Strojové učení a rozpoznávání
Dokumentace k projektu

David Hudák (xhudak03, xmracn00, xpleva07)

28. dubna 2023

1 Úvod

V tomto projektu se zabýváme rozpoznáváním různých osob na základě jejich krátkých zvukových nahrávek a fotek. Cílem je vytvořit a experimentovat s klasifikátory, které na základě těchto údajů budou schopny určit (klasifikovat), o kterou osobu se jedná, a to na základě trénovacích/testovacích dat poskytnutých předem. Oproti jiným projektům v ostatních předmětech je zásadním řešeným problémem obecně velmi malé množství poskytnutých dat (185 trénovacích a 62 testovacích párů obrázek-nahrávka).

Text je rozdělen na sekci 2 obsahující popis základní informací nezbytných ke zprovoznění a následnému spouštění vytvořených skriptů pro trénování a především pak pro klasifikaci. Následující dvě sekce 3 a 4 popisují přístupy ke klasifikaci obrázků, respektive klasifikaci zvukových nahrávek. Následující sekce 5 popisuje princip, jakým je vytvářena celková klasifikace a poslední sekce 6 shrnuje poznatky, kterých jsme během práce na projektu dosáhli.

2 Nutné nástroje a spuštění

Vytvořené klasifikátory fungují v prostředí Pythonu 3.10, k jejich rozjetí je nezbytné mít nainstalováno následující balíky a nástroje:

- Numpy pro běžné matematické operace
- Torch pro implementaci neuronové sítě
- OpenCV a Pickle pro práci se soubory
- Další nástroje a knihovny použité v knihovně ikrlib.py
  - glob, matplotlib, scipy, imageio

K práci s projektem je nutné spouštět s příkazem příslušícímu nainstalovanému Pythonu soubory **main.py** a **final.py**. První zmíněný obsahuje trénování a testování použitých nástrojů, tj. neuronové sítě a algoritmu GMM[1]. Vždy se tak spustí trénování buď již existujícího modelu, nebo se vytvoří model nový[2]. Druhý zmíněný vytvoří ze souborů ve složce určené přiřazením do argumentu skriptu `--path` odevzdávané soubory obsahující jednotlivé n-tice jména souboru, odhadované třídy a pravděpodobnosti jednotlivých tříd (ty se tisknou do adresáře, kde se nachází skript).

Požadovanou prerekvizitou pro fungování projektu (alespoň v oblasti trénování) je nutné mít v podsložce `data` složky `SRC` požadované složky train a dev a jejich podsložky s řečníky tak, jak je to v zadání projektu.

[1] Gaussian Mixture Model

[2] Podle existence souboru cnn.pkl v podsložce models. GMM se trénuje vždy nanovo.

3 Zpracování obrazových vstupů

Co se týká obrazových vstupů, tak jako nejpřímější cesta se zdají být neuronové sítě, tudíž jsme je využili. Při trénování těchto sítí jsme pracovali s tím, že train soubory jsme používali jako čistě trénovací data a dev data čistě jako testovací. Evaluace modelů jsme prováděli tak, že jednak se při učení tiskla aktuální hodnota chybové funkce a správně ohodnocených vzorů, jednak na konci každého trénování (později častěji) jsme tiskli úspěšnost na testovacím datasetu (dev). Následující podsekce se postupně zabývají charakteristikou vstupních dat, navrženými architekturami, datovými augmentacemi a regularizací a dropout vrstvami.

3.1 Vstupy a redukce kanálů

Vstupní obrázky jsou v základu 3-kanálové (RGB). Z důvodu velkého množství vstupů první vrstvy plně propojené sítě bylo v prvotní fázi experimentů pracováno s variantou, kdy s pomocí knihovny OpenCV byly původní obrázky převáděny do greyscale obrázků (hodnoty 0-255). To, byť se to v rámci trénování s jednoduššími sítěmi vyplatilo (nebylo potřeba mít tolik vstupních vah) a klasifikace byla rychlá a v rámci trénovacího datasetu přesná, nakonec není ve finálním řešení využíváno. Argumentujeme to tak, že odstraňováním barevných kanálů přicházíme o určitou informaci, přičemž z důvodu malé datové sady jimi není úplně vhodné plýtvat.

Dalším potenciálním zásahem do trénovacích a testovacích dat mohla být normalizace na hodnoty od -1 do 1. Tu jsme zkoušeli, ale nedosáhli jsme znatelných změn, tudíž jsme od ní upustili.

3.2 Architektury navržených sítí

Architektura sítě musí splňovat dva navzájem protichůdné požadavky – musí být dostatečně velká, aby byla schopna rozlišit data, ale nesmí být příliš velká, aby si to jen nepamatovala (generalizovala). V první moment jsme se pokusili vytvořit síť, která obsahuje pouze pár plně propojených vrstev s aktivační funkcí ReLU. Toto řešení se moc neosvědčilo, a to jak ve směru trénovacích dat, tak především pak testovacích. Řešení v případě testovacích dat nikdy nedosahovalo ani 10 správně ohodnocených vzorů ze 62.

Alespoň v oblasti klasifikace trénovacích dat se zdařilo si výrazně polepšit použitím leaky ReLU aktivační funkce (oproti ReLU s mírným poklesem v lehce záporných číslech na záporné části osy X). Z důvodu jednak lepšího učení vzdálených příznaků (zachycení souvislostí), jednak z důvodu nutnosti použití $80 \times 80 = 6400$ nastavitelných parametrů ve vstupní vrstvě (krát 3 kanály), jsme následně přistoupili ke konvolučním neuronovým sítím. Ty za použití 2 vrstev konvoluce s ReLU a maxpoolingem, 2 skrytých plně propojených vrstev s leaky ReLU a jedné výstupní plně propojené vrstvy zakončené aktivační funkcí softmax dosáhly vysokých hodnot při učení na trénovacím datasetu. Model jsme kombinovali s křížovou entropií a učením s pomocí optimalizátoru ADAM.

Bohužel, zmíněná síť na testovacím datasetu dosahovala pouze asi třetinové úspěšnosti (něco přes 20 správně uhodnutých vzorů ze 62), jelikož docházelo k přetrénování – přílišnému pamatování trénovacího datasetu bez schopnosti generalizace. Při dalších mnoha úpravách modelu (odebíráním, přidáváním vrstev, změnou aktivačních funkcí, normalizací, změn počtu a nastavení počátečních parametrů) nedošlo k výraznému zlepšení, a tedy jsme se rozhodli neměnit model, nýbrž trénování. Implementaci použitého modelu lze nalézt v souboru **neural_network.py**.

3.3 Datové augmentace

Jedním z prvních způsobů, jak bojovat proti přetrénováním, který jsme implementovali, bylo zavedení datových augmentací. Ty jsou dvojího rázu – nejdřív přidání gaussovského šumu do obrázků (experimentovali jsme s různými hodnotami směrodatné odchylky) a následně náhodné

(opět s normálním rozdělením) rotace libovolným směrem. Ukázalo se, že při opakované tvorbě a spouštení trénování se dařilo postupně pomalu ale jistě dosáhnout řádově lepších výsledků.

Nejlepší natrénované sítě pak dosahovaly i jenom s tímto přístupem až 50 správně uhodnutých obrázků ze 62. Pro top 3 klasifikace (správné řešení se nachází mezi třemi nejvyššími výstupy) mnohdy dosahovalo až 60 správně klasifikovaných vzorů (mnohdy to bylo ale na úkor správnosti celkové klasifikace). V kombinaci se zpracováním zvuků, jak bude uvedeno v následujících sekcích, se to ukázalo jako nejlepší varianta.

Zásadním objektem experimentů bylo také množství vygenerovaných obrázků a jejich (ne)kombinace s původními daty. Jako nejefektivnější se ukázalo vždy vytvořit zhruba dvojnásobek původních trénovaích dat (něco přes 300) se zavedením šumu bez přidání původních dat, natrénovat, zahodit, vytvořit nová data, trénovat atd. stále dokola.

3.4 Dropout, regularizace

Dalšími přirozenými způsoby, jak se vyhnout přeučení, jsou techniky dropoutu a regularizace. První zmíněná technika spočívá v "nahrazení"některých vrstev tzv. dropout vrstvami, kdy s určitou nastavitelnou pravděpodobností se konkrétní neurony v dané vrstvě při každé iteraci vypnou a nepoužijí pro klasifikaci ani trénování, přičemž jejich váhy jsou následně násobeny pravděpodobností p, se kterou byly vypnuty[3]. S touto technikou se nám nezdařilo dosáhnout jakkoliv lepších výsledků a ve finálním modelu není zahrnuta.

Druhá technika spočívá v přidání (přičtení) regularizačního členu do objektivní funkce. Ty se dělí na L2 a L1 regularizace, kdy použita byla první zmíněná. Ta spočívá k přičtení sumy kvadrátu všech vah v síti. To vede k tomu, že síť uměle oslabuje svoje váhy, čímž omezuje možnost vzniku přeučení[4]. Tato technika se ukázala jako lehce nápomocná a ve finální implementaci je její použití zahrnuto. Ukázalo se také, že se vyplácí trénovat síť nejprve s nižší regularizací a větším krokem a při dalších iteracích krok zmenšit a zvýšit regularizaci.

Jednou z dalších variant, která se tohoto tématu dotýká, je použití testovací (validační) sady právě k úpravě parametrů regularizace či learning ratu či případné předčasné zastavení v momentu, kdy modelu začne klesat testovací přesnost. Bohužel se nezdařilo ukázat, že by to v našem případě mělo nějaký vliv.

3.5 Potenciální vylepšení

V rámci této části projektu se nám podařilo natrénovat model, který je schopen správně klasifikovat zhruba 83 % testovacích dat (52 ze 62) a uvidíme, jak správně bude klasifikovat ostrá data. Přesto se nejedná o model jakkoliv dokonalý a je spousta přístupů, kterými řešení zlepšit.

Jednou z technik, které se probíraly v předmětu SUR, je jednoznačně lepší extrakce příznaků. Použití například PCA či jiného nástroje pro extrakci toho nejdůležitějšího z obrázků by zcela jednoznačně mohlo vést ke zlepšení klasifikace.

Dalším velmi výrazným vylepšením by mohlo být dodání více skutečných dat. Augmentace sice tento problém do nějaké míry řeší, přesto ne úplně dokonale a buď nafocení dalších fotek, či případně využití testovací sady pro trénování by mohlo pomoci. Také jak bylo zmíněno v předchozí podsekci, prostor k experimentování zbývá i ve využití části datasetu jako validačního, kdy bychom automatizovaně použili úspěšnost k úpravám modelu, změnám kroku učení a regularizace. Například bychom mohli zkoumat, jak moc malý model si pro trénování můžeme dovolit, aniž bychom neznemožnili síti se naučit rozpoznávat obličeje.

[3] V případě Torche, se kterým jsme pracovali, se dropout neupravoval dle pravděpodobnosti p, nýbrž se vypínal pořád.

[4] Naopak při vysokém nastavení L2 regularizace docházelo k podučení, s čímž jsme se při projektu též setkali.

Jednou z možností, které byly často zmiňovány na přednáškách, by mohlo být využití informace o nahrávacím sezení a nějakým způsobem kombinovat trénování a testování vždy v rámci daného sezení, či případně jej jinak využít.

Jako další (nikoliv ale poslední) variantou, kterou bychom určitě v případě trénování zvážili, by bylo použití již předtrénovaného modelu, který umí pracovat s lidskými obličeji. Problémem je, že přesně tuto variantu zadání explicitně zakazuje, tudíž to nebylo možné.

4 Zpracování zvukových nahrávek

Tato sekce se zabývá klasifikací řečníka na základě poskytnutých zvukových nahrávek. K vyhodnocování jsme přistoupili stejným způsobem jako u obrazových vstupů – rozdělení datasetu na trénovací a testovací s tím, že po každém trénování se ukázala úspěšnost jak na trénovacím, tak na testovacím datasetu.

4.1 Extrakce příznaků

K extrakci příznaků byla použita metoda MFCC (Mel-frequency capstral coeficients), jenž rozdělí nahrávku do vektoru příznaků 13 kapstrálních koeficientů. Byl pokus extrakci příznaků udělat za pomocí metod PCA a LDA, ale tyto metody se ukázaly jako neefektivní. Je to primárně kvůli tomu, že se nepodařila u těchto metod efektivně separovat příznaky do kategorií. Nahrávky s extrahovanými příznaky byly uloženy do polí `trains` a `tests` podle toho, zda nahrávka patřila k trénovací či testovací sadě. Tyto položky se dále využívaly při trénování a ověřování přesnosti modelu.

4.2 Trénování modelu

Trénovaným modelem byl směs gausovských rozložení (GMM) a k trénování se využil EM algoritmus. Jedná se o algoritmus podobný viterbi trénování (každý komponent reprezentuje separátní třídu, ke kterému se přiřazují individuální body. Tato přiřazení pak slouží k úpravě výsledného GMM). Na rozdíl od viterbi trénování se ale body nepřiřazují ke komponentům tvrdě, ale měkce. Výpočet nových komponentů je tedy váženým průměrem vlivů všech komponent a algoritmus tak vede k lepšímu učení.

Pro učení GMM bylo použito 100 iterací a bylo použito 13 komponent. Model natrénovaný za pomocí takového nastavení měl na testovacím datasetu přesnost kolem 70 %. Na trénovacím datasetu mělo takové nastavení uspěšnost dosahující 100 %, takže přidáním dalších komponent by nedošlo k výraznému zlepšení. Každý model, který byl zkoušen, se na trénovacích datech do jisté míry přetrénoval (na trénovacím datasetu byla úspěšnost klasifikace dosahující 100 %, na testovacím již výše zmíněných 70 %). Nejednalo se ale o nepřiměřeně velký pokles úspěšnosti.

4.3 Klasifikace nahrávky podle natrénovaného modelu

K získání pravděpodobností byla využita funkce z `ikrlib` nazvaná `logpdf_gmm` pro námi natrénovaný GMM model a pro příznaky extrahované z nahrávky. Tato funkce je původně určena k vypočítání log-likelihood pro klasifikaci mezi dvěma třídami, a tím pádem pro nás vrací velmi vysoké negativní hodnoty. Je tak potřeba tyto hodnoty upravit. Nejdříve tyto pravděpodobnosti upravíme tak, aby se rovnaly inverzní hodnotě podílu pravděpodobnosti pro danou třídu a součtu pravděpodobností pro všechny třídy. Hodnota se převrací z toho důvodu, že v původně vracených pravděpodobnostech představuje nižší hodnota vyšší pravděpodobnost klasifikace do třídy. Poté se od upravených hodnot odečte minimální likelihood. Takový mechanismus tu existuje proto, aby klasifikátor byl více jistý klasifikací do dané třídy. Poté se každá hodnota vydělí sumou všech

hodnot, čímž vzniknou pravděpodobnosti klasifikace od 0 do 1 a jejichž suma je 1. Nahrávka je pak klasikovaná do třídy určené argmaxem těchto hodnot.

K ověřování úspěšnosti natrénovaného modelu pro trénovací a testovací datasety se používají metody `fit_and_verify_train` a `fit_and_verify_test`. Tyto metody provádí klasifikaci za pomocí natrénovaného modelu do specifické třídy dle metody popsané výše a ověření, zda byla klasifikace provedena správně, Obě metody vrací podíl úspěšných klasifikací.

4.4 Potenciální zlepšení

Pro klasifikaci zvukových nahrávek nebyla použita augmentace, která by mohla potenciálně zvětšit množství trénovacích dat a tím pádem zvýšit přesnost natrénovaného modelu. Také by potenciálním zlepšením mohl být lepší způsob extrakce příznaků (například delta MFCC) či na výstupu klasifikačních pravděpodobností. Jednou z navrhovaných úprav na přednáškách také mohlo být ořezávání prvních několika sekund z důvodu ticha (řečník stiskl tlačítko, počkal, a až pak mluvil). Ořezávat nahrávky by se dalo buď konstantně (projít je ručně a říct si, že třeba první tři sekundy se nikdy nemluví), nebo na základě energií odstranit místa, která nejsou významná (problematická by v tomto mohla být situace, kdy je na pozadí nějaký výrazný šum, například větrák či nekvalitní mikrofon).

5 Vytvořené klasifikátory

V rámci řešení projektu jsme vytvořili několik způsobů, jak se dá klasifikovat. První dva jsou základní, tedy buď využít některý z klasifikátorů zmíněných v předchozích dvou sekcích (soubory **photo_neural.txt** a **voice_gmm.txt**, nebo je nějakým způsobem kombinovat. V rámci řešení projektu jsme pak dosáhli tří různých způsobů, jakým kombinovat.

Prvním je jednoduše odvozené pravděpodobnosti vzájemně vynásobit. Problémem je, že zatímco naše natrénovaná neuronová síť vždy klasifikovala velmi ostře (nějaká třída dosahovala výrazně vyšší pravděpodobnosti), tak GMM vždy testovací data označovala spíše nejistě, na první pohled hodnoty vypadaly jako vygenerované z rovnoměrného rozdělení. To nás při řešení přivedlo k tomu nějakým způsobem posilovat výstupní hodnoty z GMM, a to tak, že jsme je násobili nějakou hodnotou v rozsahu 10 až 30. S tímto přístupem jsme dosáhli vrcholu zhruba na 52 správně klasifikovaných testovacích datech ze 62. Výsledky jsou v souboru **combined_products.txt**.

Druhým principem bylo vypočítat průměrné hodnoty jednotlivých klasifikací (sečíst dva vektory a následně jejich hodnoty podělit dvěma). To se ukázalo jako mírně lepší řešení, které vedlo při správně zvoleném násobení zvukové klasifikace k až 53 správně klasifikovaným vzorům. Výsledky jsou v souboru **combined_averages.txt**.

Třetím principem kombinace klasifikátorů je způsob, který je v zásadě nejsnazší, ale současně i mnohdy nejúčinnější. Tentokrát jsme ale odvozovali pravděpodobnosti na výběrem na základě toho, který ze dvou klasifikátorů si je svou klasifikací více jistý. Tedy například pokud zvuk řekne nejvyšší pravděpodobnost 0.59 pro třídu 1 a obrázek nejvyšší pravděpodobnost 0.61 pro třídu 16, pak je zvolena třída 16. Spolu s průměrem se jedná o nejlepší řešení, které s ním soupeřilo a nakonec dosáhlo 56 úspěšně odhadnutých vzorů.Výsledky jsou v souboru **combined_compares.txt**.

6 Shrnutí a závěr

Na množství dat, které nám byly poskytnuty, se povedlo vytvořit relativně funkční klasifikátory, které v kombinaci dosáhly kolem 90% úspěšnosti na testovacím datasetu. Nejlepším přístupem se ukázala kombinace GMM a neuronové sítě kombinované na principu porovnání, kdy zvolená třída byla zvolena výběrem maximální hodnoty z dvojice výstupu GMM/neuronová síť.