Documentatie Surgical Mask Detection-Support Vector Machine

Student: Epure Cosmin-Andrei

Grupa:231

In urmatoarele randuri vom detalia cele folosite in rezolvarea proiectului din cadrul laboratorului de Inteligenta artificiala, in partea de Machine Learning.

Mai intai de toate, o scurta prezentare a cerintei proiectului:

Trebuie sa antrenam o masina sa recunoasca daca un vorbitor are gura acoperita cu o masca chirurgicala sau nu.

Pentru antrenarea masinii vom folosi un fisier text “train.txt”, care contine pe prima coloana id-ul fisierului audio(.wav) care va fi importat sip e a doua coloana 0 sau 1 reprezentand label-ul folosit pentru antrenare(0 - nu are masca, 1 - are masca).

Dupa ce masina a fost antrenata, urmeaza testarea acesteia. Pentru testare avem alt fisier text, de aceasta data acesta are doar o singura coloana(aceea cu id-ul) si trebuie sa ii fie asociat fiecarei coloane label-ul 0 sau 1 bazat pe predictiile masinii noastre.

Support Vector Machine:

Pentru implementare am ales metoda Support Vector Machine furnizata de biblioteca sklearn.

Un model SVM este o reprezentare a datelor asemenea unor puncte intr-un spatiu, asezate (sau “mapate”) astfel incat acestea sa fie impartite in doua “ferestre”, fiecare fereastra reprezentand o categorie, iar spatiul dintre cele doua ferestre sa fie cat mai larg.

Inainte de aplicarea acestei metoda este nevoie sa importam datele.

*MEL-frequency cepstral coefficients(MFCC) = “a represantion of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a non linear mel scale of frequency”(Wikipedia)

Importarea datelor:

(Pentru a incarca fisierele audio intr-un format favorabil am folosit biblioteca librosa.)

Importarea datelor a fost realizata liniar. Fiecare fisier .wav a fost incarcat pe rand, folosind metoda librosa.load()(*aceasta incarca fisierele .wav sub forma:[audio time series, sampling rates*]).[audio time series] ale fiecarui fisier vor fi folosite ca “feature-uri”.

-Prin audio time series intelegem “un vector bidimensional care contine valorile numerice (ale vectorului prin care fisierul audio poate fi reprezentat) in functie de timp.”(gaussianwaves.com)

-Prin sampling rates intelegem “de cate ori pe secunda un sunet este “probat”(techterms.com)

Pentru usurarea training-ului audio time sieres-urile au fost si sampling rates-urile au fost convertie in “mel-frequency cepstral coefficients”* prin functia librosa.feature.mfcc() iar apoi au fost “approximate” cu metoda mean() din biblioteca numpy(*aceasta returneaza media fiecarui vector dat) si adaugate intr-o lista x_train[]*.

La final acestea sunt salvate intr-un document .npy (np.save) pentru usurarea rularilor ulterioare ale programului.

Label-urile au fost salvate anterior prin aceeasi metotda (citite pe rand) in fisiere .npy.

Training:

Cum am zis mai sus, pentru training am folosit SVM-uri. Inainte de a incepe training-ul, fiecare feature a fost normalizat folosind metoda normalize_data() (laboratorul 5).

Prin intermediul datelor normaliazte vom putea sa antrenam masina prin intermedil metodei fit()(In acest caz i-am dat valoare 100 hipermarametrului C, deoarece vrem cea mai buna acuratete la final). Apoi folosind metoda pickle.dump() salvam modelul pentru usurarea rularii programelor.

Dupa antrenarea masinii, aceasta este pregatita sa faca predictii pe fisierele audio incarcate ulterior prin intermediul metodei predict(), ce

primeste ca parametru matricea normalizata de valori ale mfcc-urilor ale fisierelor audio pentru care dorim sa facem predictia.

Afisarea se face sub forma “nume_fisier”,”label”.

Accuracy:0.53

Kernel:Linear(timp scurt de asteptare)

Parametru C:100—0.53