Speech Recognition – Intro and DTW

Jan Černocký \{cernocky|ihubeika\}@fit.vutbr.cz

FIT BUT Brno
Speech Recognition

Goal: given an unseen speech signal estimate what was said

Classification:

• Isolated words – cell phone voice control. Need a voice activity detector or push-to-talk.

• Continuous words (constrained vocabulary) – e.g. figures in a telephone number or credit card number. The recognition is usually conducted by a network or a simple grammar.

• Large vocabulary continuous speech recognition LVCSR – hardest task. Requires information on acoustics but also the structure of the language (language model) and pronunciation dictionary. Works with smaller units than words (60 thousand words cannot be learned...) – phonemes, context dependent phonemes.
Structure of a recognizer

speech → feature extraction → acoustic matching → decoding

acoustic models or patterns

"bla bla bla"

language model
Parameterization

- Data size reduction.
- Discarding the components we are not interested in (pitch, mean value, phase)
- Usually based on spectral analysis (Mel-frequency cepstral coefficients) or LPC analysis (LPC cepstrum).
- Framing (quasi-stationarity)
- Parameters have to be convenient for the recognizer (uncorrelated parameters)
- See the lecture on parameterization!
The result of parameterization is a sequence of vectors: \( O = [o(1), o(2), \ldots, o(T)] \)
Acoustic matching — variability everywhere !!!

**parameter space** - a human never says one thing twice in exactly the same way ⇒ parameter vectors are **always different**. Methods working on text fail ⇒ How to do it?

1. Calculation of distance between two vectors.

2. Statistical modeling.
reference vectors

test vector

selected reference vector
winning distribution
**timing** – people never say one thing with the same timing.
timing n.1 - Dynamic time warping - path

dtw path, D=0.24094
timing n.2 - Hidden Markov models - state sequence

Markov model $M$

Observation sequence

Speech Recognition – Intro and DTW    Jan Černocký, Valentina Hubeika, DCGM FIT BUT Brno
Decoding

- Isolated words: very simple (select the maximum probability or the minimum distance).
- LVCSR: very difficult (acoustic models (tri-phones), language model, pronunciation model) - Viterbi algorithm, $A^*$-search, best-first decoding, finite state automata (!), search space narrowing (beam-search), etc.
Isolated words boundary detection:

- push-to-talk...
- speech activity detector – e.g. detector based on energy:
How does it work?

dictionary

word1
word2
word3

... 

wordN

speech signal

recognizer

"the word was word2..."
• the dictionary contains reference parameter matrices for the words of interest:

\[ R_1 \ldots R_N \]

• a test parameter matrix comes as an input to the recognizer \( O \)
• the task is to determine which reference word corresponds to the test word.

If the words were represented by only one vector, it would be simple:

\[
d(o, r_i) = \sqrt{\sum_{k=1}^{P} [o(k) - r_i(k)]^2}.
\]

A minimum distance would be chosen.
Words are however represented by more than one vector (a sequence): The task is to determine the distance (similarity) of the reference vector of the length $R$:

$$ R = [r(1), \ldots, r(R)] \quad (1) $$

and the test sequence of the length $T$:

$$ O = [o(1), \ldots, o(T)] \quad (2) $$

Calculating of distances between single vectors? How do they correspond to each other? Words are almost never represented by the sequence of the same length $R \neq T$. 
Linear Alignment

$$D(O, R) = \sum_{i=1}^{R} d[o(w(i)), r(i)]$$  \hspace{1cm} (3)$$

where $w(i)$ is defined so that the alignment is linear.

But this is not going to work in most cases... here though, still working:
...not working for this example (error of VAD):
In the best case the alignment is *conducted* by the distances between single vectors ⇒ **Dynamic Time Warping (DTW)**.

Define a time variable $k$ and *two* transformation functions:

- $r(k)$ for the reference sequence.
- $t(k)$ for the test sequence.

The vector alignment can be represented by a **path**. The number of steps of the path is denoted as $K$. The reference is represented by the vertical axes and the test sequence is represented by the horizontal axes. According to the path, the functions $r(k)$ and $t(k)$ are estimated. The functions trace single sequences.
The path $C$ is unambiguously given by its length $K_C$ and the course of the functions $r_C(k)$ and $t_C(k)$. For this path the distance between the sequence $O$ and $R$ is given as:

$$D_C(O, R) = \sum_{k=1}^{K_C} d[o(t_C(k)), r(r_C(k))] W_C(k) N_C$$  \hspace{1cm} (4)$$

where $d(o(\cdot), r(\cdot))$ is length of the vectors, $W_C(k)$ is the weight corresponding to the $k$-th step and $N_C$ is weight dependent normalization factor.

The distance between the sequences $O$ and $R$ is given as minimum distance over the set of all possible paths (all possible lengths, all possible courses):

$$D(O, R) = \min_{\{C\}} D_C(O, R).$$  \hspace{1cm} (5)$$

We need to solve 3 things:

1. allowed courses of the functions $r(k)$ and $t(k)$. The path isn’t allowed to take the opposite course, or skip frames, etc.
2. define normalization factors and the weighting function.
3. efficient and fast algorithm to calculate $D(O, R)$. 
1. **Start and terminal points**

\[
\begin{align*}
  r(1) &= 1 \quad \text{beginning} \\
  t(1) &= 1 \quad \text{beginning} \\
  r(K) &= R \quad \text{end} \\
  t(K) &= T \quad \text{end}
\end{align*}
\]

(6)

2. **Local correlation and local slope**

\[
0 \leq r(k) - r(k - 1) \leq R^* \\
0 \leq t(k) - t(k - 1) \leq T^*
\]

(7)

In practice \( R^*, T^* = 1, 2, 3 \).

- \( R^*, T^* = 1 \): each vector should be considered at least once. \( r(k) = r(k - 1) \) denotes repeated use.
- \( R^*, T^* > 1 \): Vector(s) can be skipped.
3. **Global path restriction in DTW**: restriction using lines:

\[
1 + \alpha [t(k) - 1] \leq r(k) \leq 1 + \beta [t(k) - 1] \\
R + \beta [t(k) - T] \leq r(k) \leq R + \alpha [t(k) - T]
\]
\[ r(k) = \beta[t(k)-1]+1 \]

\[ r(k) = \alpha[t(k)-1]+1 \]

\[ r(k) = t(k)+w \]

\[ r(k) = t(k)-w \]

\[ r(k) = \alpha[t(k)-T]+R \]

\[ r(k) = \beta[t(k)-T]+R \]
Weighting function $W(k)$ depends on local path progress. 4 types:

- **type a)** symmetric: $W_a(k) = [t(k) - t(k - 1)] + [r(k) - r(k - 1)]$.

- **type b)** asymmetric:
  - b1) $W_{b1}(k) = t(k) - t(k - 1)$
  - b2) $W_{b2} = r(k) - r(k - 1)$

- **type c)** $W_c(k) = \min \{t(k) - t(k - 1), r(k) - r(k - 1)\}$

- **type d)** $W_d(k) = \max \{t(k) - t(k - 1), r(k) - r(k - 1)\}$


\[ N = \sum_{k=1}^{K} W(k) \]  

(9)

For weighting function a) normalization factor is:

\[ N_a = \sum_{k=1}^{K} [t(k) - t(k-1) + r(k) - r(k-1)] = t(K) - t(0) + r(K) - r(0) = T + R \]  

(10)

For weighting function b1) je normalization factor \( N = T \).
For weighting function b2) je normalization factor \( N = R \).
For weighting function c), d) factor is strongly dependent on the path progress, better use constraint: \( N = T \).
The table presents types of local restrictions and corresponding factors $\alpha$ and $\beta$. Meaning of $g(n, m)$ will be explained later.
<table>
<thead>
<tr>
<th>Type</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>Type ( w(k) )</th>
<th>( g(n, m) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>0</td>
<td>( \infty )</td>
<td>a</td>
<td>( \min { g(n, m - 1) + d(n, m), g(n - 1, m - 1) + 2d(n, m), g(n - 1, m) + d(n, m) } )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>d</td>
<td>( \min { g(n, m - 1) + d(n, m), g(n - 1, m - 1) + d(n, m), g(n - 1, m) + d(n, m) } )</td>
</tr>
<tr>
<td>II.</td>
<td>( \frac{1}{2} )</td>
<td>2</td>
<td>a</td>
<td>( \min { g(n - 1, m - 2) + 3d(n, m), g(n - 1, m - 1) + 2d(n, m), g(n - 2, m - 1) + 3d(n, m) } )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>d</td>
<td>( \min { g(n - 1, m - 2) + d(n, m), g(n - 1, m - 1) + d(n, m), g(n - 2, m - 1) + d(n, m) } )</td>
</tr>
<tr>
<td>III.</td>
<td>$\frac{1}{2}$</td>
<td>2</td>
<td>a</td>
<td>$\min \begin{cases} g(n - 1, m - 2) + 2d(n, m - 1) + d(n, m) \ g(n - 1, m - 1) + 2d(n, m) \ g(n - 2, m - 1) + 2d(n - 1, m) + d(n, m) \end{cases}$</td>
</tr>
<tr>
<td>------</td>
<td>--------------</td>
<td>---</td>
<td>----</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>IV.</td>
<td>$\frac{1}{2}$</td>
<td>2</td>
<td>b1</td>
<td>$\min \begin{cases} g(n - 1, m) + kd(n, m) \ g(n - 1, m - 1) + d(n, m) \ g(n - 1, m - 2) + d(n, m) \end{cases}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>where $k = 1$ for $r(k - 1) \neq r(k - 2)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$k = \infty$ for $r(k - 1) = r(k - 2)$</td>
</tr>
</tbody>
</table>
Minimum distance computation

\[ D(O, R) = \min_{\{C\}} D_C(O, R). \]  

(11)

is simple, when normalization factor \( N_C \) is no function of path and we can write:

\[ N_C = N \quad \text{for} \quad \forall C \]

\[ D(O, R) = \frac{1}{N} \min_{\{C\}} \sum_{k=1}^{K_C} d[o(t_C(k)), r(r_C(k))] W_C(k) \]  

(12)

Procedure is the following:

1. the cell \( d \) of the size \( T \times R \) contains distances between the reference and the test vector, all by all.

2. define cell \( g \) with partial cumulated distance. Compared to cell \( d \), \( g \) has zero row and zero column, that are initialized to:

\[ g(0, 0) = 0, \quad a \quad g(0, m \neq 0) = g(n \neq 0, 0) = \infty. \]
3. partial cumulated distance (for each point) is calculated as:

\[ g(m, n) = \min_{\forall \text{predecessors}} \left[ g(\text{predecessor}) + d(m, n)w(k) \right] \]  

(13)

- predecessors are given by the restriction path table.
- weight \( w(k) \) corresponds to the \([m, n]\) point pass (from the predecessor).
- relations for the partial cumulated distance are tabled

4. Final minimum normalized distance is thus given by :

\[ D(O, R) = \frac{1}{N} g(T, R) \]  

(14)
**Example**

<table>
<thead>
<tr>
<th></th>
<th>d</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref.</td>
<td>4 3 2</td>
<td>ref.</td>
</tr>
<tr>
<td></td>
<td>2 3 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 2 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 1 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>test</th>
<th>0</th>
<th>inf</th>
<th>inf</th>
<th>inf</th>
</tr>
</thead>
</table>

Speech Recognition – Intro and DTW  
Jan Černocký, Valentina Hubeika, DCGM FIT BUT Brno
Result:

- given distance $D = \frac{1}{3+4} 7 = 1$.
- we can “step the optimal path” backward (the path has 5 steps): $t(k) = [1 \ 2 \ 2 \ 3 \ 3]$, $r(k) = [1 \ 1 \ 2 \ 3 \ 4]$.
Recap: what do we want? Given a word $O$ and a set of classes; we want to decide to which class, $\omega_r$, the word belongs. We dispose of $\tilde{N}$ classes, representing words (e.g. “one”, “two”, “three”, etc.).
Training – creating references or classes of patterns

during training, we dispose of data sequences from one or more speakers and we know two
which class each of them belong.

1. **simple:** each class $\omega_r$ is represented by one reference $R_r$.

2. **advanced:** each class $\omega_r$ is represented by several references: $R_{r,1} \ldots R_{r,\tilde{N}_r}$. These
can be stored (in the vocabulary) as generated or normalized to have same length:

$$
\bar{R} = \frac{1}{\tilde{N}_r} \sum_{r=1}^{N} \left[ \frac{1}{\tilde{N}_r} \sum_{i=1}^{\tilde{N}_r} R_{r,i} \right], \tag{15}
$$

where $R_{r,i}$ is the length of the $i$-th sample of the class $\omega_r$.

3. **average class pattern** $\omega_r$:
   - linear averaging – average of the linearly aligned vectors. Danger: can result in
     nonsense pattern . . .
   - dynamic averaging:
     (a) select a sample with appropriate length;
     (b) average samples aligned to this length using DTW.
4. training using *clustering*. Clusters are created to minimize within class variability and maximize across-class variability. There are several algorithms available, e.g. Mac Queen algorithm: align all the references to one cluster; the most distanced samples are split off the cluster thus forming new clusters; the data are realigned, etc.. Clusters are represented by centroids \( R_{ri} \). Advantage over averaging is that classes can have more complicated structure.
If each class is represented by one reference, classification is easy:

\[ \omega_r^{\star} = \arg \min_r D(O, R_r) \quad \text{pro} \quad r = 1, \ldots, N \]  

(16)

When classes are presented each by several references, we can approach one of the two solutions:

1. **1-NN** nearest neighbor:

\[ \omega_r^{\star} = \arg \min_{r,i} D(O, R_{r,i}) \quad \text{pro} \quad r = 1, \ldots, N \]

\[ i = 1, \ldots, N_r \]  

(17)
2. \textbf{k-NN} \( k \) nearest neighbors:

- for each class, calculate all distances \( D(O, R_{r,i}) \) and sort them from the best to the worst:

\[
D(O, R_{r(1)}) \leq D(O, R_{r(2)}) \leq \ldots \leq D(O, R_{r(N_r)})
\] (18)

- sample \( O \) is assigned to the class \( \omega_r \) according to the average distance of the \( k \) nearest neighbors:

\[
\omega_r^* = \arg \min_r \frac{1}{k} \sum_{i=1}^{k} D(O, R_{r(i)})
\] (19)