

## L7. Dynamic Time Warping (DTW)

(Alinierea Dinamică în Timp)

Una dintre dificultățile întâlnite în recunoașterea vorbirii constă în faptul că, în ciuda includerii (mai mult sau mai puțin) a aceluiași sunet în aceeași ordine în cadrul unor înregistrări diferite ale aceluiași cuvinte, nu vor exista sincronizări /potriviri exacte în ceea ce privește duratele fiecărui subcuvânt în cadrul cuvântului. Prin urmare, eforturile de a recunoaște cuvinte prin potrivirea lor cu șabloane vor conduce la rezultate inexacte dacă nu există o aliniere temporală. Deși a fost înlocuită pe scară largă de către modelele Markov ascunse (HMM), tehnica de programare dinamică DTW este folosită la recunoașterea vocală pentru a estompa diferențele temporale între rostirile test și șabloane (templates). Principiul pe care stă la baza acestei tehnici este reprezentat de existența unor 'pași' în spațiu (cadre temporale în rostirile test, respectiv cadre temporale în template-uri/modele) și de a găsi în acel spațiu calea care maximizează potrivirea locală între cadrele temporale aliniate care fac subiectul unor constrângeri implicite pașilor permisi. 'Costul de similaritate' total găsit prin acest algoritm este o indicație a cât de bună este potrivirea între rostirile-test și șabloane (template), aceasta informație putând fi utilizată apoi pentru a alege șablonul (template-ul) care se potrivește cel mai bine.

Codul și exemplul dat este o implementare simplă a alinierii DTW pentru fișiere de sunet, pentru a evalua similaritatea între sunete și pentru a alinia tempo-ul unui fișier audio pentru a se potrivi cu al unei referințe, de exemplu pentru a sincroniza două pronunții (rostiri) ale aceluiași cuvânt.

Rutinele furnizate sunt:

`simmx.m` - facilitează calcularea matricii de matching local total, respectiv pentru a calcula distanța dintre oricare pereche de cadre (vectori acustici) din semnalul de test și semnalul model/template.

`dp.m` - implementează algoritmul de programare dinamică, simplu, care permite 3 pași (constrângeri locale) - (1,1), (0,1) și (1,0) - cu ponderi egale.

`dp2.m` - versiune alternativă experimentală care permite 5 pași (constrângeri locale) - (1,1), (0,1), (1,0), (1,2), și (2,1) - cu ponderi diferite pentru a indica 'cale abrupte', dar fără o limitare rigidă în ceea ce privește regiunile în care se găsesc potrivirile. Sintaxa etc. la fel ca la `dp.m`

`dpfast.m` - versiune rapidă care utilizează o versiune MEX (`dpcore`) pentru a executa bucla internă ne-vectorizabilă, de până la de 200 de ori mai rapid! Permite specificarea de către utilizator a matricii pas/cost.

`dpcore.c` - cod C sursă ptr. rutina MEX care eficientizează ca viteză rutina `dpfast.m` (`help mex`).

### Mersul lucrării

1. Studiați principiul metodei DTW din cursul 9.
2. Analizați aplicația `dtw.m` și rutinele folosite.
3. Modificați aplicația astfel încât fișierele de lucru (\*.wav) să se introducă la execuția aplicației.
4. Ce parametri se utilizează pentru vectorii acustici?
5. Testați cele 3 variante ale algoritmului de programare dinamică (`dp.m`, `dp2.m`, `dpfast.m`) pentru diferite fișiere corespunzând aceleiași rostiri (ex. 929\_\*.wav). Comparați distanțele obținute funcție de duratele rostirilor. Comparați duratele de execuție a aplicațiilor (`dp.m`, `dp2.m`, `dpfast.m`) funcție de rostirile de intrare aliniate.
6. Repetați experimentele pentru rostiri diferite (ex. hall.wav; girl.wav;...)
7. Faceți capturi ale aplicației pentru diferitele situații și includeți-le în raport împreună cu rezultatele anterioare și comentariile corespunzătoare.
8. Modificați aplicația pentru a extrage coeficienții de predicție (LPC 12) ca vectori acustici. (optional).