Popis řečového modelu

V teto sekci bude popsán postup tvorby a princip fungování tohoto řečového modelu. Základní generativní model byl sestaven z ukázek a následně byly provedeny pro nedostatek času pouze drobnější změny.

Extrakce příznaků

Pro extrakci MFCC příznaků byla použita funkce *wav16khz2mfcc* z demo ukázek na stránkách předmětu. Výstup je pak slovník obsahující jméno souboru a 2d pole (T,13) kde T je počet rámců a každý z nich má 13 MFCC příznaků.

CMVN normalizace

Data jsou zarovnána na stejný dynamický rozsah odečtením střední hodnoty a podělením rozptylem dat.

Při trénování je tato normalizace provedena v rámci 1 věty, kvůli vyrovnání energie signálu každé nahrávky - aby například hlasitost a šum dané nahrávky neměly na klasifikaci vliv. Zároveň je provedena normalizace samotných nahrávek v rámci celé třídy (pro jednoho mluvčího), aby si jednotliví mluvčí byli mezi sebou rovni. Jinými slovy všechny odchylky v rámci jednotlivých nahrávek daného mluvčího, které jsou způsobeny například typem mikrofonu nebo prostředím, budou vyhlazeny a zůstanou pouze informace charakterizující hlas mluvčího, které jediné mají mít vliv na jeho klasifikaci. Při klasifikaci jsou pak vstupní data normalizována pouze v rámci 1 věty, jelikož nemáme k dispozici více nahrávek daného mluvčího - klasifikace probíhá po větách.

Redukce dimenze dat

Redukce je realizována pomocí funkce *LinearDiscriminantAnalysis* z knihovny *sklearn* se zvoleným solverem *eigen*. Ten se snaží najít takovou transformaci (natočení), která maximalizuje varianci středních hodnot tříd a minimalizuje varianci dat uvnitř tříd. Snaží se tedy najít vlastní vektory s největším vlastním číslem, které odpovídá podílu globální a lokální varianci tříd. Počet vlastních vektorů, které odpovídají výsledné dimenzi dat po transformaci se pak vybírá jako minimum z počtu tříd-1 a dimenze vstupních dat.

Generativní model

Z normalizovaných a transformovaných trénovacích dat každé třídy je vypočítán vektor středních hodnot a kovarianční matice pomocí funkce *train_gauss*.

Během klasifikace se vstupní data také normalizují (popsáno výše) a transformují stejně jako data trénovací. Poté se pro každý rámec věty určí vektor log. pravděpodobností všech tříd a vypočítá se průměr ze všech rámců. Výsledkem je tedy 1 vektor obsahující průměrnou log. pravděpodobnost pro každou třídu ze všech rámců (pro celou nahrávku). Výsledkem klasifikace je pak číslo kategorie s největší log. pravděpodobností. Jelikož se snažíme pouze určit, do které třídy dato patří a ne jaká je konkrétní pravděpodobnost, nemusíme započítávat hodnotu evidence. Apriorní pravděpodobnosti jednotlivých tříd jsou stejné, jelikož nedává

smysl zvýhodňovat třídy s větším počtem rámců v trénovací sadě - tato informace neříká nic o charakterech hlasu daných mluvčích.

Testování

Model byl trénován na všech datech ve složce *train* a validován na datech ve složce *dev* ze zadání. Zkoušel jsem použít diagonální kovarianční matici místo plné, ale přesnost model jak nad trénovacími, tak nad testovacími daty se znatelně snížila - nejspíš je mezi daty pro danou LDA transformaci nezanedbatelná korelace. Po několika pokusech vyšel jako nejlepší počet komponent LDA 11, které zlepšilo přesnost nad testovacími daty o několik procent oproti původnímu automatickému určení (popsáno výše) - opět jsem pro jistotou vyzkoušel diagonální kovarianční matici, ale opět bez úspěchu. Parametry tohoto trénování byly tedy pomocí knihovny *joblib* uloženy jako finální do souboru *model_params*.

Model byl napsán pythonu ve verzi 3.11.11. Pro maticové a vektorové operace byla využita knihovna numpy. Pro natrénování modelu stačí zavolat funkci *train_model* s cestou k datum typu *wav* (data musejí být roztříděny ve složkách s názvem dané třídy) a k souboru pro uložení parametrů. Pro otestování přesnosti modelu nad zadanými označenými daty stejně jako v předchozím případě a pro konkrétní parametry stačí zavolat funkci *evaluate_set*, která vrací vypočítanou přesnost. Pro klasifikaci neoznačených dat (všechny musí být v jedné složce) stačí zavolat funkci *classify_set*, která do zvoleného souboru vypíše výsledek klasifikace ve formátu stanoveném v zadání.

Model by se dal zlepšit přidáním více zvukových dat pomocí augumentace původních. Také by šlo ořezat počátky zvukových stop, kde dochází k zapnutí mikrofonu a nejsou zde užitečná data. Dále by šlo zkusit třídy rozdělit na více komponent například pomocí EM algoritmu s vhodnou incializací například pomocí k-means clustering.