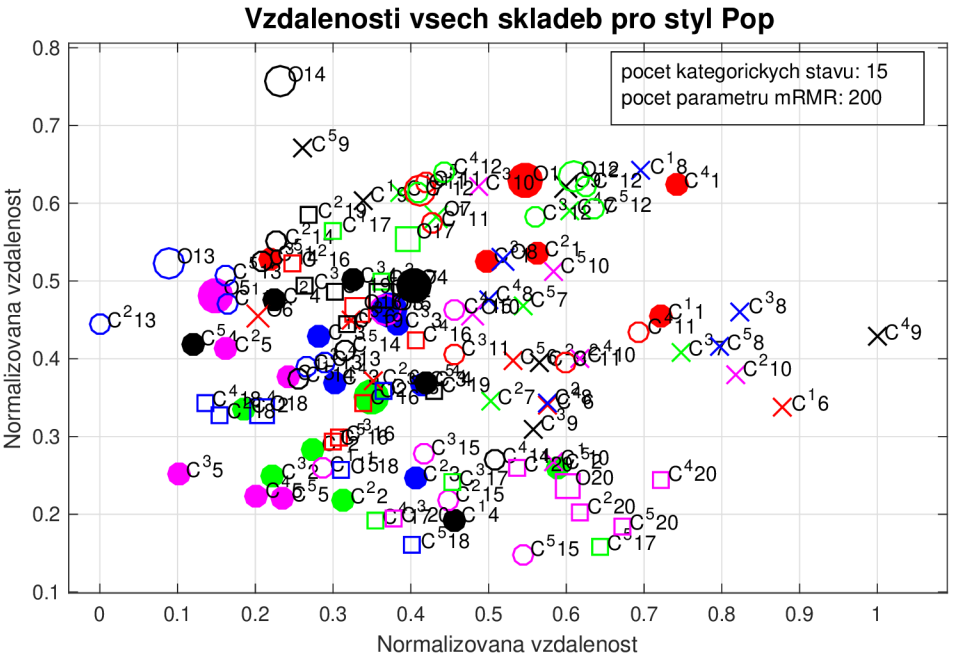

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Pop
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 27.556
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 21.3407 minut
chyba [-]
iterace [-]

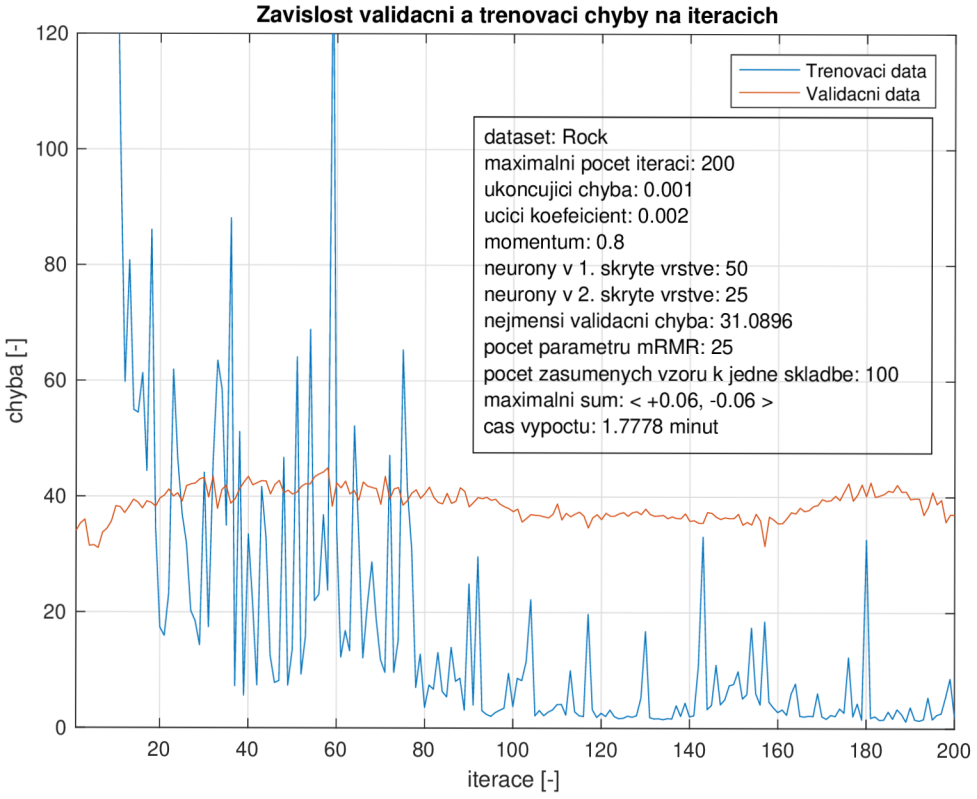

Vzdalenosti vsech skladeb pro styl Pop
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost

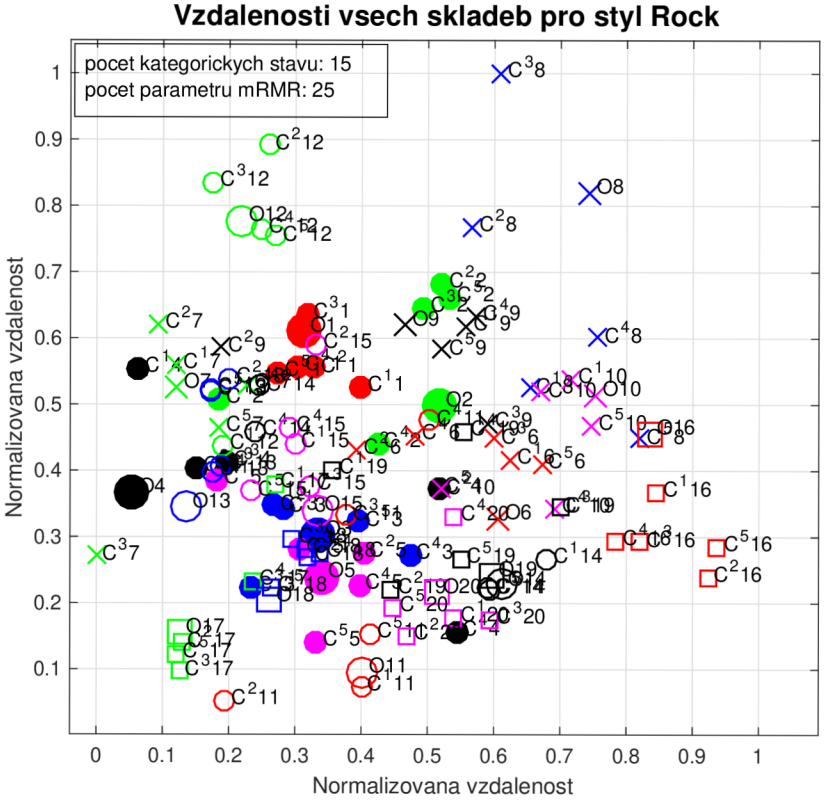

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Pop
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 300
neurony v 2. skryte vrstve: 150
nejmensi validacni chyba: 29.412
pocet parametru mRMR: 150
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 13.9551 minut
chyba [-]
iterace [-]

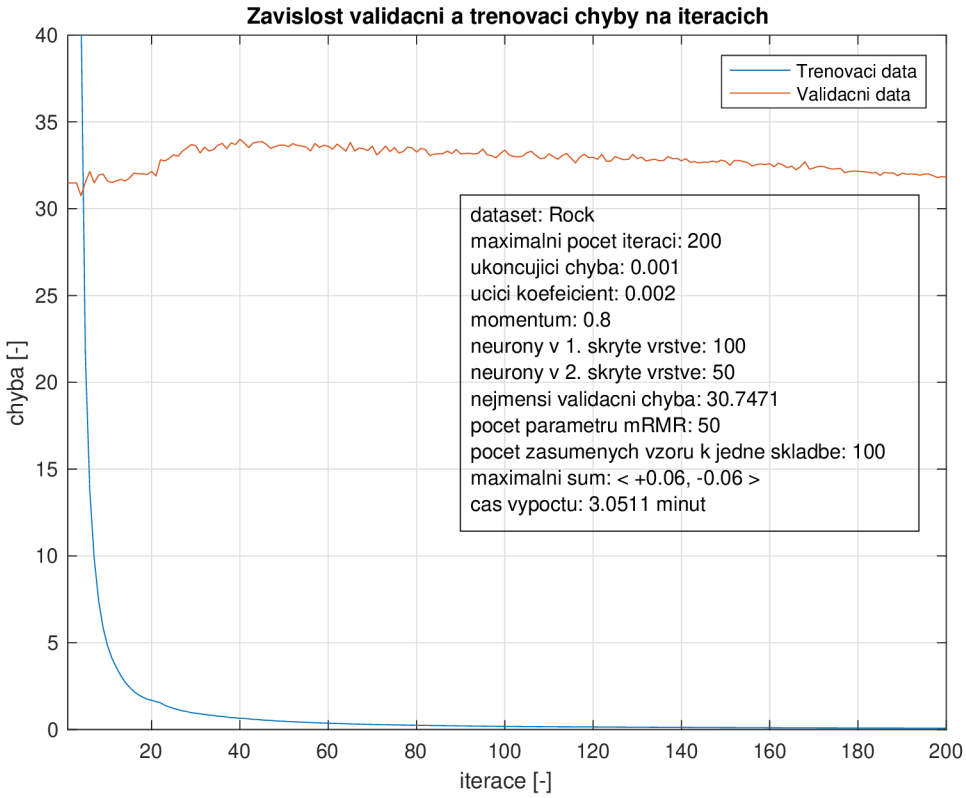

Vzdalenosti vsech skladeb pro styl Pop
pocet kategorickych stavu: 15
pocet parametru mRMR: 150
Normalizovana vzdalenost
Normalizovana vzdalenost

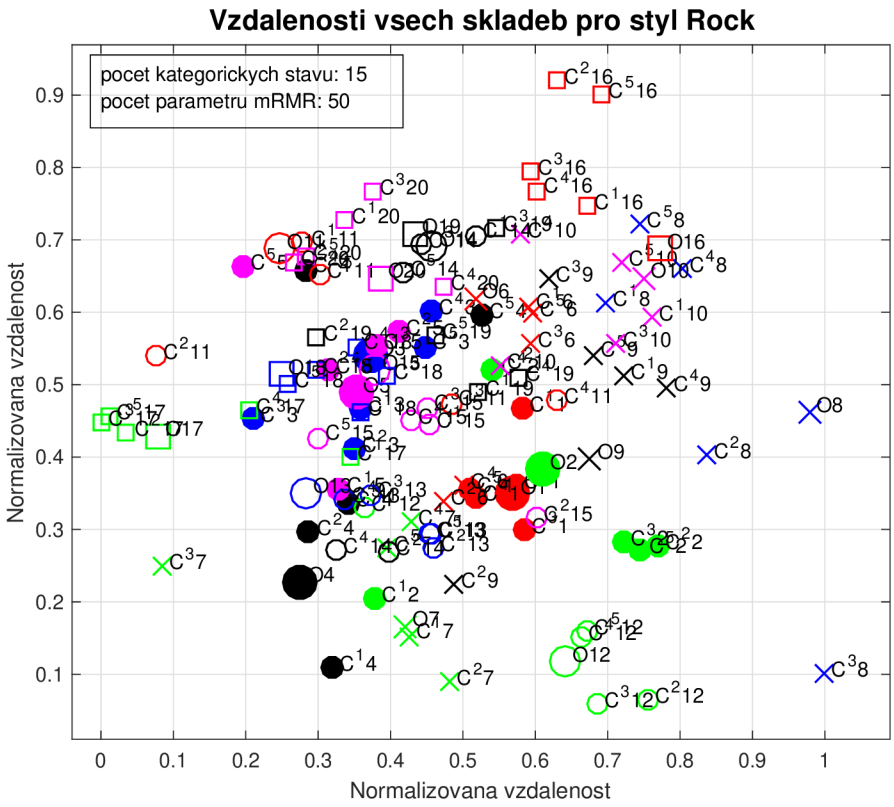

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Pop
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 25.4591
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 13.3561 minut
chyba [-]
iterace [-]

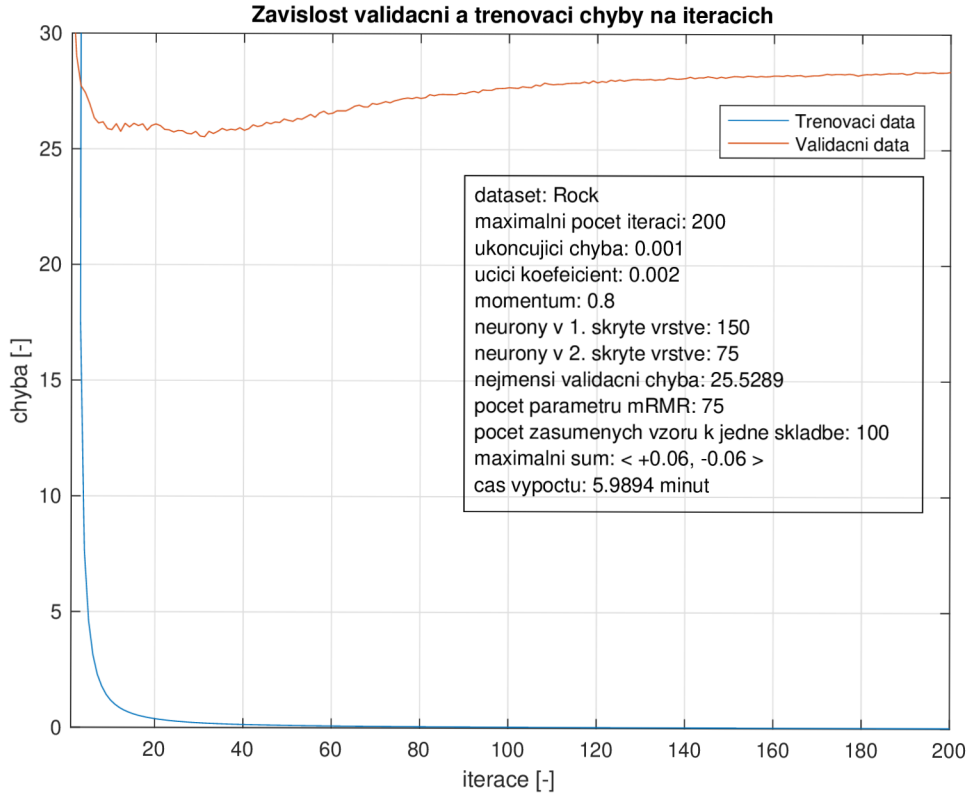

Vzdalenosti vsech skladeb pro styl Pop
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

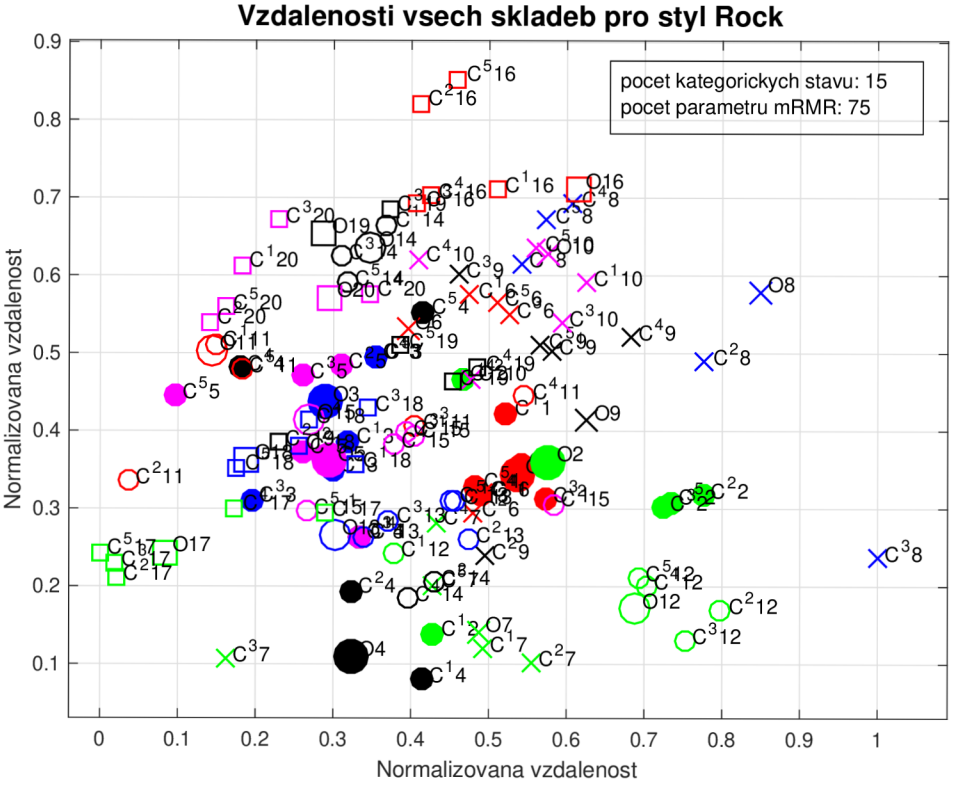

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Rock
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 31.0896
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 1.7778 minut
chyba [-]
iterace [-]

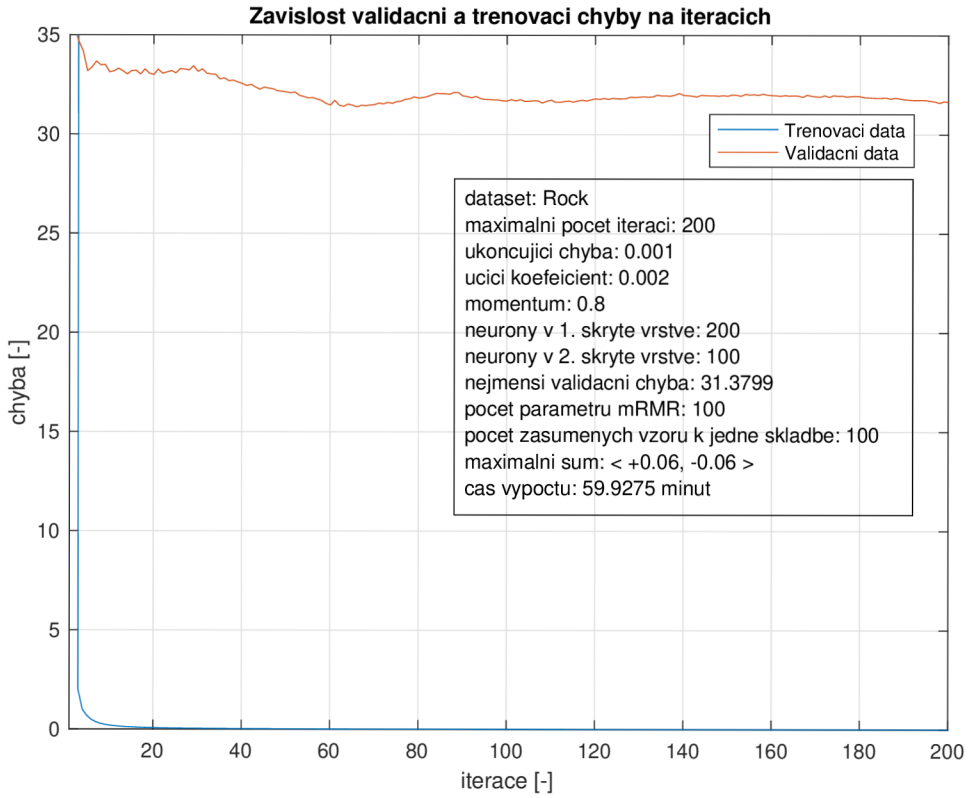

Vzdalenosti vsech skladeb pro styl Rock
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

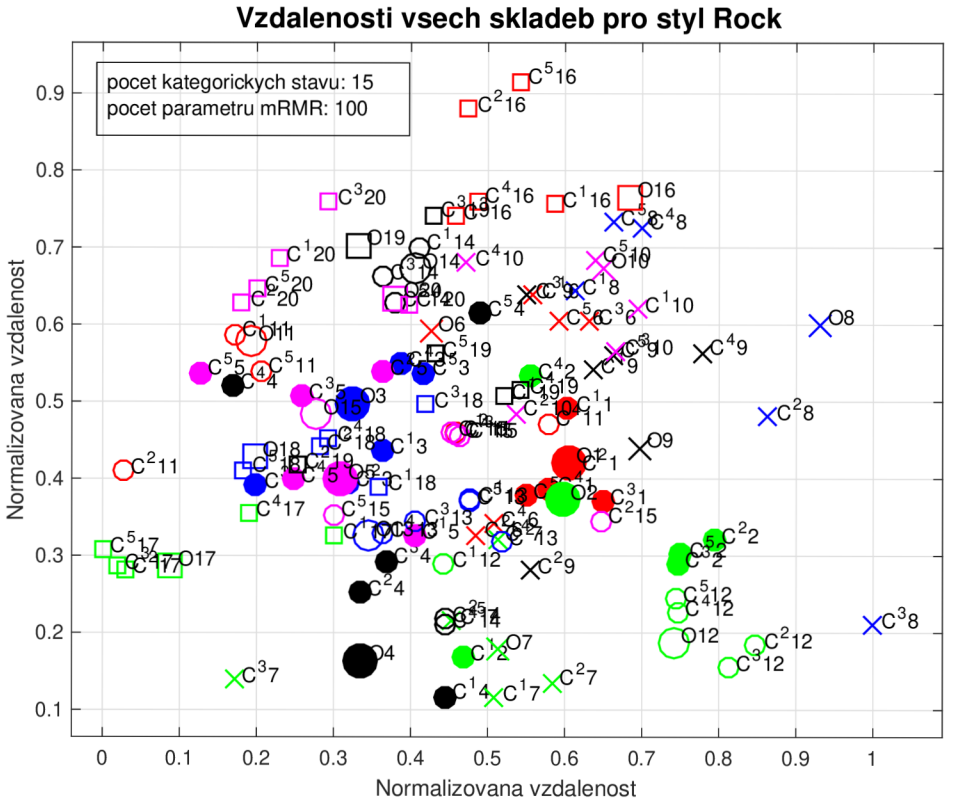

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Rock
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 30.7471
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 3.0511 minut
chyba [-]
iterace [-]

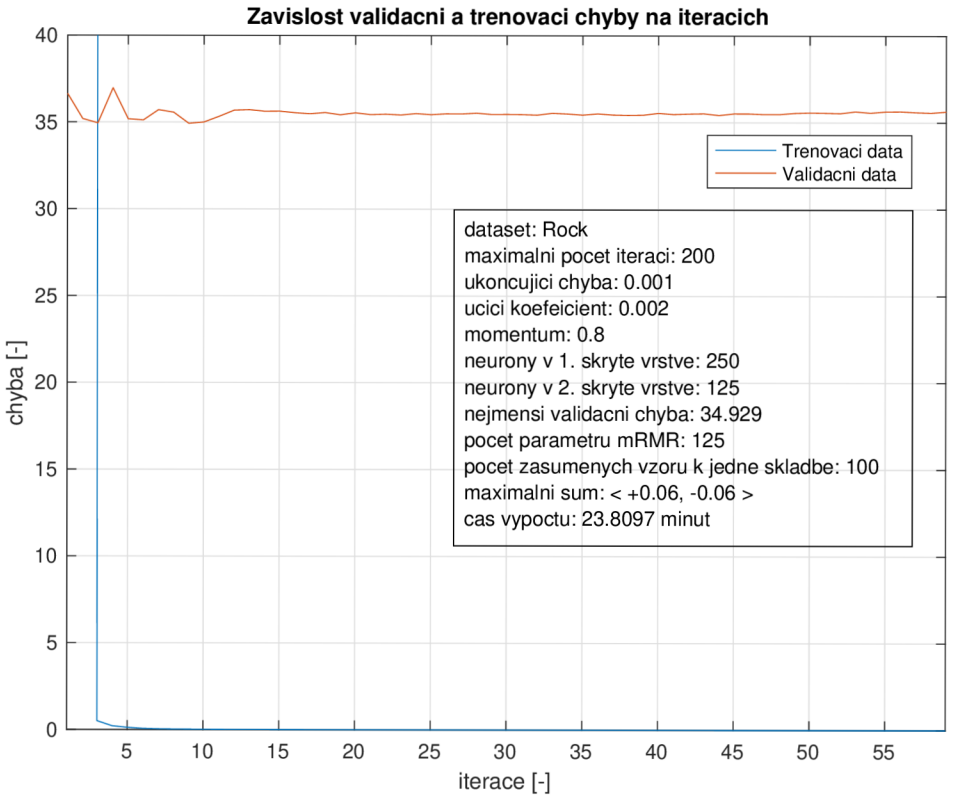

Vzdalenosti vsech skladeb pro styl Rock
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

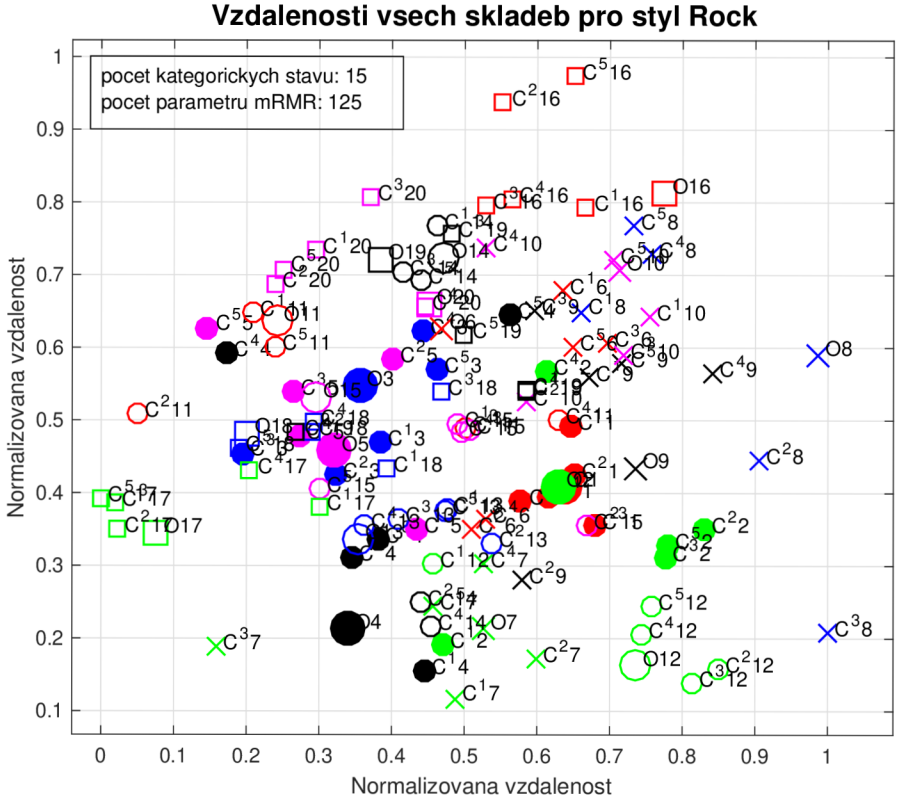

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Rock
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 25.5289
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 5.9894 minut
chyba [-]
iterace [-]

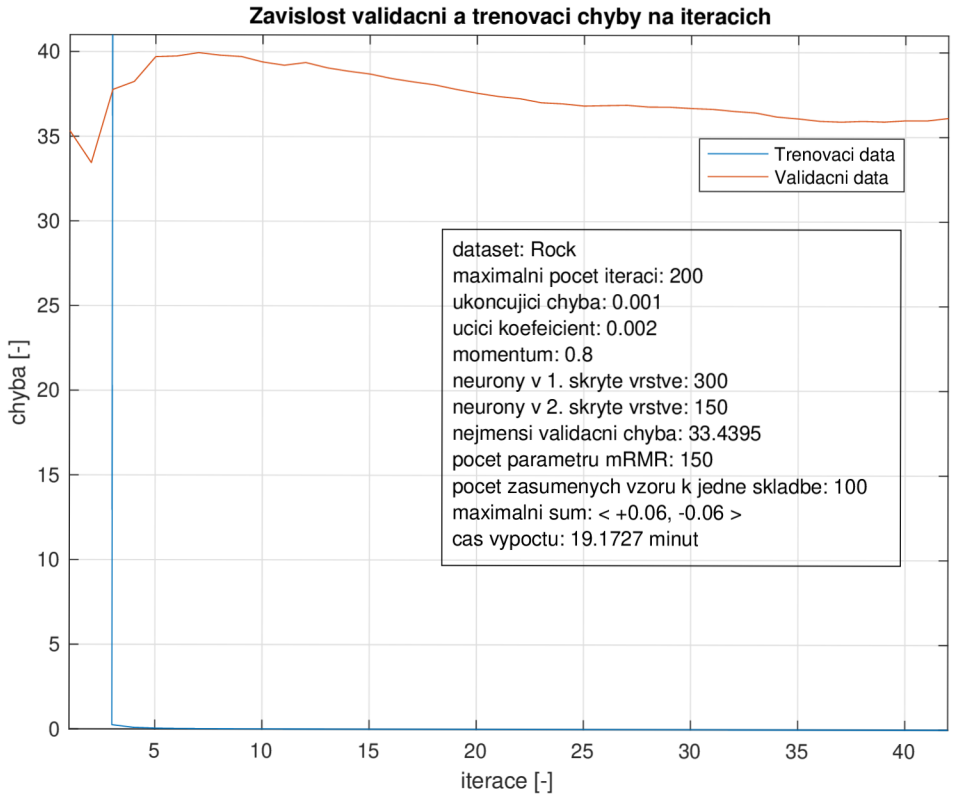

Vzdalenosti vsech skladeb pro styl Rock
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost

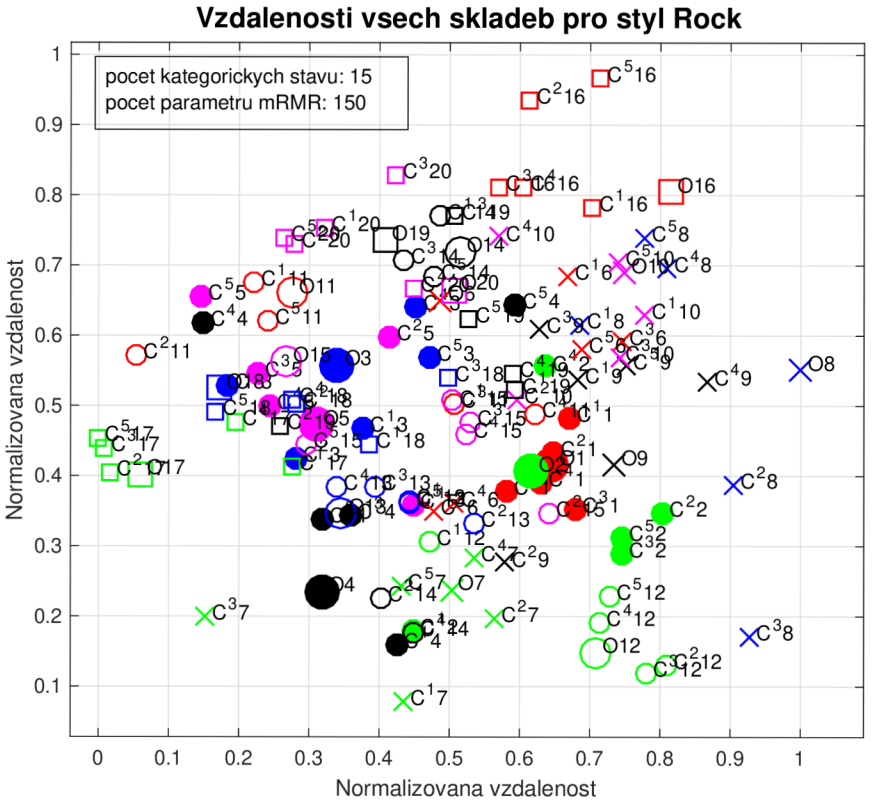

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Rock
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 31.3799
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 59.9275 minut
chyba [-]
iterace [-]

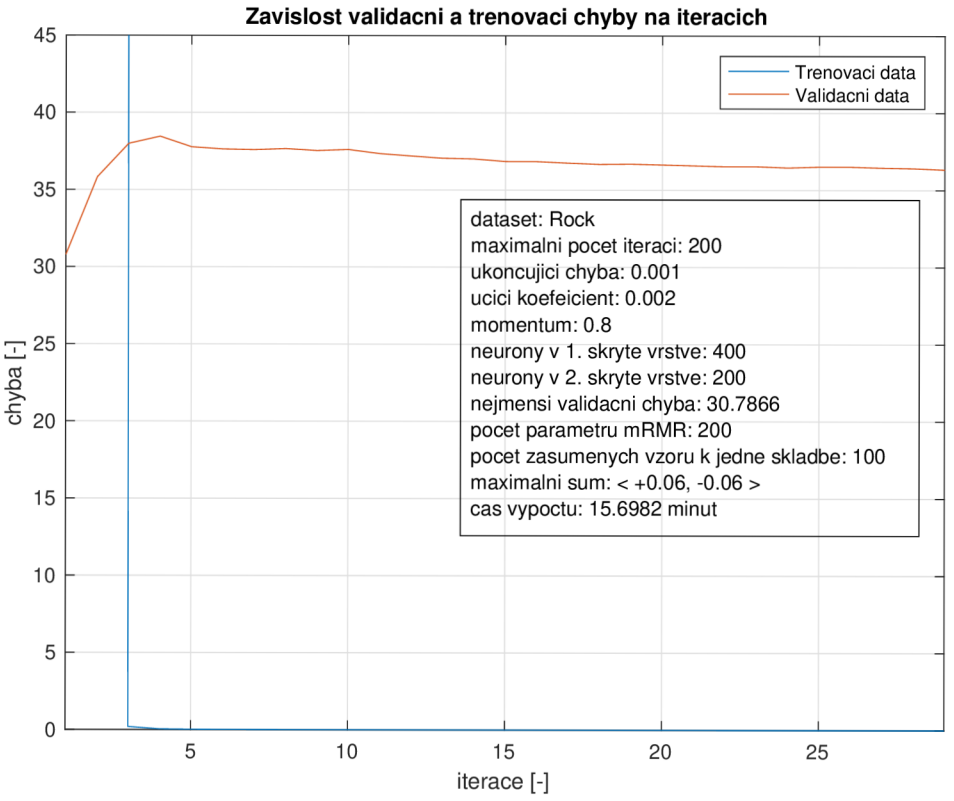

Vzdalenosti vsech skladeb pro styl Rock
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost

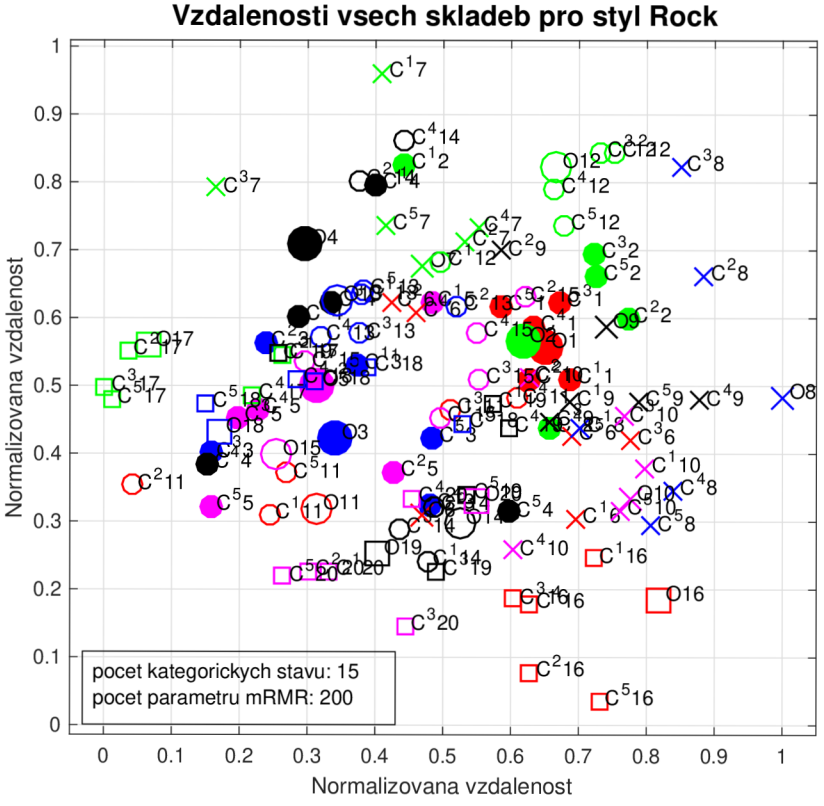

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Rock
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 34.929
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 23.8097 minut
chyba [-]
iterace [-]

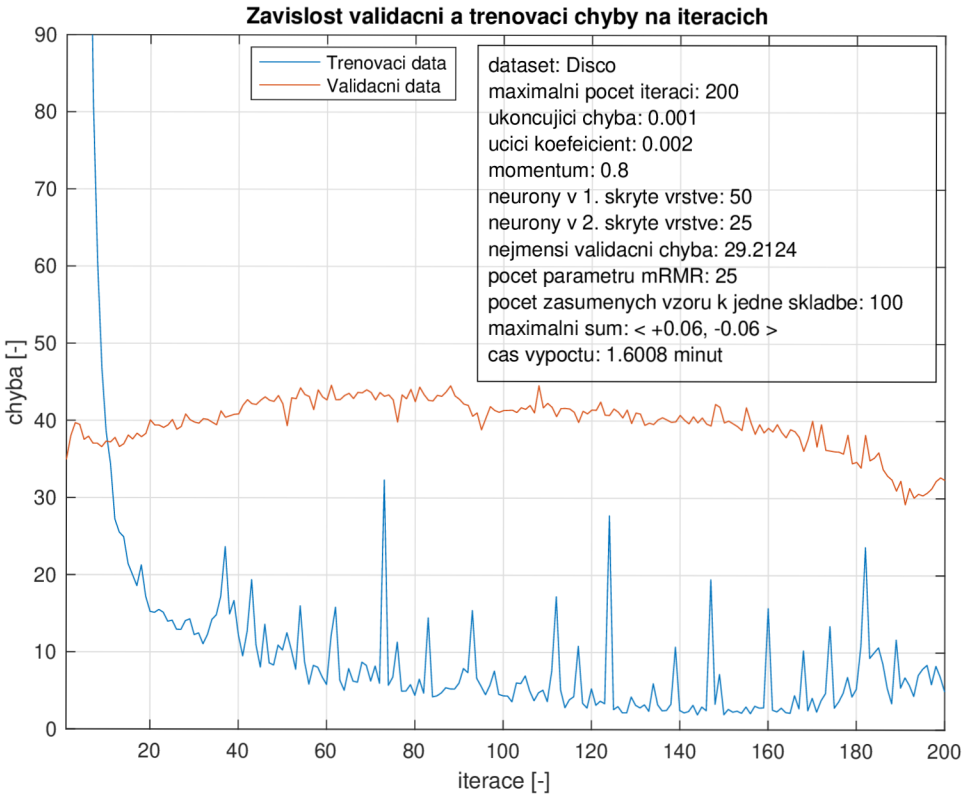

Vzdalenosti vsech skladeb pro styl Rock
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost

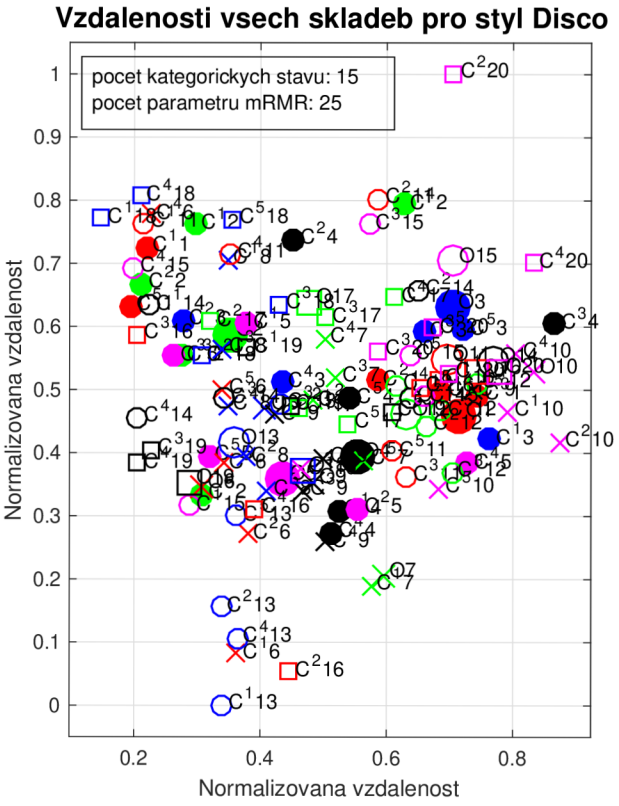

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Rock
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 300
neurony v 2. skryte vrstve: 150
nejmensi validacni chyba: 33.4395
pocet parametru mRMR: 150
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 19.1727 minut
chyba [-]
iterace [-]

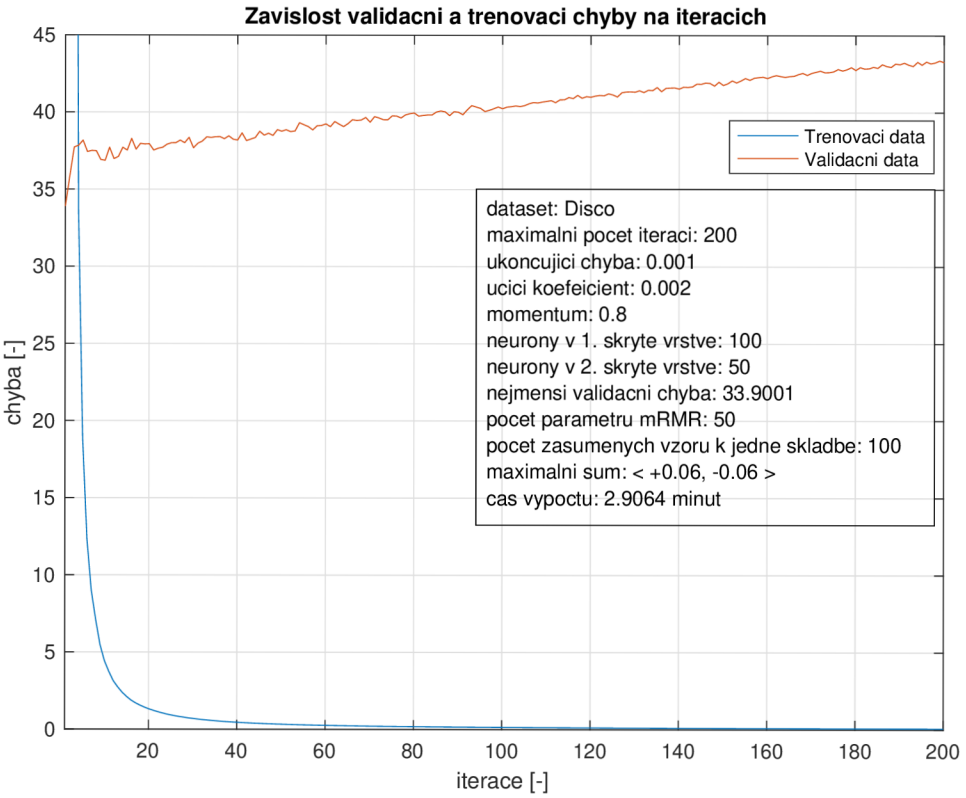

Vzdalenosti vsech skladeb pro styl Rock
pocet kategorickych stavu: 15
pocet parametru mRMR: 150
Normalizovana vzdalenost
Normalizovana vzdalenost

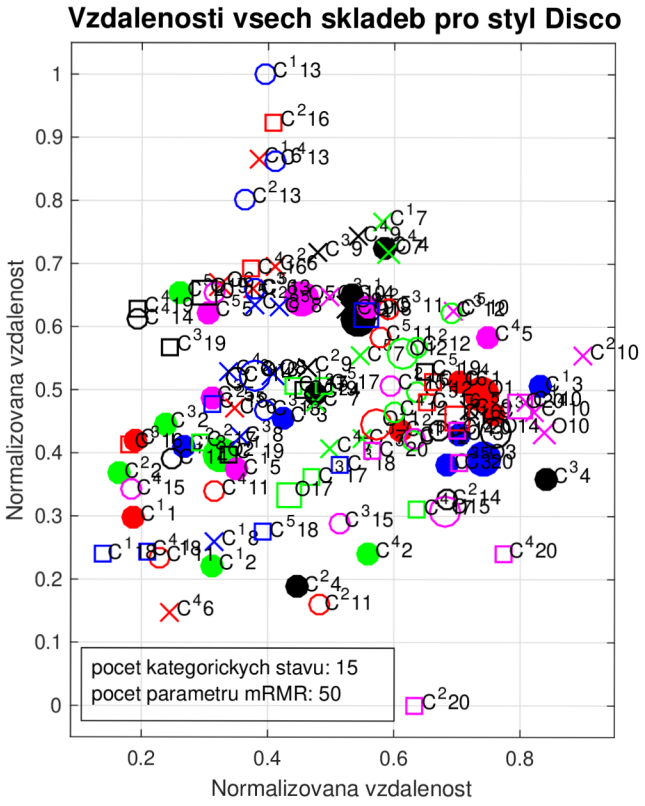

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Rock
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 30.7866
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 15.6982 minut
chyba [-]
iterace [-]

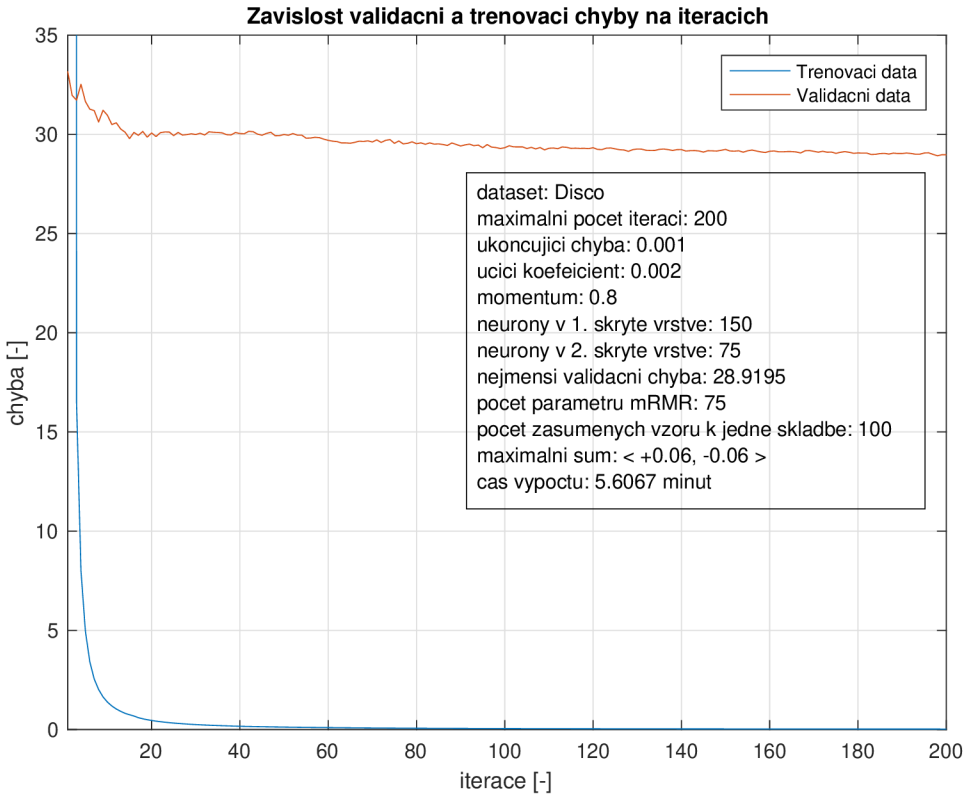

Vzdalenosti vsech skladeb pro styl Rock
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

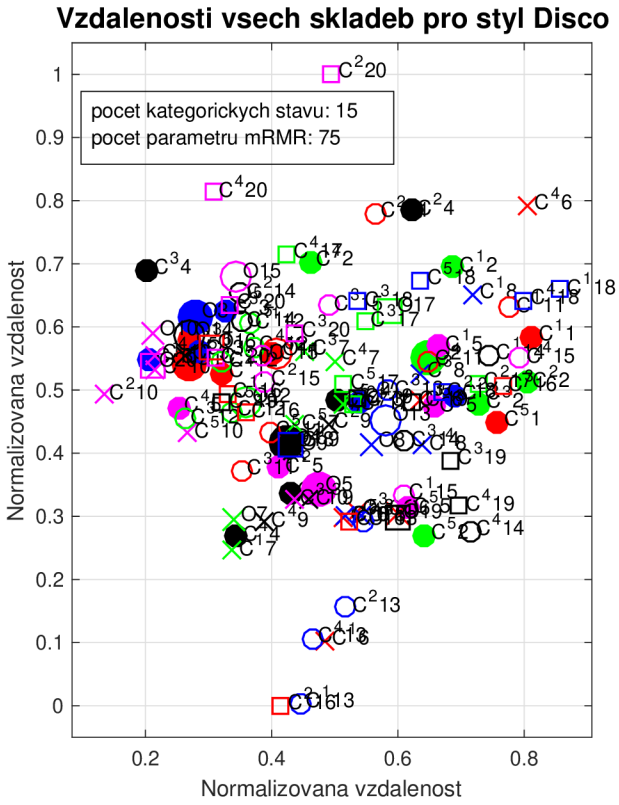

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Disco
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 29.2124
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 1.6008 minut
chyba [-]
iterace [-]

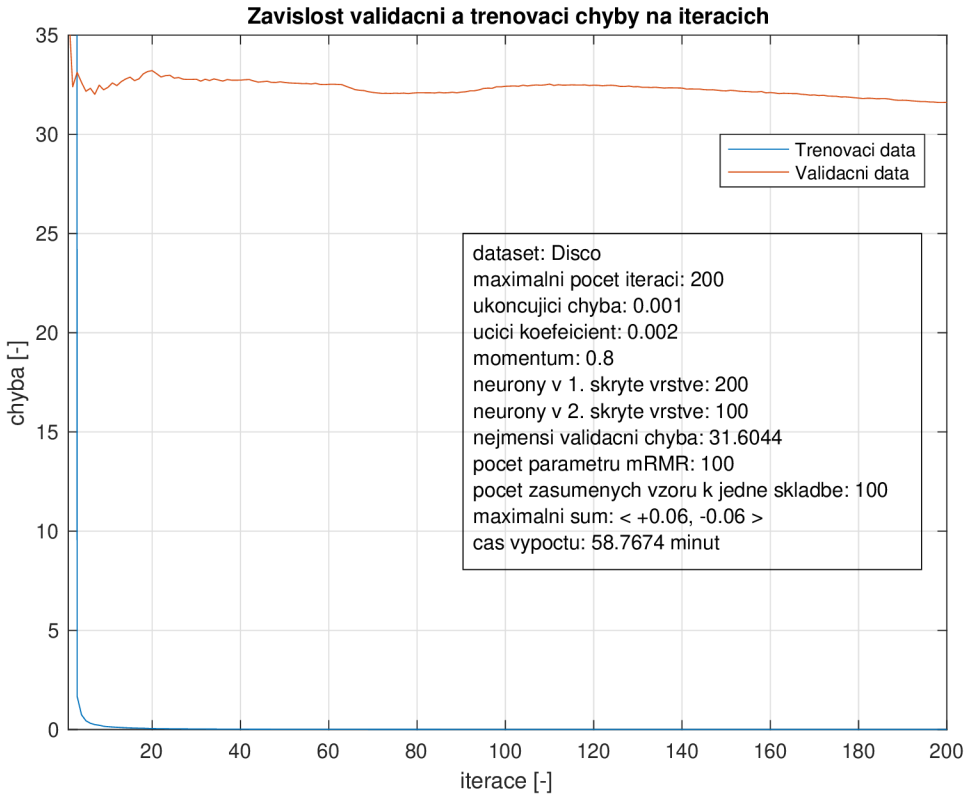

Vzdalenosti vsech skladeb pro styl Disco
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

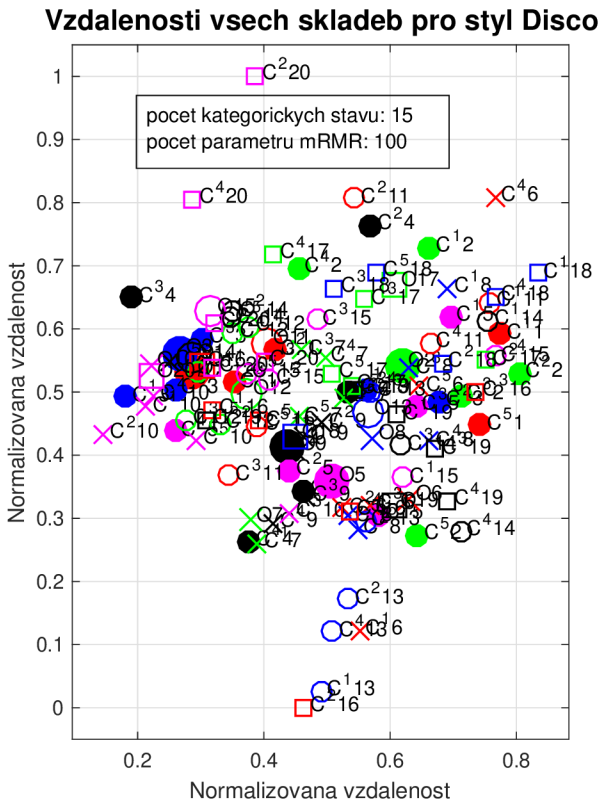

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Disco
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 33.9001
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 2.9064 minut
chyba [-]
iterace [-]

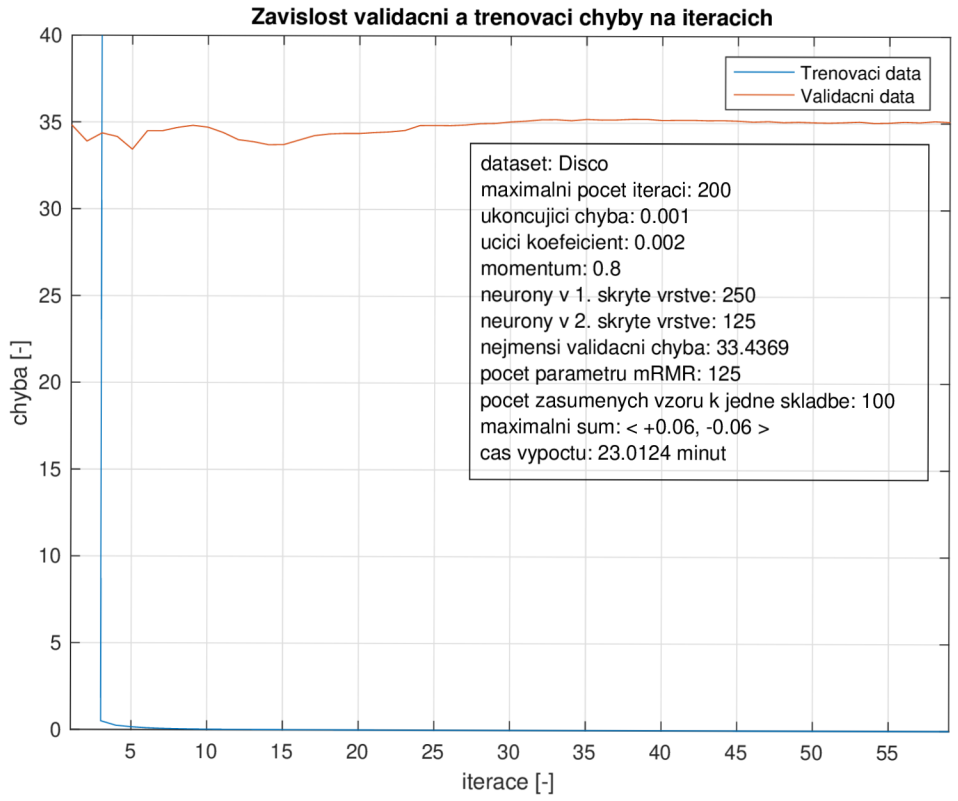

Vzdalenosti vsech skladeb pro styl Disco
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

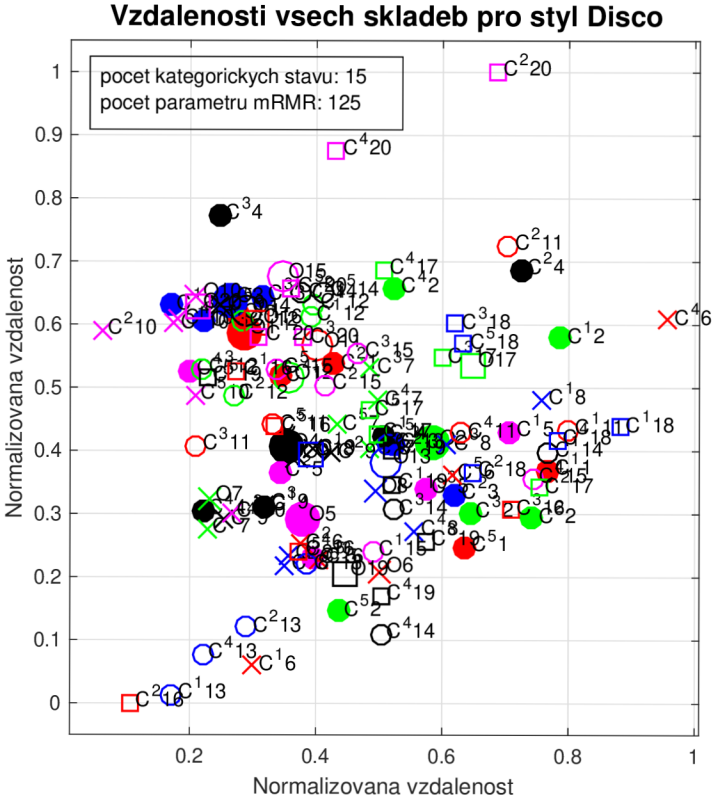

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Disco
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 28.9195
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 5.6067 minut
chyba [-]
iterace [-]

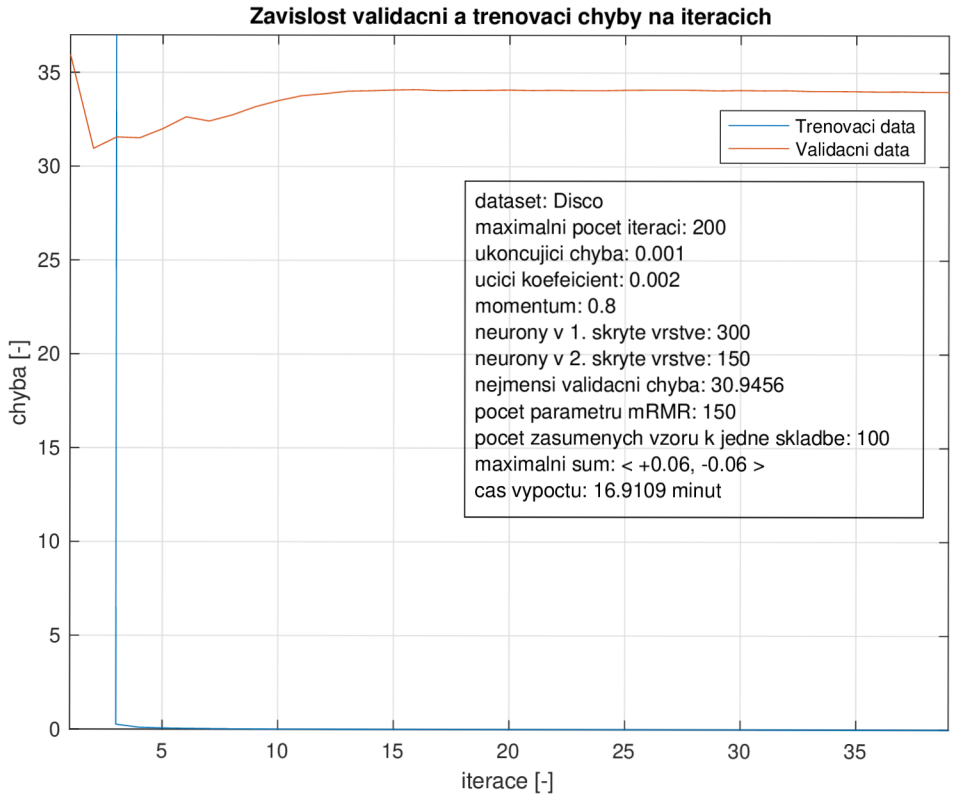

Vzdalenosti vsech skladeb pro styl Disco
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost

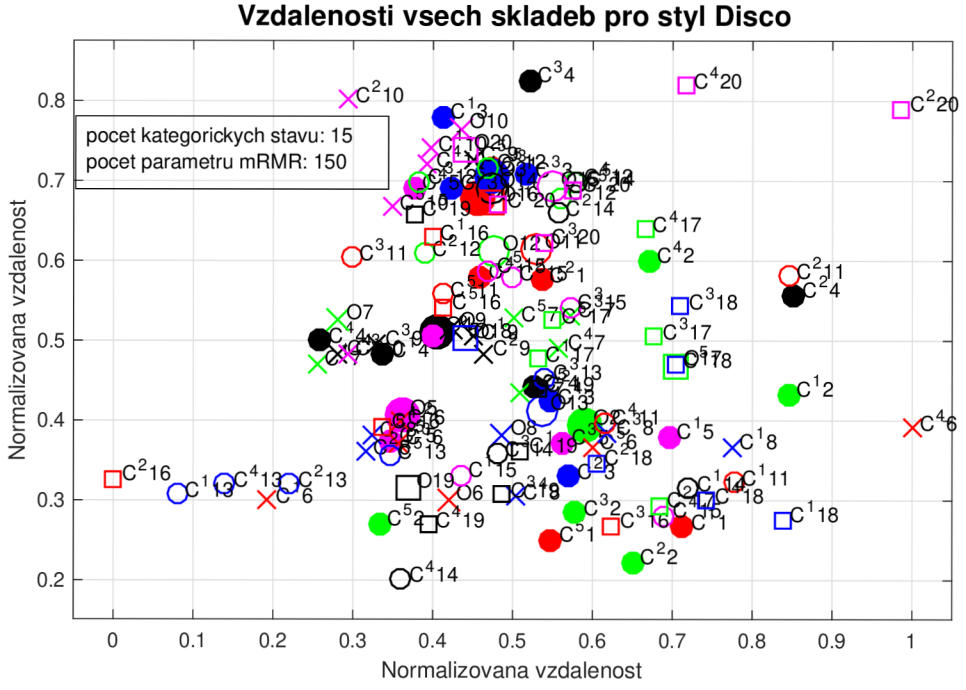

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Disco
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 31.6044
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 58.7674 minut
chyba [-]
iterace [-]

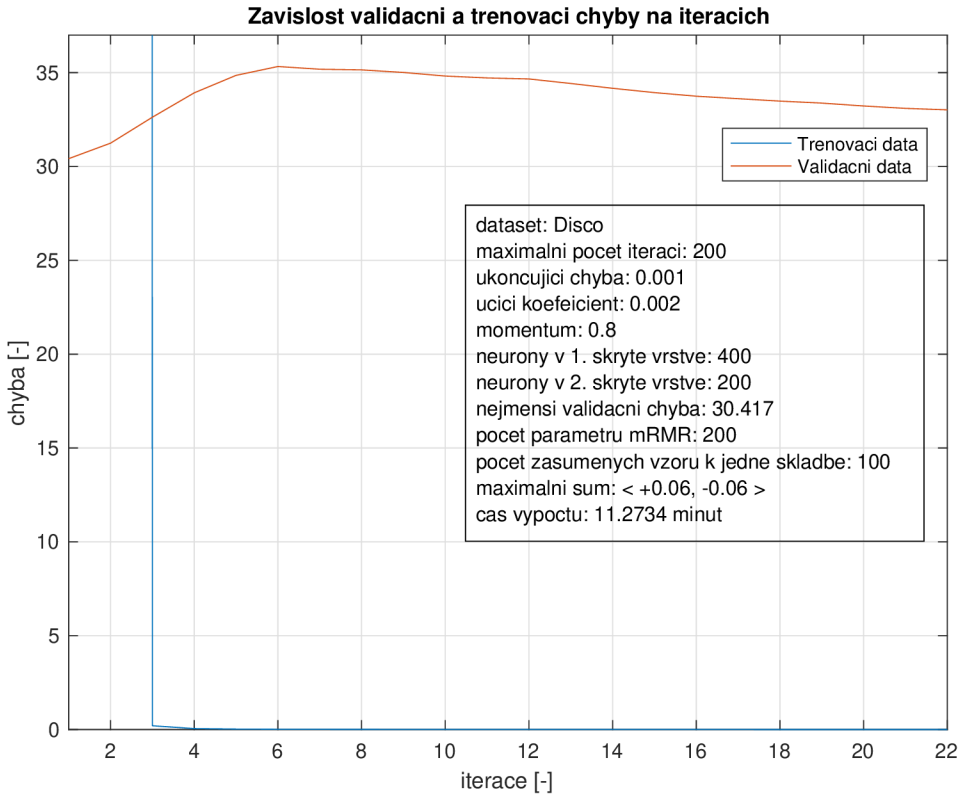

Vzdalenosti vsech skladeb pro styl Disco
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost

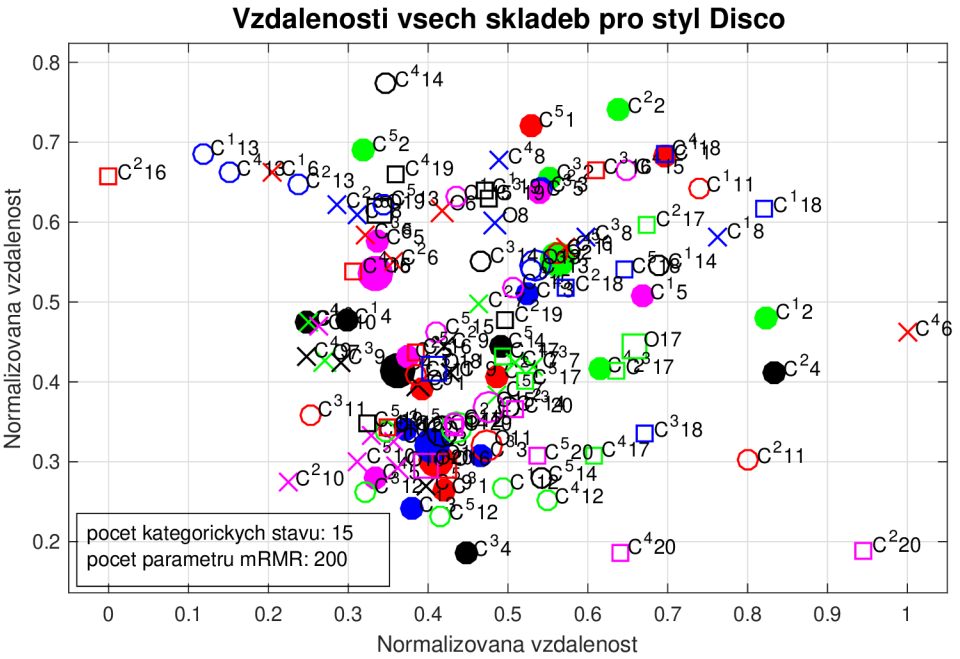

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Disco
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 33.4369
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 23.0124 minut
chyba [-]
iterace [-]

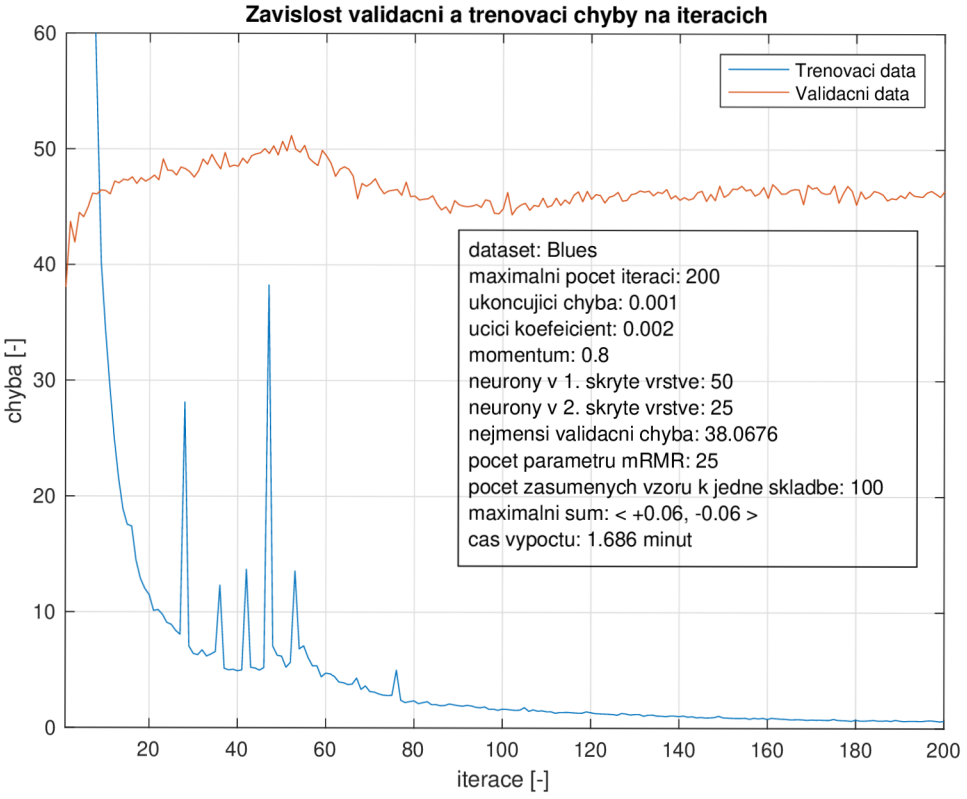

Vzdalenosti vsech skladeb pro styl Disco
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost

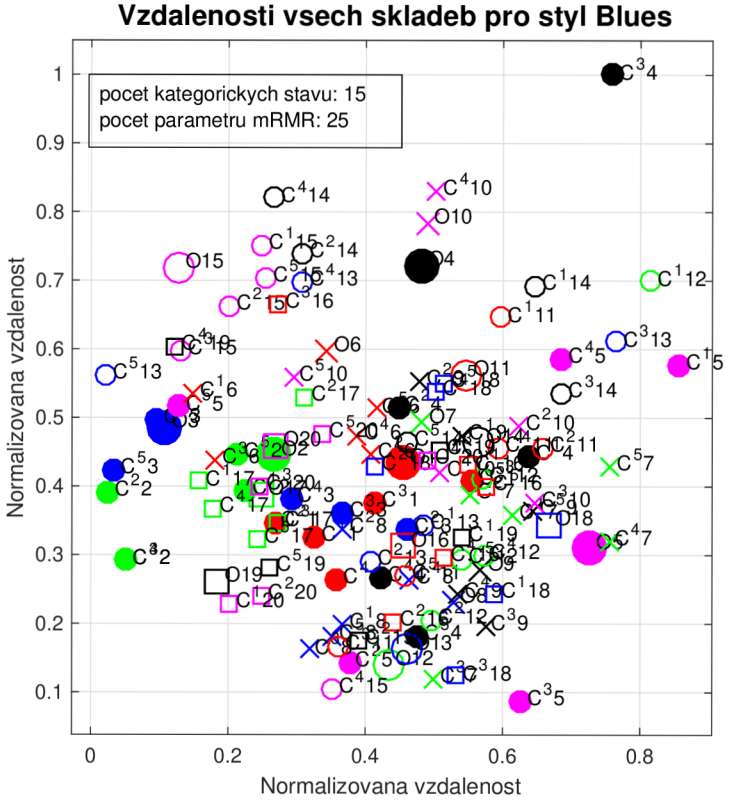

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Disco
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 300
neurony v 2. skryte vrstve: 150
nejmensi validacni chyba: 30.9456
pocet parametru mRMR: 150
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 16.9109 minut
chyba [-]
iterace [-]

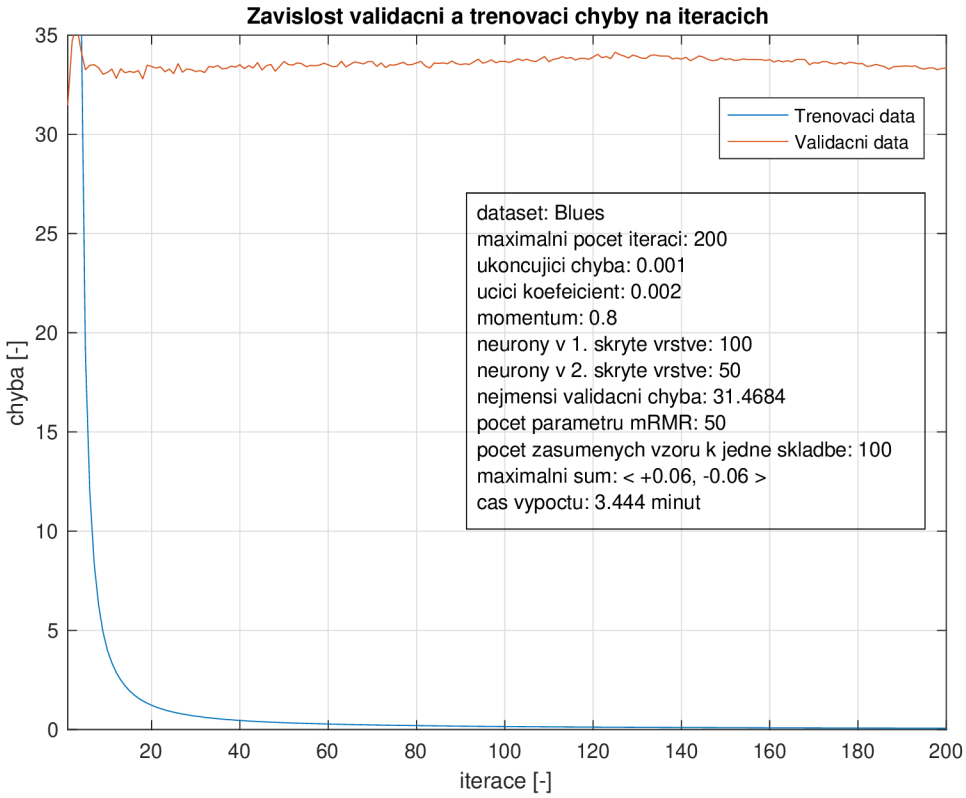

Vzdalenosti vsech skladeb pro styl Disco
pocet kategorickych stavu: 15
pocet parametru mRMR: 150
Normalizovana vzdalenost
Normalizovana vzdalenost

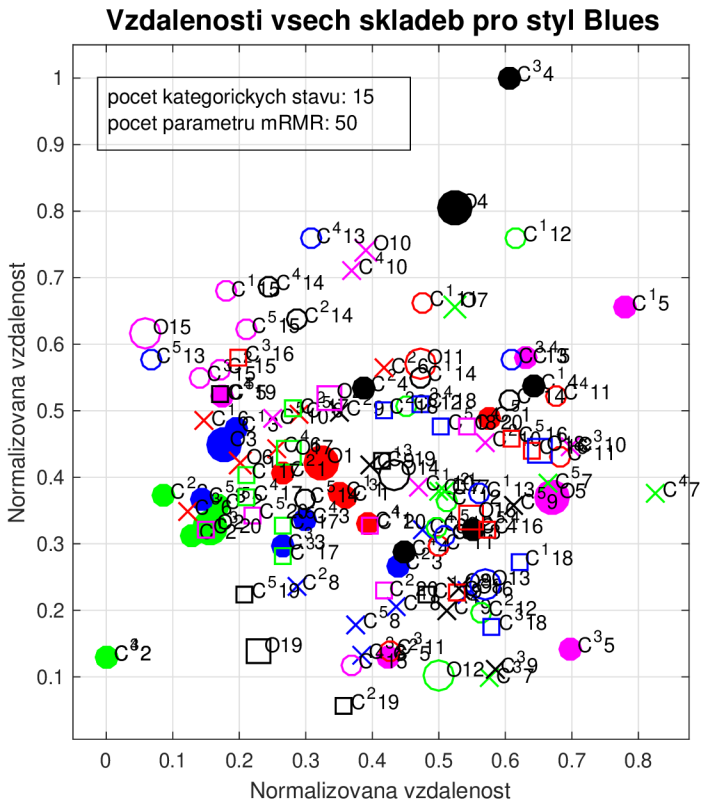

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Disco
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 30.417
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 11.2734 minut
chyba [-]
iterace [-]

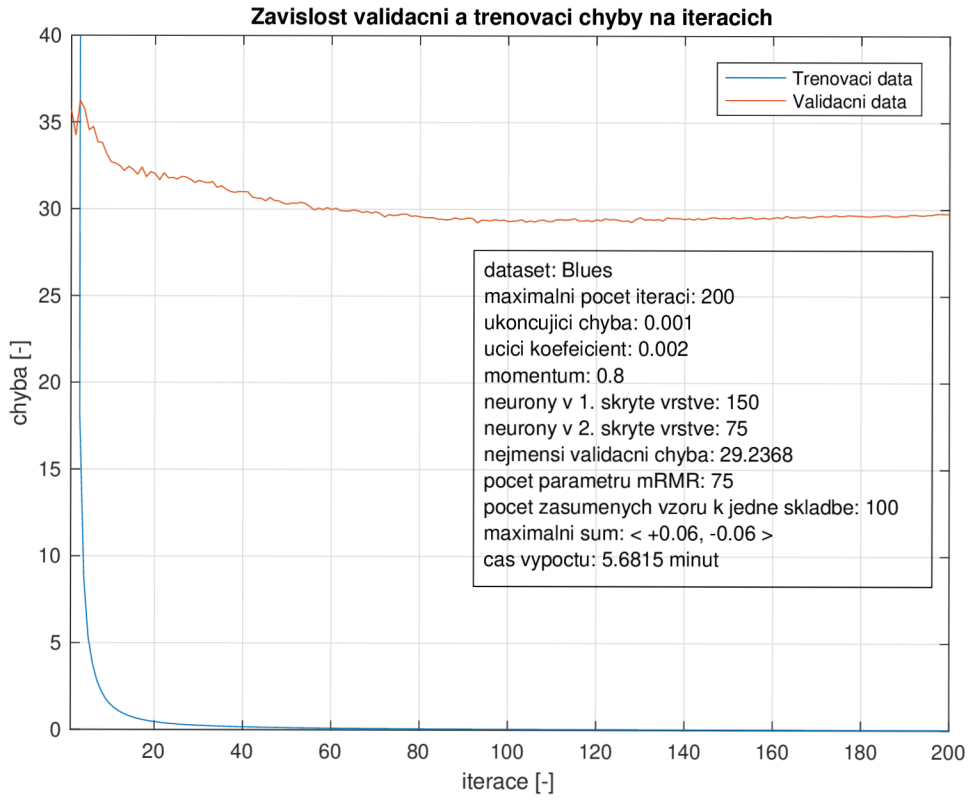

Vzdalenosti vsech skladeb pro styl Disco
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

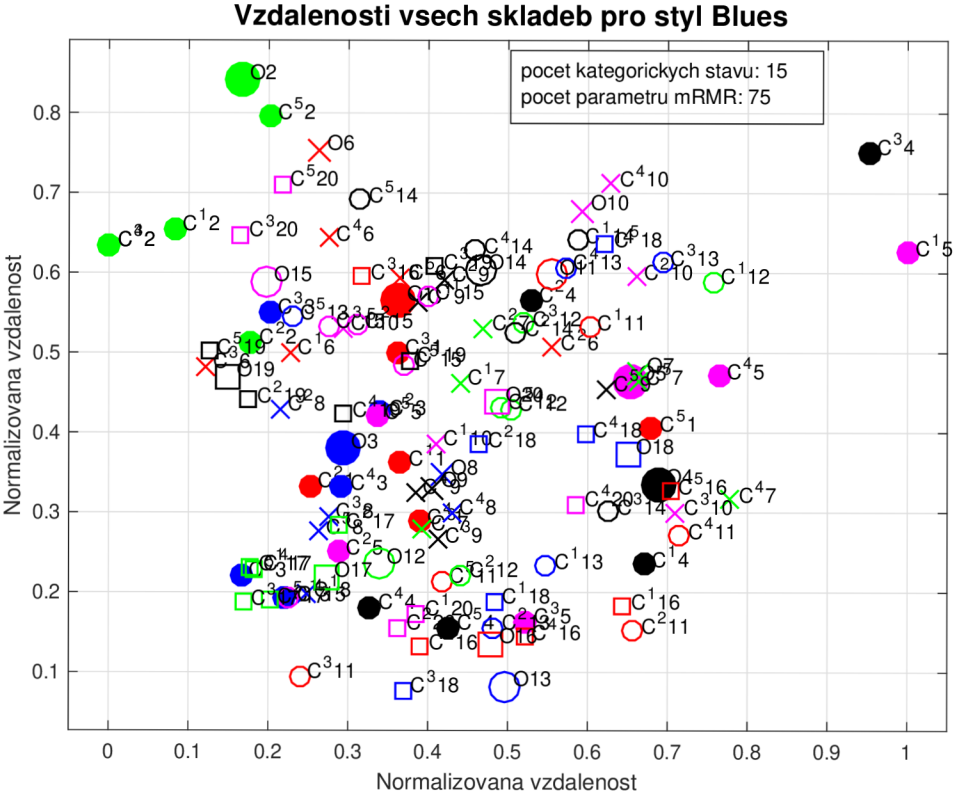

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Blues
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 38.0676
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 1.686 minut
chyba [-]
iterace [-]

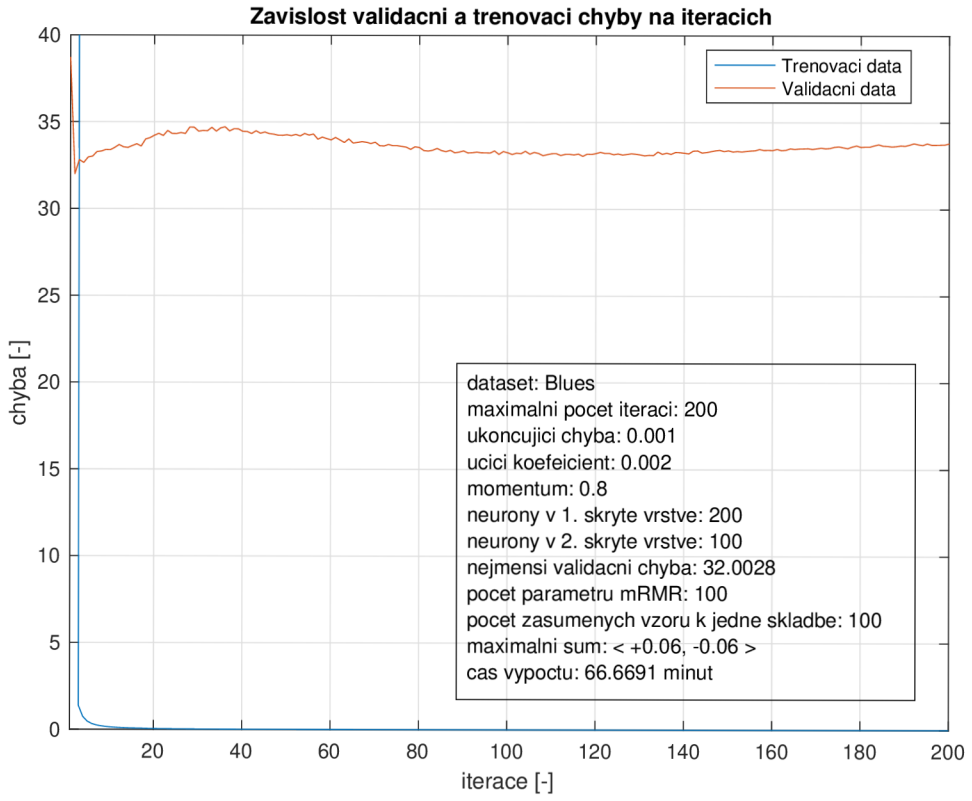

Vzdalenosti vsech skladeb pro styl Blues
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

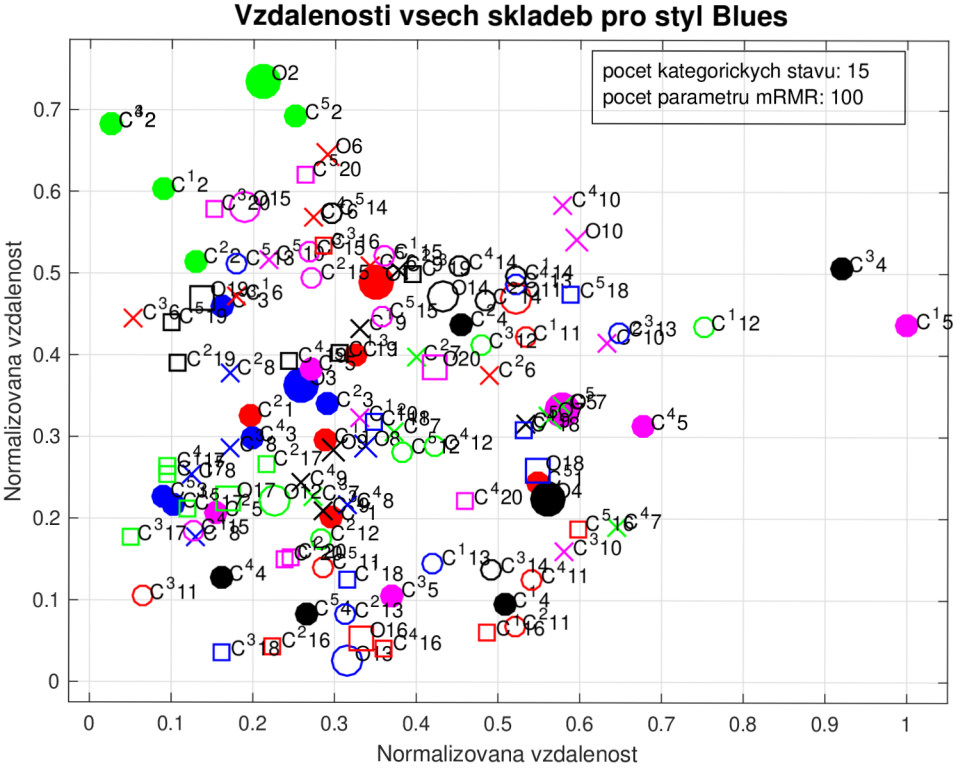

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Blues
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 31.4684
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 3.444 minut
chyba [-]
iterace [-]

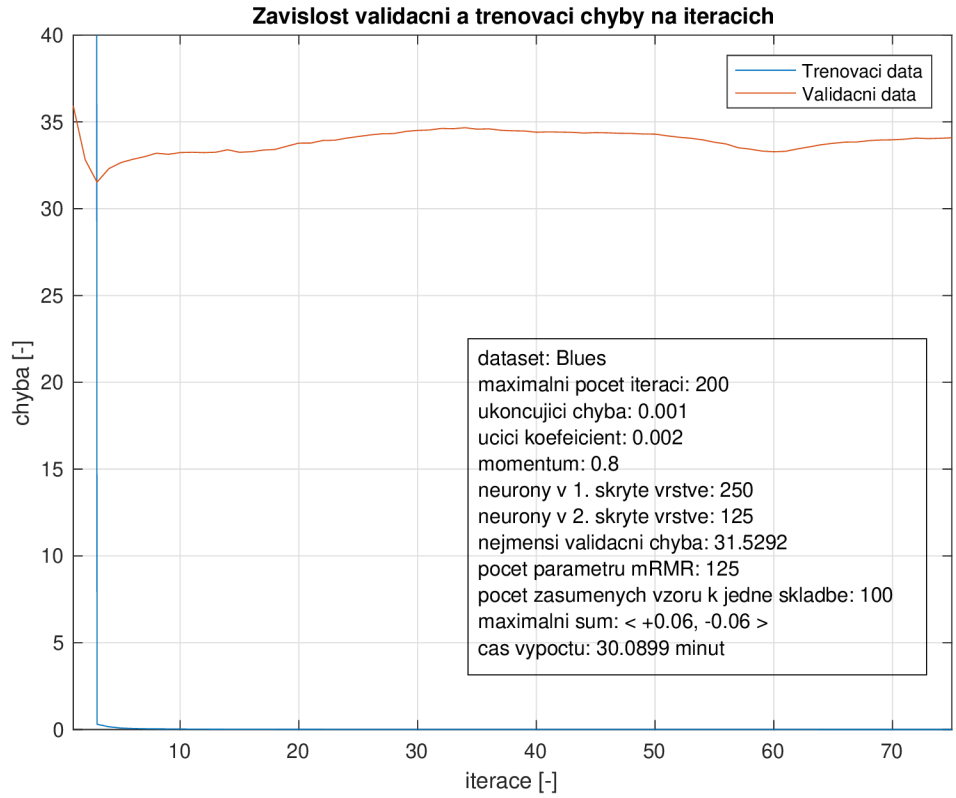

Vzdalenosti vsech skladeb pro styl Blues
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

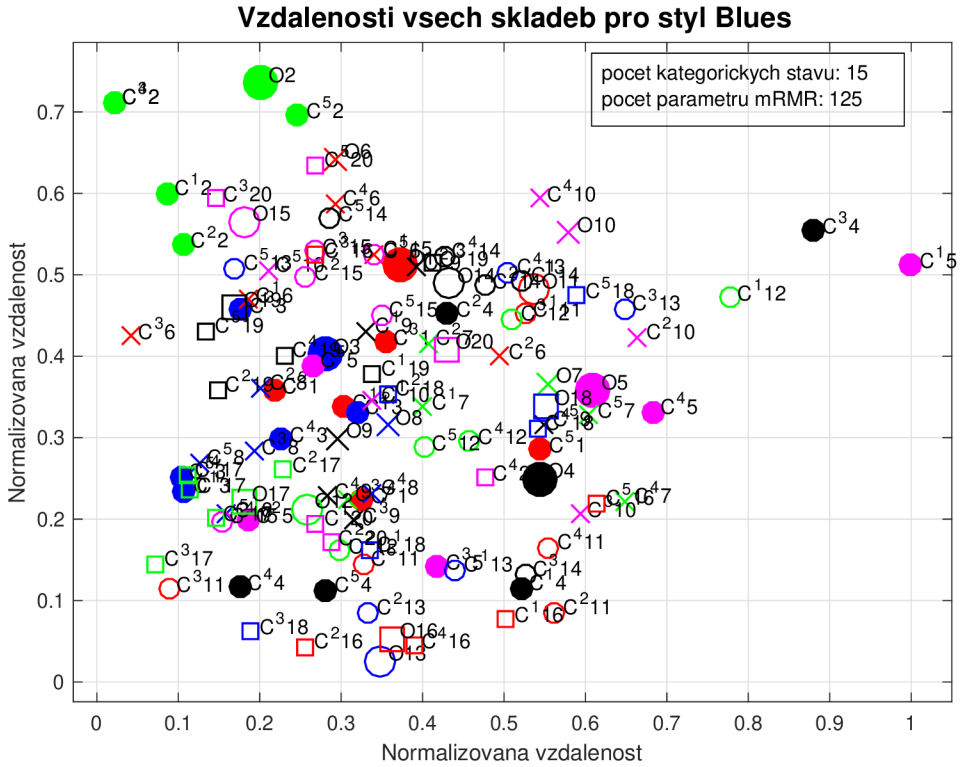

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Blues
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 29.2368
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 5.6815 minut
chyba [-]
iterace [-]

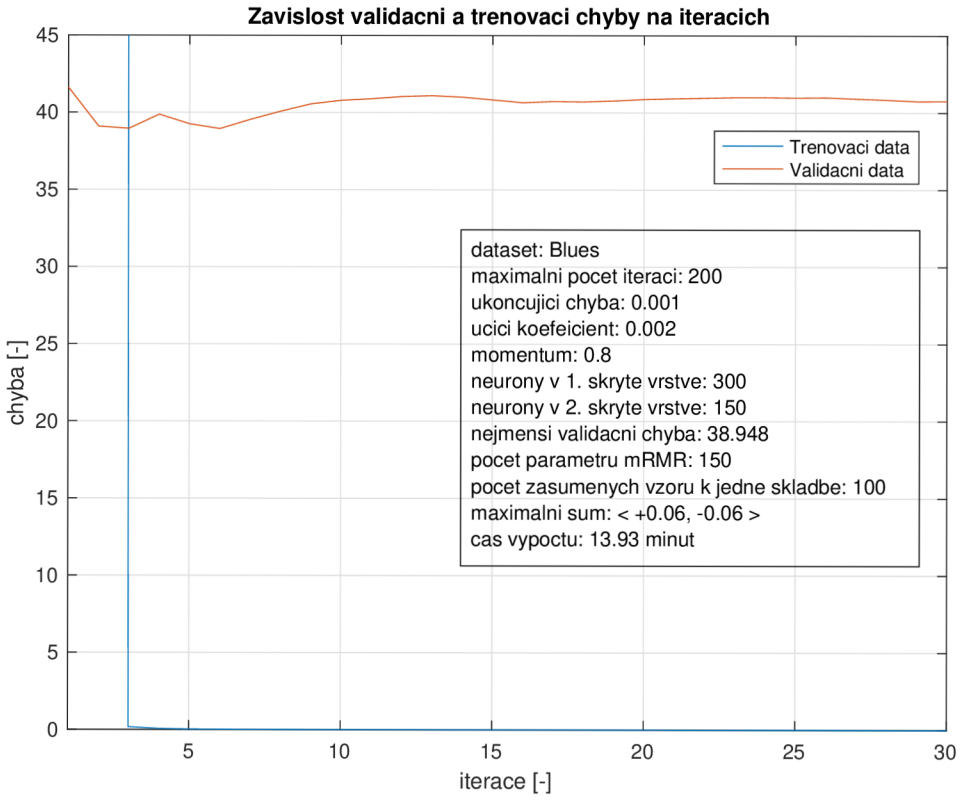

Vzdalenosti vsech skladeb pro styl Blues
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost

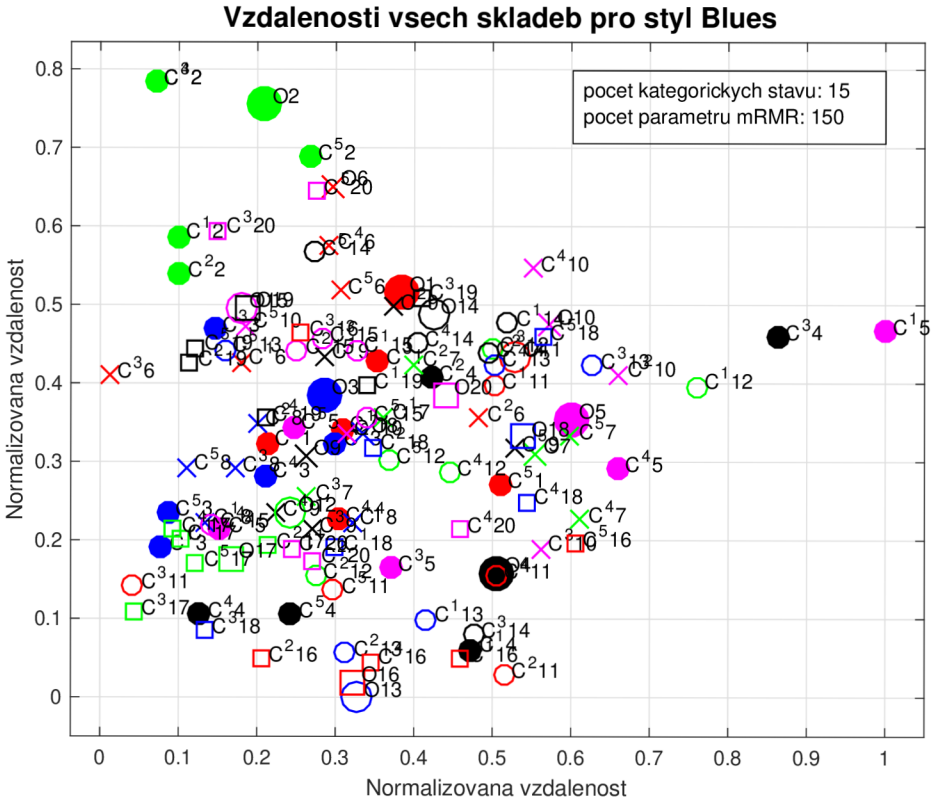

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Blues
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 32.0028
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 66.6691 minut
chyba [-]
iterace [-]

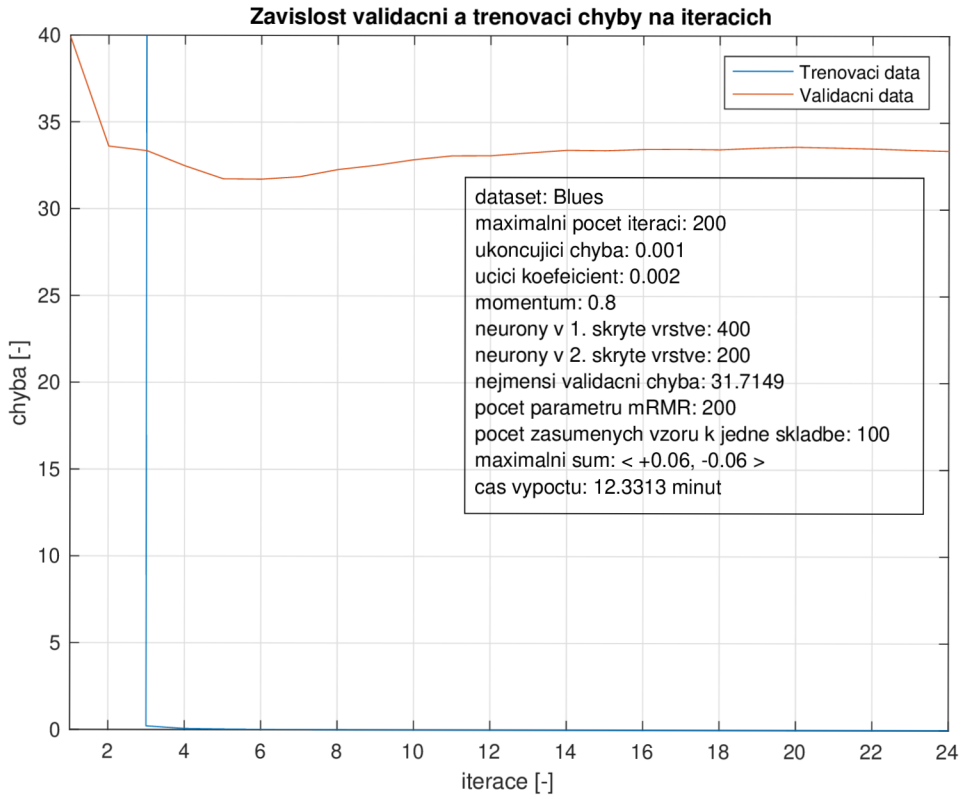

Vzdalenosti vsech skladeb pro styl Blues
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost

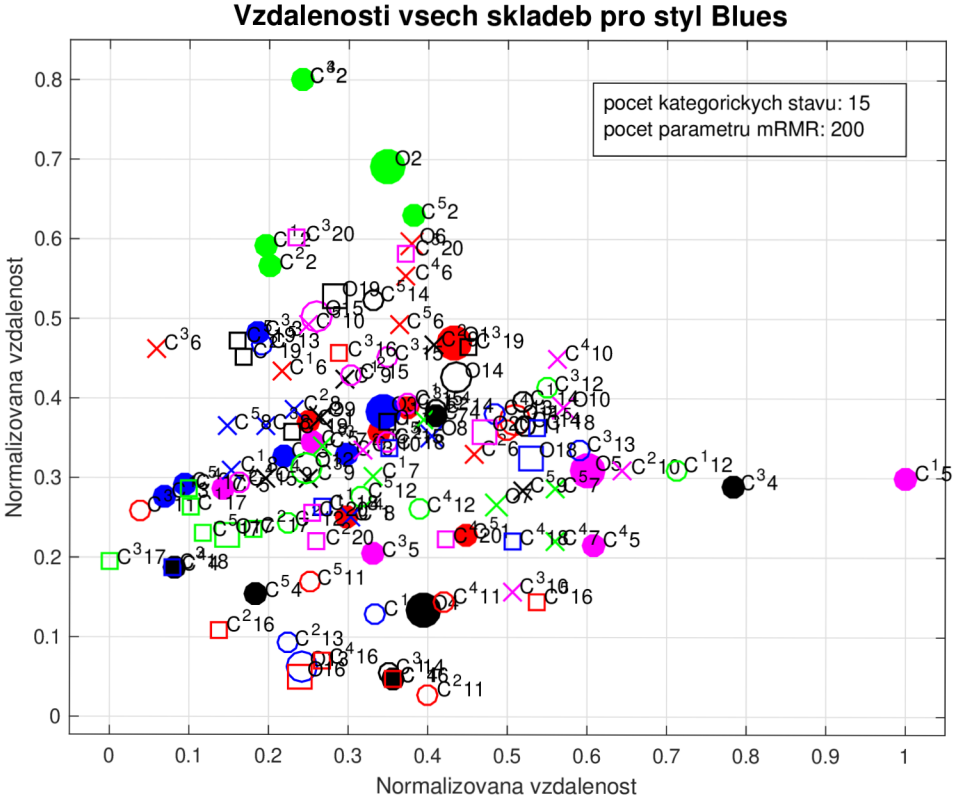

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Blues
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 31.5292
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 30.0899 minut
chyba [-]
iterace [-]

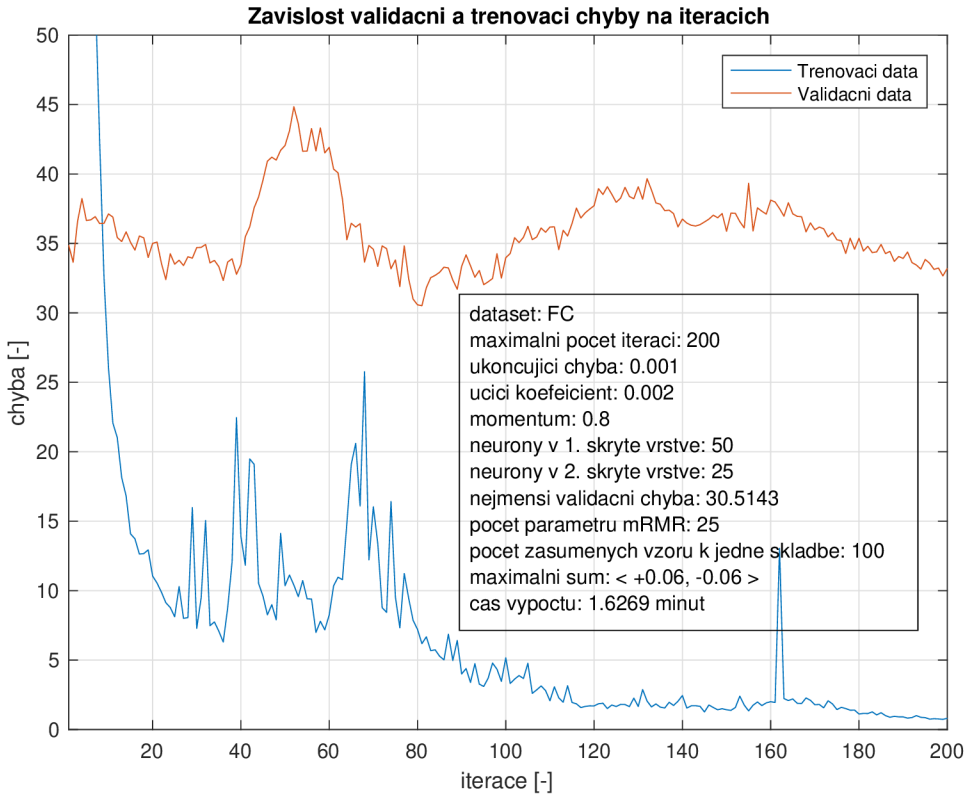

Vzdalenosti vsech skladeb pro styl Blues
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost

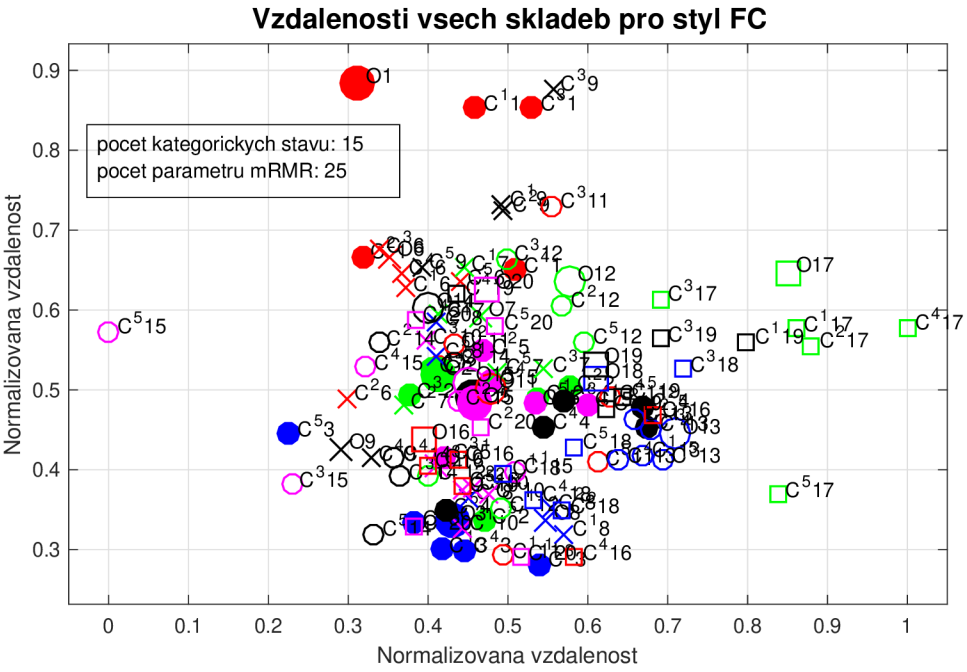

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Blues
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 300
neurony v 2. skryte vrstve: 150
nejmensi validacni chyba: 38.948
pocet parametru mRMR: 150
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 13.93 minut
chyba [-]
iterace [-]

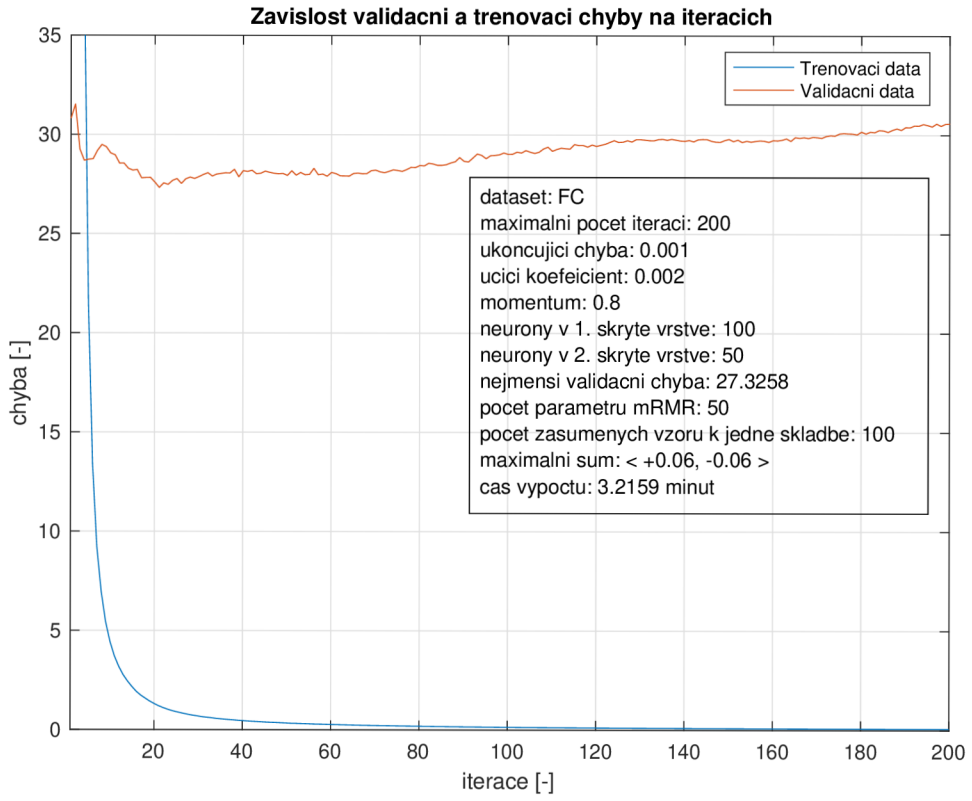

Vzdalenosti vsech skladeb pro styl Blues
pocet kategorickych stavu: 15
pocet parametru mRMR: 150
Normalizovana vzdalenost
Normalizovana vzdalenost

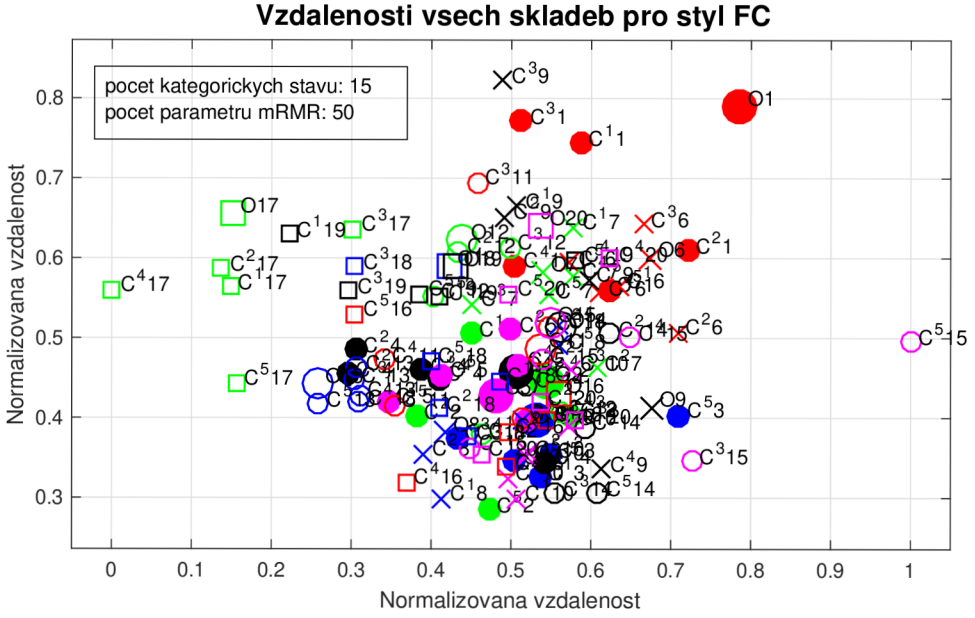

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Blues
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 31.7149
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 12.3313 minut
chyba [-]
iterace [-]

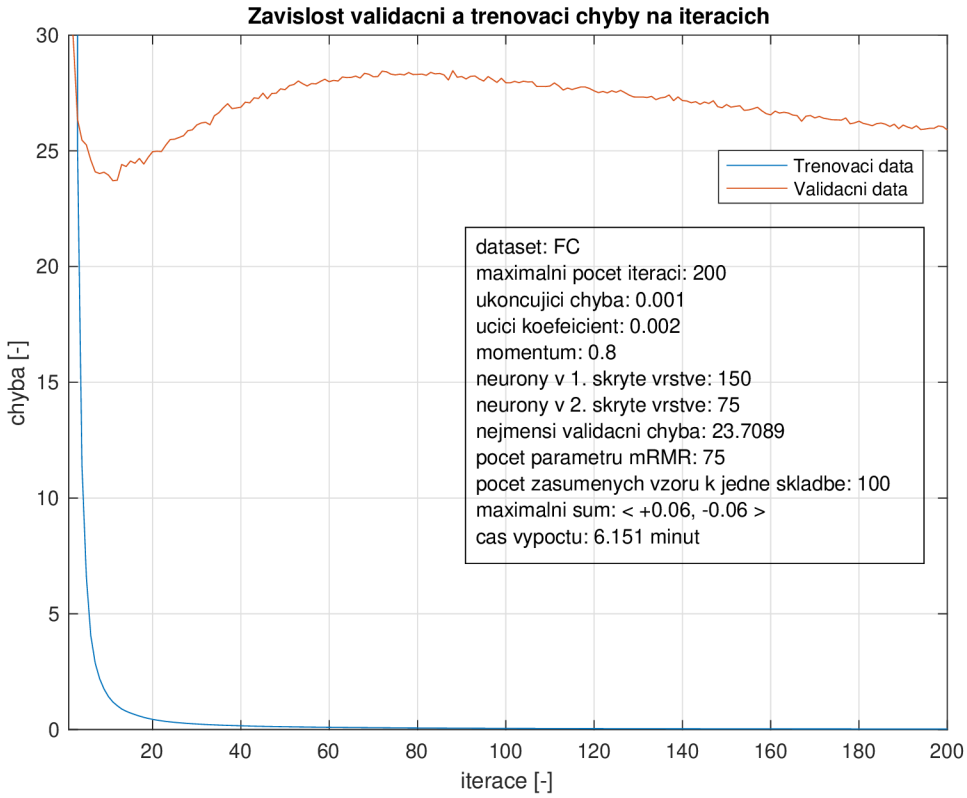

Vzdalenosti vsech skladeb pro styl Blues
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

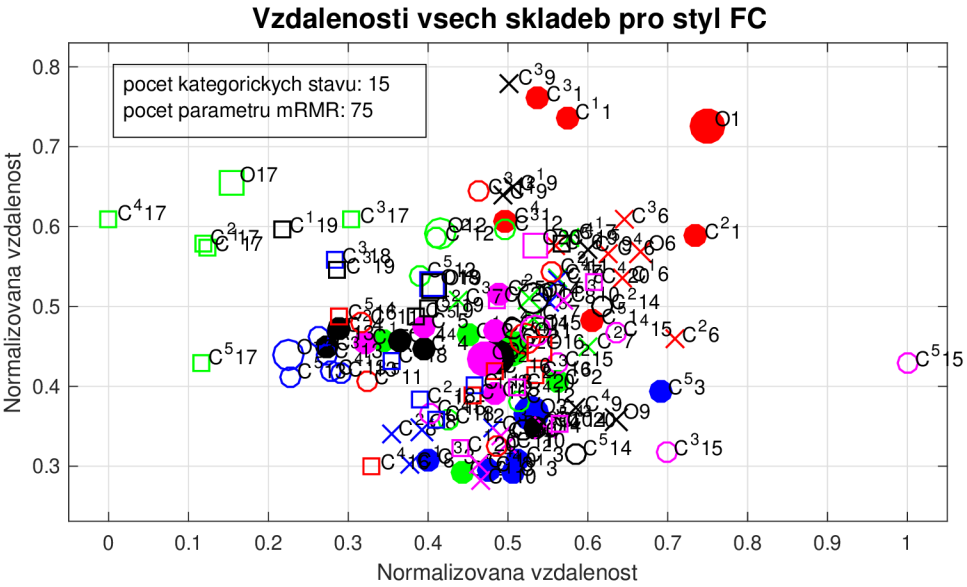

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: FC
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 30.5143
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 1.6269 minut
chyba [-]
iterace [-]

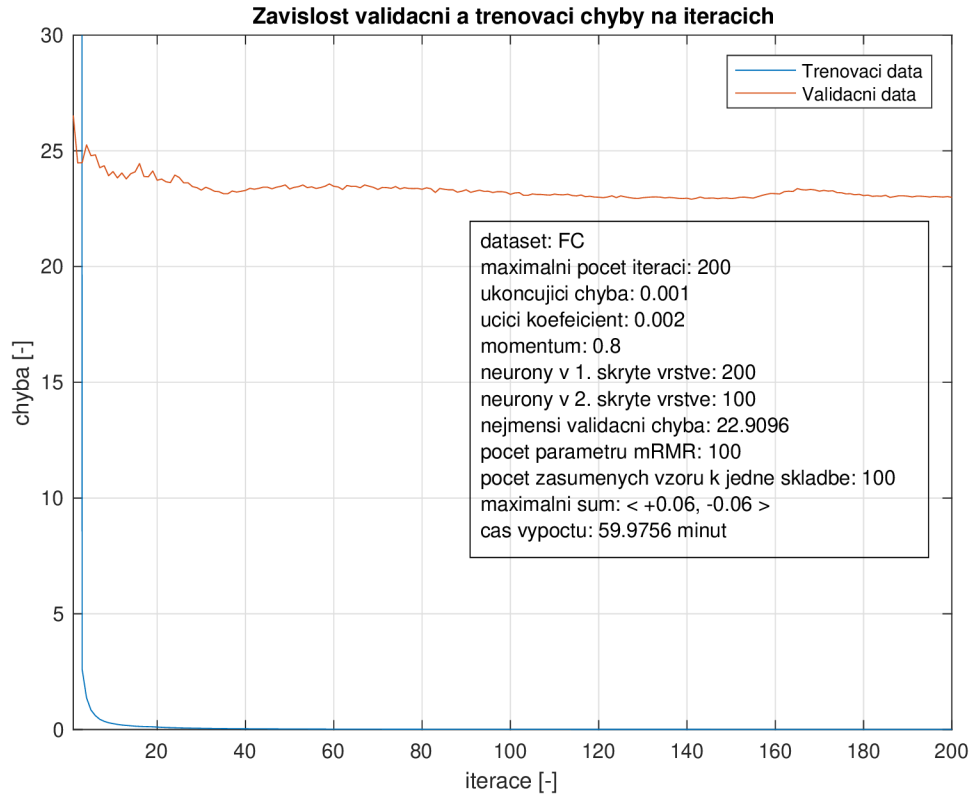

Vzdalenosti vsech skladeb pro styl FC
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

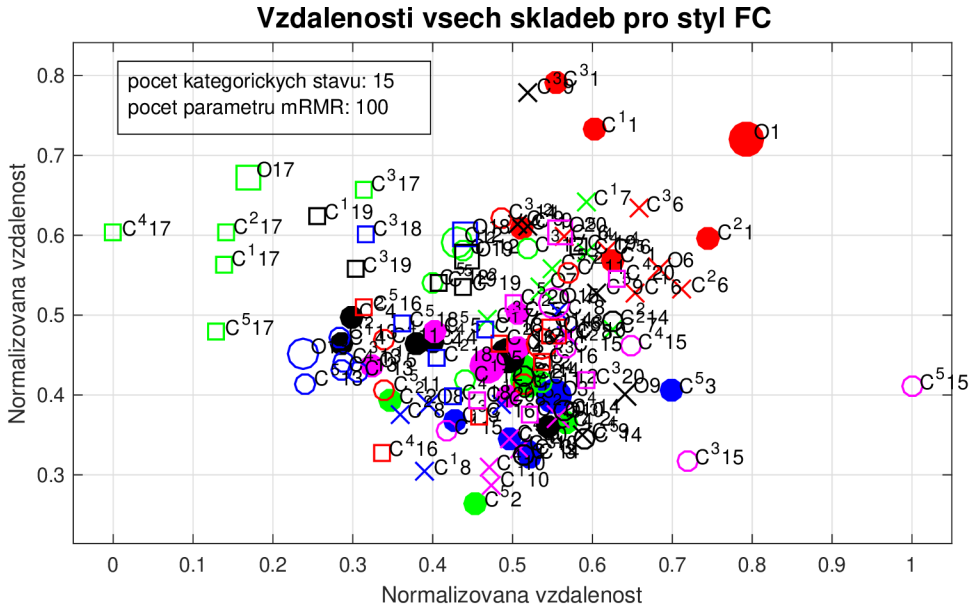

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: FC
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 27.3258
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 3.2159 minut
chyba [-]
iterace [-]

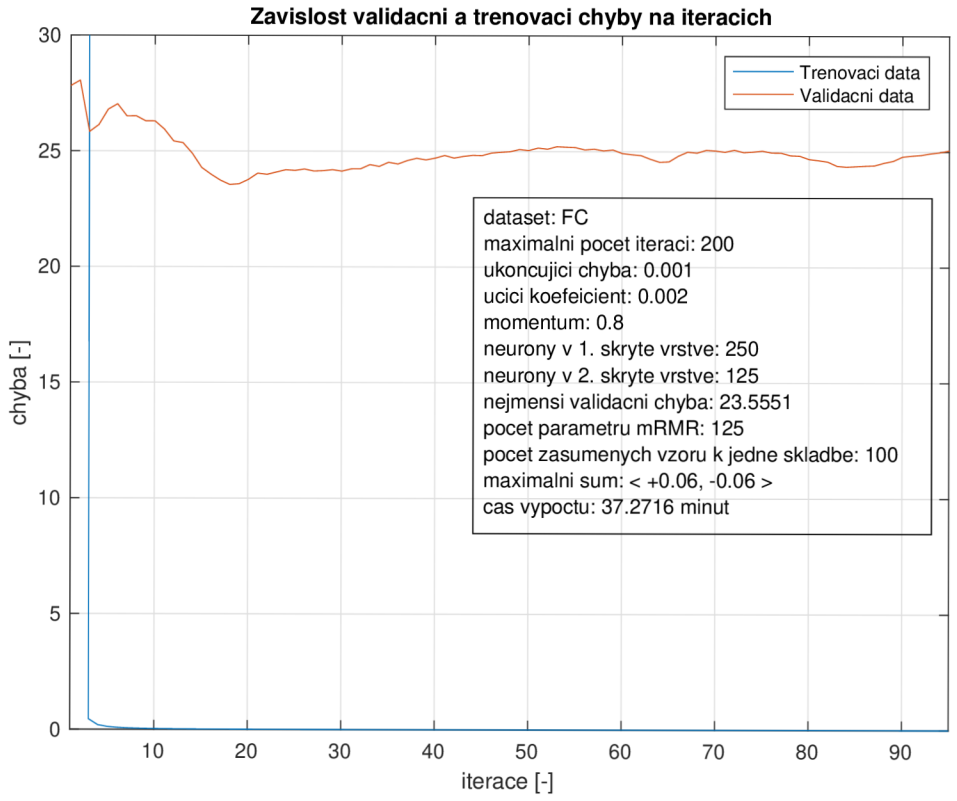

Vzdalenosti vsech skladeb pro styl FC
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

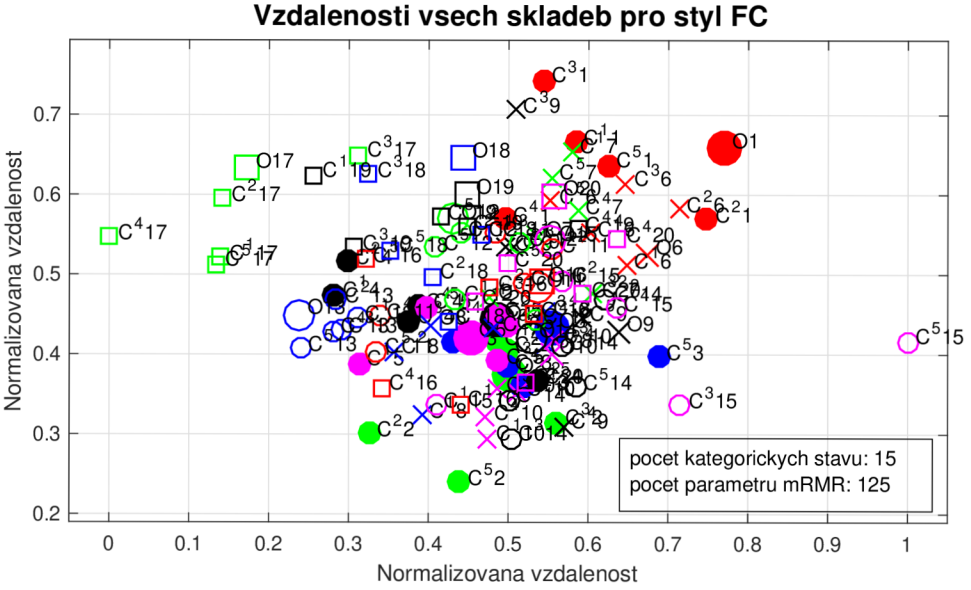

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: FC
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 23.7089
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 6.151 minut
chyba [-]
iterace [-]

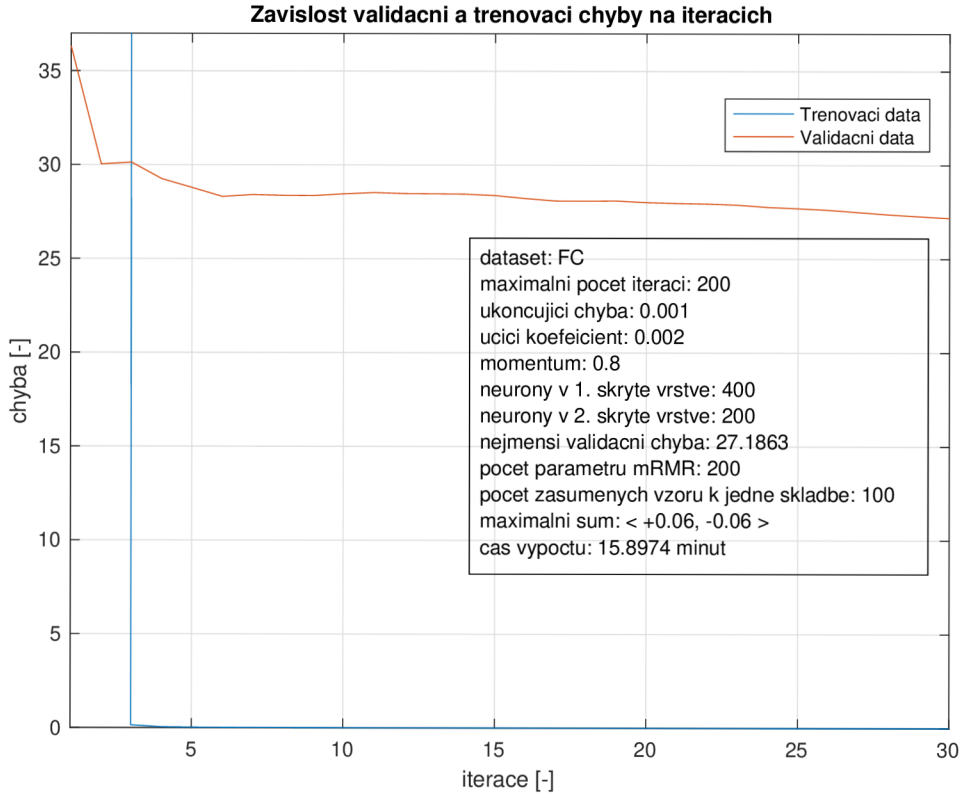

Vzdalenosti vsech skladeb pro styl FC
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost

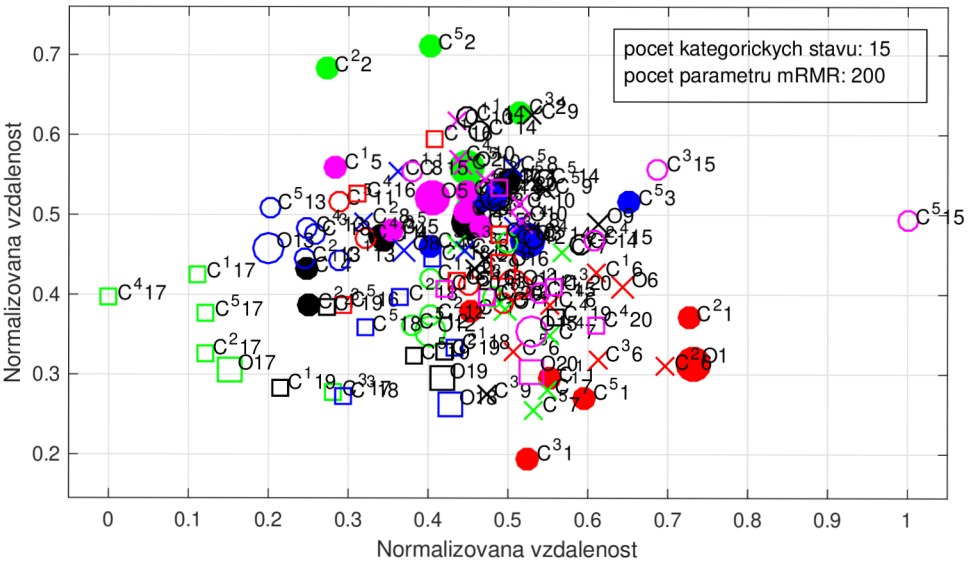

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: FC
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 22.9096
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 59.9756 minut
chyba [-]
iterace [-]

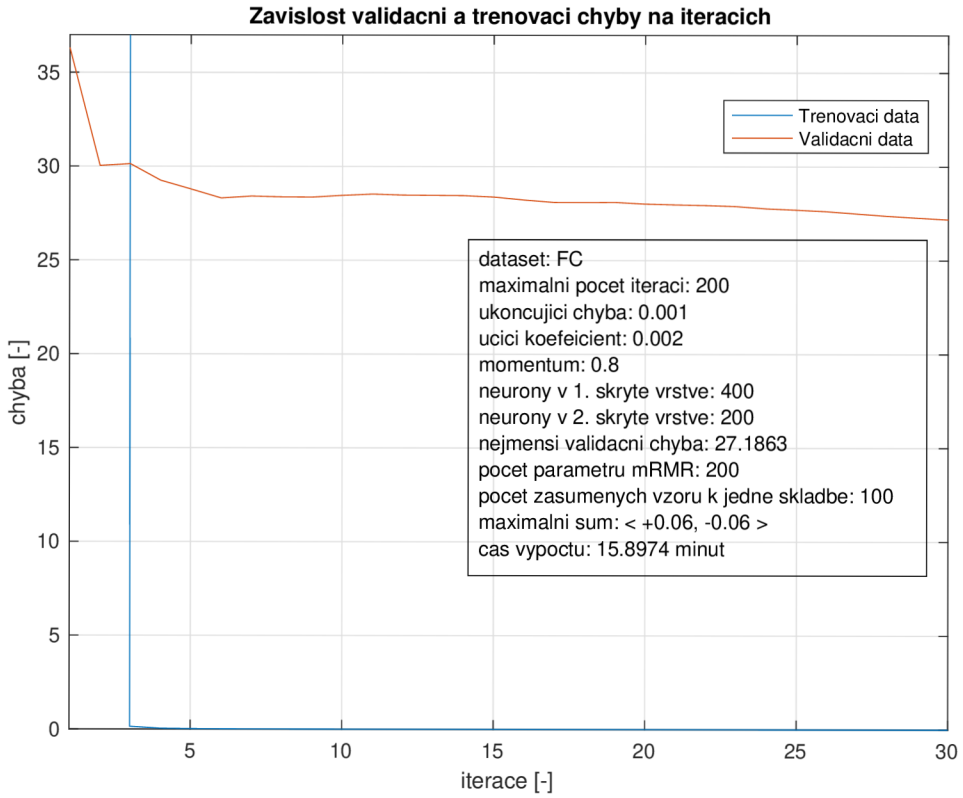

Vzdalenosti vsech skladeb pro styl FC
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost

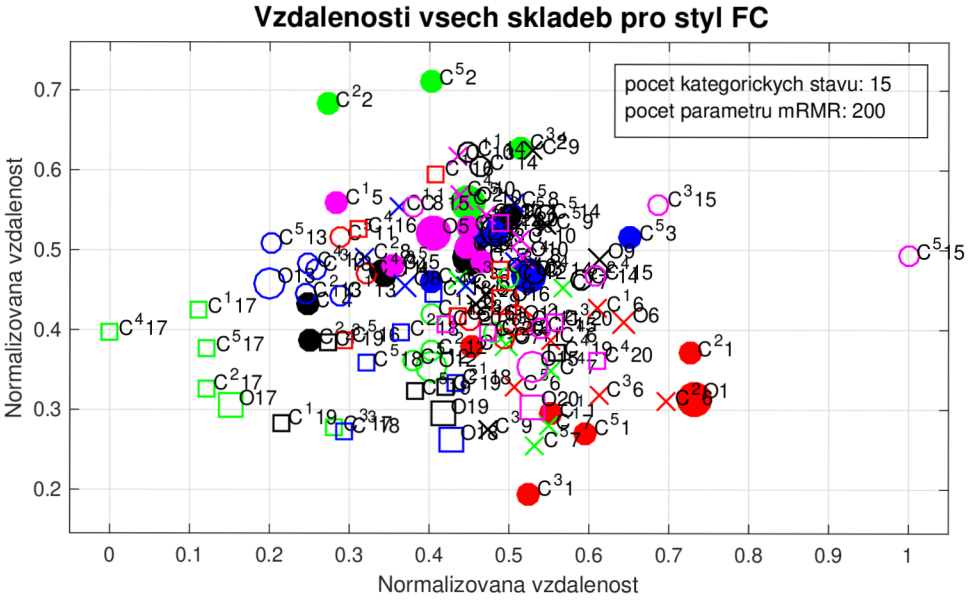

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: FC
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 23.5551
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 37.2716 minut
chyba [-]
iterace [-]

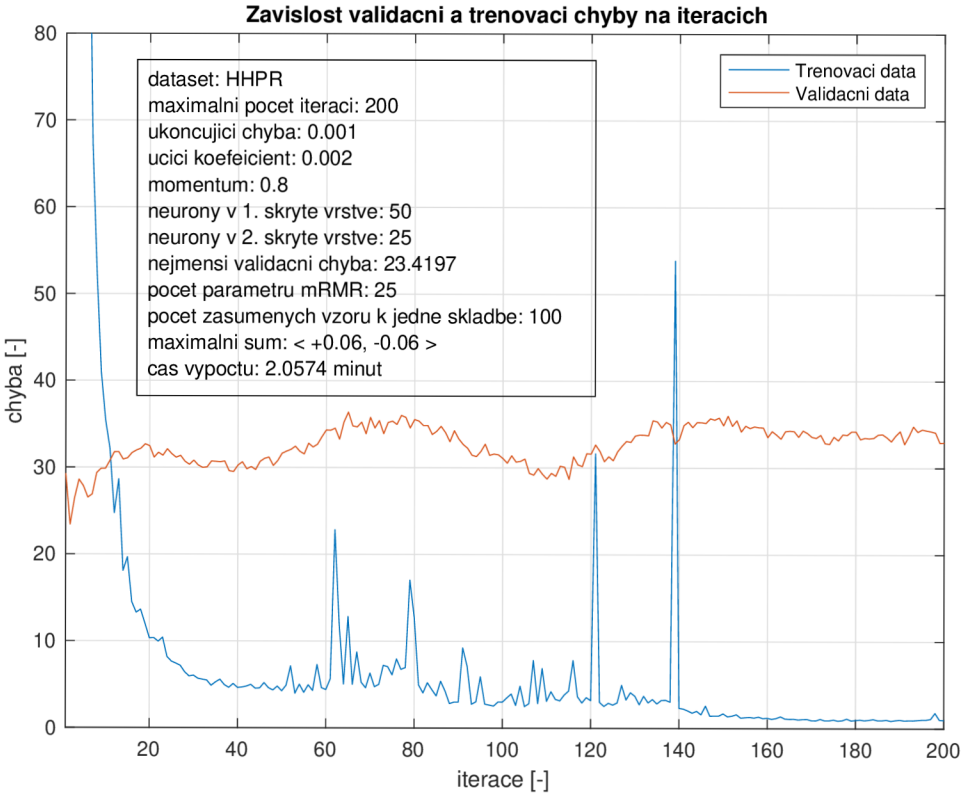

Vzdalenosti vsech skladeb pro styl FC
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost

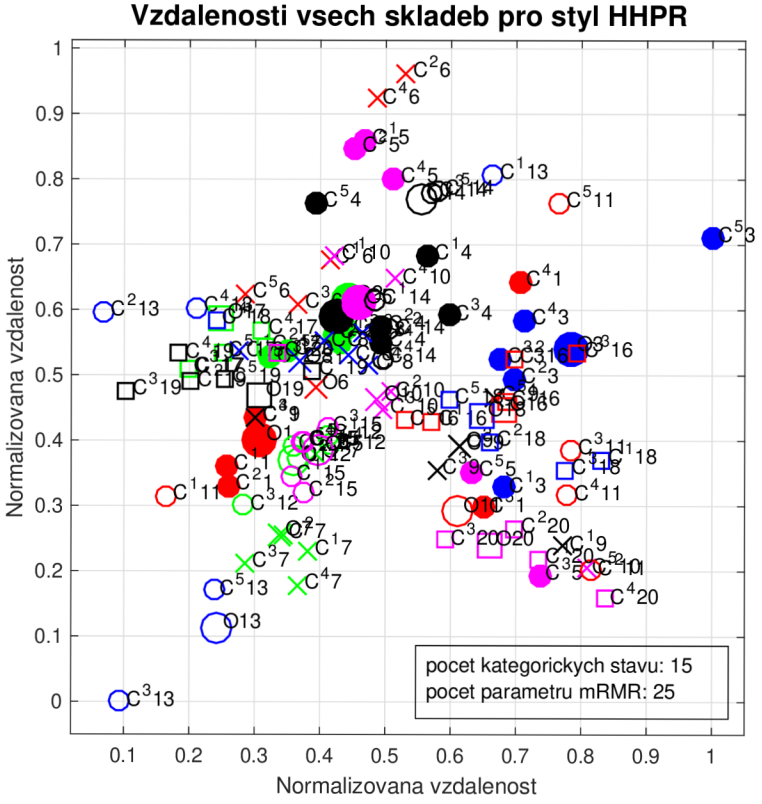

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: FC
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 27.1863
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 15.8974 minut
chyba [-]
iterace [-]

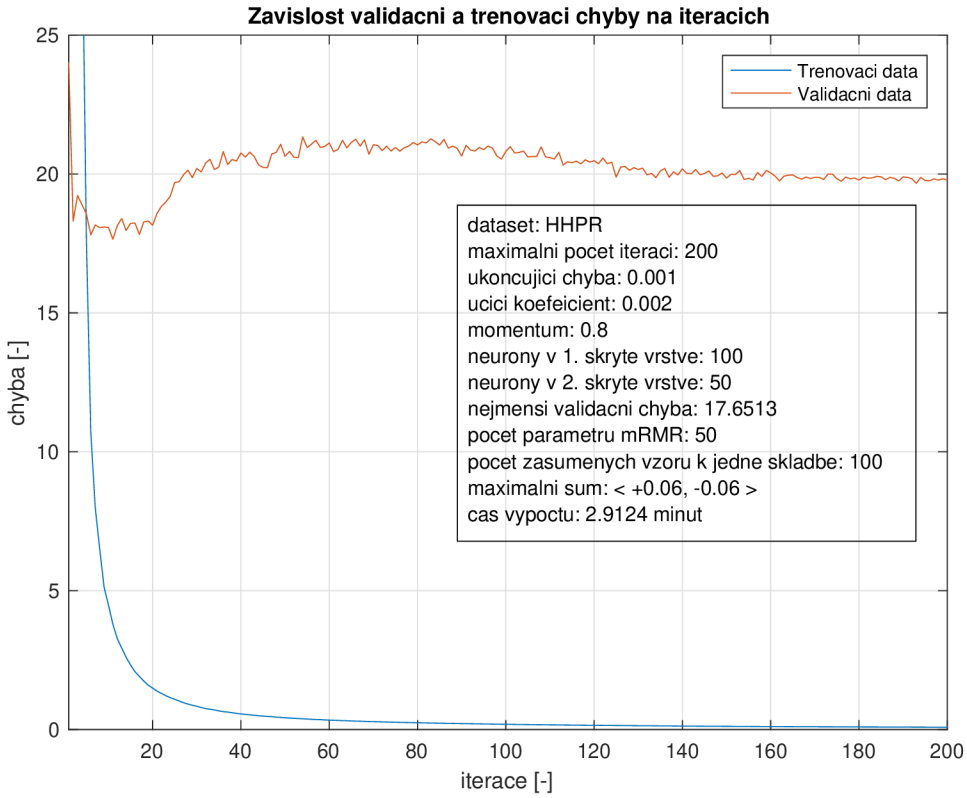

Vzdalenosti vsech skladeb pro styl FC
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

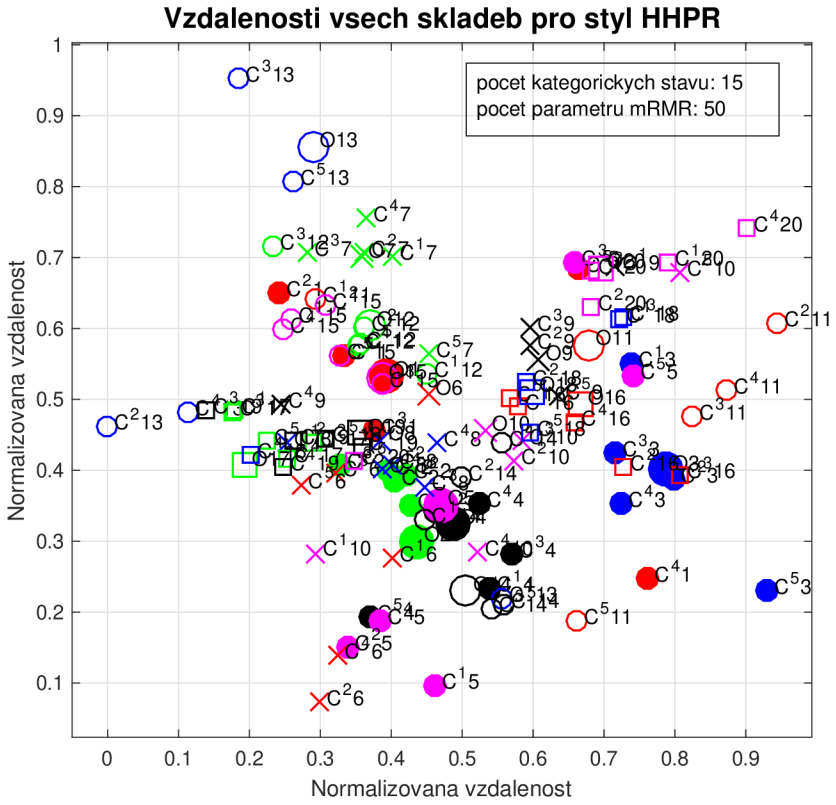

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: FC
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 27.1863
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 15.8974 minut
chyba [-]
iterace [-]

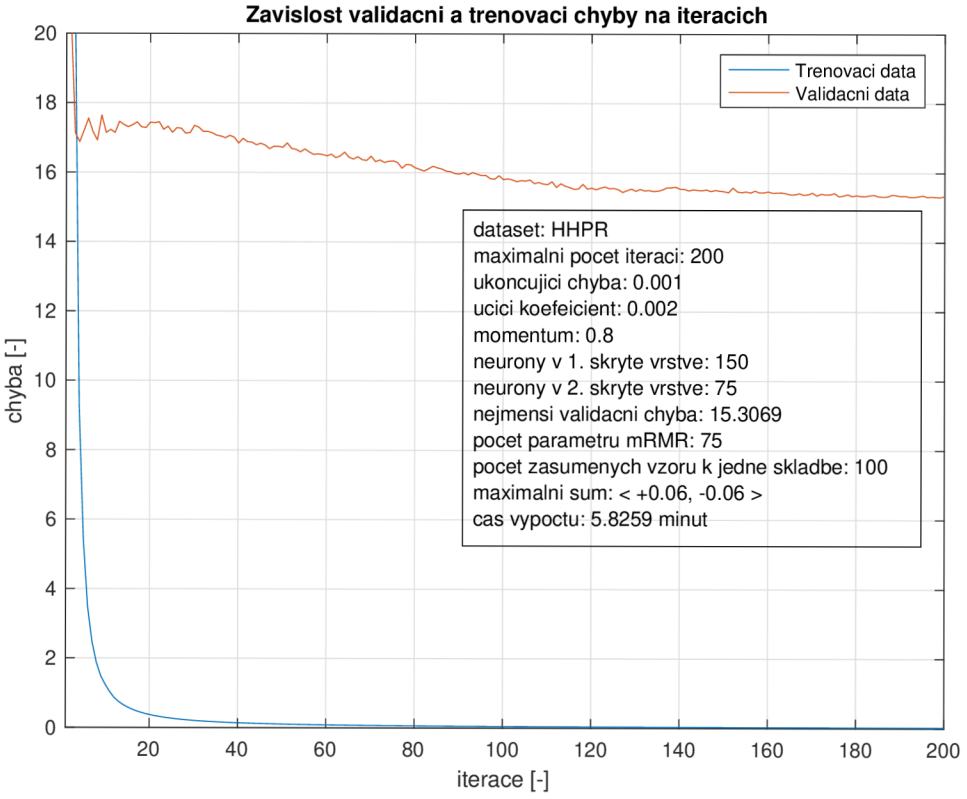

Vzdalenosti vsech skladeb pro styl FC
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

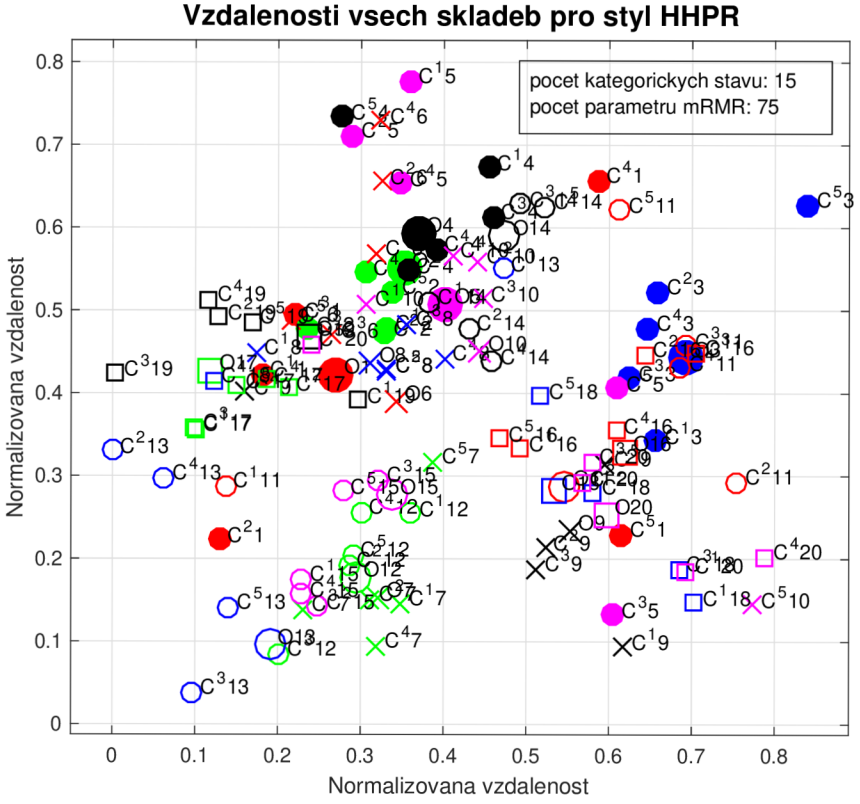

Zavislost validacni a trenovaci chyby na iteracich
dataset: HHPR
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 23.4197
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 2.0574 minut
Trenovaci data
Validacni data
chyba [-]
iterace [-]

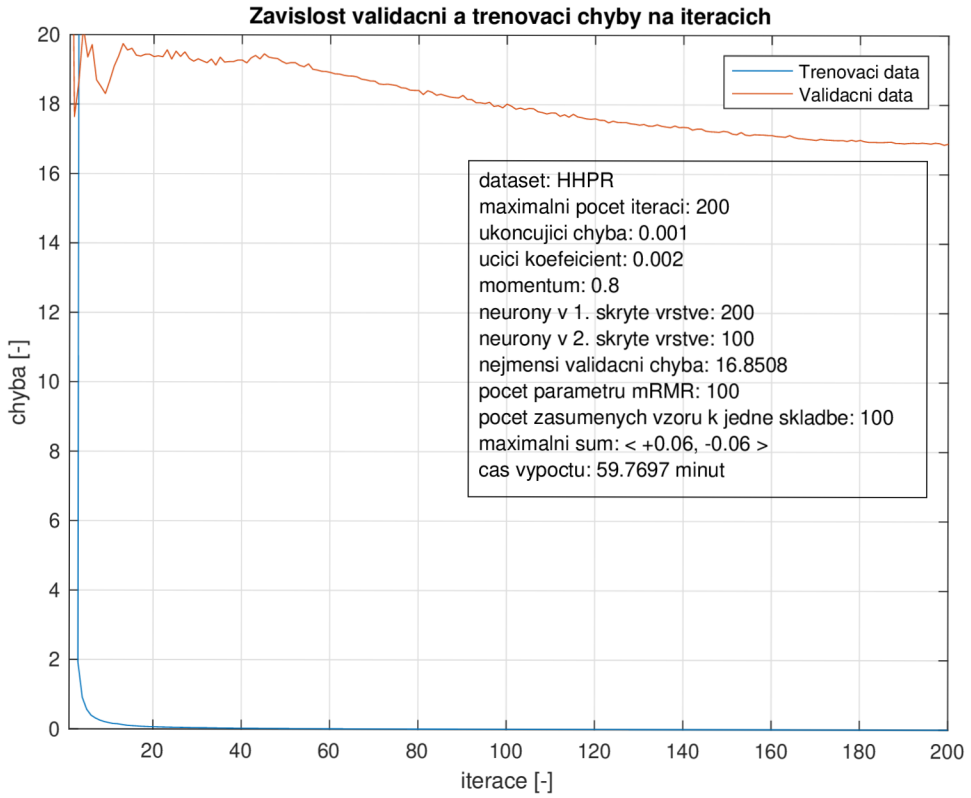

Vzdalenosti vsech skladeb pro styl HHPR
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

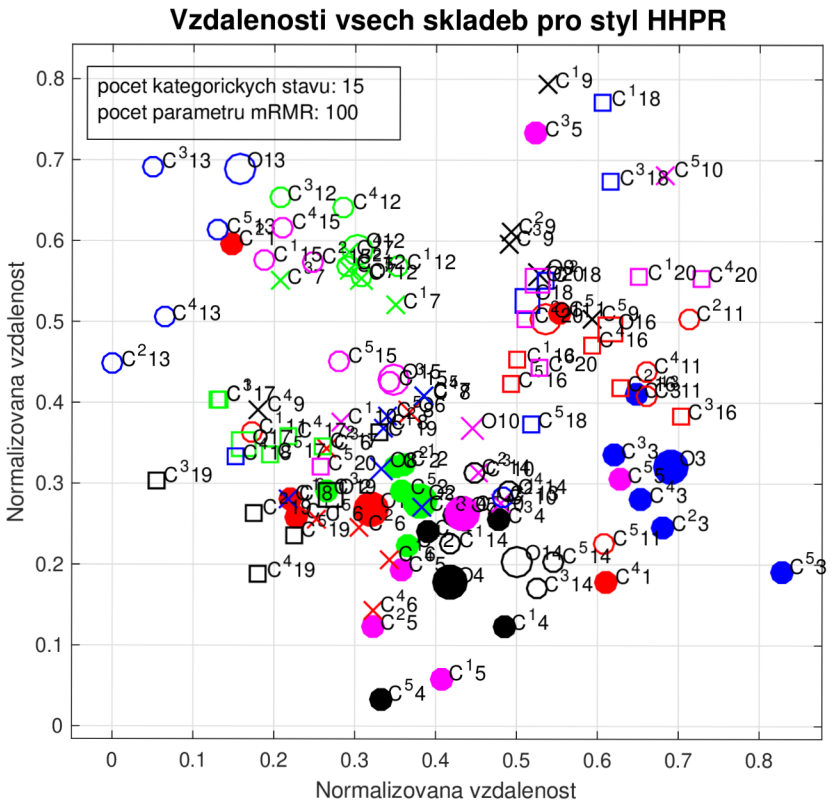

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: HHPR
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 17.6513
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 2.9124 minut
chyba [-]
iterace [-]

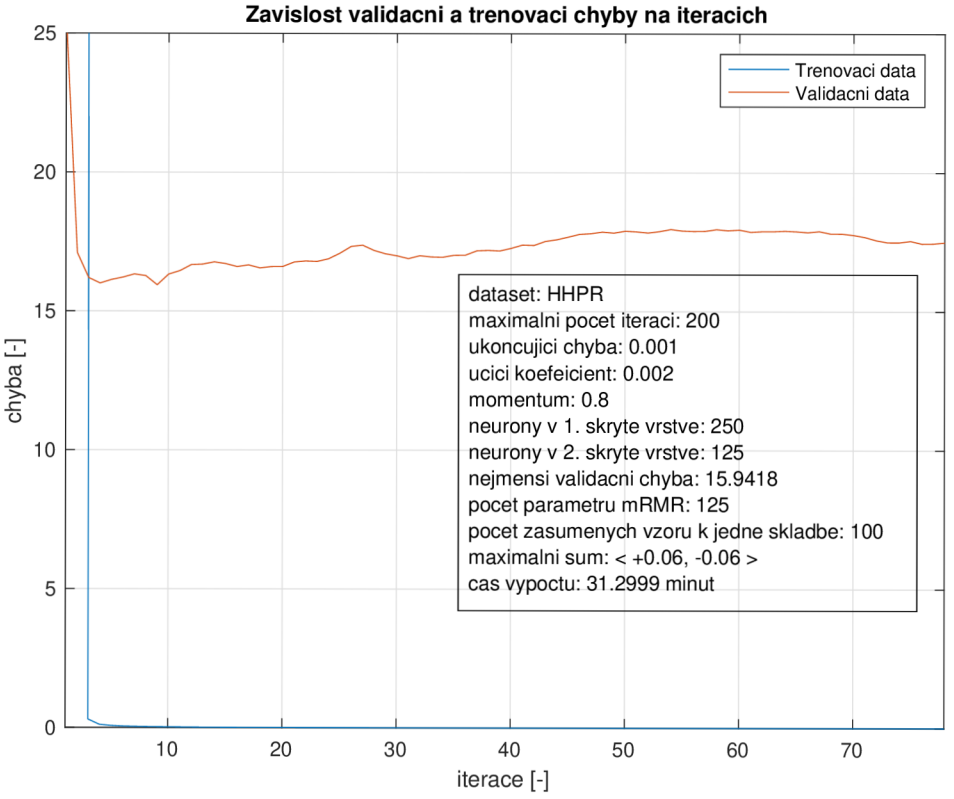

Vzdalenosti vsech skladeb pro styl HHPR
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

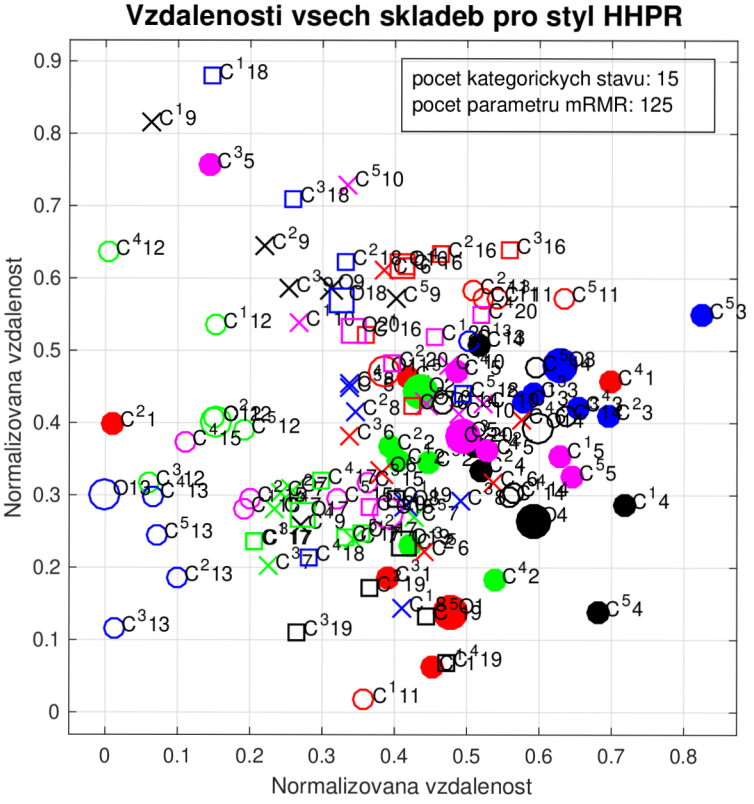

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: HHPR
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 15.3069
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 5.8259 minut
chyba [-]
iterace [-]

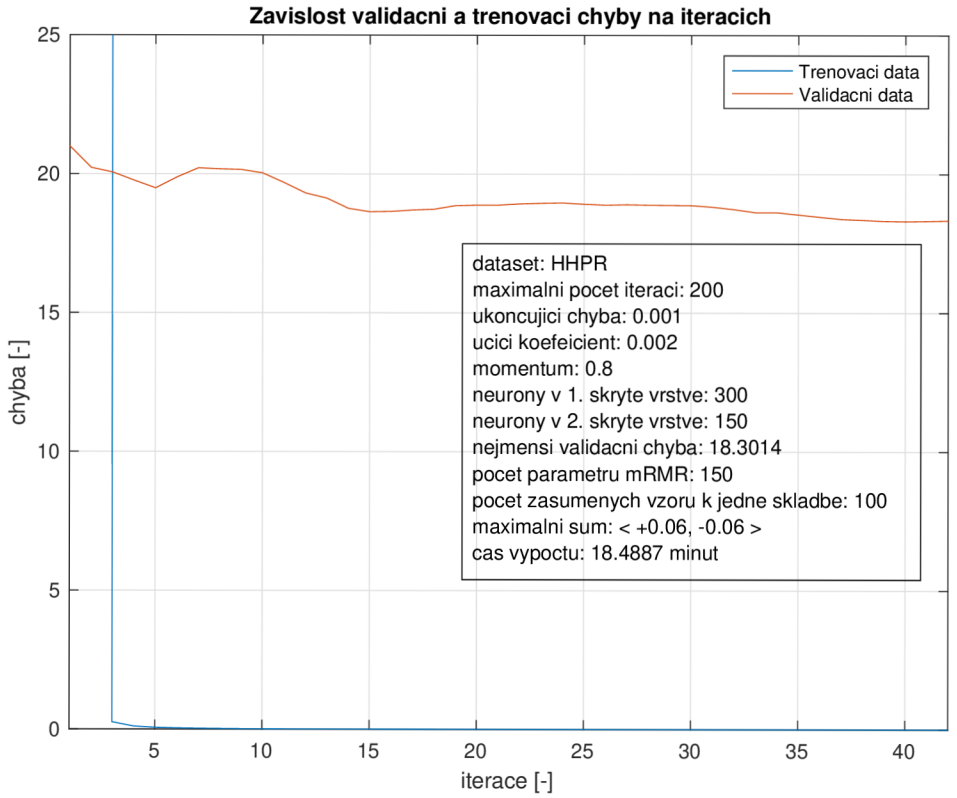

Vzdalenosti vsech skladeb pro styl HHPR
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost

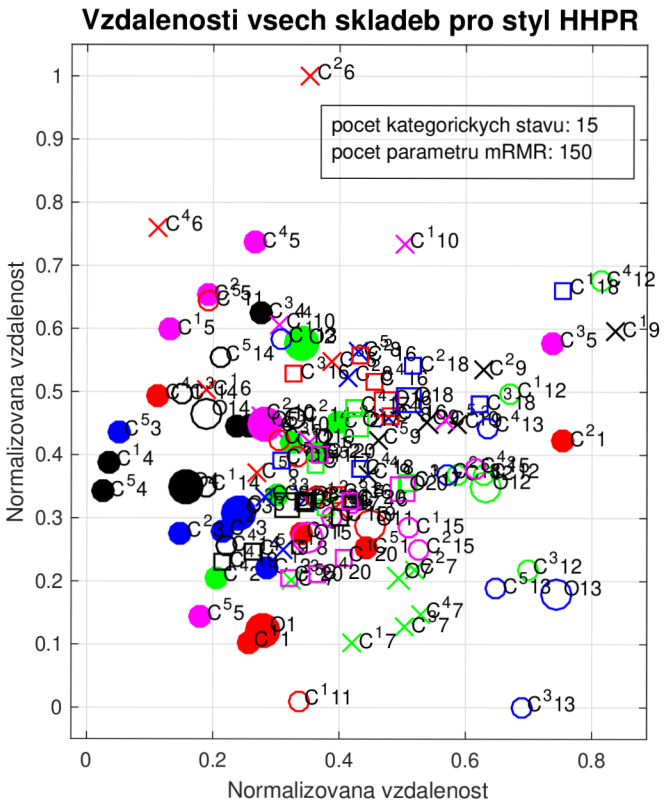

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: HHPR
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 16.8508
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 59.7697 minut
chyba [-]
iterace [-]

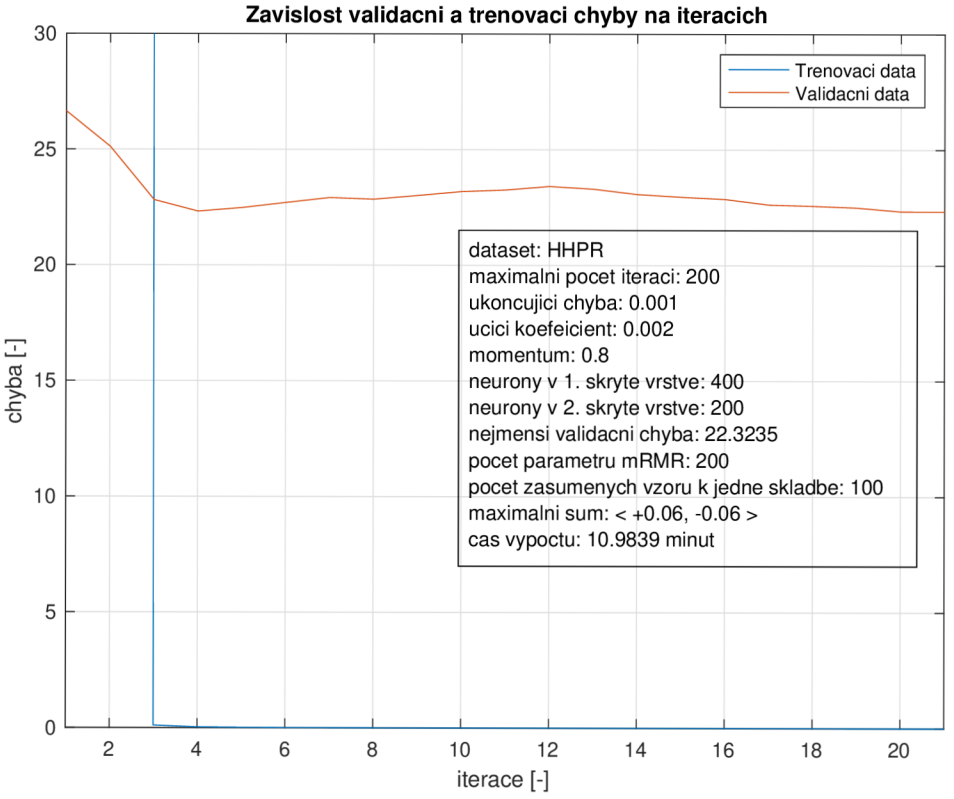

Vzdalenosti vsech skladeb pro styl HHPR
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost

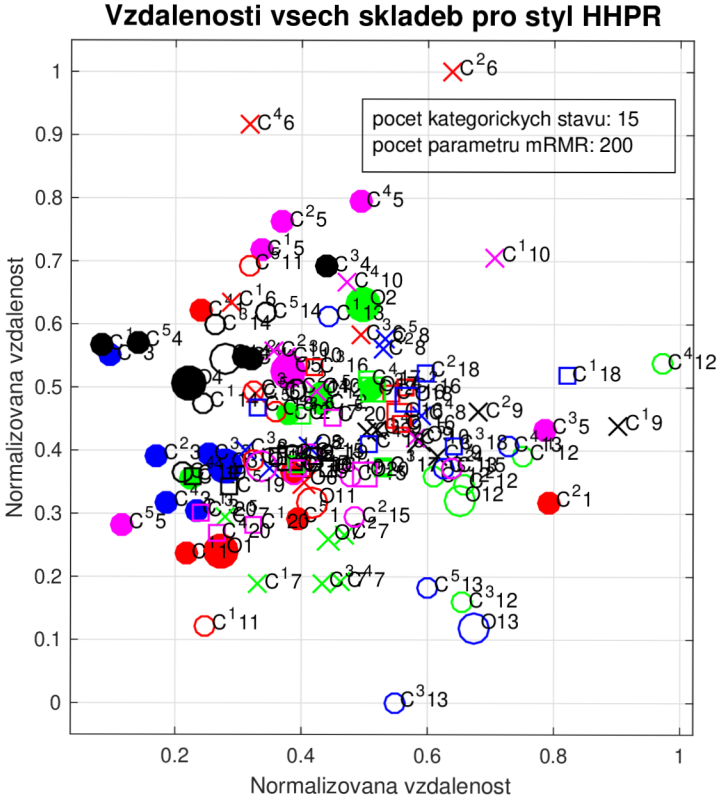

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: HHPR
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 15.9418
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 31.2999 minut
chyba [-]
iterace [-]

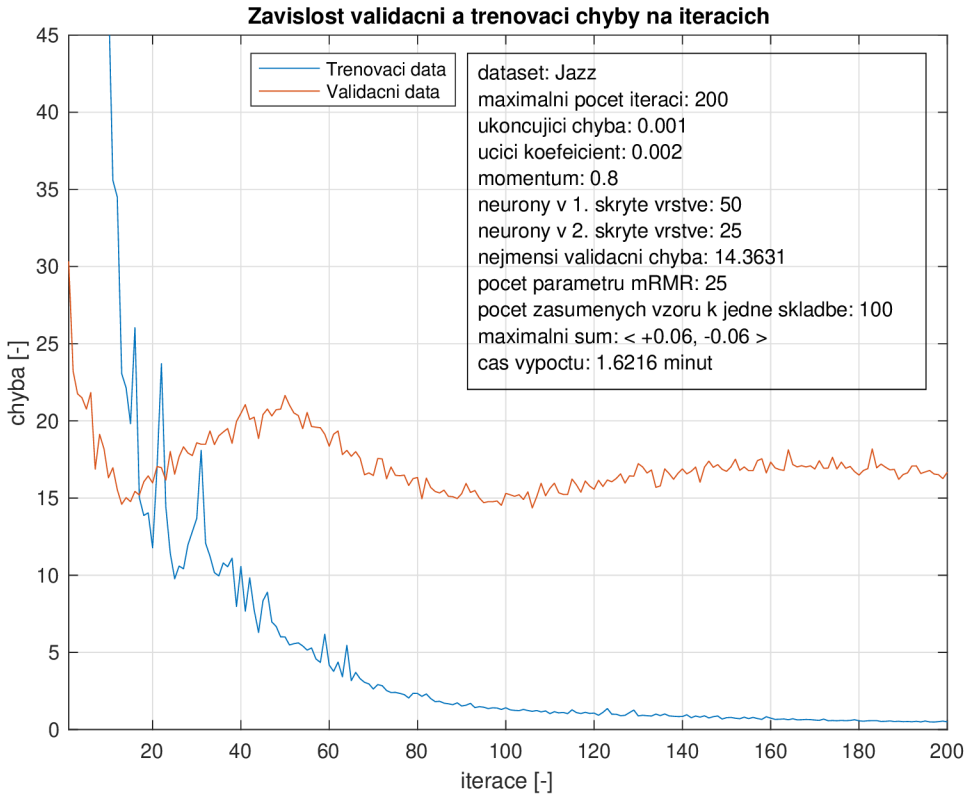

Vzdalenosti vsech skladeb pro styl HHPR
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost

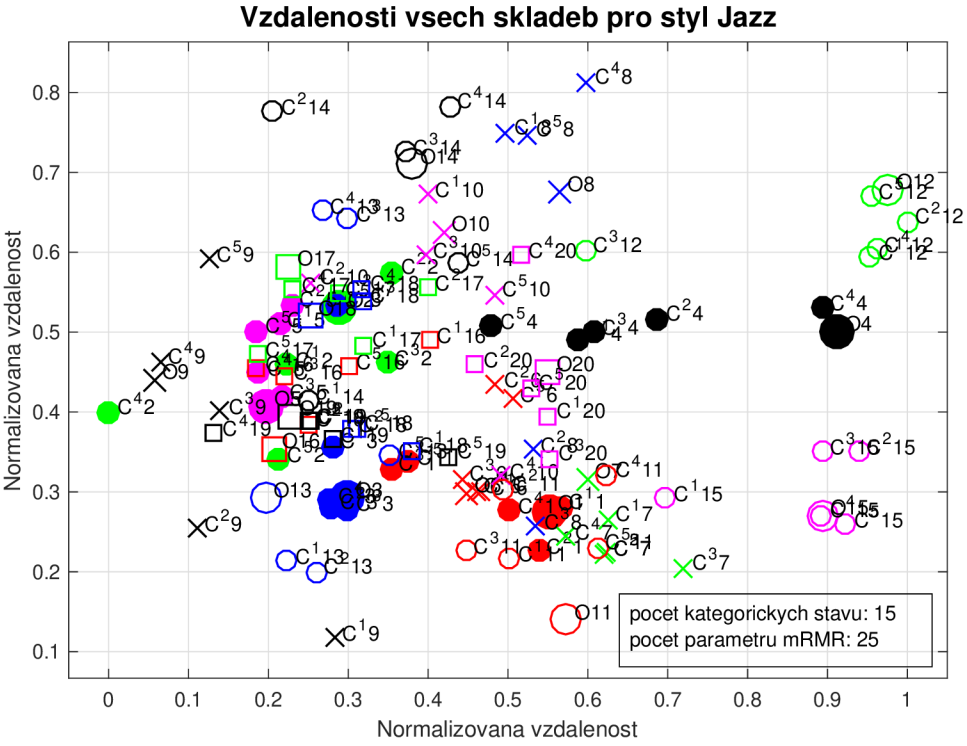

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: HHPR
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 300
neurony v 2. skryte vrstve: 150
nejmensi validacni chyba: 18.3014
pocet parametru mRMR: 150
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 18.4887 minut
chyba [-]
iterace [-]

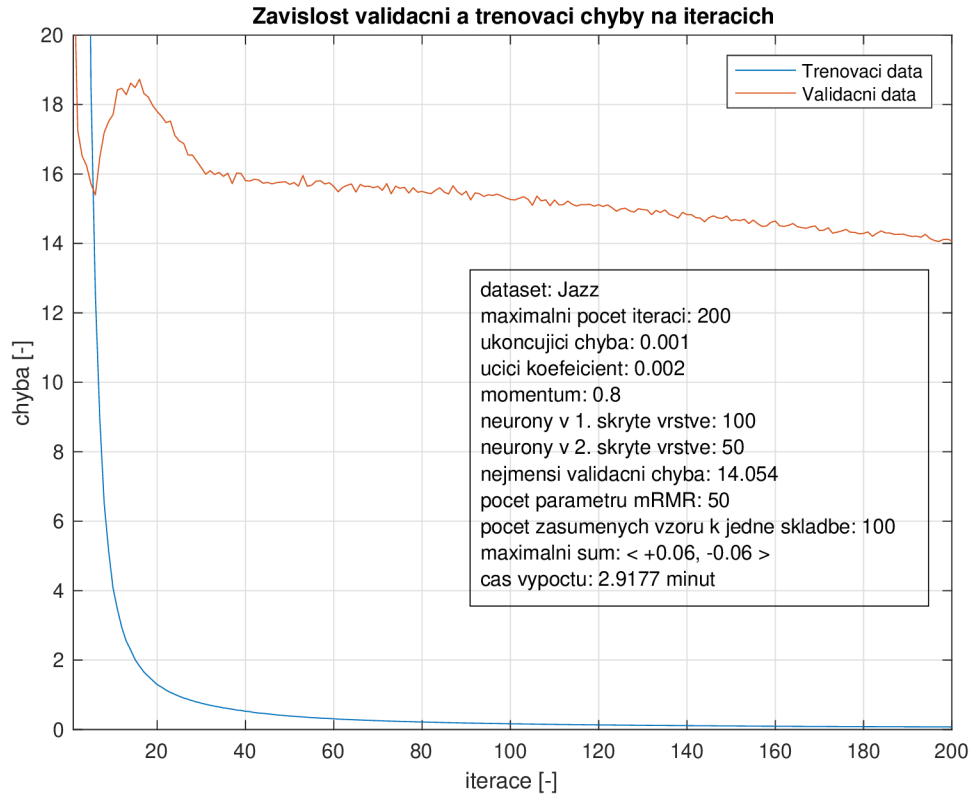

Vzdalenosti vsech skladeb pro styl HHPR
pocet kategorickych stavu: 15
pocet parametru mRMR: 150
Normalizovana vzdalenost
Normalizovana vzdalenost

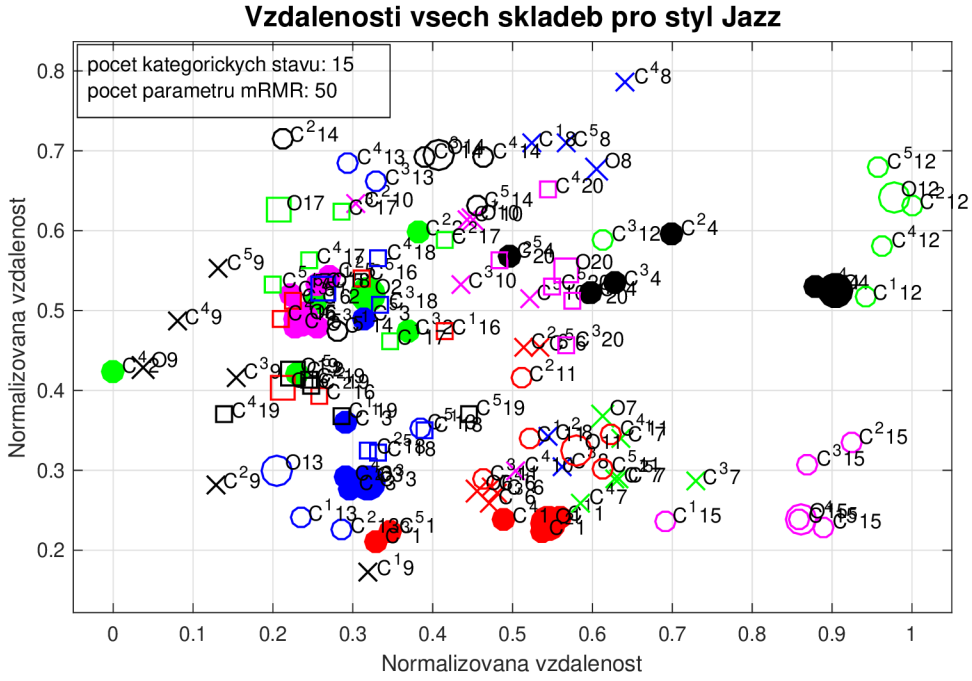

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: HHPR
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 22.3235
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 10.9839 minut
chyba [-]
iterace [-]

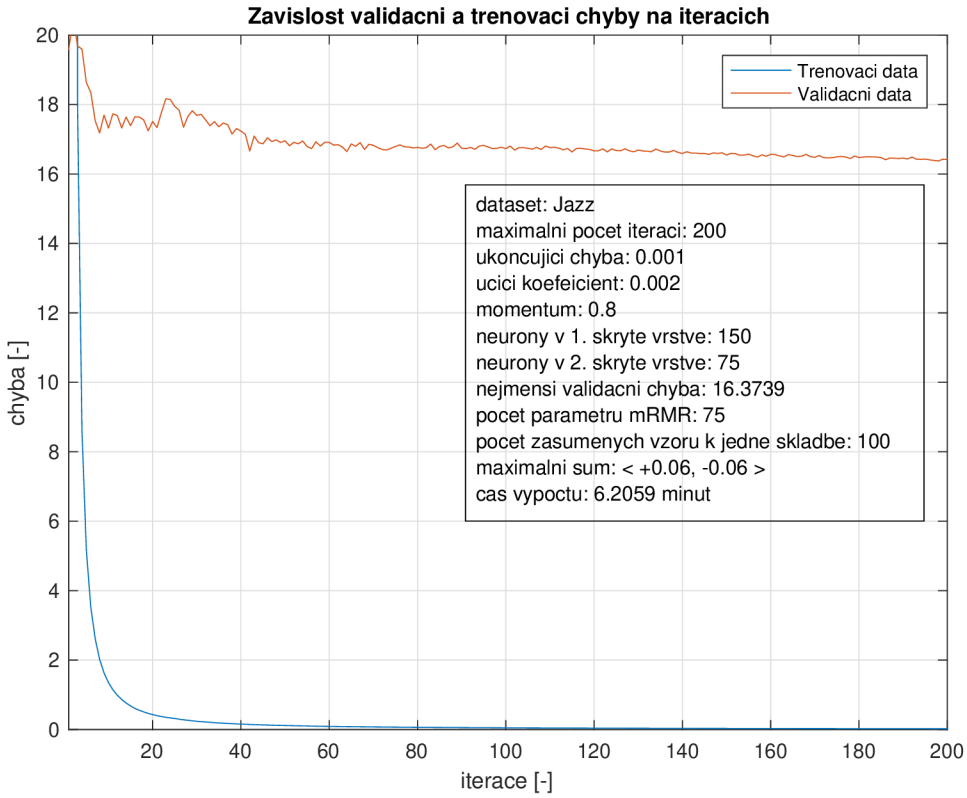

Vzdalenosti vsech skladeb pro styl HHPR
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

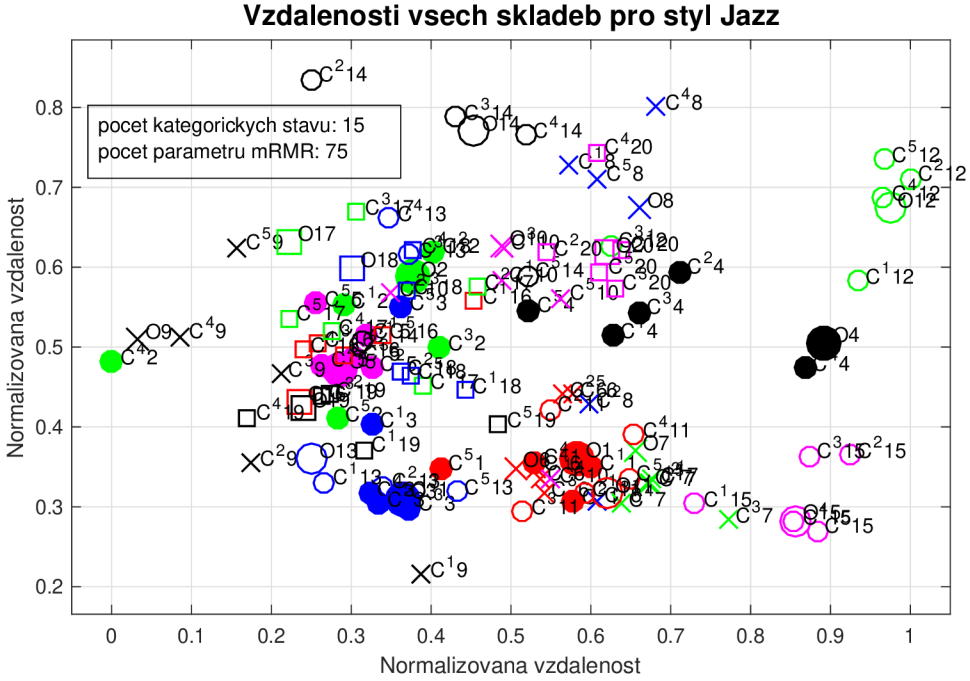

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Jazz
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 14.3631
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 1.6216 minut
chyba [-]
iterace [-]

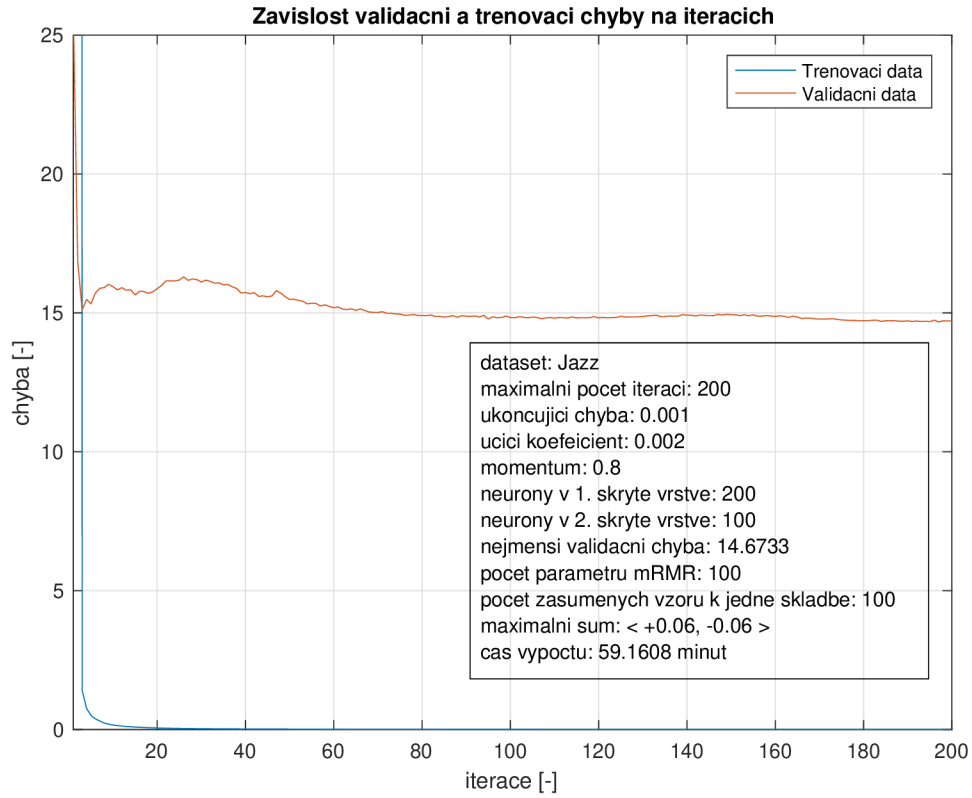

Vzdalenosti vsech skladeb pro styl Jazz
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

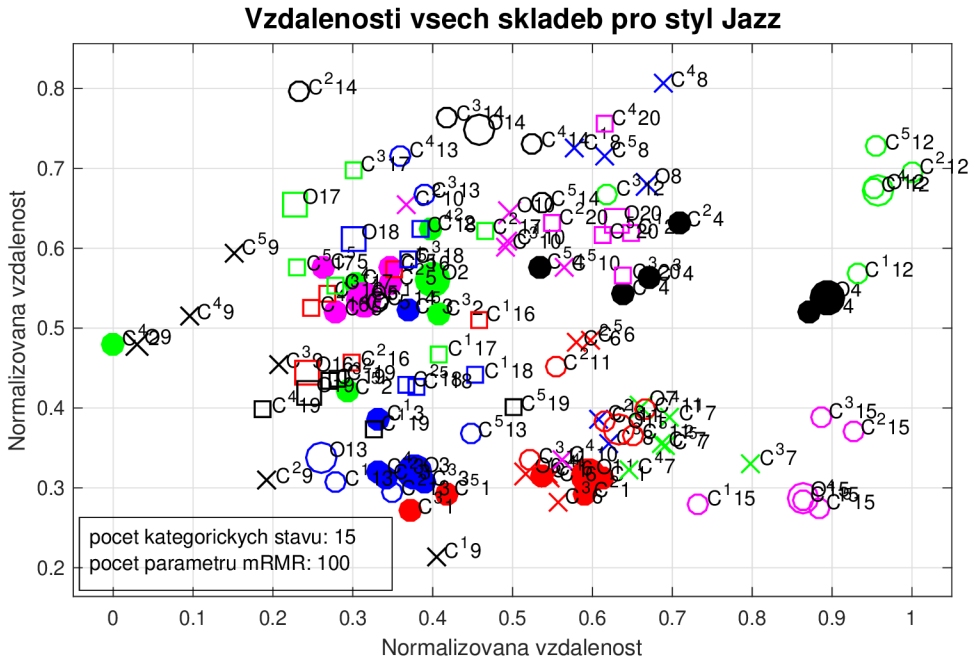

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Jazz
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 14.054
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 2.9177 minut
chyba [-]
iterace [-]

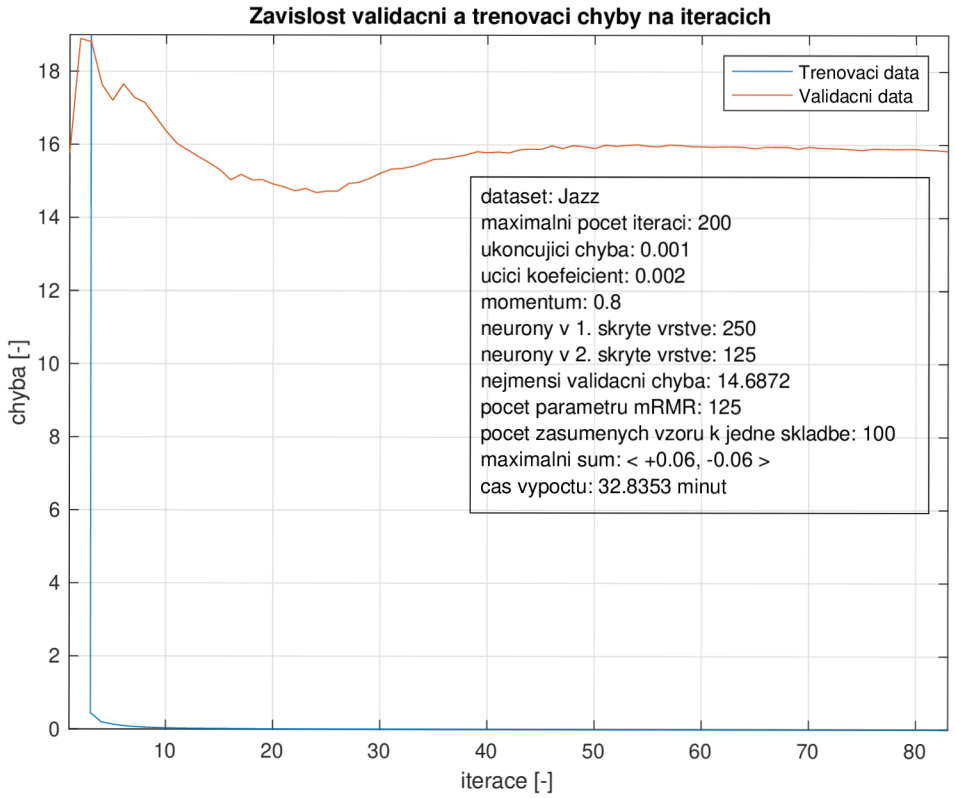

Vzdalenosti vsech skladeb pro styl Jazz
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

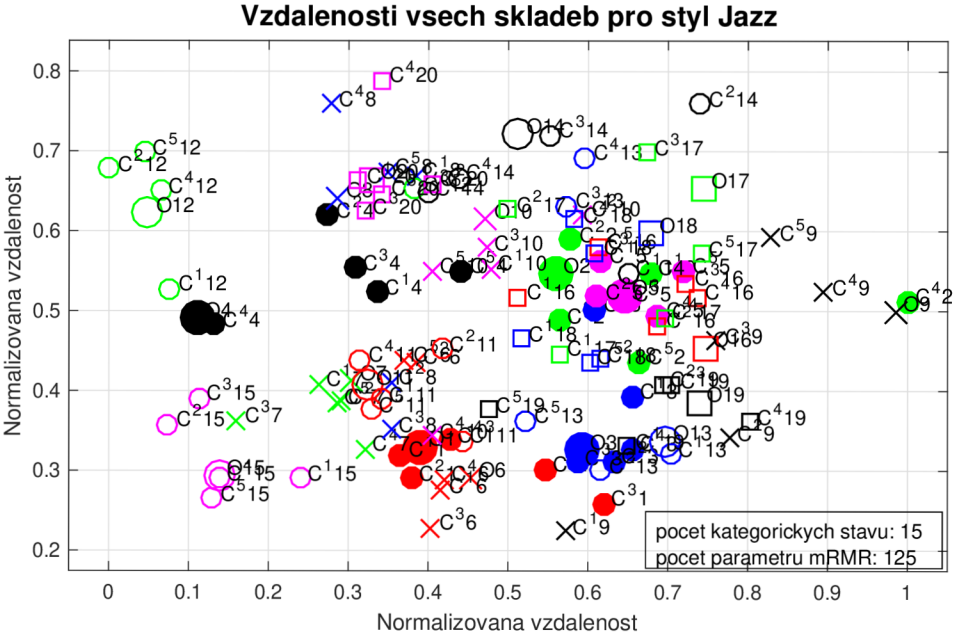

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Jazz
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 16.3739
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 6.2059 minut
chyba [-]
iterace [-]

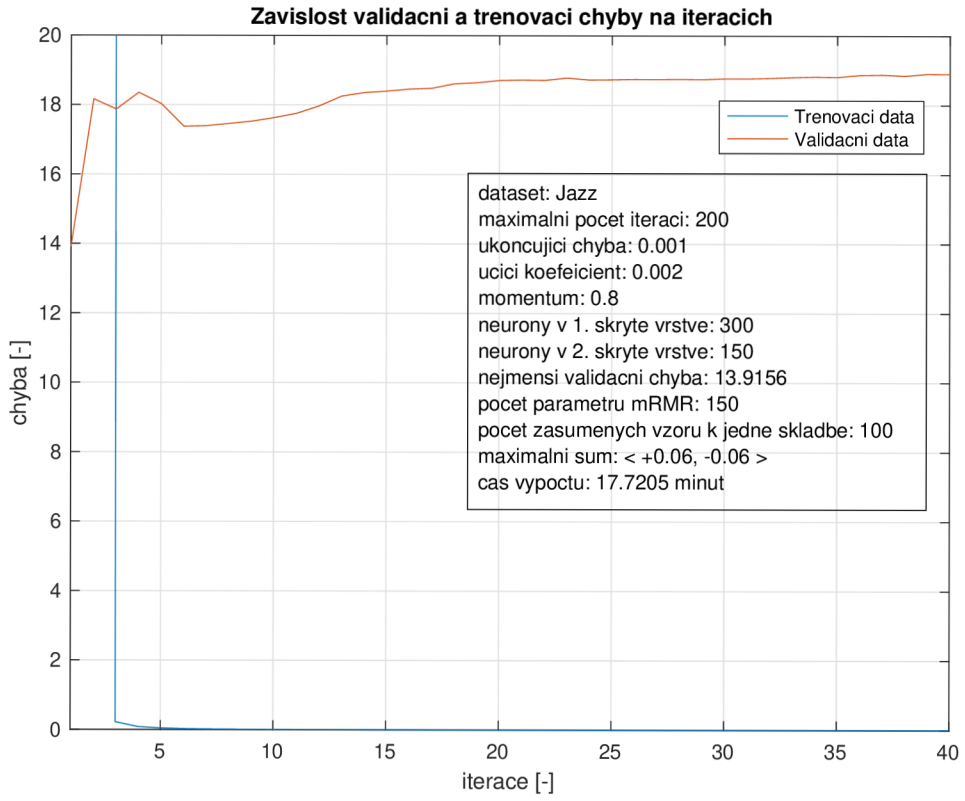

Vzdalenosti vsech skladeb pro styl Jazz
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost

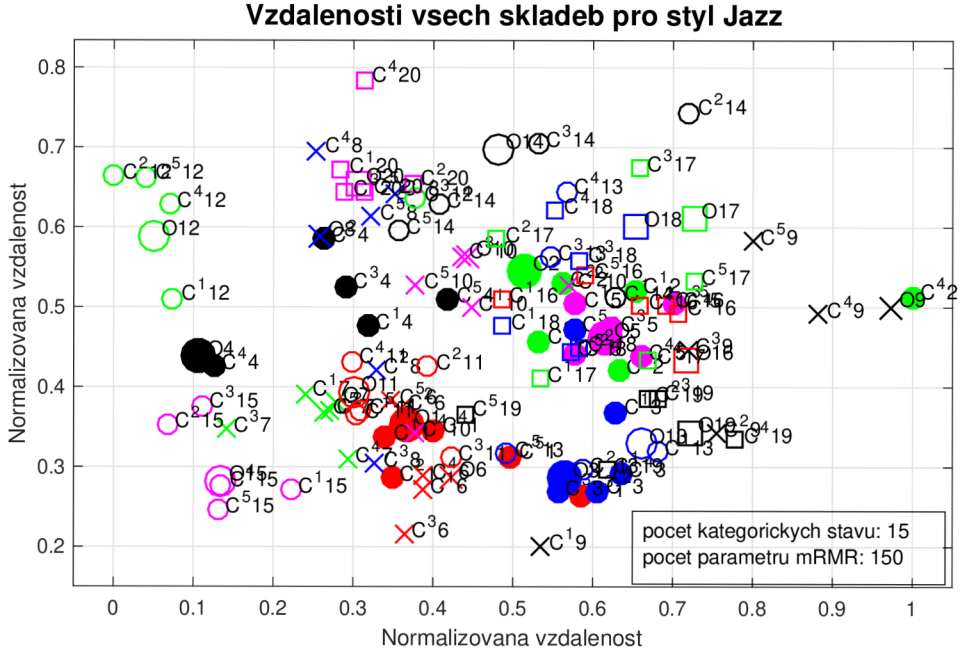

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Jazz
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 14.6733
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 59.1608 minut
chyba [-]
iterace [-]
0
5
10
15
20
25
20
40
60
80
100
120
140
160
180
200

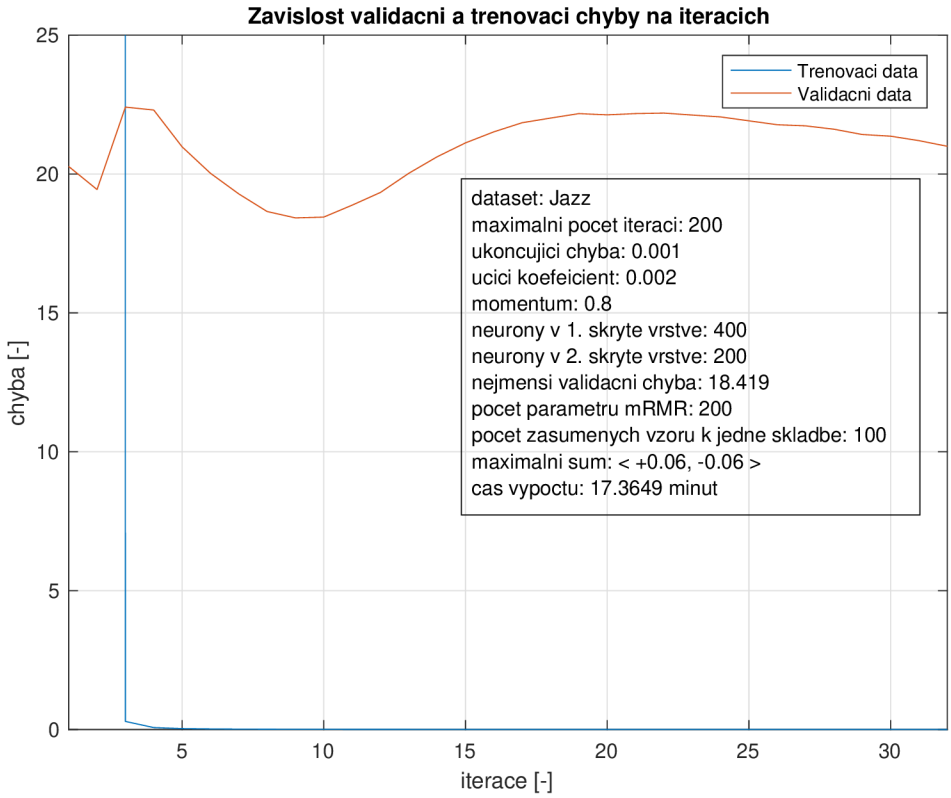

Vzdalenosti vsech skladeb pro styl Jazz
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

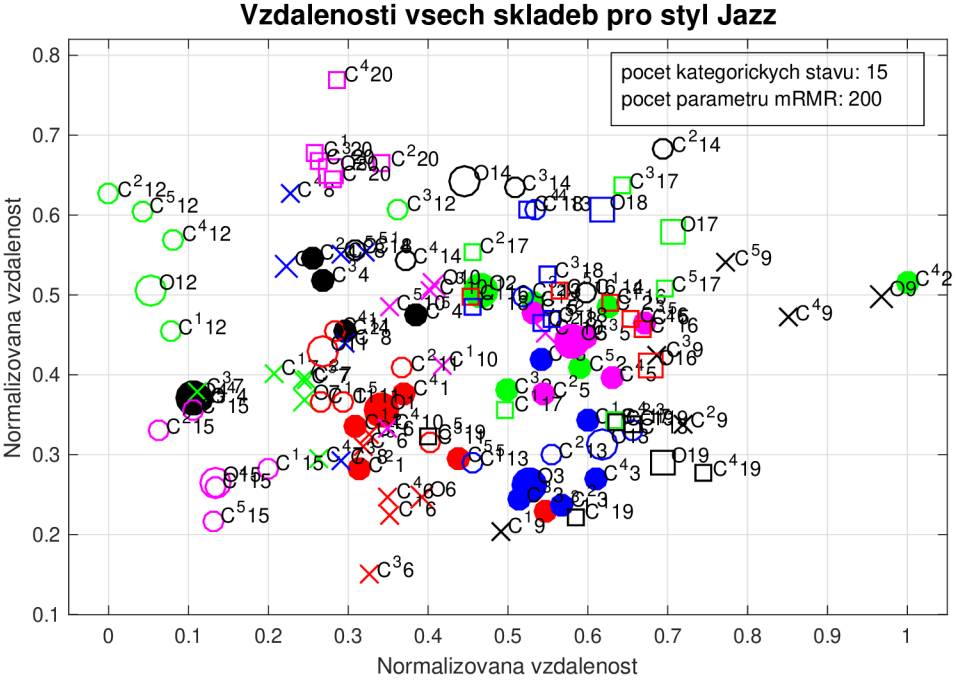

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Jazz
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 14.6872
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 32.8353 minut
chyba [-]
iterace [-]

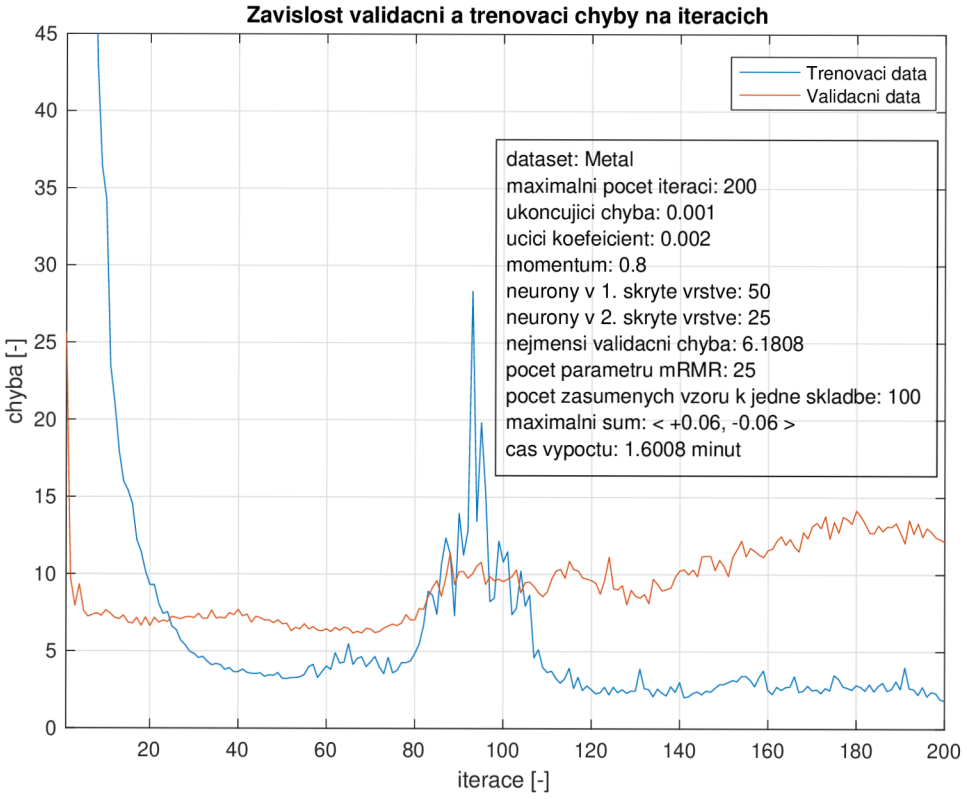

Vzdalenosti vsech skladeb pro styl Jazz
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost

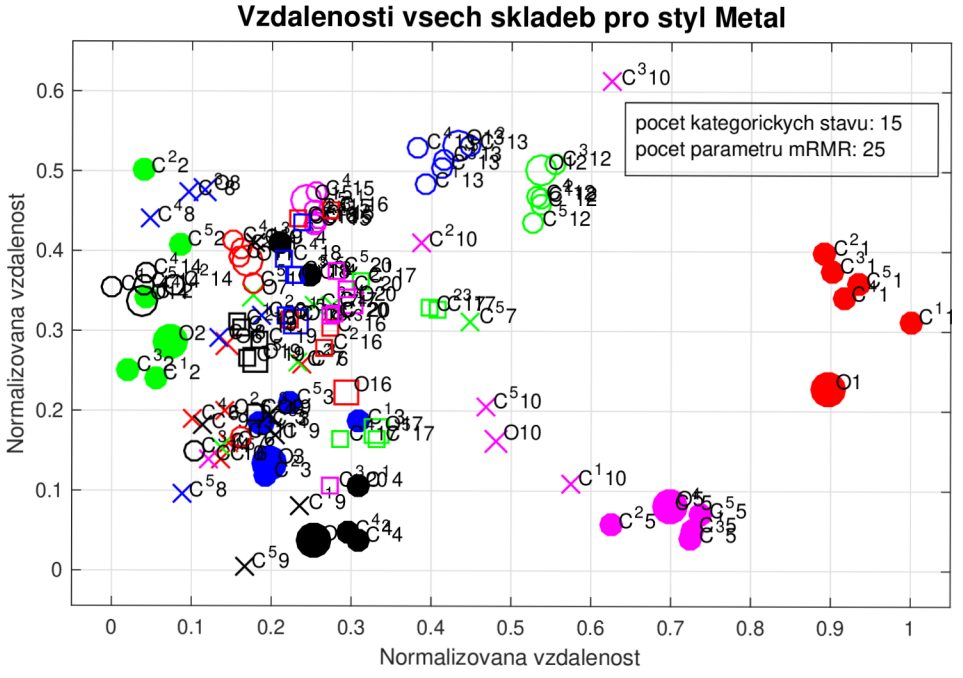

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Jazz
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 300
neurony v 2. skryte vrstve: 150
nejmensi validacni chyba: 13.9156
pocet parametru mRMR: 150
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 17.7205 minut
chyba [-]
iterace [-]

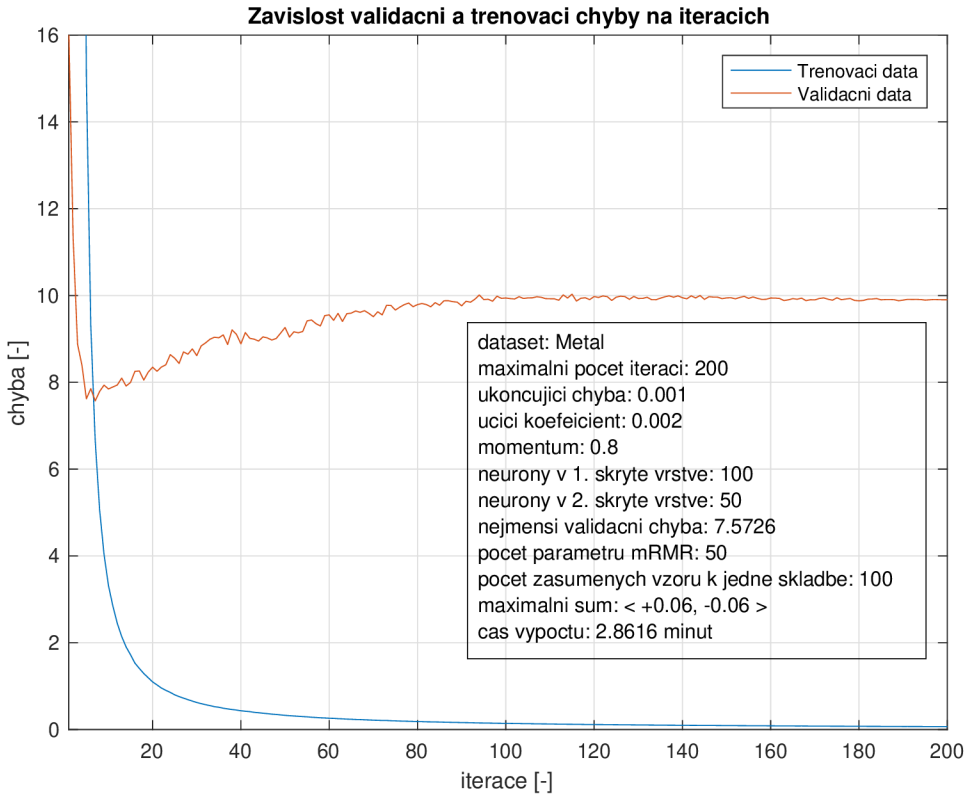

Vzdalenosti vsech skladeb pro styl Jazz
pocet kategorickych stavu: 15
pocet parametru mRMR: 150
Normalizovana vzdalenost
Normalizovana vzdalenost

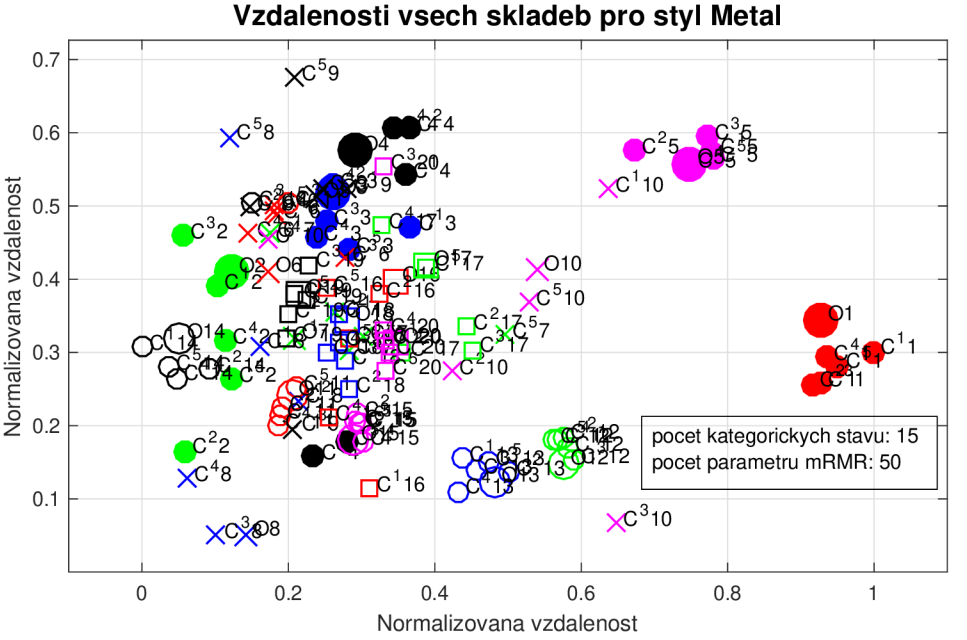

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Jazz
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 18.419
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 17.3649 minut
chyba [-]
iterace [-]
0
5
10
15
20
25
30

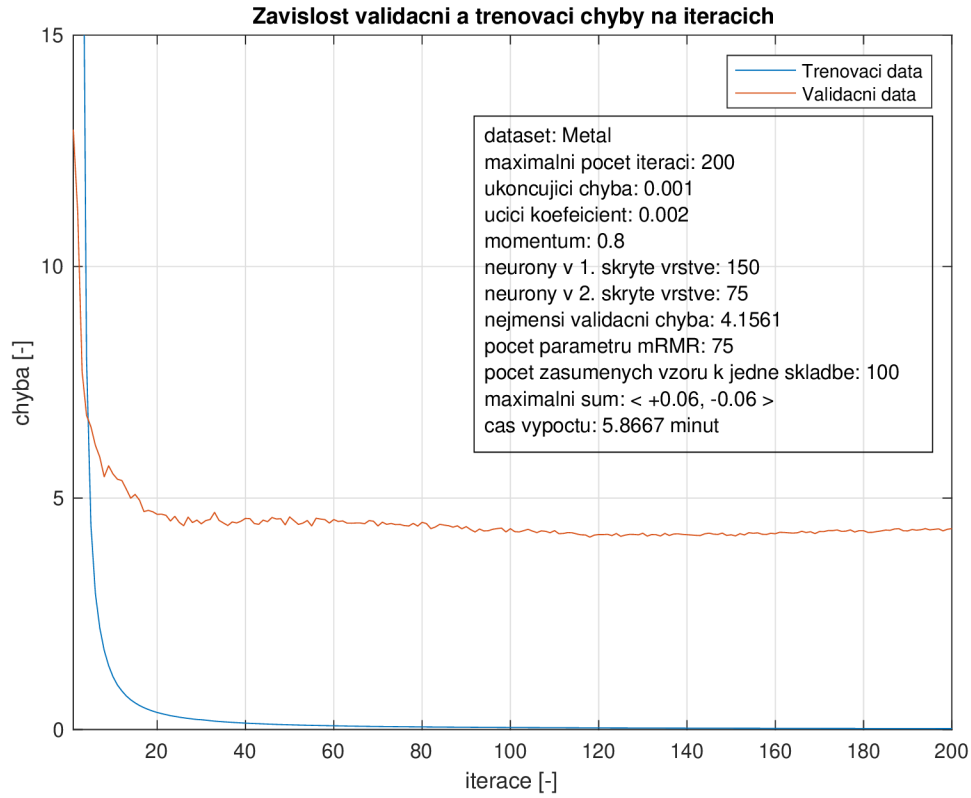

Vzdalenosti vsech skladeb pro styl Jazz
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

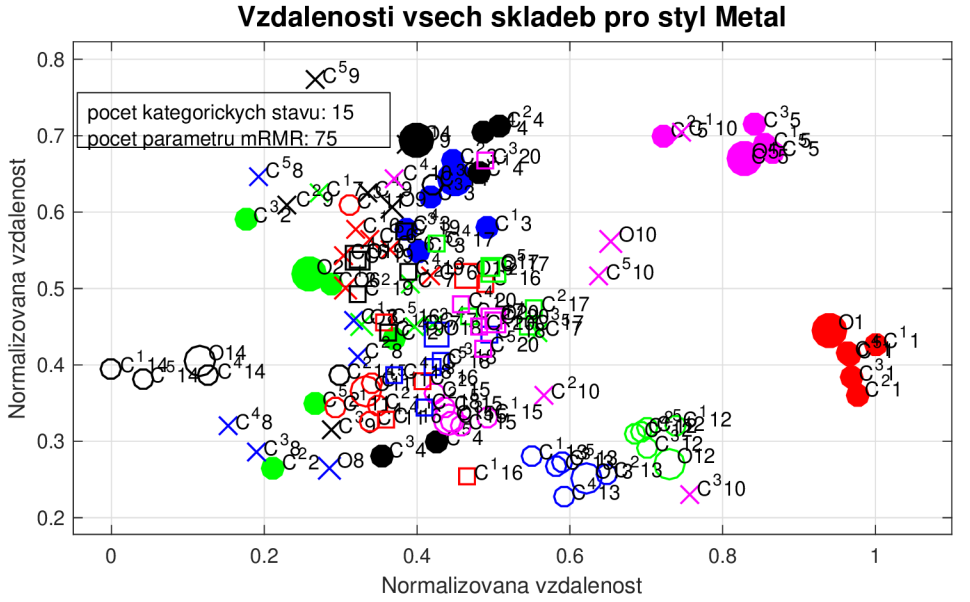

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Metal
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 6.1808
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 1.6008 minut
chyba [-]
iterace [-]

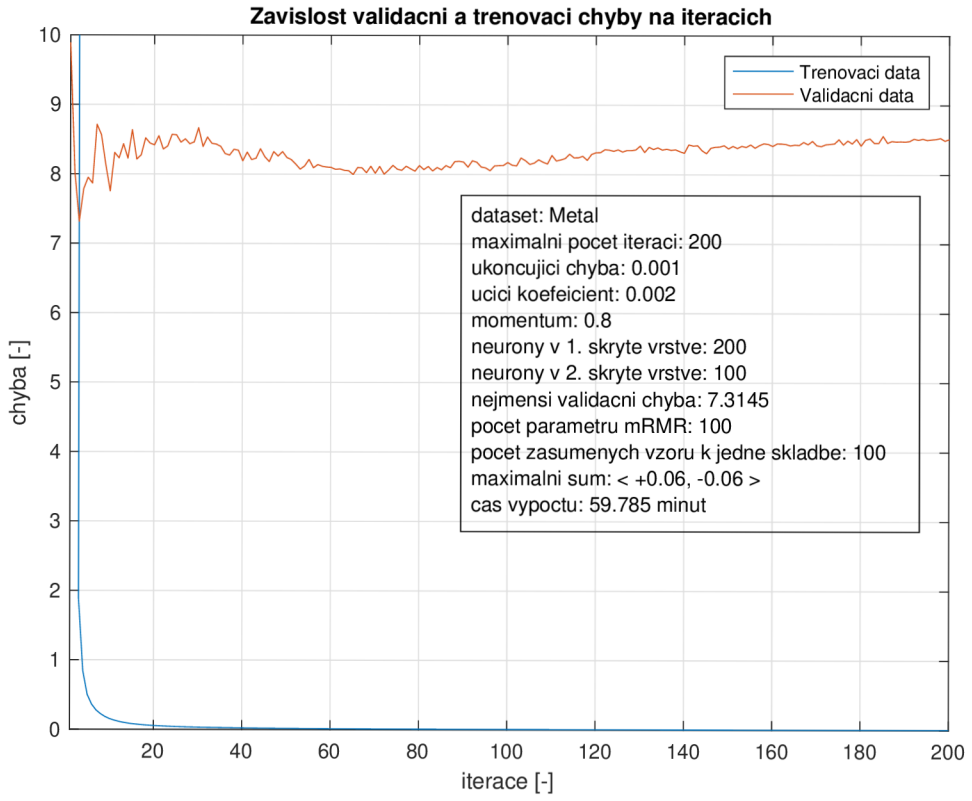

Vzdalenosti vsech skladeb pro styl Metal
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

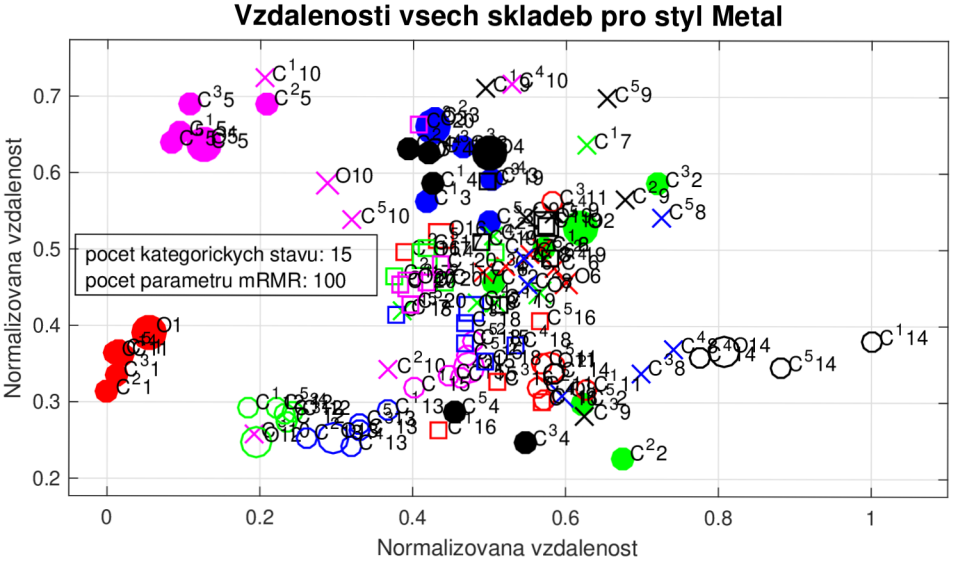

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Metal
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 7.5726
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 2.8616 minut
chyba [-]
iterace [-]

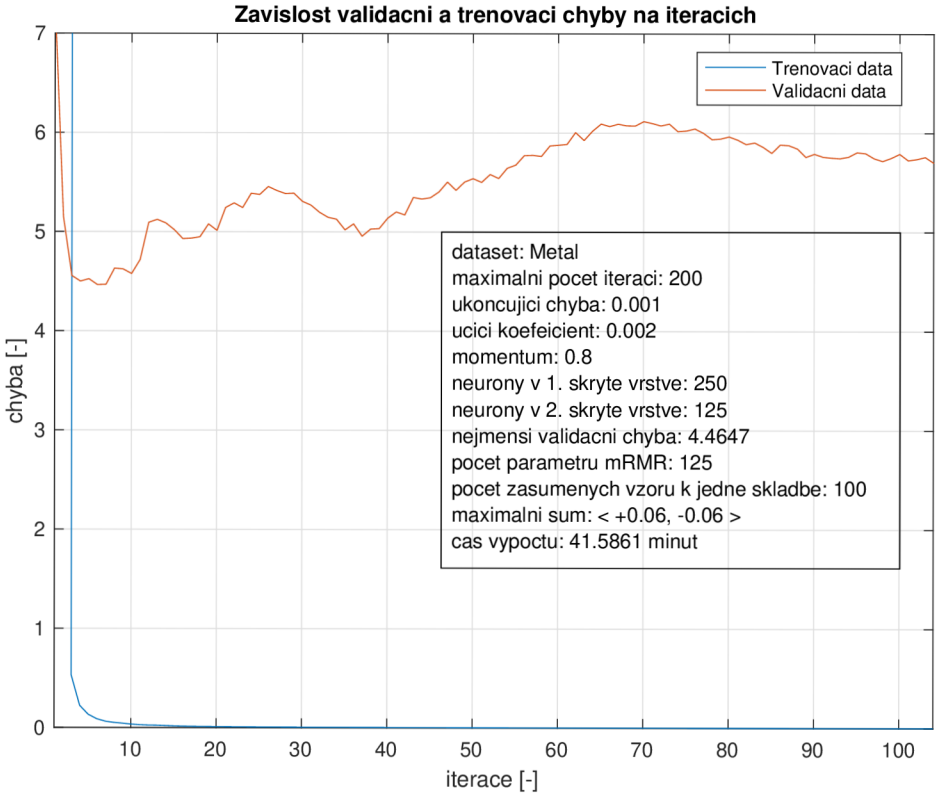

Vzdalenosti vsech skladeb pro styl Metal
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

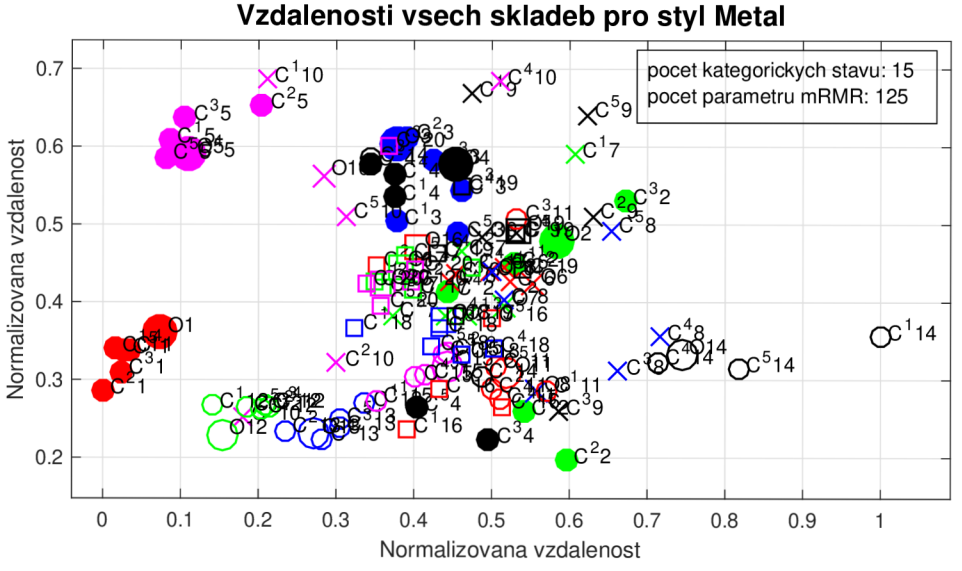

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Metal
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 4.1561
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 5.8667 minut
chyba [-]
iterace [-]
0
5
10
15
20
40
60
80
100
120
140
160
180
200

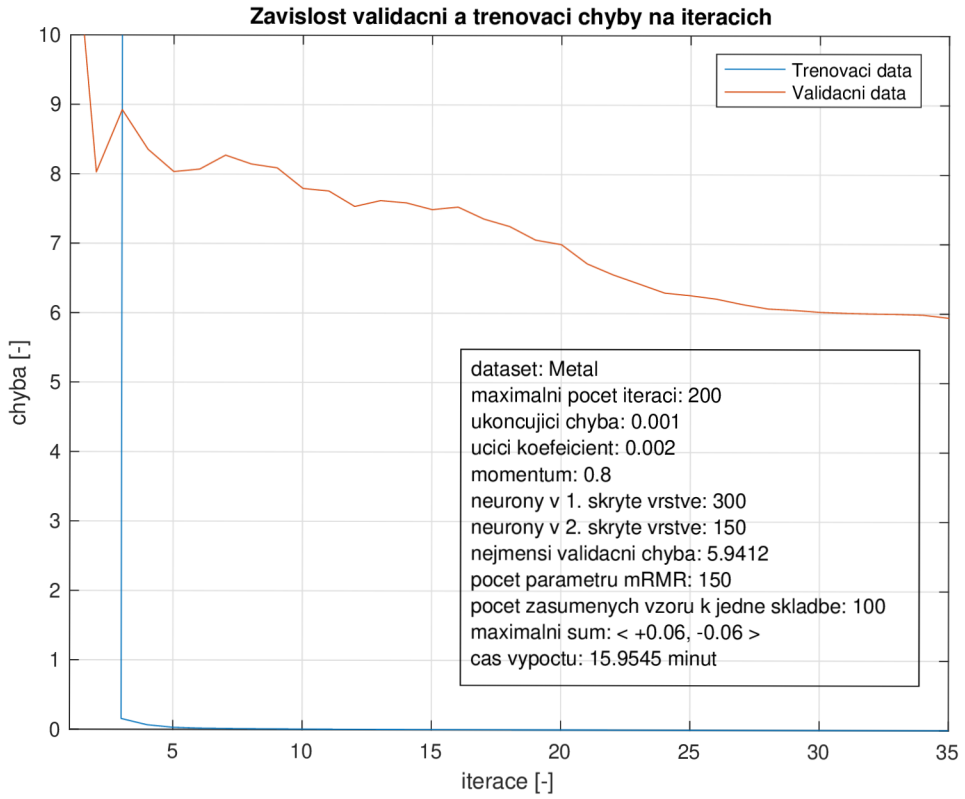

Vzdalenosti vsech skladeb pro styl Metal
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
0.2
0.4
0.6
0.8
1

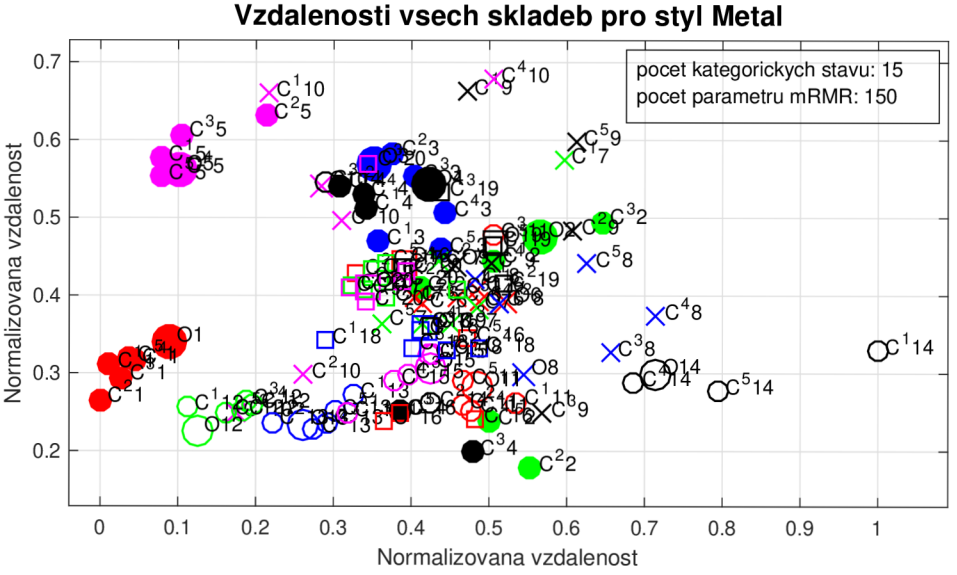

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Metal
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 7.3145
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 59.785 minut
chyba [-]
iterace [-]

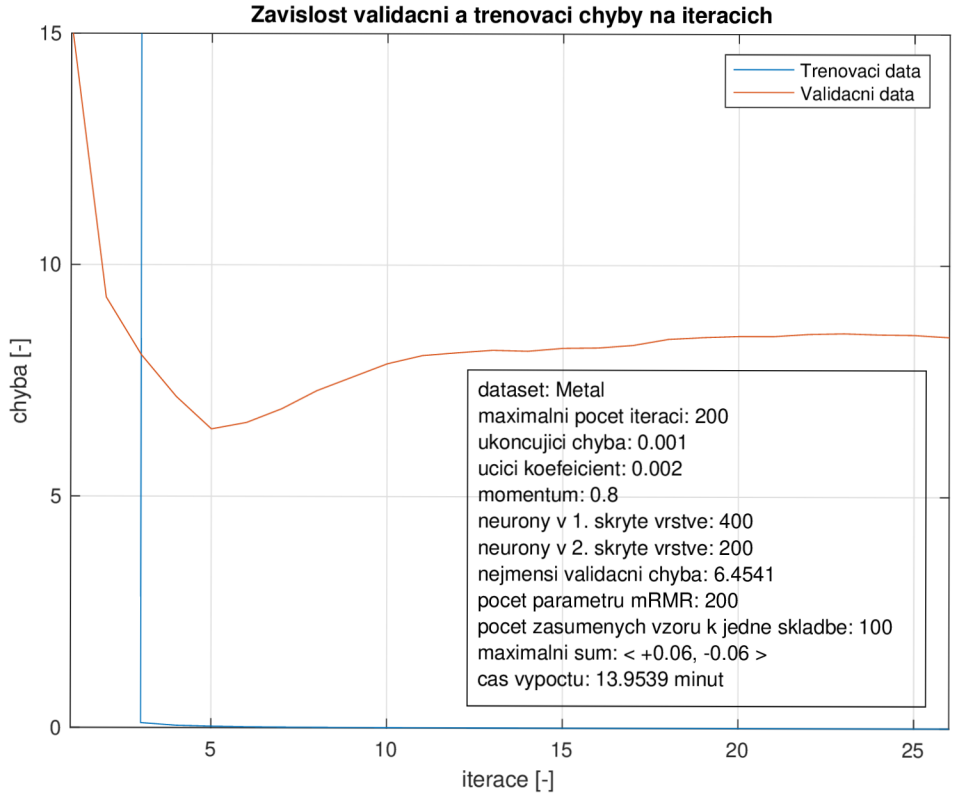

Vzdalenosti vsech skladeb pro styl Metal
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost

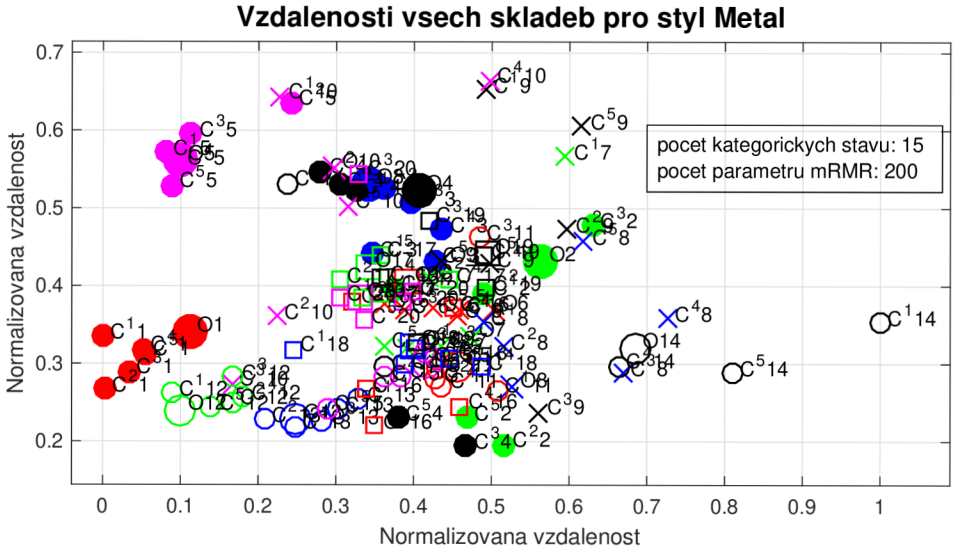

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Metal
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 4.4647
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 41.5861 minut
chyba [-]
iterace [-]

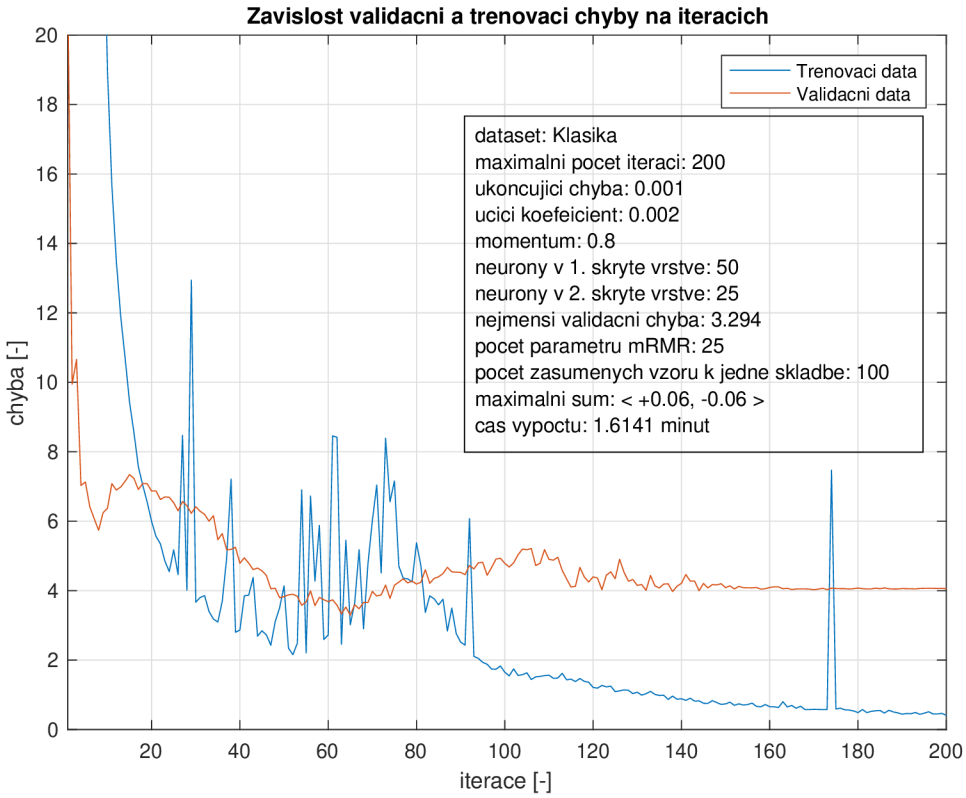

Vzdalenosti vsech skladeb pro styl Metal
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost

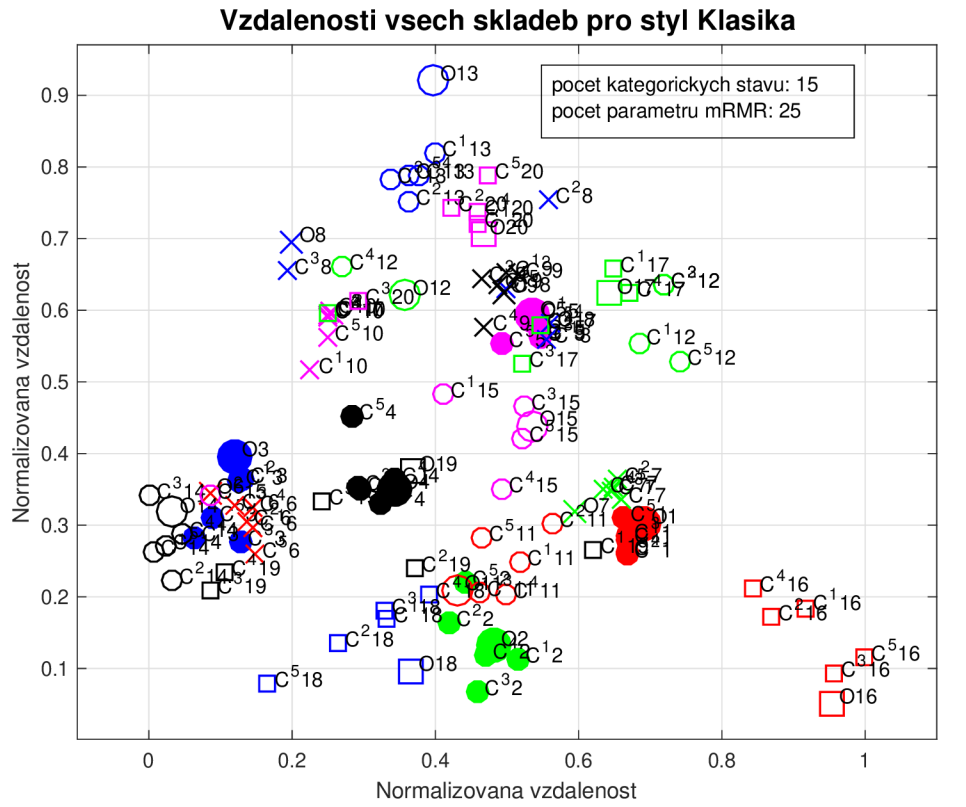

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Metal
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 300
neurony v 2. skryte vrstve: 150
nejmensi validacni chyba: 5.9412
pocet parametru mRMR: 150
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 15.9545 minut
chyba [-]
iterace [-]

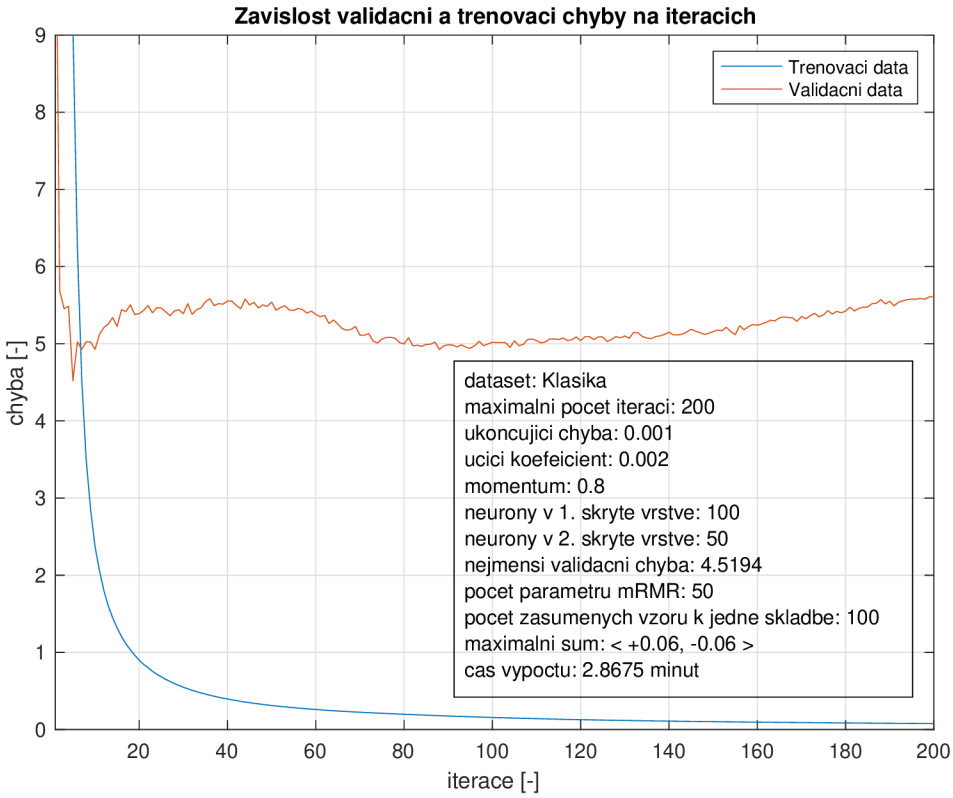

Vzdalenosti vsech skladeb pro styl Metal
pocet kategorickych stavu: 15
pocet parametru mRMR: 150
Normalizovana vzdalenost
Normalizovana vzdalenost

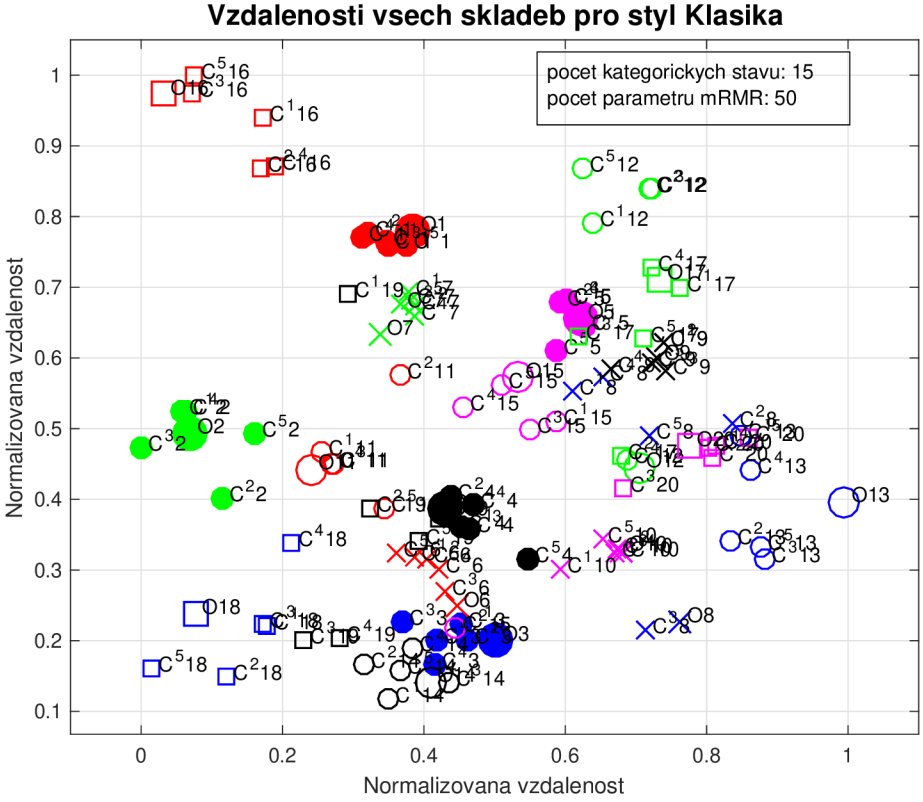

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
chyba [-]
iterace [-]
dataset: Metal
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 6.4541
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 13.9539 minut

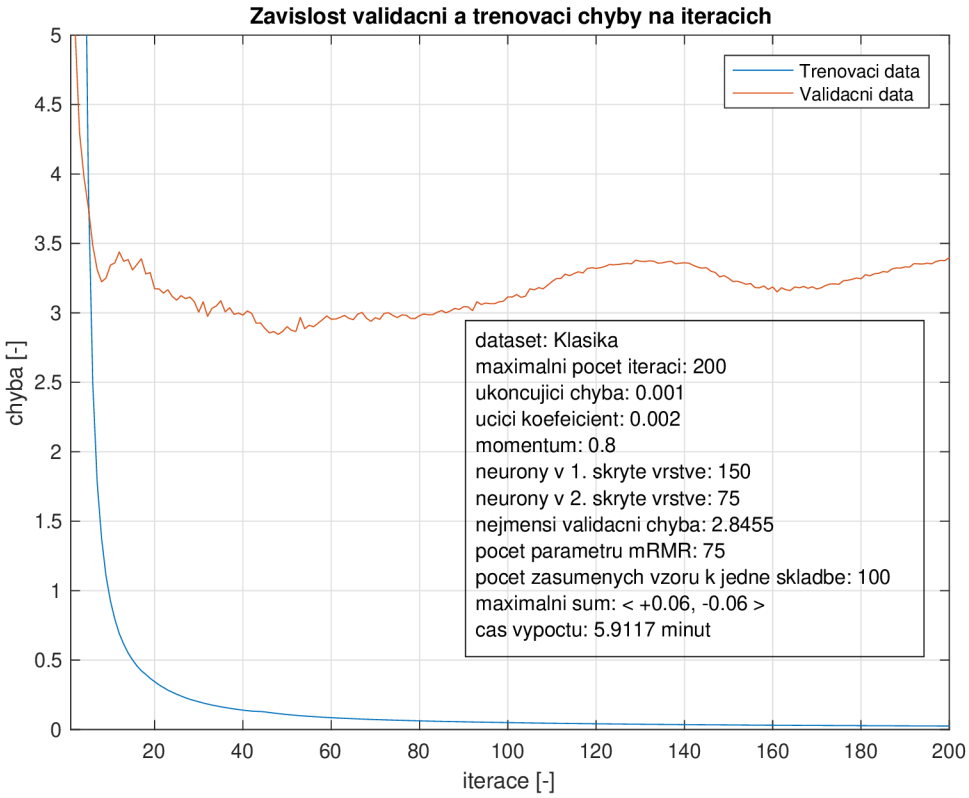

Vzdalenosti vsech skladeb pro styl Metal
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

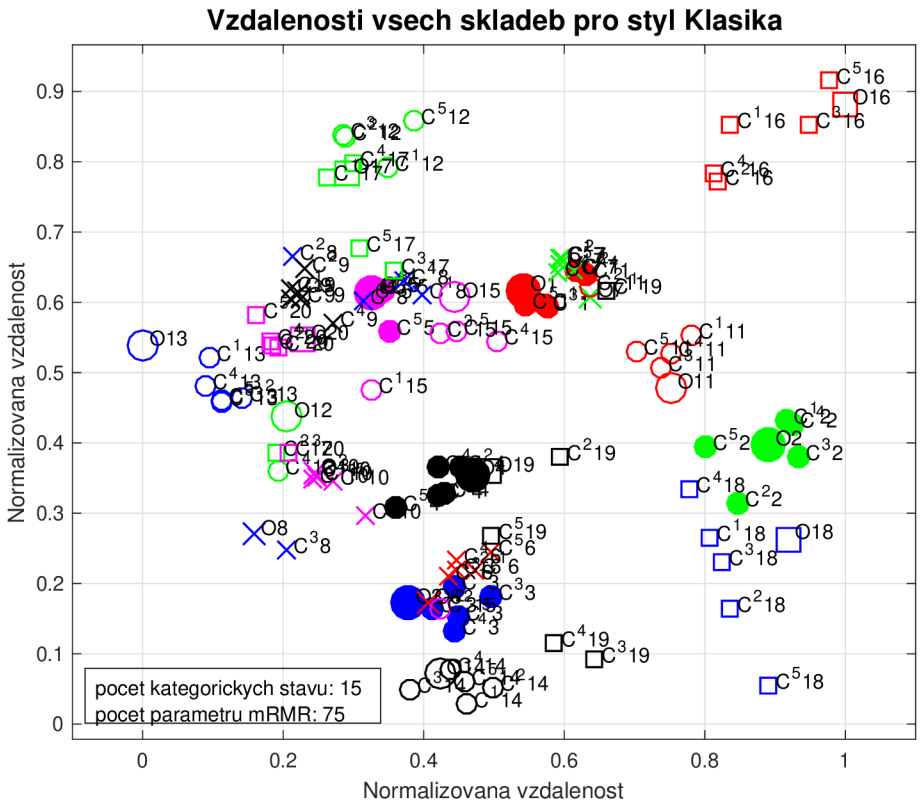

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Klasika
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 3.294
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 1.6141 minut
chyba [-]
iterace [-]

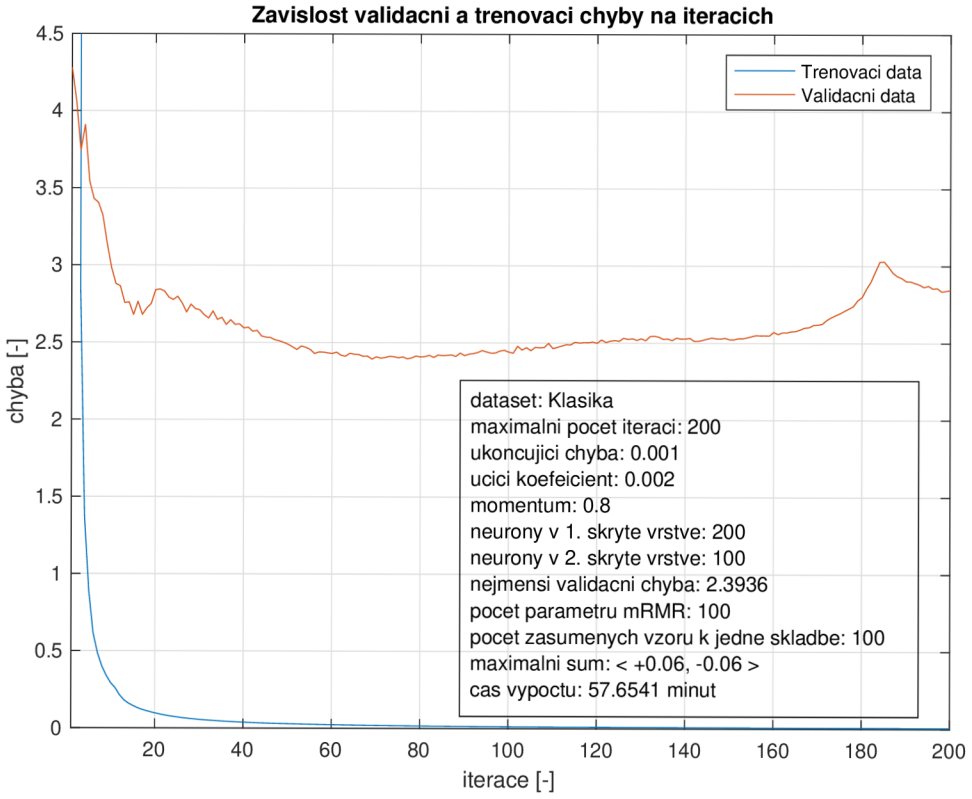

Vzdalenosti vsech skladeb pro styl Klasika
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

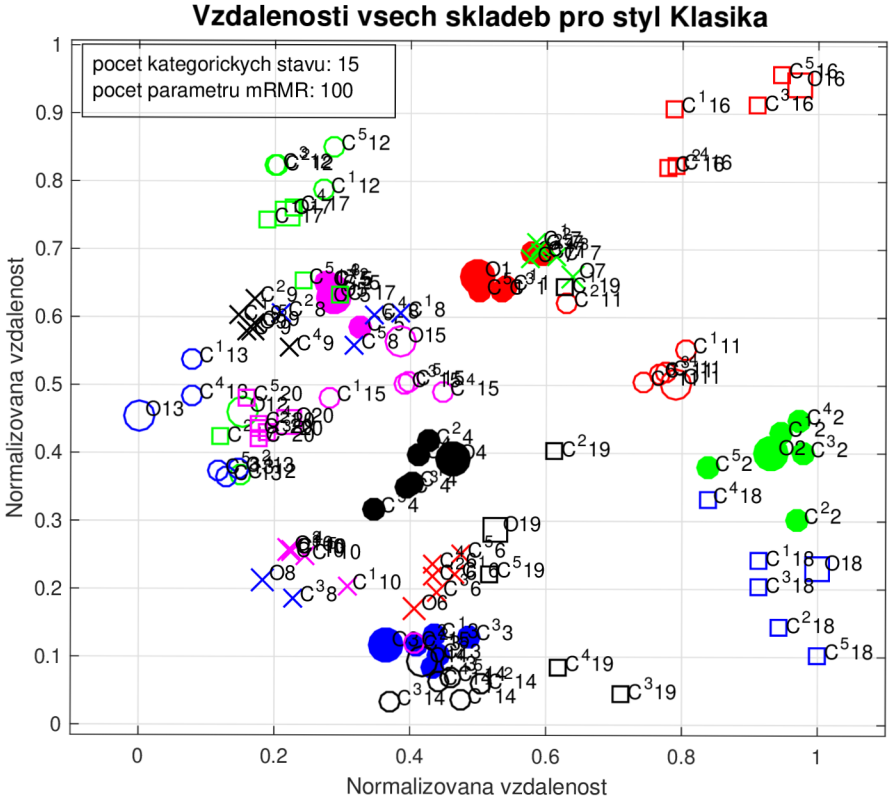

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Klasika
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 4.5194
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 2.8675 minut
chyba [-]
iterace [-]

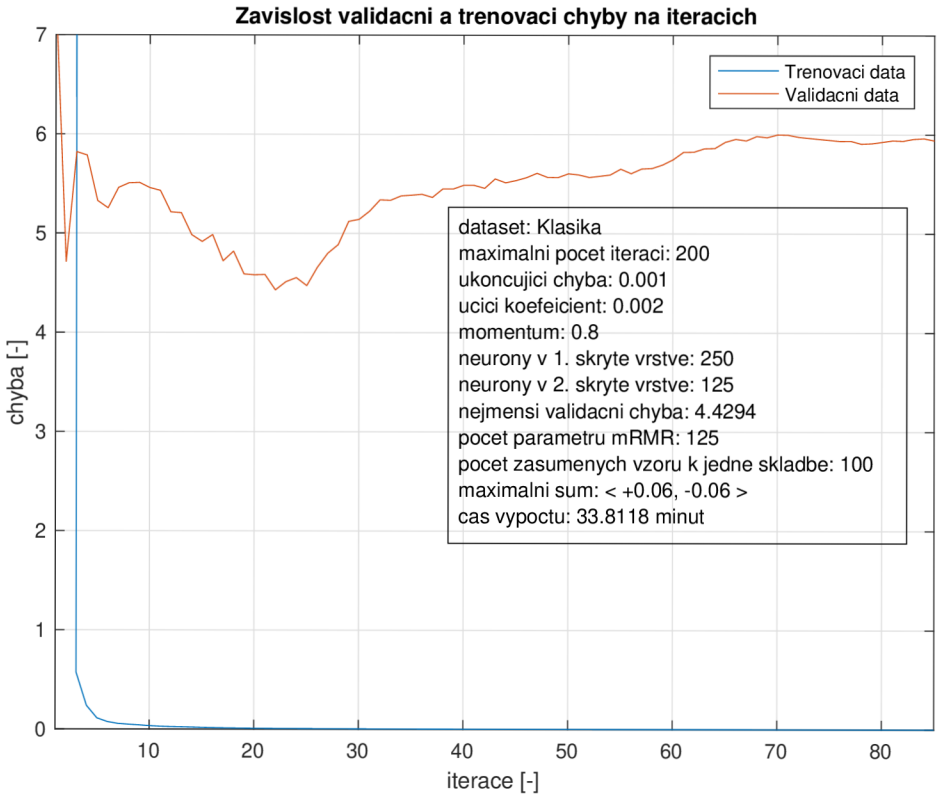

Vzdalenosti vsech skladeb pro styl Klasika
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

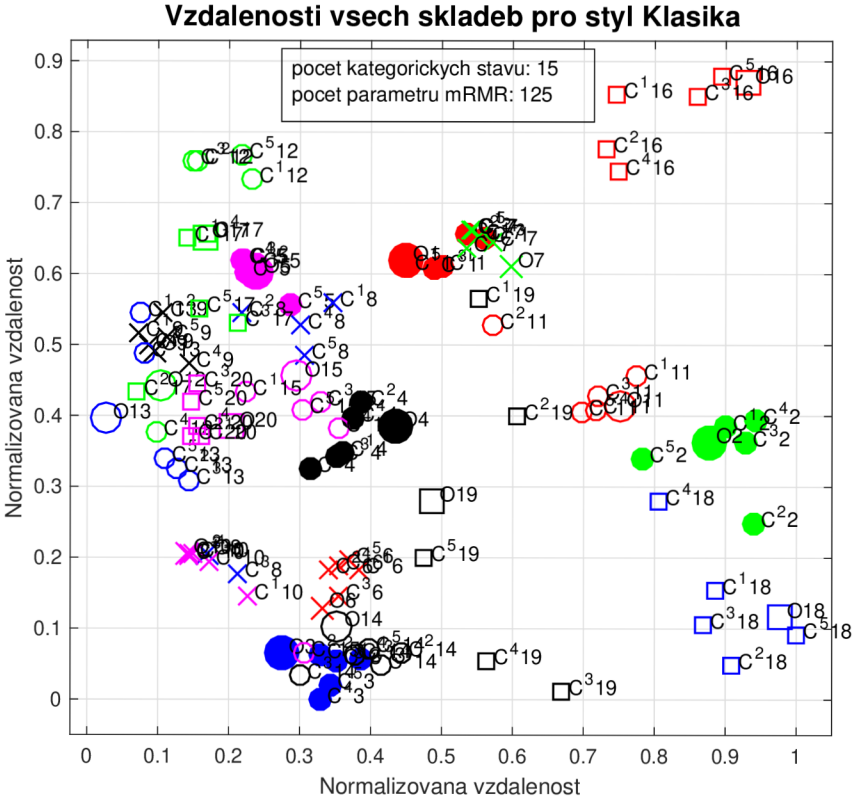

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Klasika
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 2.8455
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 5.9117 minut
chyba [-]
iterace [-]

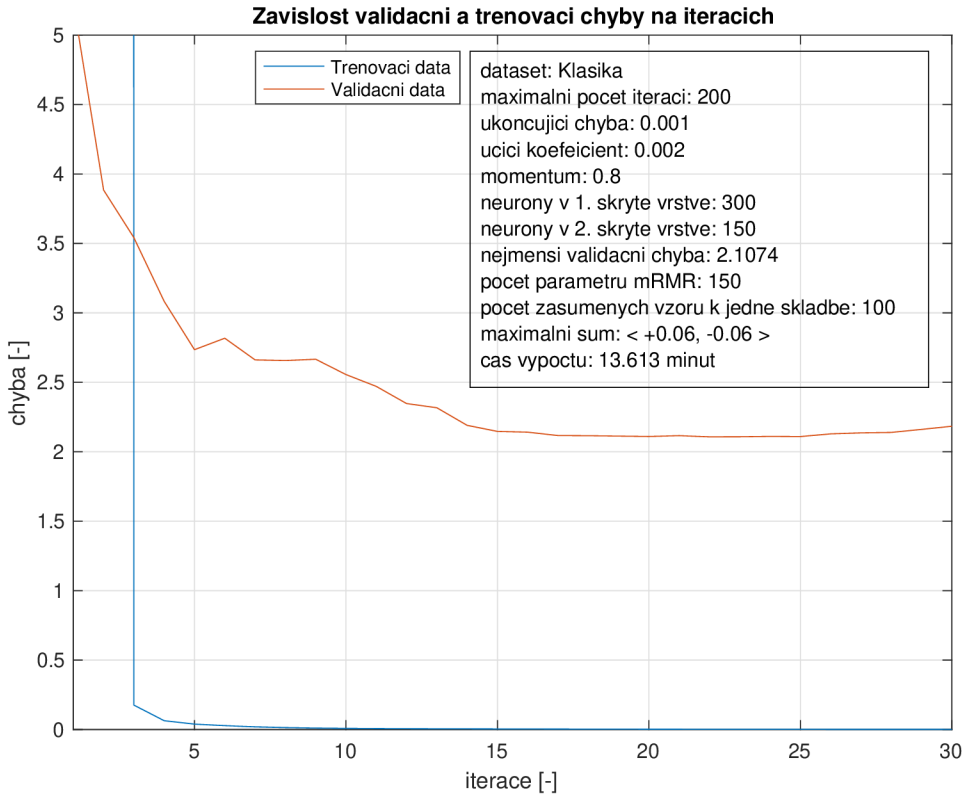

Vzdalenosti vsech skladeb pro styl Klasika
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost

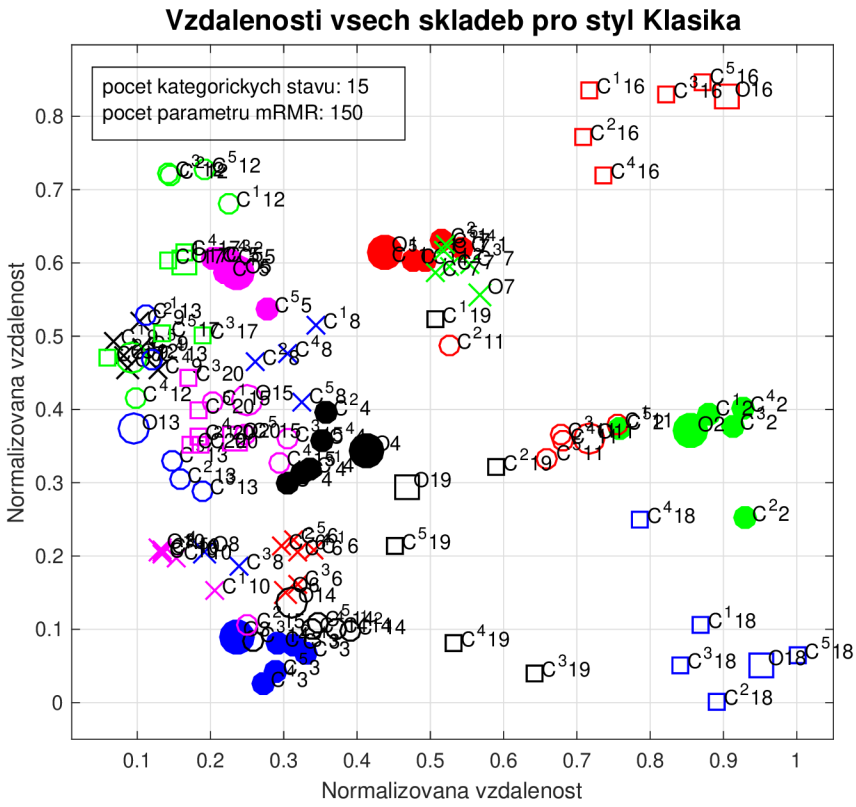

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Klasika
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 2.3936
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 57.6541 minut
chyba [-]
iterace [-]

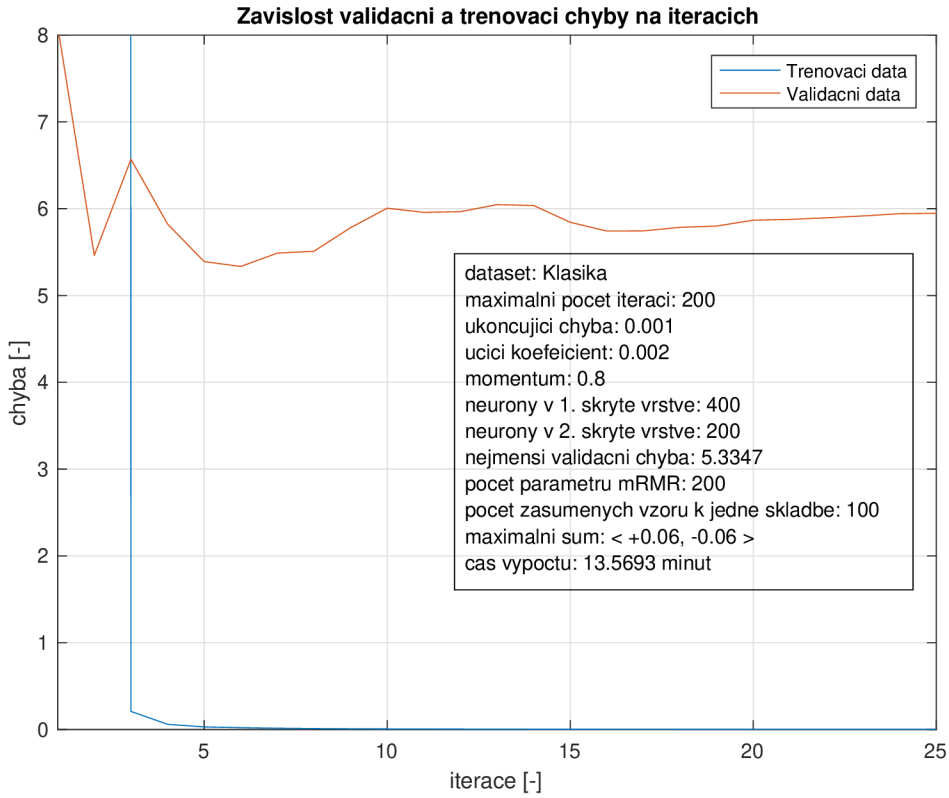

Vzdalenosti vsech skladeb pro styl Klasika
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost

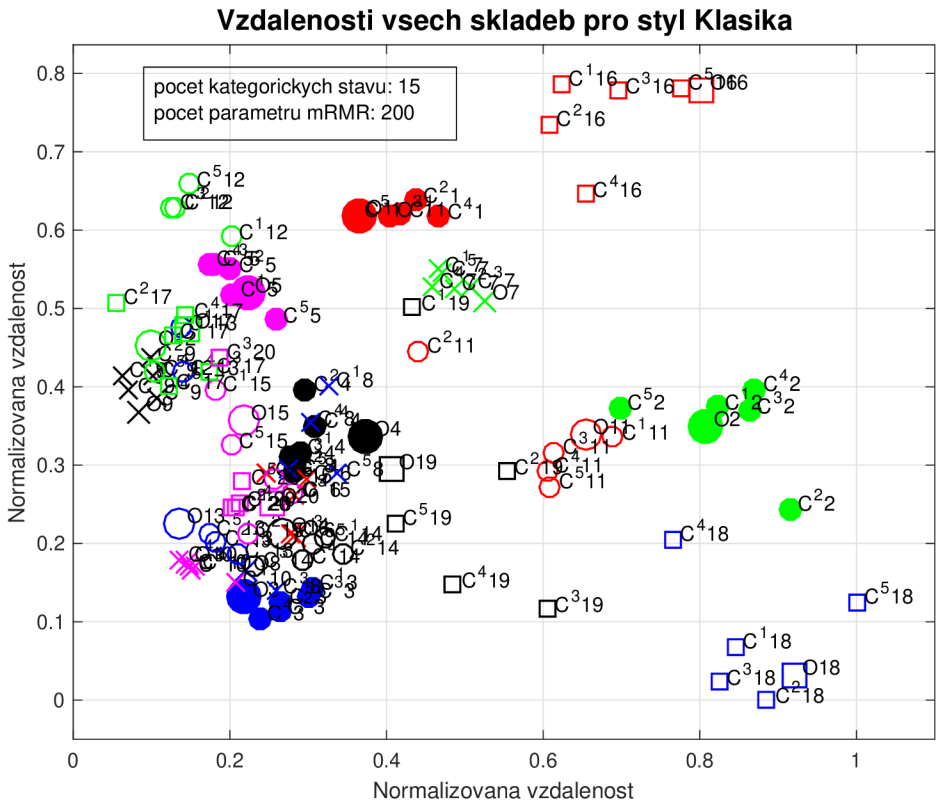

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Klasika
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 4.4294
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 33.8118 minut
chyba [-]
iterace [-]

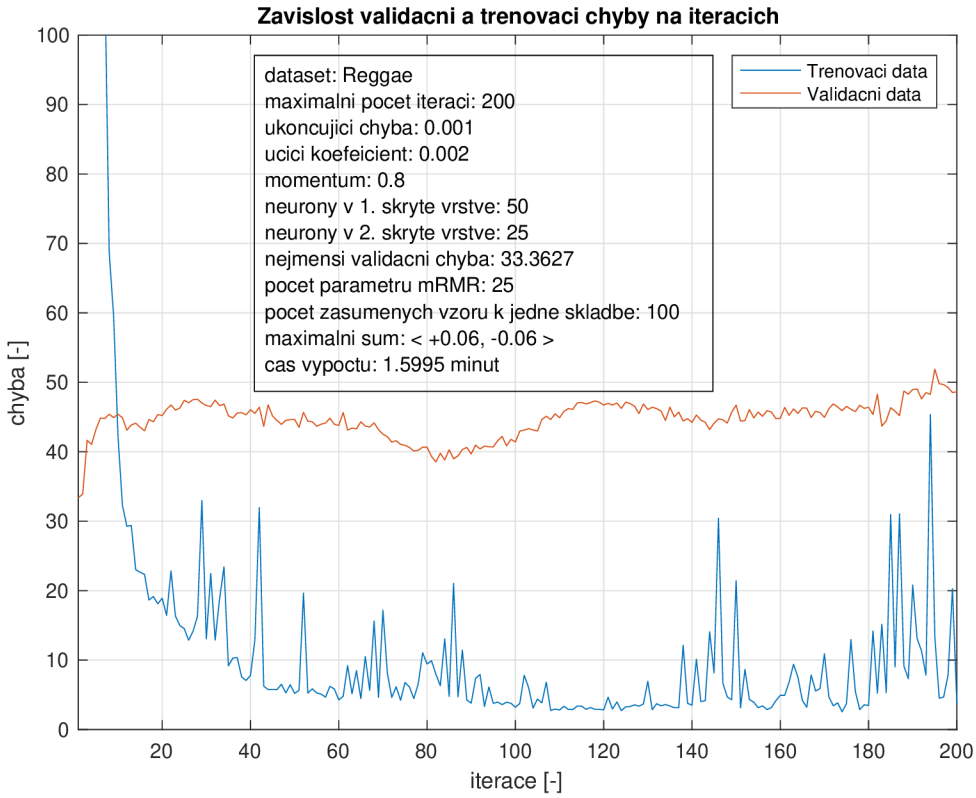

Vzdalenosti vsech skladeb pro styl Klasika
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost

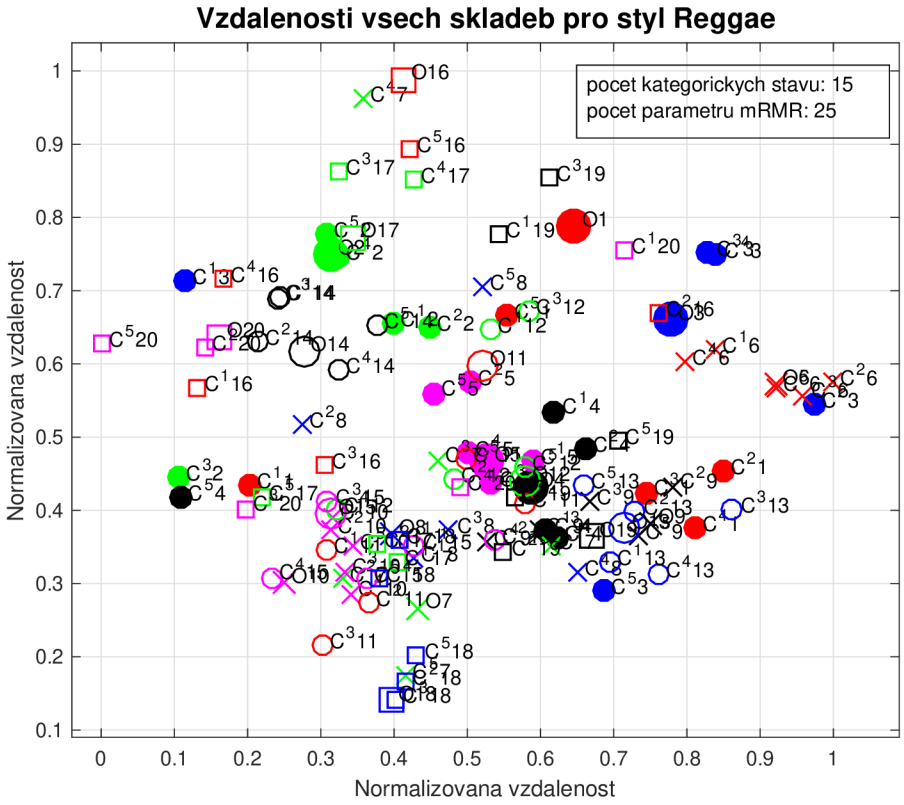

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Klasika
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 300
neurony v 2. skryte vrstve: 150
nejmensi validacni chyba: 2.1074
pocet parametru mRMR: 150
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 13.613 minut
chyba [-]
iterace [-]

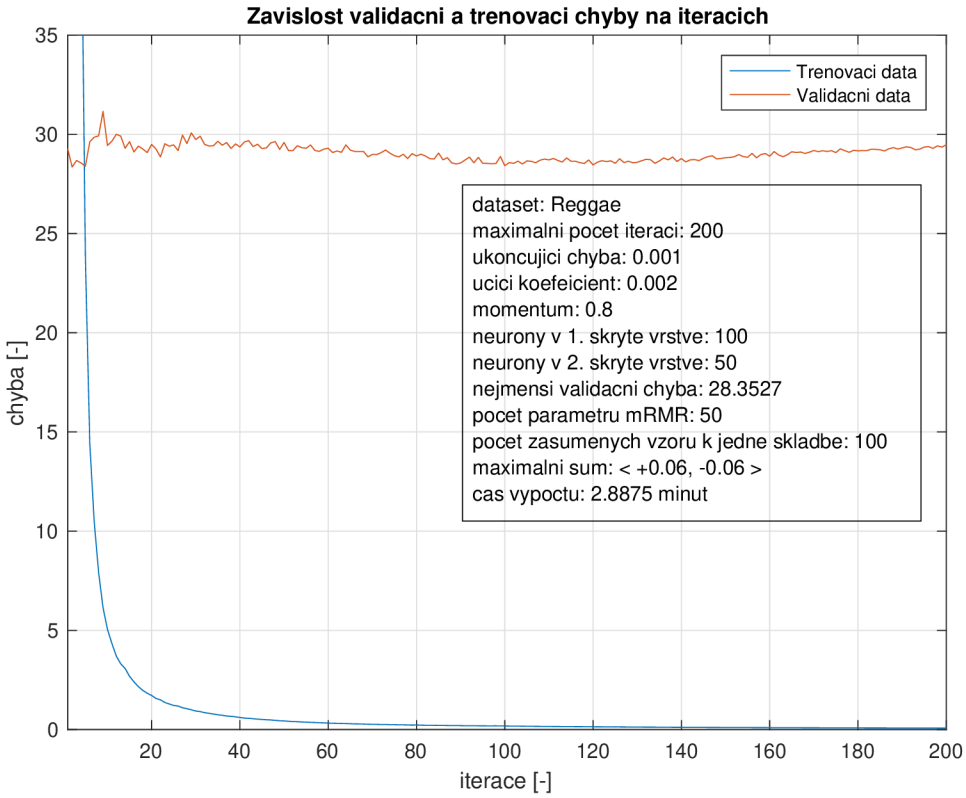

Vzdalenosti vsech skladeb pro styl Klasika
pocet kategorickych stavu: 15
pocet parametru mRMR: 150
Normalizovana vzdalenost
Normalizovana vzdalenost

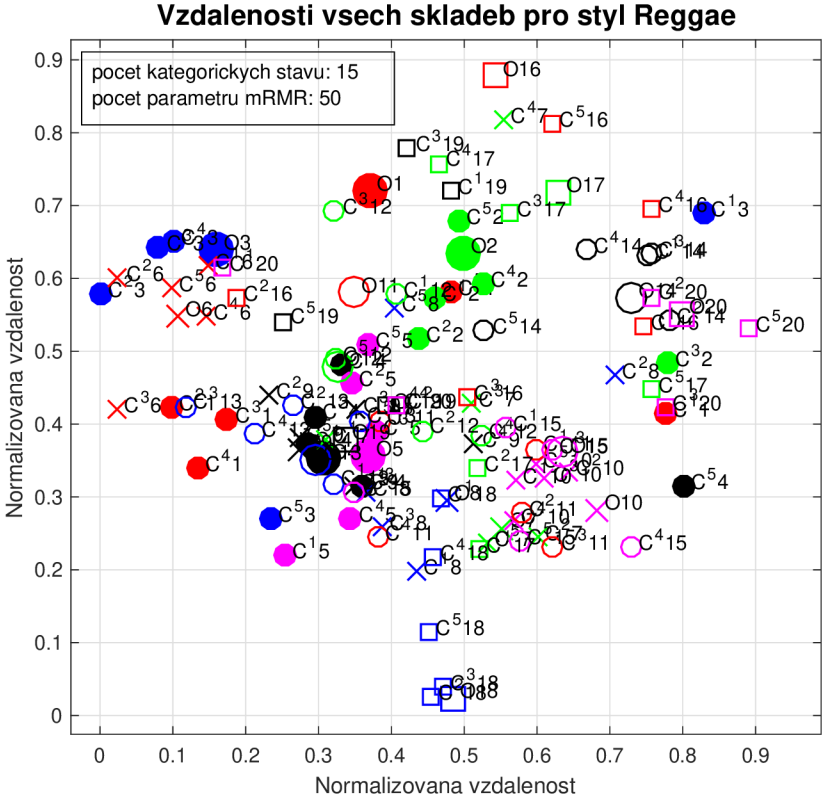

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Klasika
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 5.3347
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 13.5693 minut
chyba [-]
iterace [-]

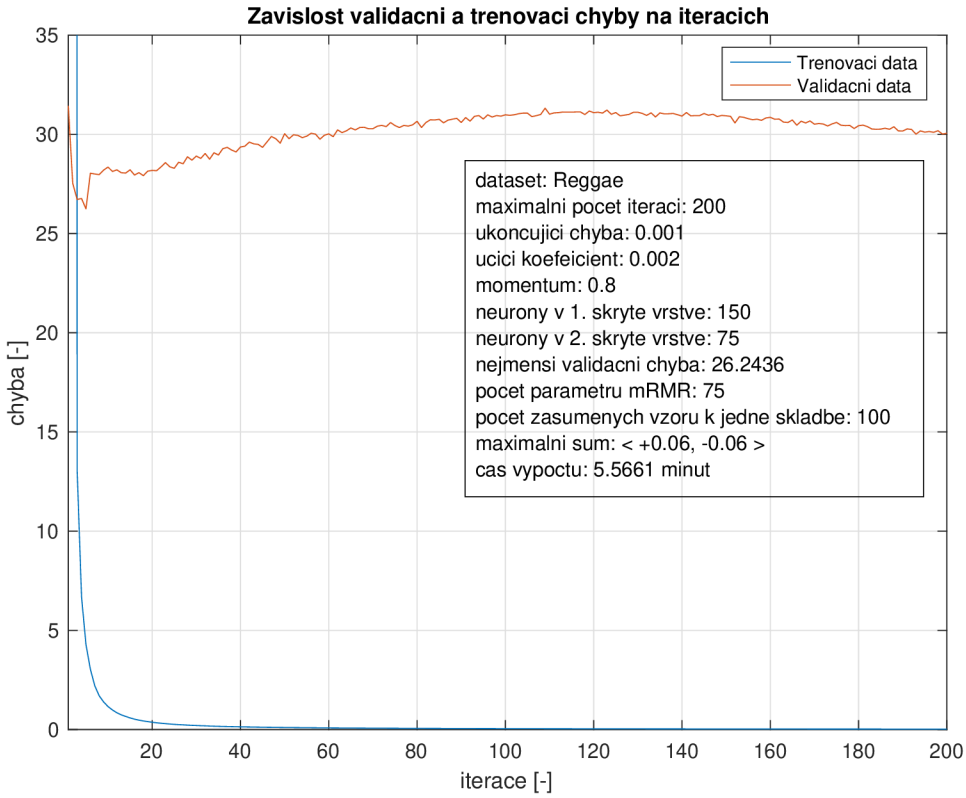

Vzdalenosti vsech skladeb pro styl Klasika
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

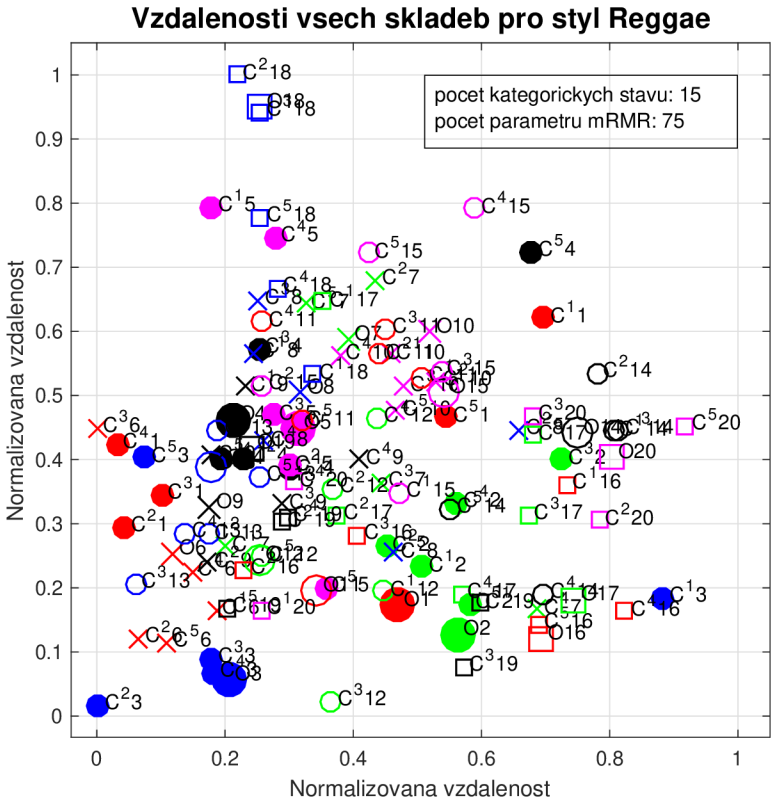

Zavislost validacni a trenovaci chyby na iteracich
dataset: Reggae
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 50
neurony v 2. skryte vrstve: 25
nejmensi validacni chyba: 33.3627
pocet parametru mRMR: 25
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 1.5995 minut
Trenovaci data
Validacni data
chyba [-]
iterace [-]

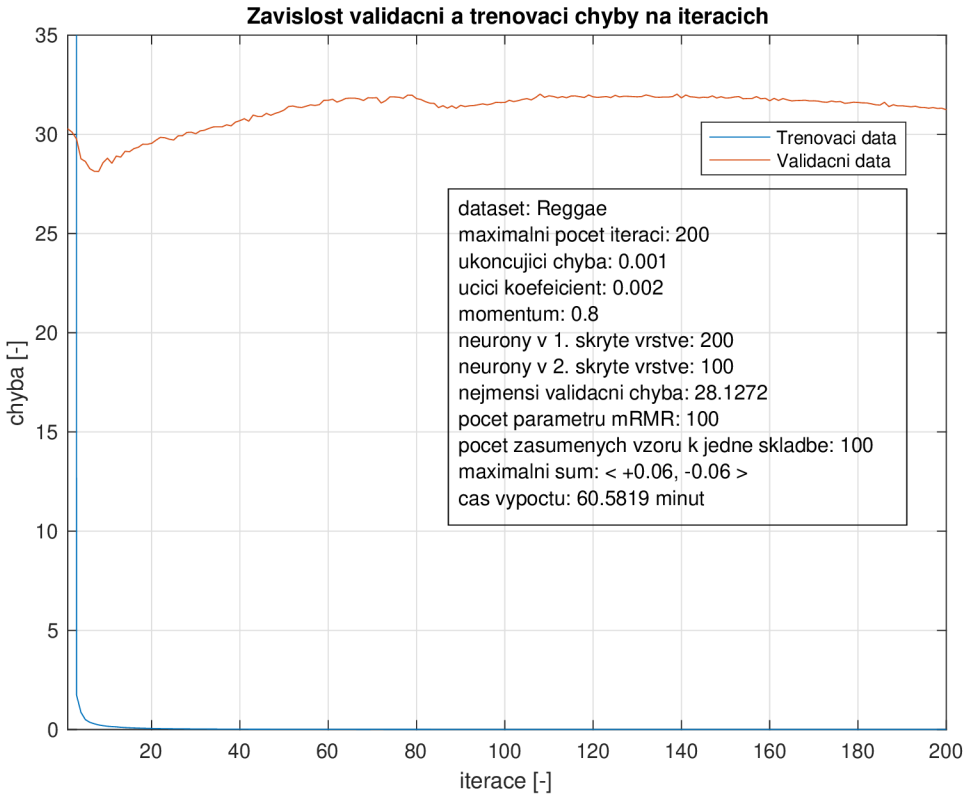

Vzdalenosti vsech skladeb pro styl Reggae
pocet kategorickych stavu: 15
pocet parametru mRMR: 25
Normalizovana vzdalenost
Normalizovana vzdalenost

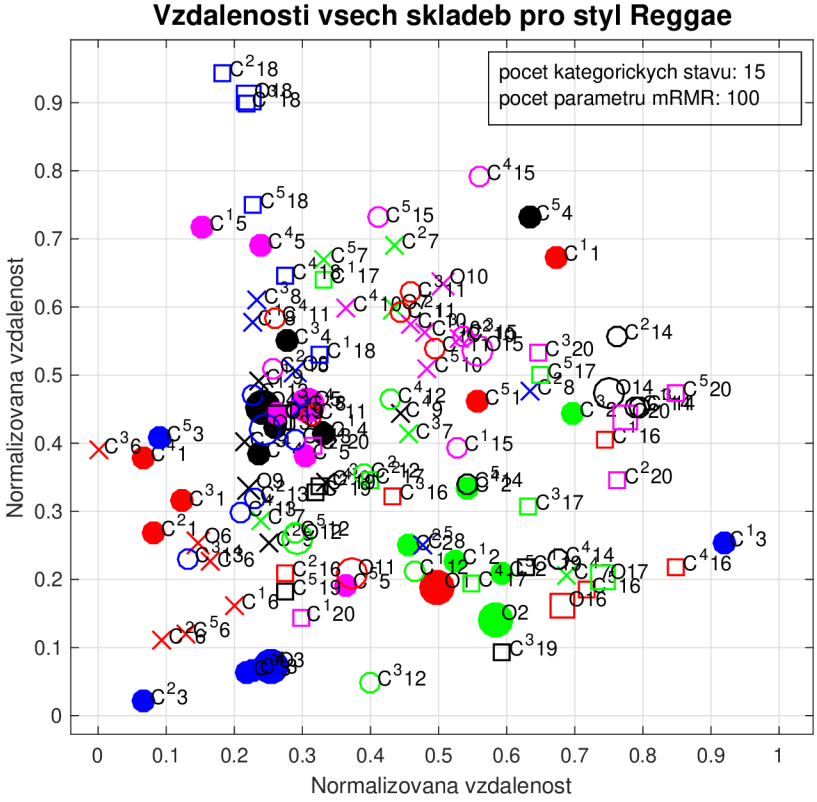

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Reggae
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 100
neurony v 2. skryte vrstve: 50
nejmensi validacni chyba: 28.3527
pocet parametru mRMR: 50
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 2.8875 minut
chyba [-]
iterace [-]

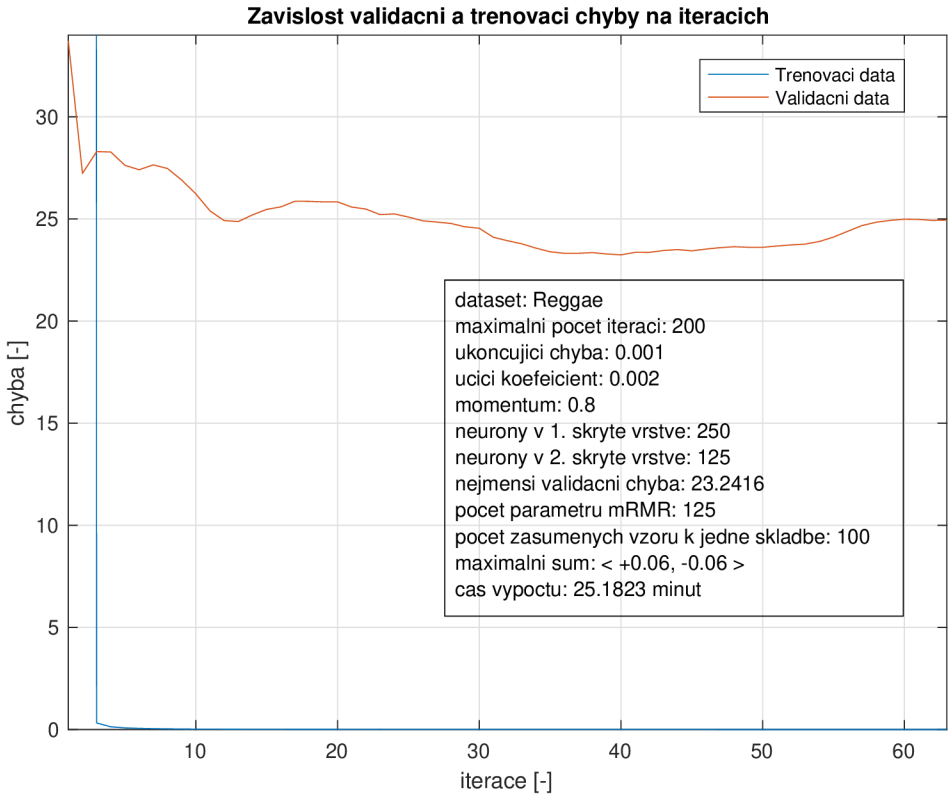

Vzdalenosti vsech skladeb pro styl Reggae
pocet kategorickych stavu: 15
pocet parametru mRMR: 50
Normalizovana vzdalenost
Normalizovana vzdalenost

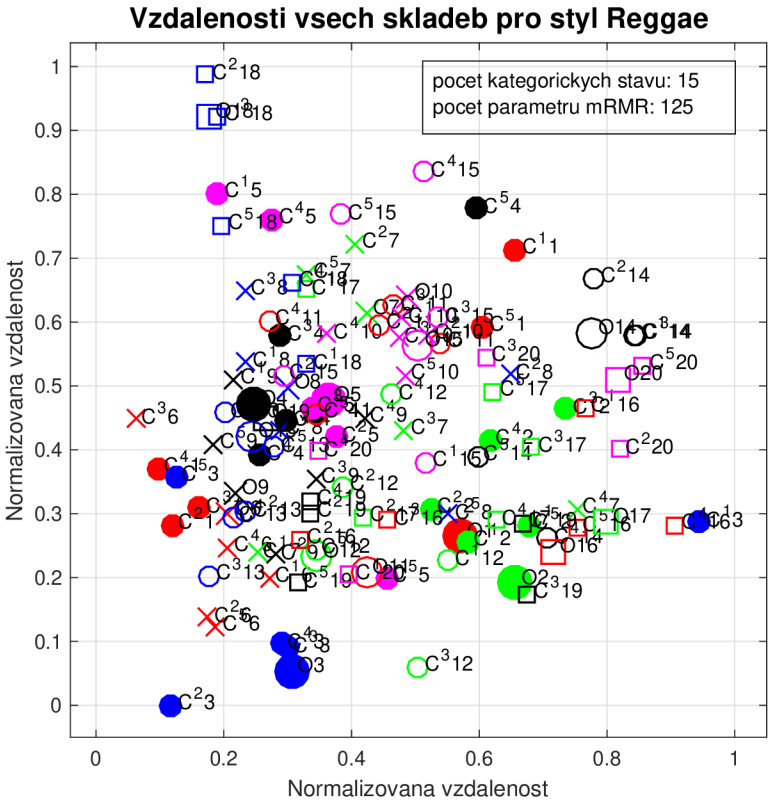

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Reggae
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 150
neurony v 2. skryte vrstve: 75
nejmensi validacni chyba: 26.2436
pocet parametru mRMR: 75
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 5.5661 minut
chyba [-]
iterace [-]
0
5
10
15
20
25
30
35
20
40
60
80
100
120
140
160
180
200

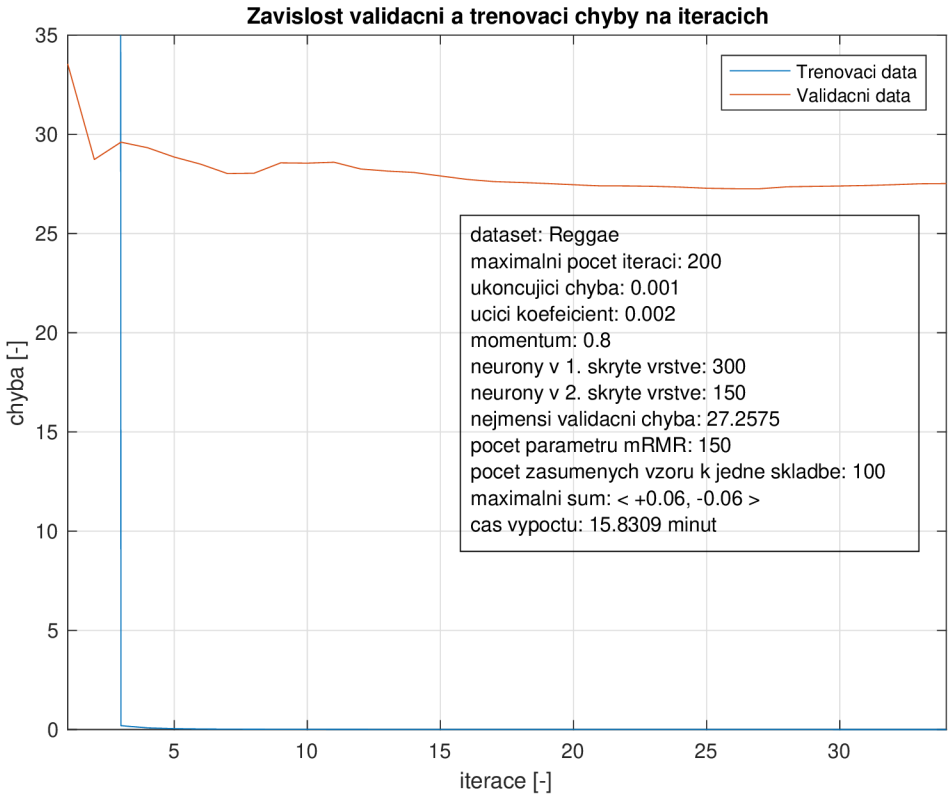

Vzdalenosti vsech skladeb pro styl Reggae
pocet kategorickych stavu: 15
pocet parametru mRMR: 75
Normalizovana vzdalenost
Normalizovana vzdalenost
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1

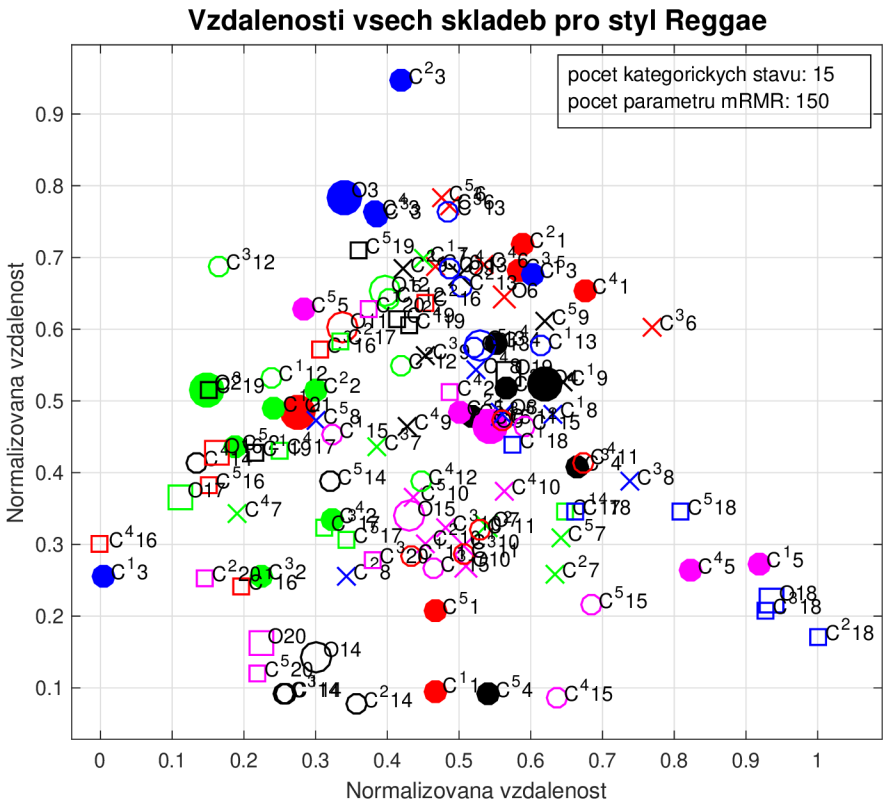

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Reggae
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 200
neurony v 2. skryte vrstve: 100
nejmensi validacni chyba: 28.1272
pocet parametru mRMR: 100
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 60.5819 minut
chyba [-]
iterace [-]

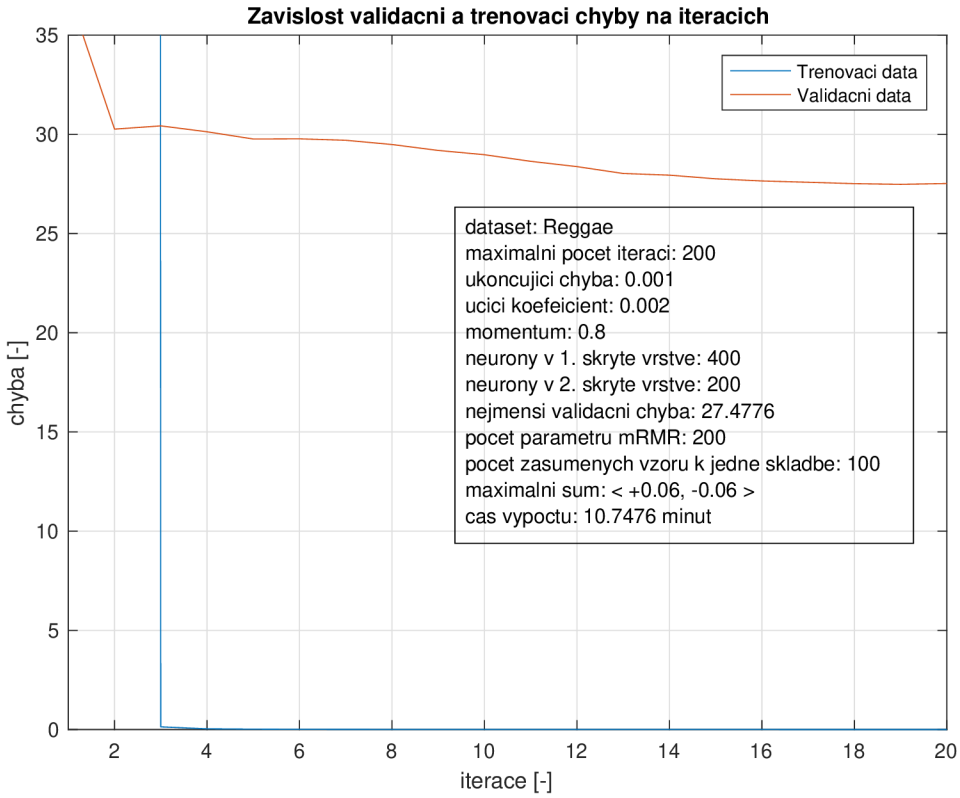

Vzdalenosti vsech skladeb pro styl Reggae
pocet kategorickych stavu: 15
pocet parametru mRMR: 100
Normalizovana vzdalenost
Normalizovana vzdalenost

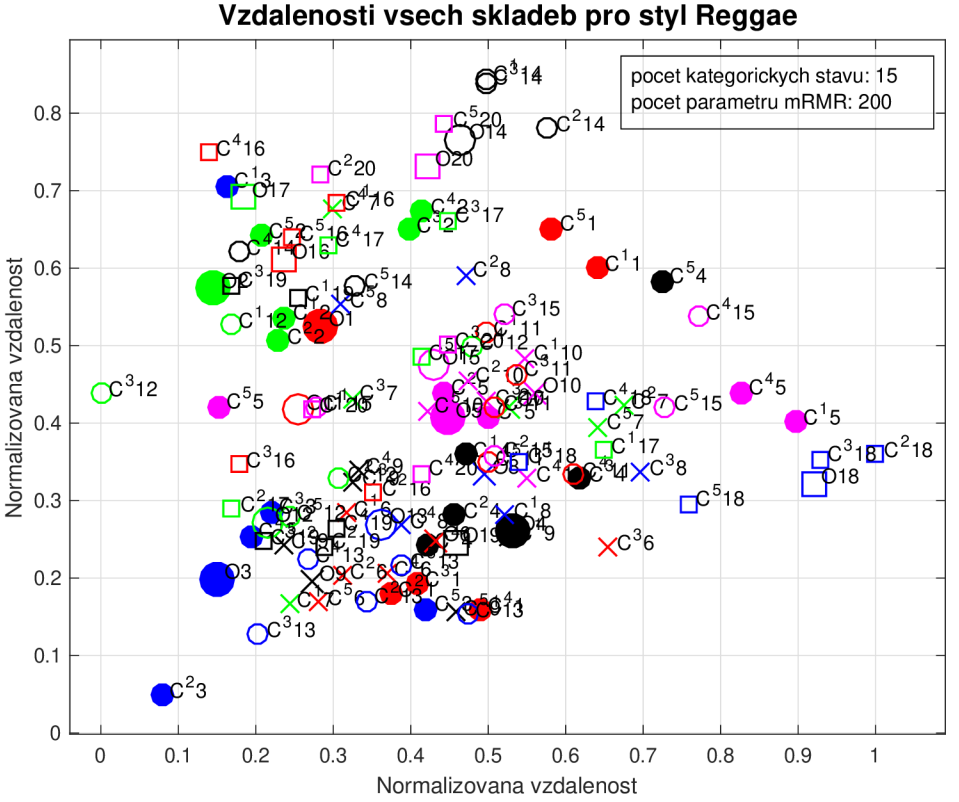

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Reggae
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 250
neurony v 2. skryte vrstve: 125
nejmensi validacni chyba: 23.2416
pocet parametru mRMR: 125
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 25.1823 minut
chyba [-]
iterace [-]


Vzdalenosti vsech skladeb pro styl Reggae
pocet kategorickych stavu: 15
pocet parametru mRMR: 125
Normalizovana vzdalenost
Normalizovana vzdalenost


Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Reggae
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 300
neurony v 2. skryte vrstve: 150
nejmensi validacni chyba: 27.2575
pocet parametru mRMR: 150
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 15.8309 minut
chyba [-]
iterace [-]


Vzdalenosti vsech skladeb pro styl Reggae
pocet kategorickych stavu: 15
pocet parametru mRMR: 150
Normalizovana vzdalenost
Normalizovana vzdalenost

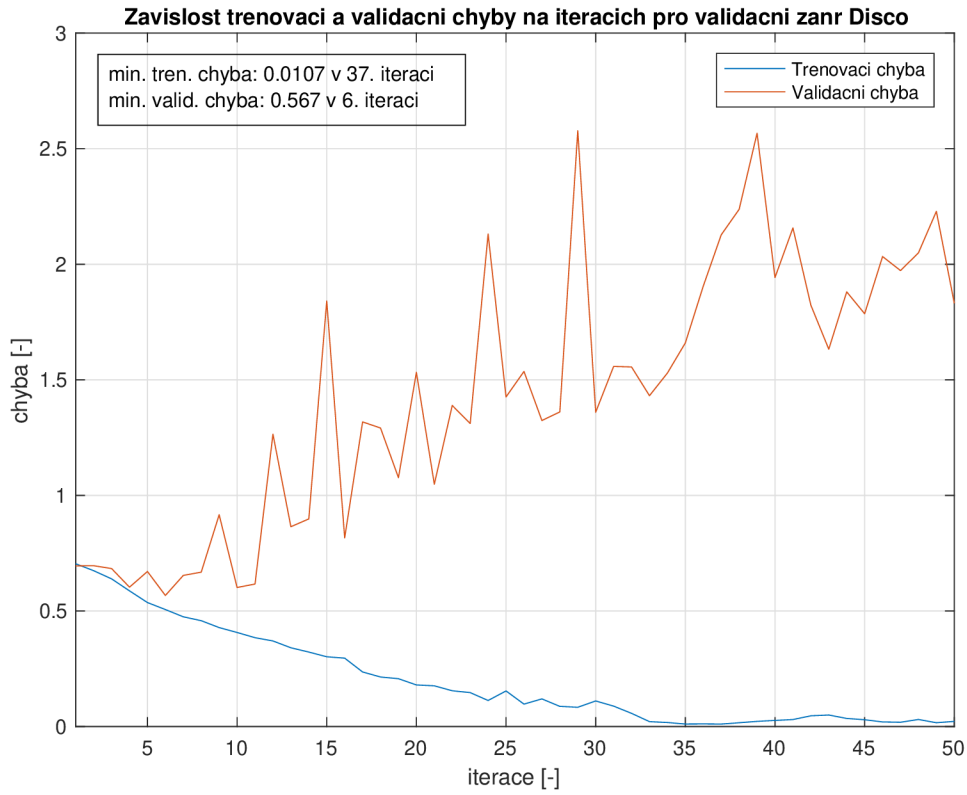

Zavislost validacni a trenovaci chyby na iteracich
Trenovaci data
Validacni data
dataset: Reggae
maximalni pocet iteraci: 200
ukoncujici chyba: 0.001
ucici koefeicient: 0.002
momentum: 0.8
neurony v 1. skryte vrstve: 400
neurony v 2. skryte vrstve: 200
nejmensi validacni chyba: 27.4776
pocet parametru mRMR: 200
pocet zasumenych vzoru k jedne skladbe: 100
maximalni sum: < +0.06, -0.06 >
cas vypoctu: 10.7476 minut
chyba [-]
iterace [-]

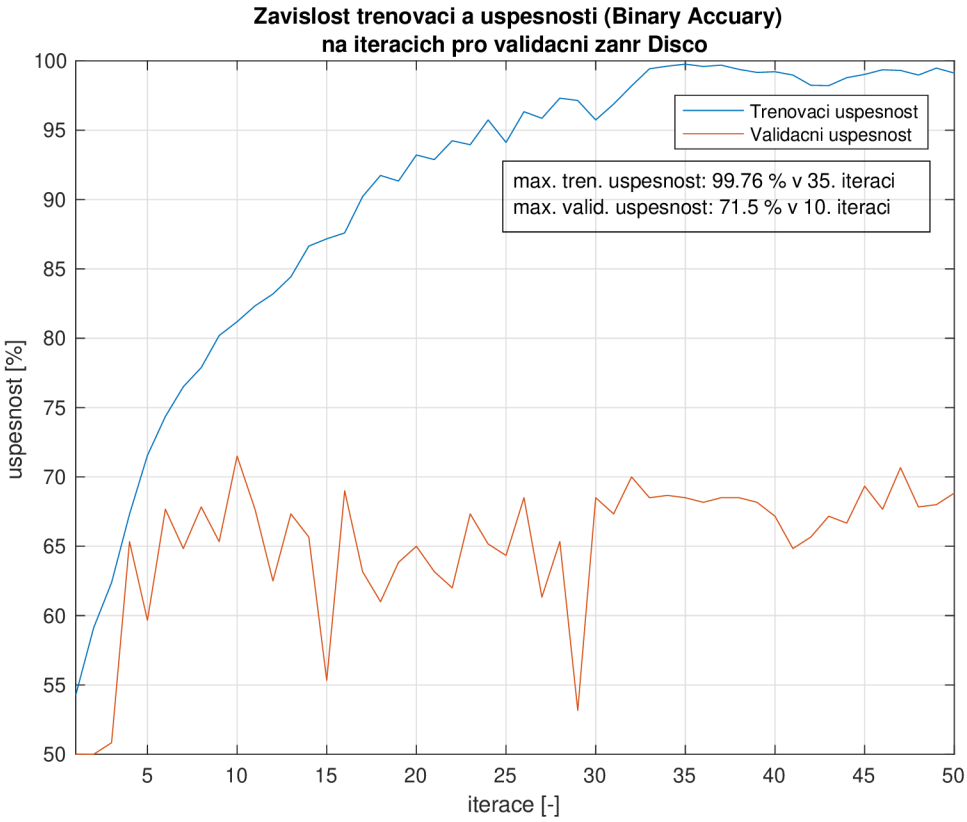

Vzdalenosti vsech skladeb pro styl Reggae
pocet kategorickych stavu: 15
pocet parametru mRMR: 200
Normalizovana vzdalenost
Normalizovana vzdalenost

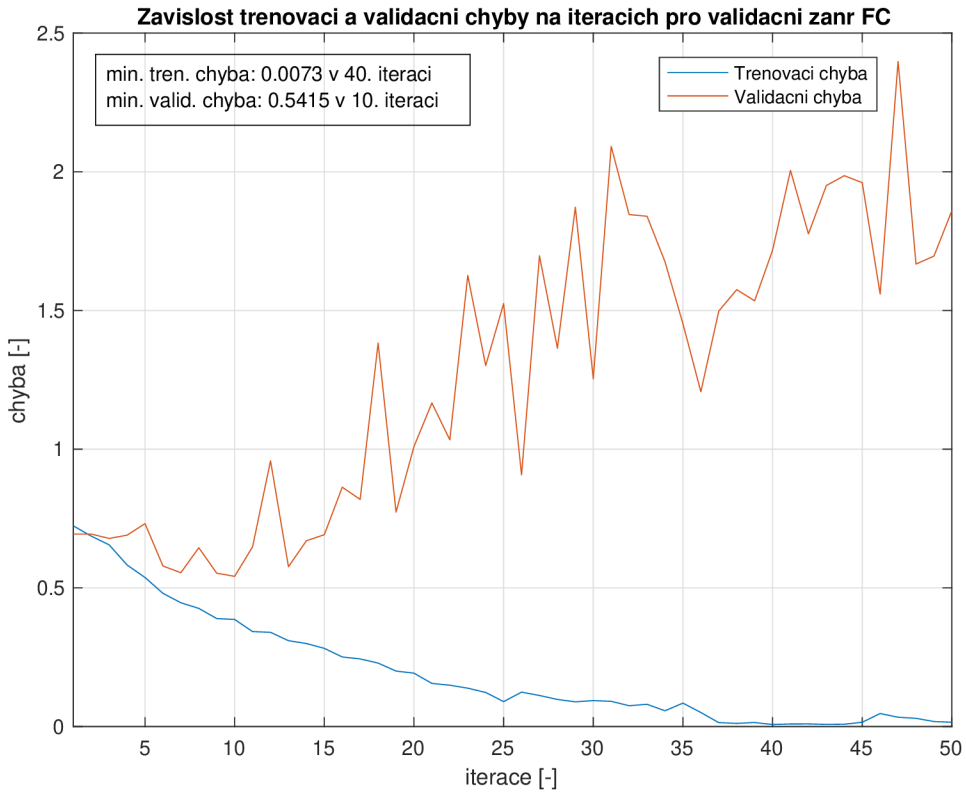
C Vyhodnocení metody založené na rozpoznávání CSM

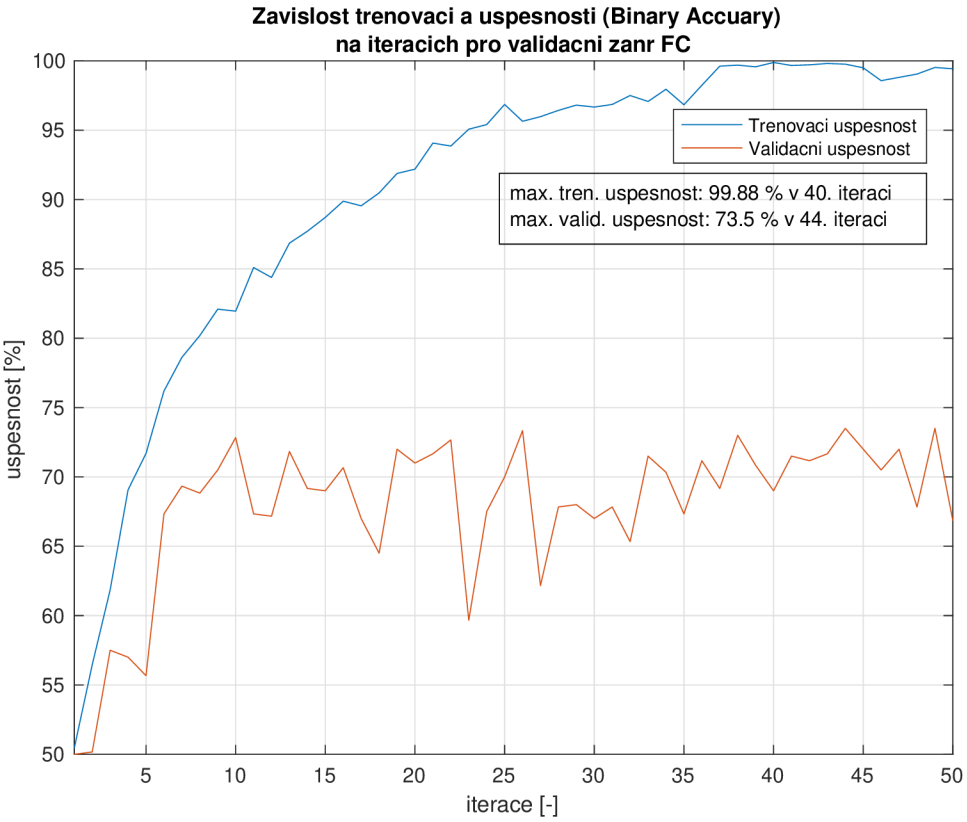
		CNN s nejmenší validační chybou					CNN v poslední trénovací iteraci				
valid. množina	metoda vyhodnocení	**správně určených**	**MAP**	**MNIT5**	**MR1**	**Preciznost [%]**	**správně určených**	**MAP**	**MNIT5**	**MR1**	**Preciznost [%]**
metal	max p	805	0,54	2,35	9,44	55,9	**794**	**0,50**	**2,49**	**6,63**	**55,1**
	min kosin.	864	0,57	2,60	**6,38**	60	391	0,24	0,63	17,11	27,2
	min P. kor.	**901**	**0,60**	**2,75**	7,30	**62,3**	483	0,30	1,01	12,86	33,5
disco	max p	726	0,48	2,03	5,79	50,4	**596**	**0,35**	**1,70**	**6,97**	**41,32**
	min kosin.	685	0,45	1,85	8,01	47,6	397	0,25	0,65	15,16	27,57
	min P. kor.	**759**	**0,50**	**2,16**	**5,71**	**52,7**	448	0,28	0,87	11,63	31,11
folk	max p	489	**0,32**	1,03	**13,07**	34	**439**	**0,26**	**1,08**	**17,45**	**30,5**
	min kosin.	469	0,30	0,95	14,80	32,6	309	0,20	0,29	27,74	21,5
	min P. kor.	**497**	**0,32**	**1,07**	14,70	**34,5**	358	0,23	0,49	19,95	24,9
pop	max p	803	0,53	2,35	5,85	55,7	**687**	**0,44**	**1,92**	**7,62**	**47,7**
	min kosin.	812	0,56	2,38	**3,95**	56,4	450	0,28	0,87	14,47	31,3
	min P. kor.	**842**	**0,56**	**2,51**	4,83	**58,5**	565	0,35	1,35	8,13	39,2
blues	max p	112	0,04	0,33	42,35	7,78	**639**	**0,42**	**1,72**	**10,74**	**44,4**
	min kosin.	379	0,24	0,58	**17,67**	26,32	525	0,33	1,19	12,18	36,5
	min P. kor.	**407**	**0,26**	**0,70**	17,72	**28,26**	556	0,35	1,32	11,14	38,6

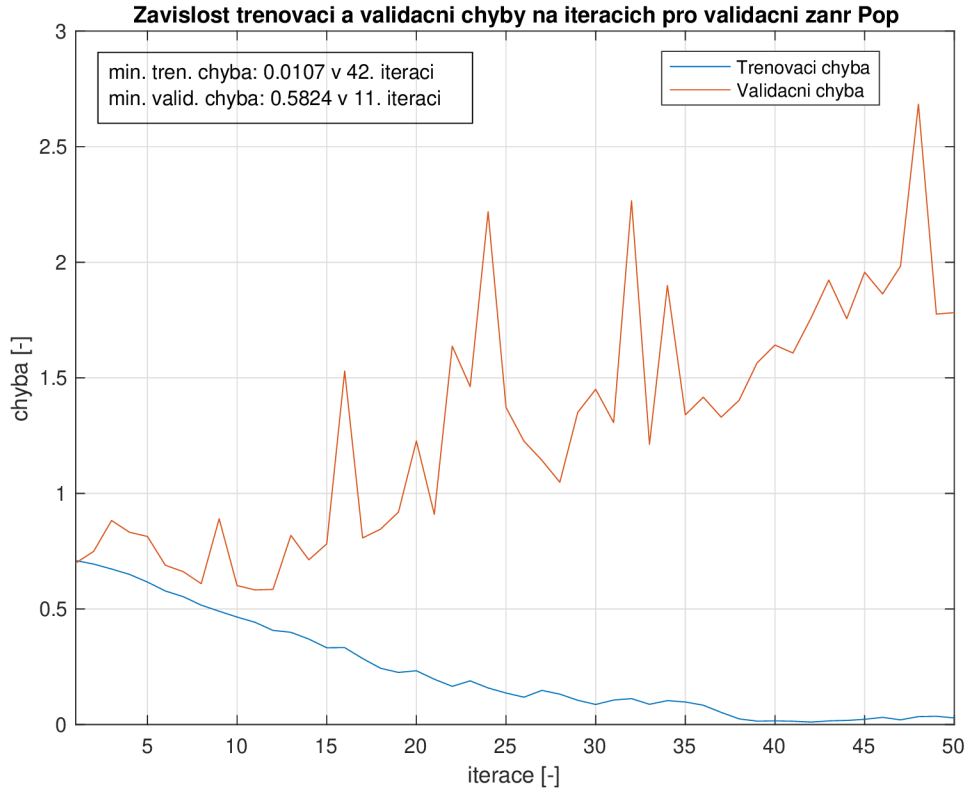
Tab. C.1: Srovnání vyhodnocovacích metod 10 modelů
CNN s 5 odlišnými trénovacími a testovacími datasety

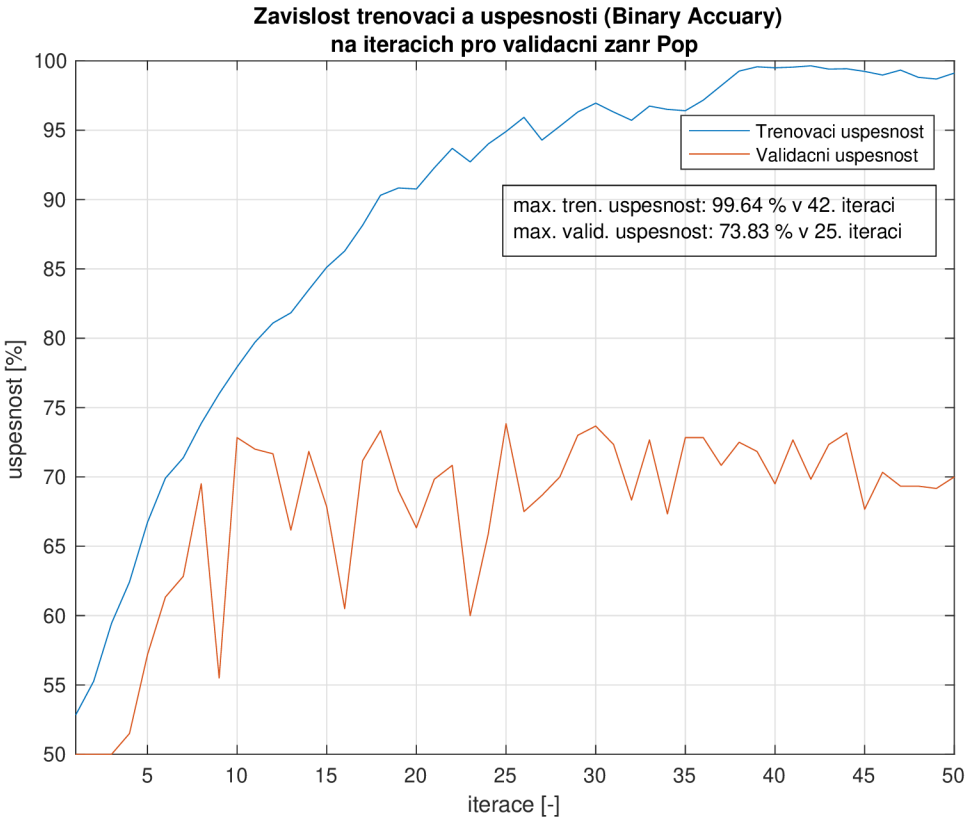

Zavislost trenovaci a validacni chyby na iteracich pro validacni zanr Disco
min. tren. chyba: 0.0107 v 37. iteraci
min. valid. chyba: 0.567 v 6. iteraci
Trenovaci chyba
Validacni chyba
chyba [-]
iterace [-]

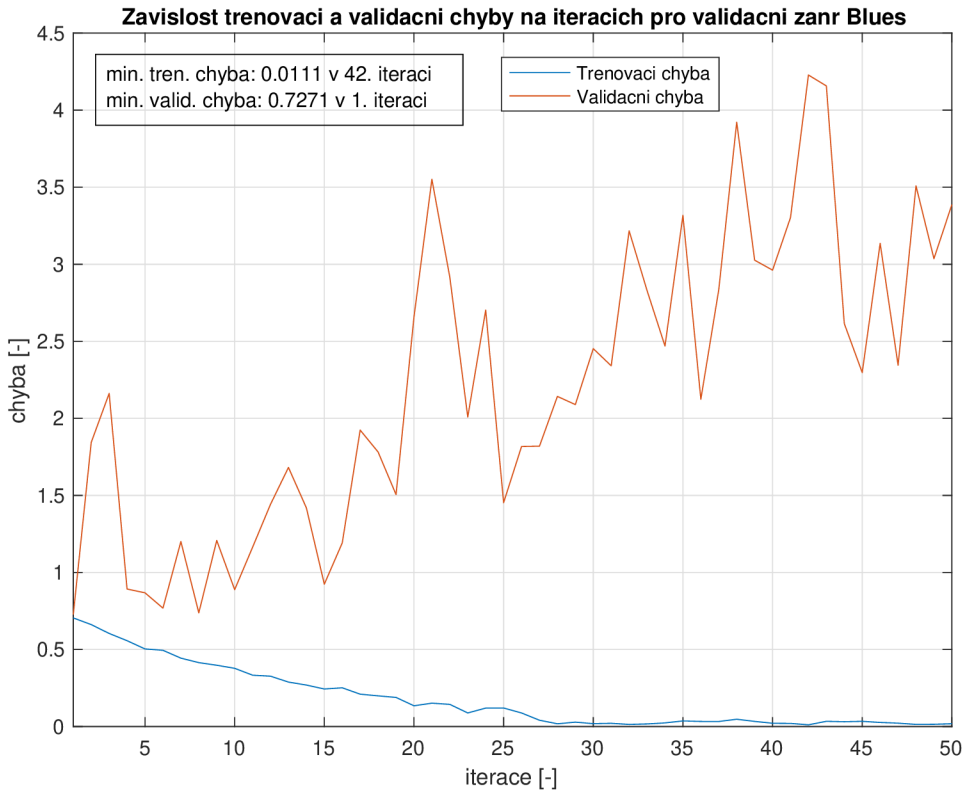

Zavislost trenovaci a uspesnosti (Binary Accuary)
na iteracich pro validacni zanr Disco
Trenovaci uspesnost
Validacni uspesnost
max. tren. uspesnost: 99.76 % v 35. iteraci
max. valid. uspesnost: 71.5 % v 10. iteraci
uspesnost [%]
iterace [-]

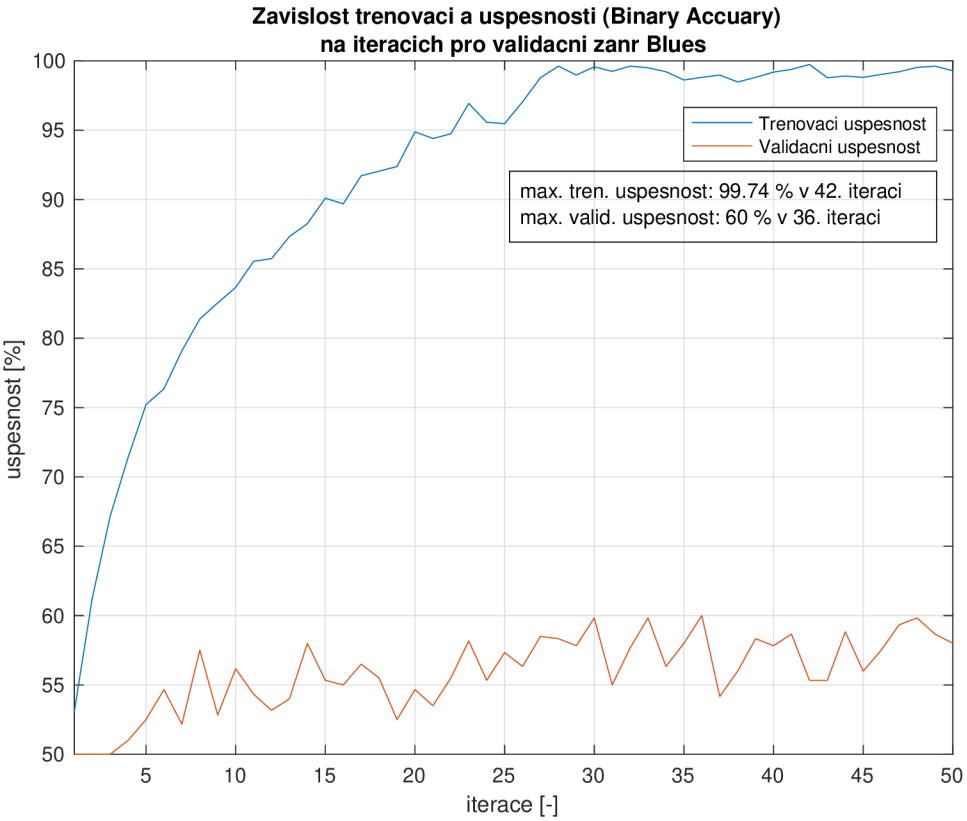

Zavislost trenovaci a validacni chyby na iteracich pro validacni zanr FC
min. tren. chyba: 0.0073 v 40. iteraci
min. valid. chyba: 0.5415 v 10. iteraci
Trenovaci chyba
Validacni chyba
chyba [-]
iterace [-]
0
0.5
1
1.5
2
2.5
5
10
15
20
25
30
35
40
45
50


Zavislost trenovaci a uspesnosti (Binary Accuary)
na iteracich pro validacni zanr FC
Trenovaci uspesnost
Validacni uspesnost
max. tren. uspesnost: 99.88 % v 40. iteraci
max. valid. uspesnost: 73.5 % v 44. iteraci
uspesnost [%]
iterace [-]
50
55
60
65
70
75
80
85
90
95
100
5
10
15
20
25
30
35
40
45
50


Zavislost trenovaci a validacni chyby na iteracich pro validacni zanr Pop
min. tren. chyba: 0.0107 v 42. iteraci
min. valid. chyba: 0.5824 v 11. iteraci
Trenovaci chyba
Validacni chyba
chyba [-]
iterace [-]
0
0.5
1
1.5
2
2.5
3
5
10
15
20
25
30
35
40
45
50


Zavislost trenovaci a uspesnosti (Binary Accuary)
na iteracich pro validacni zanr Pop
Trenovaci uspesnost
Validacni uspesnost
max. tren. uspesnost: 99.64 % v 42. iteraci
max. valid. uspesnost: 73.83 % v 25. iteraci
uspesnost [%]
iterace [-]
50
55
60
65
70
75
80
85
90
95
100
5
10
15
20
25
30
35
40
45
50


Zavislost trenovaci a validacni chyby na iteracich pro validacni zanr Blues
min. tren. chyba: 0.0111 v 42. iteraci
min. valid. chyba: 0.7271 v 1. iteraci
Trenovaci chyba
Validacni chyba
chyba [-]
iterace [-]


Zavislost trenovaci a uspesnosti (Binary Accuary)
na iteracich pro validacni zanr Blues
Trenovaci uspesnost
Validacni uspesnost
max. tren. uspesnost: 99.74 % v 42. iteraci
max. valid. uspesnost: 60 % v 36. iteraci
uspesnost [%]
iterace [-]