Text Algorithms (4AP)
Lecture: Time warping and sound …

Jaak Vilo

2008 fall

Soundex distance metric

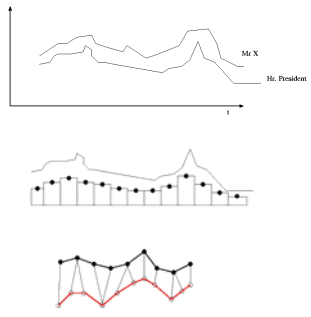
- Soundex is a coarse phonetic indexing scheme, widely used in genealogy. This approach focuses upon individuals names and as such has not been provably applied to a more general context.
- Soundex allows phonetic mispellings to be easily evaluated, for instance the names *John*, *Johne* and *Jon* are often geneologically the same person. This is a term based evaluation where each term is given a Soundex code, each soundex code consists of a letter and three numbers between 0 and 6, e.g. **"Chapman" is "C155".** The letter is always the first letter of the surname. The numbers hash together the rest of the name. This approach is very promising for disabiguation of translitterated/misspelt names, i.e non english names represented as ASCII are frequently misrepresented. The basic algorithm for soundex is now detailed.

Soundex cnt.

- A soundex code consists of the first letter of the surname, followed by three digits. The digits are based on the consonants as in the following table, (this can differ in implementations):
- 1) B,P,F,V
  2) C,S,K,G,J,Q,X,Z
  3) D,T
  4) L
  5) M,N
  6) R
- The vowels are not used. If two or more adjacent (not separated by a vowel) letters have the same numeric value, only one is used. This also applies if the first (which is used for the letter in the code), and the second letter in name have the same value; the second letter would not be used to generate a digit. If there are not three digits after the consonants are convert, the code is filled out with zeros. The name Sue has no consonants after the S, so the soundex code would be S000.
- This Metric is included in the SimMetric open source library.

Time warping

- Previous examples all worked on strings or character sequences
- But if we had a sequence of numercial values, e.g. a time course?
- Are two numerical time courses similar?
- Usage:
- Speech recognition
- Comparison of dynamic processes, e.g. the gene expression changes



Dynamic Time Warping and Minimum Distance Paths for Speech Recognition

Isolated word recognition:

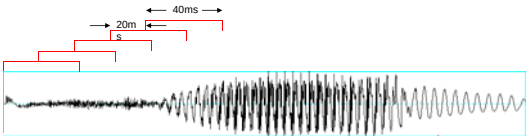
- Task :
  - Want to build an isolated 'word' recogniser e.g. voice dialling on mobile phones
- Method:
  1. Record, parameterise and store vocabulary of reference words
  2. Record test word to be recognised and parameterise
  3. Measure distance between test word and each reference word
  4. Choose reference word 'closest' to test word

6

Words are parameterised on a frame-by-frame basis

Choose frame length, over which speech remains reasonably stationary

Overlap frames e.g. 40ms frames, 10ms frame shift



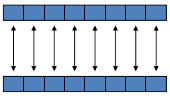
We want to compare frames of test and reference words
i.e. calculate distances between them

7

Calculating Distances

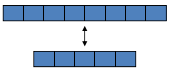
- Easy:

  Sum differences between corresponding frames



- Problem:

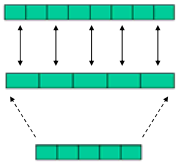
  Number of frames won't always correspond



8

- Solution 1: Linear Time Warping

  Stretch shorter sound



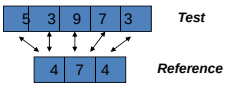
- Problem?

  Some sounds stretch more than others

9

- Solution 2:

  Dynamic Time Warping (DTW)



Using a dynamic alignment, make most similar frames correspond

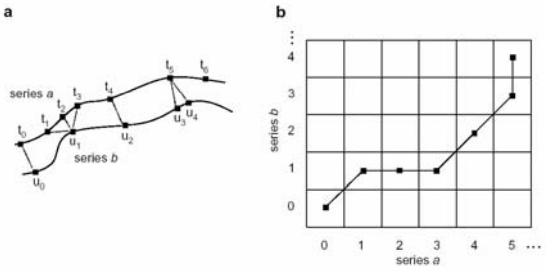
Find distances between two utterences using these corresponding frames

10

- Discretise the time (e.g. A/D mapping)
- Search for best match between sequences
- Can use the formulas from integral calculus
- substitution cost is replaced by mathematical formulas and interpolations
- Usually just narrowing and widening, no deletion or insertion
- Let be the discrete time unit T

- $T_{ij} = \min \begin{cases} T_{i-1,j} + 1/2 \cdot T \cdot w(ai,bj) \\ T_{i-1,j-1} + T \cdot w(ai,bj) \\ T_{i,j-1} + 1/2 \cdot T \cdot w(ai,bj) \end{cases}$

- **w(ai,bj) could be, for example, |ai - bj|**
- One can define several mappings and costs, these lead to slightly modified recurrence formulas



Demo

```
vilo@muhu:~/TextAlgorithms$ head -15 time_warp.pl
@S1 = qw( 0 5 3 9 7 3 4 3 8 ) ;
@S2 = qw( 0 4 7 4 7 8 9 ) ;
print join "\t" , @S1 , "\n" ; print join "\t" , @S2 , "\n" ;

for ( $j = 0 ; $j < @S1 ; $j++ ) {  $D[0][$j] =  $D[0][$j-1] + abs( $S1[$j] - $S2[0] )  }
for ( $i = 0 ; $i < @S2 ; $i++ ) {  $D[$i][0] =  $D[$i-1][0] + abs( $S2[$i] - $S1[0] )  }

for ( $i = 1 ; $i < @S2 ; $i++ ) {
 for ( $j = 1 ; $j < @S1 ; $j++ ) {
   $D[$i][$j] = minval(
            $D[$i-1][$j-1] + abs( $S2[$i] - $S1[$j] ) ,
            $D[$i-1][$j] +   abs( $S2[$i] - $S1[$j] ) / 2 ,
            $D[$i][$j-1] +   abs( $S2[$i] - $S1[$j] ) / 2 ,
    ) ;
 }
vilo@muhu:~/TextAlgorithms$
```

```
vilo@muhu:~/TextAlgorithms$ perl time_warp.pl
0   5   3   9   7   3   4   3   8
0   4   7   4   7   8   9
D( 0,5,3,9,7,3,4,3,8 -:- 0,4,7,4,7,8,9 ) = 6.5
        0   5   3   9   7   3   4   3   8
0   0   5   8   17  24  27  31  34  42
4   4   1   1.5 4   5.5 6   6   6.5 8.5
7   11  2   3.5 3.5 3.5 5.5 7   8.5 7.5
4   15  2.5 3   5.5 5   4.5 4.5 5   7
7   22  3.5 5   5   5   6.5 6   7   6
8   30  5   7.5 5.5 5.5 8   8   9.5 6
9   39  7   10  5.5 6.5 9.5 10.5 12.5 6.5
vilo@muhu:~/TextAlgorithms$
```

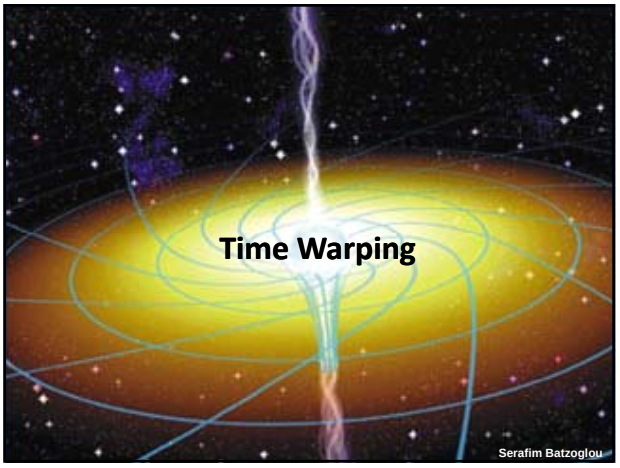
	0	5	3	9	7	3	4	3	8
0	0	5	8	17	24	27	31	34	42
4	4	1	1.5	4	5.5	6	6	6.5	8.5
7	11	2	3.5	3.5	3.5	5.5	7	8.5	7.5
4	15	2.5	3	5.5	5	4.5	4.5	5	7
7	22	3.5	5	5	5	6.5	6	7	6
8	30	5	7.5	5.5	5.5	8	8	9.5	6
9	39	7	10	5.5	6.5	9.5	10.5	12.5	6.5

- Aach J, Church GM. Aligning gene expression time series with time warping algorithms. Bioinformatics. 2001 Jun;17(6):495-508. PubMed:11395426
- **Time Warps, String Edits, and Macromolecules**: The Theory and Practice of Sequence Comparison (The David Hume Series) David Sankoff (Editor), Joseph Kruskal (Editor). Paperback - 407 pages (28 January, 2000) Cambridge University Press; ISBN: 1575862174Reviews: Book Description
- Time Warps, String Edits and Macromolecules is a young classic in computational science, scientific analysis from a computational perspective. The computational perspective is that of sequence processing, in particular the problem of recognizing related sequences. The book is the first, and still best compilation of papers explaining how to measure distance between sequences, and how to compute that measure effectively. This is called string distance, Levenshtein distance, or edit distance. The book contains lucid explanations of the basic techniques; well-annotated examples of applications; mathematical analysis of its computational (algorithmic) complexity; and extensive discussion of the variants needed for weighted measures, timed sequences (songs), applications to continuous data, comparison of multiple sequences and extensions to tree-structures. In molecular biology the sequences compared are the macromolecules DNA and RNA. Sequence distance allows the recognition of homologies (correspondences) between related molecules. One may interpret the distance between molecular sequences in terms of the mutations necessary for one molecule to evolve into another. A further application explores methods of predicting the secondary structure (chemical bonding) of RNA sequences. In speech recognition speech input must be compared to stored patterns to find the most likely interpretation (e.g., syllable). Because speech varies in tempo, part of the comparison allows for temporal variation, and is known as "time-warping". In dialectology Levenshtein distance allows analysis of the learned variation in pronunciation, its cultural component. Levenshtein distance introduces a metric which allows more sophisticated analysis than traditional dialectology's focus on classes of alternative pronunciations. A similar application is the study of bird song, where degrees of distance in song are seen to correspond to the divergence of bird populations. A final application area is software, where Levenshtein distance is employed to located differing parts of different versions of computer files, and to perform error correction.

Following is borrowed from

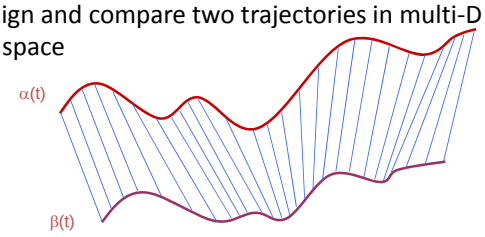
- **Serafim Batzoglou,**
  Associate Professor
- http://ai.stanford.edu/~serafim/
- *http://ai.stanford.edu/~serafim/cs262/Spring2003/Slides/Lecture4.ppt*

Serafim Batzoglou



Time Warping

Align and compare two trajectories in multi-D space

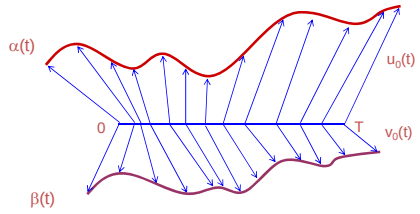


- Additive random error
- Variations in speed from one segment to another

Time Warping

Definition: $\alpha(u)$, $\beta(u)$ are connected by an approximate continuous time warping (u_0, v_0), if:

u_0, v_0 are strictly increasing functions on [0, T], and
$\alpha(u_0(t)) \cong \beta(v_0(t))$ for $0 \le t \le T$



Time Warping

How do we measure how "good" a time warping is?

Let's try:

$$\int_0^T w(\alpha(u_0(t)), \beta(v_0(t))) \, dt$$

However, an equivalent time warping ($u_1(s)$, $v_1(s)$), is given by:

$s = f(t)$; $\quad f: [0, T] \to [0, S]$

has score

$$\int_0^S w(\alpha(u_1(s)), \beta(v_1(s))) \, ds = \int_0^T w(\alpha(u_0(t)), \beta(v_0(t))) \, f'(t) \, dt$$

This is arbitrarily different

Time Warping

This one works:

$$d(u_0, v_0) = \int_0^T w(\alpha(u_0(t)), \beta(v_0(t))) \, [(u'_0(t) + v'_0(t))/2] \, dt$$

Now, if $s = f(t)$; $t = g(s)$, and $g = f^{-1}$,

$$\int_0^S w(\alpha(u_1(s)), \beta(v_1(s))) \, (u'_1(s) + v'_1(s))/2 \, ds =$$

$f(t) = f(g(s)) = s$;
$f'(t) = f'(g(s)) \, g'(s) = 1$, therefore $g'(s) = 1/f'(t)$
$u_0(t) = u_0(g(s))$, therefore $u'_0(t) = u'_0(g(s)) \, g'(s)$

$$\int_0^T w(\alpha(u_0(t)), \beta(v_0(t))) \, (u'_0(t)+v'_0(t))/2 \; g'(s) \, f'(t) \, dt =$$
$$\int_0^T w(\alpha(u_0(t)), \beta(v_0(t))) \, [(u'_0(t) + v'_0(t))/2] \, dt$$

Time Warping

From continuous to discrete:

Let's discretize the signals:

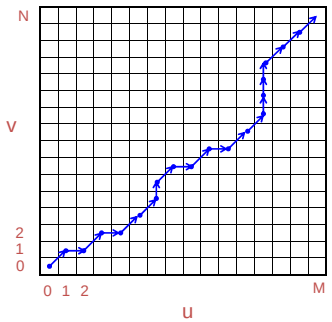
$\alpha(t)$: $a = a_0 \ldots\ldots a_M$
$\beta(t)$: $b = b_0 \ldots\ldots b_N$

Definition:
a, **b** are connected by an approximate discrete time warping (u, v), if u and v are weakly increasing integer functions on $1 \le h \le H$, such that

$$a_{u[h]} \cong b_{v[h]} \text{ for all } h = 1 \ldots\ldots H$$

Moreover, we require $\quad u[0] = v[0] = 0$;
$u[H] = M$;
$v[h] = N$

Time Warping



Define possible steps:

$(\Delta u, \Delta v)$ is the possible difference of u and v between steps h-1 and h

$$(\Delta u, \Delta v) = \begin{cases} (1, 0) \\ (1, 1) \\ (0, 1) \end{cases}$$

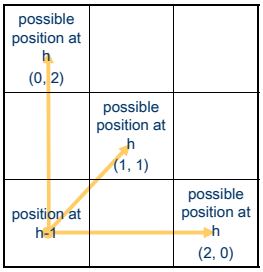
Time Warping

Alternatively:

$(\Delta u, \Delta v) = \begin{cases} (2, 0) \\ (1, 1) \\ (0, 2) \end{cases}$

Advantage:

Every time warp has the same number of steps



Time Warping

Discrete objective function:

For $0 \le i = u[h] \le M$; $0 \le j = v[h] \le N$,
Define $w(i, j) = w(a_{u[h]}, b_{v[h]})$

Then,

$D(u, v) = \Sigma_h w(u[h], v[h]) (\Delta u + \Delta v)/2$

In the case where we allow (2, 0), (1, 1), and (0, 2) steps,

$D(u, v) = \Sigma_h w(u[h], v[h])$

Time Warping

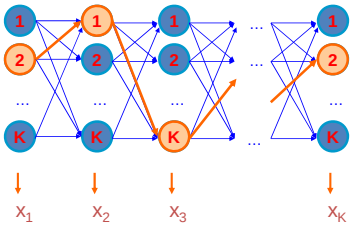
Algorithm for optimal discrete time warping:

Initialization:
$D(i, 0) = \infty \Sigma_{i'<i} w(i, 0)$
$D(0, j) = \infty \Sigma_{j'<j} w(0, j)$
$D(1, j) = D(i, 1) = w(i, j) + w(i-1, j-1)$

Iteration:
For i = 2……M
  For j = 2……N

$D(i, j) = \min \begin{cases} D(i-2, j) + w(i, j) \\ D(i-1, j-1) + w(i, j) \\ D(i-2, j) + w(i, j) \end{cases}$

Hidden Markov Models



Outline for our next topic

- Hidden Markov models – the theory
- Probabilistic interpretation of alignments using HMMs

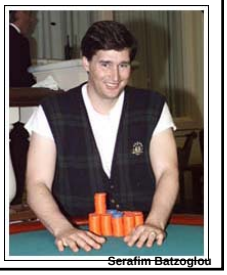
Later in the course:
- Applications of HMMs to biological sequence modeling and discovery of features such as genes

Example: The Dishonest Casino

A casino has two dice:
- Fair die
  P(1) = P(2) = P(3) = P(5) = P(6) = 1/6
- Loaded die
  P(1) = P(2) = P(3) = P(5) = 1/10
  P(6) = 1/2

Casino player switches back-&-forth between fair and loaded die once every 20 turns

Game:
1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2



Question # 1 – Evaluation

GIVEN

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344

QUESTION

How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem in HMMs

Serafim Batzoglou

Question # 2 – Decoding

GIVEN

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344

QUESTION

What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the **DECODING** question in HMMs

Serafim Batzoglou

Question # 3 – Learning

GIVEN

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344

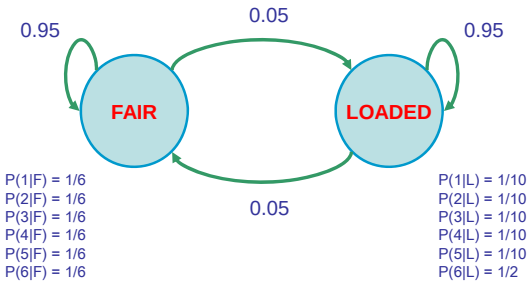
QUESTION

How "loaded" is the loaded die? How "fair" is the fair die? How often does the casino player change from fair to loaded, and back?

This is the **LEARNING** question in HMMs

Serafim Batzoglou

The dishonest casino model



Serafim Batzoglou

Definition of a hidden Markov model

Definition: A hidden Markov model (HMM)

- Alphabet $\Sigma = \{ b_1, b_2, \ldots, b_M \}$
- Set of states $Q = \{ 1, \ldots, K \}$
- Transition probabilities between any two states

  a_{ij} = transition prob from state i to state j

  $a_{i1} + \ldots + a_{iK} = 1$, for all states i = 1…K

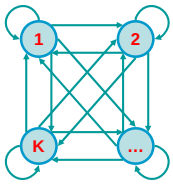
- Start probabilities a_{0i}

  $a_{01} + \ldots + a_{0K} = 1$

- Emission probabilities within each state

  $e_i(b) = P(x_i = b \mid \pi_i = k)$

  $e_i(b_1) + \ldots + e_i(b_M) = 1$, for all states i = 1…K

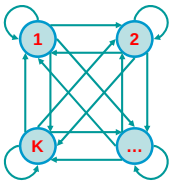


Serafim Batzoglou

A Hidden Markov Model is memory-less

At each time step t,
the only thing that affects future states
is the current state π_t

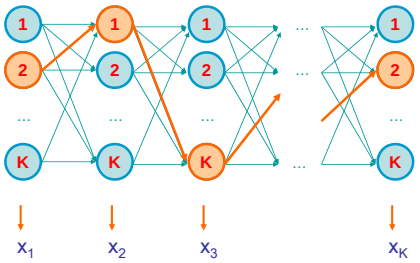
$P(\pi_{t+1} = k \mid$ "whatever happened so far") =
$P(\pi_{t+1} = k \mid \pi_1, \pi_2, \ldots, \pi_t, x_1, x_2, \ldots, x_t)$ =
$P(\pi_{t+1} = k \mid \pi_t)$



Serafim Batzoglou

A parse of a sequence

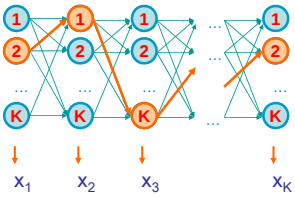
Given a sequence $x = x_1 \ldots\ldots x_N$,
A parse of x is a sequence of states $\pi = \pi_1, \ldots\ldots, \pi_N$



Serafim Batzoglou

Likelihood of a parse

Given a sequence $x = x_1 \ldots\ldots x_N$
and a parse $\pi = \pi_1, \ldots\ldots, \pi_N$,



To find how likely is the parse:
(given our HMM)

$P(x, \pi) = P(x_1, \ldots, x_N, \pi_1, \ldots\ldots, \pi_N) =$
$P(x_N, \pi_N \mid \pi_{N-1})\ P(x_{N-1}, \pi_{N-1} \mid \pi_{N-2}) \ldots\ldots P(x_2, \pi_2 \mid \pi_1)\ P(x_1, \pi_1) =$
$P(x_N \mid \pi_N)\ P(\pi_N \mid \pi_{N-1})\ \ldots\ldots P(x_2 \mid \pi_2)\ P(\pi_2 \mid \pi_1)\ P(x_1 \mid \pi_1)\ P(\pi_1) =$
$a_{0\pi1}\ a_{\pi1\pi2} \ldots\ldots a_{\pi N-1\pi N}\ e_{\pi1}(x_1) \ldots\ldots e_{\pi N}(x_N)$

Serafim Batzoglou

Example: the dishonest casino

Let the sequence of rolls be:

x = 1, 2, 1, 5, 6, 2, 1, 6, 2, 4

Then, what is the likelihood of

π = Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair, Fair?

(say initial probs $a_{0Fair} = ½$, $a_{oLoaded} = ½$)

$½ \times P(1 \mid Fair)\ P(Fair \mid Fair)\ P(2 \mid Fair)\ P(Fair \mid Fair) \ldots P(4 \mid Fair) =$

$½ \times (1/6)^{10} \times (0.95)^9 = .00000000521158647211 = 0.5 \times 10^{-9}$

Serafim Batzoglou

Example: the dishonest casino

So, the likelihood the die is fair in all this run is just 0.521×10^{-9}

OK, but what is the likelihood of

π = Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded, Loaded?

$½ \times P(1 \mid Loaded)\ P(Loaded, Loaded) \ldots P(4 \mid Loaded) =$

$½ \times (1/10)^8 \times (1/2)^2\ (0.95)^9 = .00000000078781176215 = 7.9 \times 10^{-10}$

Therefore, it is after all 6.59 times more likely that the die is fair all the way, than that it is loaded all the way.

Serafim Batzoglou

Example: the dishonest casino

Let the sequence of rolls be:

x = 1, 6, 6, 5, 6, 2, 6, 6, 3, 6

Now, what is the likelihood π = F, F, …, F?

$½ \times (1/6)^{10} \times (0.95)^9 = 0.5 \times 10^{-9}$, same as before

What is the likelihood

π = L, L, …, L?

$½ \times (1/10)^4 \times (1/2)^6\ (0.95)^9 = .00000049238235134735 = 0.5 \times 10^{-7}$

So, it is 100 times more likely the die is loaded

Serafim Batzoglou

The three main questions on HMMs

1. Evaluation

   GIVEN a HMM M, and a sequence x,
   FIND Prob[x | M]

2. Decoding

   GIVEN a HMM M, and a sequence x,
   FIND the sequence π of states that maximizes P[x, π | M]

3. Learning

   GIVEN a HMM M, with unspecified transition/emission probs., and a sequence x,

   FIND parameters $\theta = (e_i(.), a_{ij})$ that maximize P[x | θ]

Serafim Batzoglou

Let's not be confused by notation

P[x | M]: The probability that sequence x was generated by the model

The model is: architecture (#states, etc)
+ parameters $\theta = a_{ij}, e_i(.)$

So, P[x | θ], and P[x] are the same, when the architecture, and the entire model, respectively, are implied

Similarly, P[x, π | M] and P[x, π] are the same

In the LEARNING problem we always write P[x | θ] to emphasize that we are seeking the θ that maximizes P[x | θ]

Serafim Batzoglou