Projekt I - Klasyfikacja cyfr (2018)

Wstęp

Celem ćwiczenia jest zbudowanie systemu automatycznej klasyfikacji cyfr wypowiadanych przez różnych mówców i oszacowanie w procesie testowania krzyżowego (*kroswalidacja*, *N* x 80% vs 20% lub metodą *leave-one-out*) jego skuteczności wyrażonej jako ERR (*Error Rate*, %) lub RR (*Recognition Rate*, %) wraz z wartością odchylenia standardowego wyniku (tylko dla testu krzyżowego 80/20) z wyznaczeniem macierzy błędów (*confusion matrix*).

Wskazówki

- Pliki zawierające nagrania samogłosek do treningu/testowania można znaleźć na stronie przedmiotu. Są to nagrania cyfr artykułowanych przez różne głosy męskie. Każdy głos wypowiada daną samogłoskę jednokrotnie. Wypowiedziana cyfra zawarta jest między *podkreślnikami* w nazwie każdego pliku.
- Poruszanie się po katalogach (i nazwach plików) najlepiej zrealizować posługując się pakietem *os (https://docs.python.org/3.6/library/os.html)*.
- Aby zminimalizować czas działania systemu najlepiej sparametryzować wszystkie pliki (i zapisać) przed wszystkimi kolejnymi operacjami i poza pętlą iterującą testy każdego pliku i pętlą iterującą kolejne permutacje kroswalidacji. Zapis/odczyt do plików można zrealizować m.in. za pomocą pakietu *pickle (https://docs.python.org/3/library/pickle.html)*.
- Można wykorzystać dowolny rodzaj parametryzacji. Sugerowane: MFCC lub PLP.
- Dowolny klasyfikator, który zapewni rozpoznanie większe od losowego. Sugerowane klasyfikatory to: GMM, ANN, SVM, kNN, kNN2, Tree, Random Forest, itp.
- Należy przemyśleć typ i złożoność klasyfikatora (np. rozmiar sieci neuronowej, liczbę komponentów GMM itp.) w kontekście posiadanych danych (liczba klas, długość bazy wzorców, jakość nagrań, dostępne zasoby obliczeniowe, moc CPU itp.).
- Warto zaprojektować system, w którym w czasie uruchomienia można zadeklarować (np. jako argument funkcji uruchamiającej cały system lub ustawienie wartości w pliku konfiguracyjnym) najważniejszych parametrów (np. długość ramki, liczbę filtrów melowych, rząd predyktora, liczbę współczynników cepstralnych, liczbę komponentów GMM, typ macierzy kowariancji, liczbę permutacji testu krzyżowego - jeśli dotyczy, i inne...). Pozwoli to na końcu na dobranie najlepszych parametrów aby osiągnąć najlepszą skuteczność.
- **Bardzo ważne jest przemyślenie i opisanie dokładne struktury całego systemu przed rozpoczęciem implementacji**. Zapobiegnie to później wielu problemom.
- Test krzyżowy powinien polegać na usunięciu 20% losowych mówców (nie losowych plików tych mówców) ze zbioru treningowego i weryfikacji skuteczności na ich podstawie (próbki testowanych mówców nie biorą udziału w treningu). Podobnie test *leave-one-out* polegać ma na usunięciu wszystkich plików jednego mówcy ze zbioru treningowego.

Zaliczenie zadania

- System realizowany będzie w trakcie 3 kolejnych spotkań laboratoryjnych.
- Proponowany plan działań
    - Lab 1 - architektura systemu, diagram Ganta, wybór metod i bibliotek, wczytanie plików i odkodowanie ich nazw, parametryzacja całego zbioru.
    - Lab 2 - Implementacja pętli testu krzyżowego (sklearn.model_selection.KFold.split) - pętli treningu/testowania, implementacja treningu klasyfikatora.
    - Lab 3 - Implementacja treningu i klasyfikatora c.d., wykonanie testów i optymalizacja klasyfikatora, test na danych ewaluacyjnych.

- Pracujemy w stałych grupach od 1 do 3 osób.
- Do zaliczenia konieczne jest ukończenie systemu i uzyskanie wyników testu skuteczności dla zbioru ewaluacyjnego.
- Ocenie podlegać będą:
    - Systematyczność w pracy nad systemem (z tygodnia na tydzień) weryfikowana na podstawie cotygodniowych raportów (PDF z, max. 2 str A4) zawierających np.: koncepcję systemu, zadania zrealizowane, zadania do zrealizowania, analizę problemów napotkanych w trakcie pracy, wnioski uzyskanych wyników, cząstkowe rezultaty, wykresy pośrednich etapów działania itp.
    - Raporty należy przesłać korzystając z Uczelnianej Platformy e-Learningowej przed rozpoczęciem się kolejnych zajęć laboratoryjnych.
    - W raporcie należy zamieścić skład zespołu, numer spotkania.
    - Raporty mają być redagowane w sposób przyrostowy (jak blog), nie jako kolejne nowe pliki.
    - Prawidłowość dobranych metod i/lub ich ciekawe i racjonalne uzasadnienie.
    - Wartości i jakość uzyskanych wyników ostatecznych (RR, ERR, macierze błędów) i konstruktywne uzasadnienie ich wartości.
    - Przed ostatnimi zajęciami zostanie opublikowany ewaluacyjny zbiór plików o zawartości podobnej do tej ze zbioru treningowego. Pliki nie będą opisane. Napisany system powinien zapisać plik csv zawierający w kolejnych liniach wpisy postaci:*nazwa pliku, rozpoznana cyfra, wartość log-likelihood*. Przykładowa zawartość pliku:

```
001.WAV,9,-35.87
002.WAV,2,-73.89
003.WAV,2,-32.99
004.WAV,3,-94.24
005.WAV,7,-94.20
006.WAV,5,-68.69
007.WAV,2,-31.14
008.WAV,7,-49.07
009.WAV,2,-77.84
010.WAV,9,-20.62
```

        - Plik posłuży do oceny jakości systemu na zupełnie obcych danych (niedostępnych w czasie treningu i testów krzyżowych).
        - Dobre wyniki testu na obcych danych będą dodatkowo punktowane przy wystawianiu oceny końcowej.