Speech Recognition using DTW and HMM

Jan Černocký, Valentina Hubeika, FIT BUT Brno

23. dubna 2009

In this lab we are going to look at two basic isolated words recognition techniques:

- Dynamic Time Warping, that compares two matrices of parameters using optimal path setting
- Hiden Markov Models, where statistical models are trained on the training set that are alter used for test segments scoring.

1 Essential terms

Reference utterance or **reference sequence** is data that are known; used during training. **Test utterance** nebo **test sequence** is unknown; used as the input to the trained recognizer. In this lab, we know the content of the test utterance as we want to test whether our recognizer is performing well.

2 Signals and Parameterization

LPC-cepstral coefficients are used as features (see the lecture on LPC). Function `c_matrice.m` implements LPC computation. In the function `a_to_cepst.m`, look at the conversion of LPC to LPCC.
We dispose of 4 training utterances and 7 test utterances. By running

`sigceps`

We load the data and apply parameterization. The result is:

signal	cepstral matrix	label
Training		
s1	c1	jedna
s2	c2	dvě
s3	c3	tři
s4	c4	čtyři
Testování		
s1t	c1t	jedna (different utterance)
s2t	c2t	dvě (different utterance)
s3t	c3t	tři (different utterance)
s4t	c4t	čtyři (different utterance)
s1l	c1l	jééédna
s2l	c2l	dvjééé
sb	cb	bedna

3 DTW

Using DTW, we compare two sequences of vectors: reference $\mathbf{R} = [\mathbf{r}(1), \dots \mathbf{r}(R)]$ of the length R and test $\mathbf{O} = [\mathbf{o}(1), \dots \mathbf{o}(T)]$ of the length T. The aim is to calculate the distance between them $D(\mathbf{O}, \mathbf{R})$. In the simplest case, the i-th w2ord is represented by the word w_i in the vocabulary $\mathbf{R}_i$, the recognizing word is given thus as:

$$i^\star = \arg\min_i D(\mathbf{O}, \mathbf{R}_i). \tag{1}$$

During DTW, we define a matrix $\mathbf{D}$ of the size $T \times R$, that is filled with the local distances between the vectors: $d(i,j) = d(\mathbf{o}(i), \mathbf{r}(j))$. Distance $d(\cdot,\cdot)$ is here the cepstral measure. Another important matrix is $\mathbf{G}$ – matrix of the partial cumulated distances. Compared to $\mathbf{D}$, matrix $\mathbf{G}$ has zeros row and zeros column, that are initialized with

the values ∞, except the cell $g(0,0) = 0$. For the local restriction of the path, type I, and for the type weights a. a. (see lecture on DTW) we can calculate next items **G**:

$$g(i,j) = \min \begin{cases} g(i-1,j) + d(i,j), \\ g(i,j-1) + d(i,j), \\ g(i-1,j-1) + 2d(i,j) \end{cases} \tag{2}$$

for $1 \leq i \leq T$ and $1 \leq j \leq R$. The last matrix item $g(T,R)$ presents the optimal distance:

$$D(\mathbf{O},\mathbf{R}) = \frac{g(T,R)}{T+R} \tag{3}$$

In matrix **G** we can find the optimal path backwards. All these operation are implemented in the function `dtw.m`. Look at the function **carefully**. The function uses the DTW distance as the input. The graphical output of the function is realized:

- the upper panel presents matrices of the local distances (vectors "all by all").
- the middle panel presents matrices of partial cumulated distances.
- the bottom panel presents the path. The panel title presents the DTW distance.

3.1 Comparison of the matrix with itself

First, compare one training utterance with itself:

```
D=dtw (c4, c4); soundsc ([s4 s4])
```

Tasks

1. How much is the calculated DTW distance ?
2. Why is the comparison path totally straight ?
3. Why do we see in the comparison matrix a "quatrefoil" in the upper right corner ?

3.2 Matrix comparison of two utterances of the same word

```
D=dtw (c4, c4t); soundsc ([s4 s4t])
```

Tasks

1. Comment of what you see.

3.3 Comparing two different matrices of the same word

Try 'jedna' and 'jééédna':

```
D=dtw (c1, c1l); soundsc ([s1 s1l])
```

Tasks

1. Can you see on the comparison path the segments of short and long 'e' ?

3.4 Real Recognition

Till now we were comparing one sequence to another, no actual recognition was done. Actual recognition has to compare the test utterance to all reference utterances. The following code will open 4 windows, put them vertically next to each other, dont overlap the Matlab interpret. We are going to recognize the utterance `s2t`, which we know contains the word 'dvě', but our recognizer has no knowledge on it.

```
figure(1); pause(1); soundsc([s2t s1]); D1=dtw (c1, c2t);
figure(2); pause(1); soundsc([s2t s2]); D2=dtw (c2, c2t);
figure(3); pause(1); soundsc([s2t s3]); D3=dtw (c3, c2t);
figure(4); pause(1); soundsc([s2t s4]); D4=dtw (c4, c2t);
```

Display all distances and choose the best one - this will indicate the recognized word.

```
D1
D2
D3
D4
[m,mini]=min([D1 D2 D3 D4]);
disp(sprintf('Rozpoznano ======= %d ========\n', mini));
```

Tasks

1. Does the result satisfy the expectations ?
2. Can you identify on the optimal paths which of the utterances contain the vowel 'e' ?

3.5 The big thing

Let us now recognize all the test utterances. To get to a new utterance press Enter. Keep the graphical windows open.

```
reco_dtw
```

Tasks

1. Check the results.
2. Do you think in real tasks (noise, more speakers, large vocabulary, continuous words) we always achieve 100% accuracy? Can you guess what is the accuracy of state-of-the-art English recognizers (telephone calls, meetings)?

4 Hidden Markov Models (HMM)

This part of the lab presents HMM recognizers. Our models are characterized by:

- N states, only $N-2$ are "emmiting", the first and the last one are used only for convenience
- Left-to-right structure. Matrix of transition probability has the nonzero items only on the diagonal.
- the emmition probabilities are modeled using

P-dimensional Gaussian distribution:

$$b_j[\mathbf{o}(t)] = \mathcal{N}(\mathbf{o}(t); \mu_j, \mathbf{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^P |\mathbf{\Sigma}_j|}} e^{-\frac{1}{2}(\mathbf{o}(t)-\mu_j)^T \mathbf{\Sigma}_j^{-1} (\mathbf{o}(t)-\mu_j)}, \tag{4}$$

As the input sequence **O** LPC-cepstral vectors are used without c_0 – same as for DTW.

4.1 Model training

For the training use:

- `initemis.m` — initializes "emmiting" probability distribution. the input is the matrix of observations and number of states, the output is matrices **MI** and **SIGMA**. All emmiting states are initialized to **same values** — global mean and variance.
- `inittran.m` — initializes matrix of transition probabilities. The input is the number of states, the output is matrix **A**. Transition probability is 0.5. Persistence probability is 0.5.
- `reestim.m` — Baum-Welch re-estimation. Input is the sequence of observations, **A**, **MI** and **SIGMA**.

  The function estimates (see slides `09_hmm.pdf`):

  – partial likelihoods $\alpha_j(t)$,.
  – partial backward likelihoods $\beta_j(t)$.

– posterior probabilities.
  Values of $L_j(t)$ has to sum up to 1.
– HMM parameters.

Each word is represented by 1 model.

4.2 Recognition

4.3 Initialization

We work with 9 states models.
Lets play with the training word "jedna". First initialize the models:

```
N=9; A=inittran(N); [MI,SIGMA]=initemis(c1,N);
A
MI
SIGMA
```

4.4 Baum-Welch retraining – 1. iteration

Rus one re-estimation cycle and look at the results:

```
[NEWA, NEWMI, NEWSIGMA, Ptot, ALFA, BETA, L] = reestim (c1, A, MI, SIGMA);
ALFA
BETA
```

... look at the $L_j(t)$.

```
T = size (c1,2); plot (1:T, L)
sum(L(2:N-1,:))
```

4.5 More iterations of Baum-Welch

Estimate new parameters from the old ones.

```
[NEWA,NEWMI,NEWSIGMA,Ptot,ALFA,BETA,L] = reestim (c1, NEWA, NEWMI, NEWSIGMA); plot (1:T, L); Ptot
```

... and so on. `Ptot` is zero. Analyze what happened.

```
L
```

Run:

```
show_bw_iters
```

look at the script `show_bw_iters.m`, press enter..

4.6 The big thing

Train models for all 4 words (look at `train_hmms.m`).

```
train_hmms
```

The estimated parameters are in `Ax, MIx, SIGMAx`, where `x` is identity of the word.
Recognition is realized by `viterbi_log.m`, look inside and then run:

```
reco_hmms
```