Rozpoznání hudebného žánru z nahrávky pomocí neuronové sítě

Lukáš Kuklínek
xkukli01@stud.fit.vutbr.cz

1 Úvod

Rozpoznání hudebního žánru patří mezi úlohy, dokáže člověk v jednoznačných případech řešit bez problémů. Strojová klasifikace stylů však již tak přímočará není. V této práci se pokusíme popsat možný přístup k řešení této úlohy. Budeme však rozpoznávat podmnožinu hudebních žánrů, které jsou víceméně dobře rozpoznatelné a nevznikají velké spory o zařazení nahrávek.

2 Návrh klasifikátoru

Samotný klasifikátor bude implementován pomocí neuronové sítě. Budeme potřebovat sadu trénovacích dat, ze které následně vyextrahujeme příznaky a za pomoci připravených anotací jimi neuronovou síť natrénujeme. Pak ji bude možno použít k rozpoznání žánru nových nahrávek.

2.1 Získání trénovacích dat a anotací

Vstupními daty jsou nahrávky písniček ve formátu `mp3` nebo `ogg`. Tyto formáty obsahují jak vlastní zvuková data, tak i metadata, především název skladby a jejího autora či interpreta.

Tyto informace použijeme k automatickému získání anotací o hudebním žánru ze serveru last.fm[1]. Anotace jsou uživately serveru zadané tagy, které je možno pomocí API získat ve formátu XML. Každý tag je ohodnocen číslem 1 až 100 na základě jeho četnosti.

2.2 Extrakce příznaků

Abychom si usnadnili práci při psaní programu, vstupní nahrávku nejdříve zkonverujeme do formátu `wav`. Tu převedeme na příznaky MFCC[2]. Tyto příznaky jsou založeny na poznatcích psychoakustiky a abstrahují další výpočet od fyzikálních vlastností zvuku [1]. Výstupem je několik málo (v našem případě 15) relativně dobře dekorelovaných koeficientů pro každých 20 ms nahrávky.

Abychom v příznacích nepostihli pouze samotné spektrum, ale i rychlost jeho změn, přidáme ke každému takovému vektoru ještě delta-MFCC, neboli rozdíl od předchozího MFCC vektoru v nahrávce. První vektor zahodíme. Vstupní vektor pro neuronovou síť tedy bude mít 30 dimenzí.

Celou trénovací sadu normalizujeme podle jejího průměru a směrodatné odchylky. Poté na základě tohoto průměru a směrodatné odchylky upravíme i klasifikovaná data [3].

$$x' = \frac{x - \mu}{\sigma}$$

2.3 Struktura neuronové sítě

Bude se jednat o třívrstvou neuronovou síť učenou pomocí zpětného šíření chyby. Počet vstupů již známe. Počet neuronů skryté vrstvy byl experimentálně určen na 18.

Jejich aktivační funkce je normální logistická sigmoida ($\lambda = 1$):

$$\mathrm{sig}(x) = \frac{1}{1 + e^{-x}}$$

Počet výstpů sítě bude roven počtu žánrů, které chceme rozpoznávat. Každý žánr tedy dostane skóre, které by mělo reflektovat, jak moc daná nahrávka do tohoto žánru náleží. Budeme tedy požadovat výstup v intervalu $\langle 0; 1\rangle$. Abychom se vyhnuli extrémním hodnotám, data získaná z last.fm jednoduše přeškálujeme do intervalu o něco menšího, třeba $\langle 0,005; 0,995\rangle$. Brzy bude zřejmé, proč je především nenulová dolní mez důležitá.

Kombinace výstupů Neuronová síť odpoví na jeden vstupní vektor vektorem výstupním. Tento však reprezenruje pouhých 20 ms nahrávky. Je nutné skóre všech vektorů nahrávky nějakým způsobem spojit ve výsledné ohodnocení. Pro průměrování skóre je, podobně jako pro pravděpodobnosti, mnohem přirozenější použít průměr geometrický nežli aritmetický. Geometrický průměr je pro množinu $\{a_1, a_2, \ldots, a_n\}$ definován takto:

$$\mathrm{gm}(A) = \left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}}$$

Velké množství násobení malých čísel by však znamenalo jednak výkonnostní penalizaci a hlavně by se brzy v neúnosné míře naakumulovaly numerické chyby.

[1] `http://last.fm/`
[2] Mel-frequency Cepstral Coefficients

Je tedy lepší daný výpočet provést v logaritmu, čímž geometrický průměr převedeme na průměr aritmetický:

$$\ln \mathrm{gm}(A) = \frac{1}{n} \ln \left(\prod_{i=1}^{n} a_i \right) = \frac{1}{n} \left(\sum_{i=1}^{n} \ln(a_i) \right)$$

Požadavek na logaritmický výstup můžeme za předpokladu příslušné úpravy (t.j. zlogaritmování) na trénovacích datech zakódovat přímo do aktivační funkce výstupní vrstvy. Tou bude logaritmus logistické sigmoidy:

$$\ln \mathrm{sig}(x) = \ln \frac{1}{1+e^{-x}} = -\ln(1+e^{-x})$$

Derivaci lze vyjádřit na základě hodnoty funkce v daném bodě:

$$(\ln \mathrm{sig}(x))' = \frac{1}{1+e^{x}} = 1 - e^{-\ln(1+e^{-x})} = 1 - e^{\ln \mathrm{sig}(x)}$$

2.4 Trénování sítě

K trénování sítě si připravíme tři sady dat: trénovací, krosvalidační a testovací. Síť je trénována pomocí algoritmu *backpropagation* po jednotlivých vrstvách. Všechny datové sady jsou nejdříve normalizovány podle průměru a směrodatné odchylky trénovacích dat. Trénovací data obsahující příznakové vektory množství nahrávek různých žánrů náhodně promícháme, aby nebyla data z jedné nahrávky, a tedy velmi podobná, předkládána mnohokrát po sobě.

Vrstva je reprezentována pomocí matice vah jednotlivých neuronů. Při učení je jí přiřazena ještě akumulační matice chyb vah. Ta je po předložení několika vzorků z trénovací sady pronásobena konstantou rychlosti učení a přičtena k matici vah vrstvy. Taktéž je přičtena malá část rozdílu aktuální a předchozí matice vah, čímž je implementována jistá setrvačnost učení aka *momentum*. Síti je předložen trénovací vektor spolu s požadovaným výstupním vektorem, je spočtena odezva a chyba se postupně předává jednotlivým vrstvám.

Zastavení trénování a také ochranu proti přetrénování řeší algoritmus *New Bob* [2]. Kelsne-li chyba sítě po jedné iteraci učení na krosvalidačních datech o méně než půl procenta, sníží se rychlost učení na polovinu. Klesne-li chyba podruhé v řadě o méně než půl procenta, učení je u konce.

Vzhledem k tomu, že počáteční váhy jsou vybrány náhodně, má cenu natrénovat více neuronoých sítí a vybrat tu nejlepší. K ohodnocení kvality výsledného klasifikátoru a porovnání s ostatními je použita metrika chyby na testovacích datech.

2.5 Klasifikace

Klasifikátor tedy na základě natrénovaných matic vah přiřadí každému vektoru nahrávky logaritmické skóre pro každý hudební žánr. Jednotlivá skóre zprůměruje a poté na tento průměr aplikuje funkci e^x, aby byl výsledek zase v rozsahu $\langle 0; 1 \rangle$. Tím dosáhneme lepší interpretovatelnosti výsledku člověkem a otevřou se i zajímavější možnosti pro vizualizaci výsledného skóre.

3 Implementační detaily

Klasifikátor je napsán v jazyce C++ a používá několik knihoven.

Knihovna *Boost.uBLAS*[3] poskytuje sadu rutin pro práci s vektory a maticemi. Je použita pro samotnou implementaci neuronové sítě. Dokáže také vektory a matice ukládat do souborů a pak je z něj načíst. Tato vlastnost je využita k serializaci a deserializaci natrénovaného klasifikátoru.

Pro extrakci příznaků MFCC je použita knihovna *Aquila DSP*[4]. Implementace se neukázala být příliš efektivní, proto byl navržen souborový formát, který uloží datovou sadu nahrávek již v podobě vektoru příznaků s vektorem očekávaného výstupu. Na této sadě je pak možno trénovat více klasifikátorů, aniž by musely být z nahrávek znovu extrahovány koeficienty MFCC.

Různé podpůrné funkce, jako práce s řetězci, procházení filesystémemu a generování pseudonáhodných čísel, jsou realizovány za pomoci standardní knihovny jazyka C++, případně příslušných knihoven ze sady *Boost*[5].

O předzpracování datové sady se stará *Makefile*[6]. Ten jednak volá skript pro získání anotací, využívající k tomuto účelu nástroj *curl*[7] a *taglib*[8], a jednak využívá příkaz *ffmpeg*[9] ke konverzi audioformátů.

4 Výsledky experimentů

Pro experimenty byla vybrána množina nahrávek různých žánrů a to konkrétně následujících pět:

1. pop
2. rock
3. classical (vážná hudba)
4. electronic (techno & spol.)
5. rap

Testovací data neobsahovala žádnou nahrávku z trénovacích ani krosvalidačních dat, obsahovala však několik nahrávek od stejných interpretů. Klasifikátor

[3] `http://www.boost.org/doc/libs/release/libs/numeric`
[4] `http://aquila-dsp.org/`
[5] `http://boost.org/`
[6] `http://www.gnu.org/software/make/`
[7] `http://curl.haxx.se/`
[8] `http://developer.kde.org/ wheeler/taglib.html`
[9] `http://www.ffmpeg.org/`

ve většině případů přiřadil správnému žánru nejvyšší skóre.

Popové skladby dostávaly často vysoké rockové či elektronické skóre (i vyšší ohodnocení než pop), protože mají těmto žánrům blízko použitím kytar či syntetizátorů. Problém nastal s rapem, který je patrně těžké odlišit od rocku či elektroniky pouze na základě spektrální analýzy. Rapeři totiž často používají různě remixované samply více či méně slavných rockových kapel. Spolehlivě je rozpoznána nahrávka Eminema, u kterého se však klasifikátor mohl natrénovat na hlas interpreta.

Naopak typicky elektronické skladby a vážná hudba jsou rozpoznány velmi dobře. Rockové skladby mají rockové skóre o dost vyšší, než skóre ostatních žánrů, ale i nerockové skladby mají často vysoké rockové skóre. To může být způsobeno rozbalancováním trénovacích dat, protože moje sbírka písní rockových interpretů je mnohem rozsáhlejší než u ostatních žánrů.

5 Uživatelský manuál

5.1 Překlad

Budou potřeba všechny knihovny popsané v sekci 3. Knihovnu *Aquila DSP* je třeba stáhnout a přeložit v podadresáři `aquila` v kořenovém adresáři projektu.

Klasifikátor poté přeložíme následovně (předpokládáme, že pracovní adresář je kořenový adresář projektu):

```
mkdir build
cd build
cmake ..
make
```

5.2 Příprava datové sady

Pro trénovací, testovací a krosvalidační data je vyčleněn adresář `data`. Ten připravíme takto:

```
# predpokladame korenovy adresar projektu
cd data
# adersar pro trenovaci data
mkdir train
cd train
# kopirujeme ci nalinkujeme trenovaci data
ln -s /nejaka/nahravka.mp3
ln -s ../scripts/features.make Makefile
make
```

Výsledkem příkazu `make` je soubor `features.dat` s normalizovanou a náhodně promíchanou datovou sadou. Provádí získání anotací ze serveru *last.fm*, převod anotací, konverzi audioformátu na `wav` a následné vytvoření samotné datové sady. Obdobný postup provedeme i pro adresáře `xval` a `test` pro krosvalidační a testovací data.

5.3 Použití

5.3.1 Trénování

Součástí distribuce projektu již je jedna natrénovaná síť. Jedná se o soubor `cls.nn` v kořenovém adresáři projektu. K natrénování vlastní sítě slouží následující příkaz:

```
# v adresari build:
./genre train \
  -l pop:rock:classical:electronic:rap \
  -o cls.nn \ # vysledny soubor se siti
  -h 18 \     # pocet skrytych neuronu
  "../data/train/features.dat" \
  "../data/test/features.dat" \
  "../data/xval/features.dat"
```

Jestliže jsou vynechána krosvalidační data, použijí se místo nich testovací. Jestliže jsou vynechána i testovací data, použje jako krosvalidační i testovací data trénovací sada (nedoporučuje se).

5.3.2 Klasifikace

```
./genre classify -f cls.nn nahravka.wav
```

V rámci jednoho volání příkazu je možno klasifikovat i více nahrávek.

6 Závěr

Klasifikátor dokázal většinu testovacích nahrávek rozpoznat, stále však je velký prostor ke zlepšení. Největší potenciál nejspíše leží v množství trénovacích dat a v lepším výběru příznaků. Spektrální analýza dvacetimilisekundových úseků nahrávky totiž nepostihuje rytmické a melodické vlastnosti zvuku. To jsou právě ty, které by dokázaly mnohem lépe diskriminovat například rock a rap.

Reference

[1] Steven B. Davis. Paul Mermelstein. *Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences*. IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(4), pp. 357–366. 1980.

[2] Bob New. Kevin Shaw. *Neural Networks – Setup and Configuration*. Oregon Graduate Institute.

[3] Warren S. Sarle. *Neural Network FAQ*. 2002. URL: `<ftp://ftp.sas.com/pub/neural/FAQ.html>`.