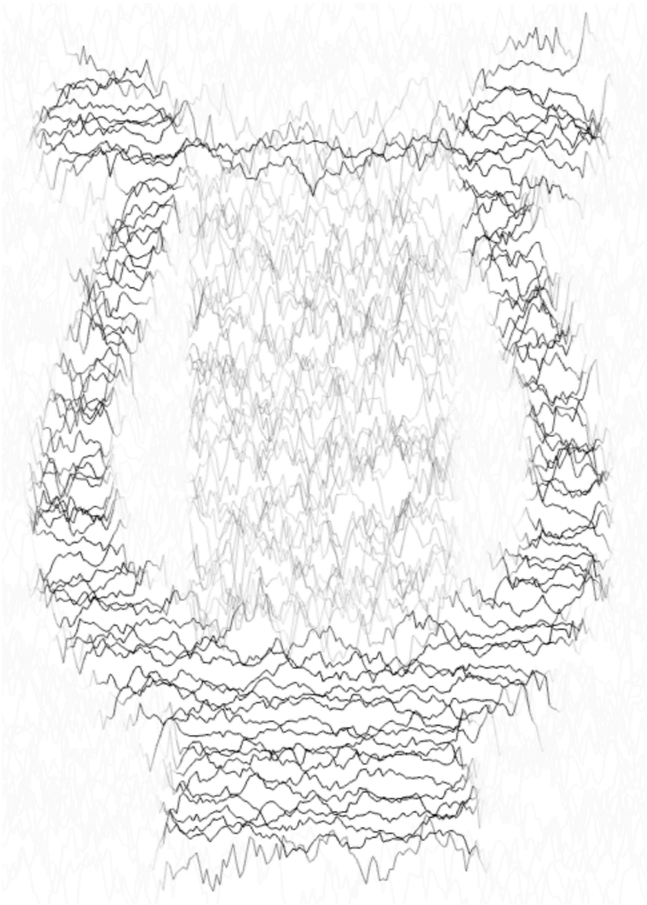
University of Illinois @ Urbana-Champaign



CS 498PS – Audio Computing Lab

Temporal Modeling and Basic Speech Recognition

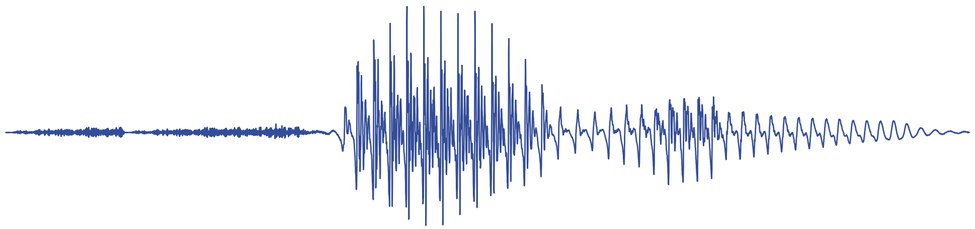
Paris Smaragdis
paris@illinois.edu
paris.cs.illinois.edu

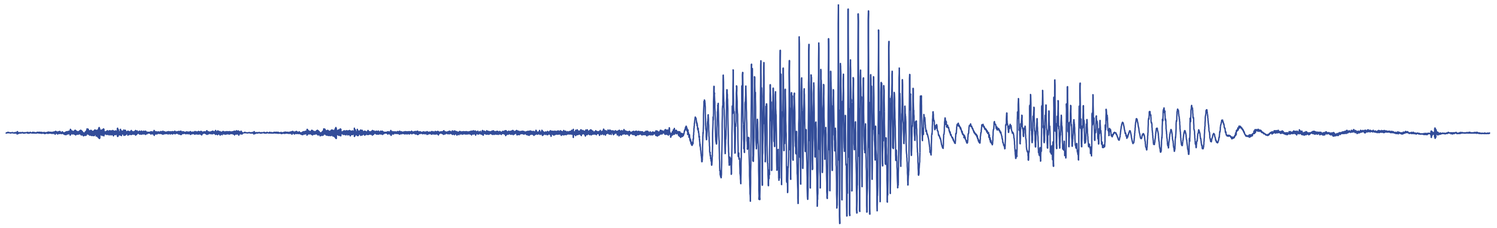
Today's lecture

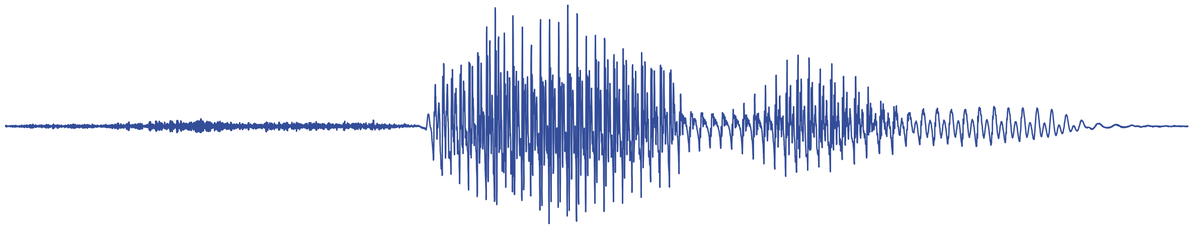
- Recognizing time series
  - Basic speech recognition
- Dynamic Time Warping
- Hidden Markov Models

A new problem?

- Are these the same utterances?







How do we recognize the spoken digit?

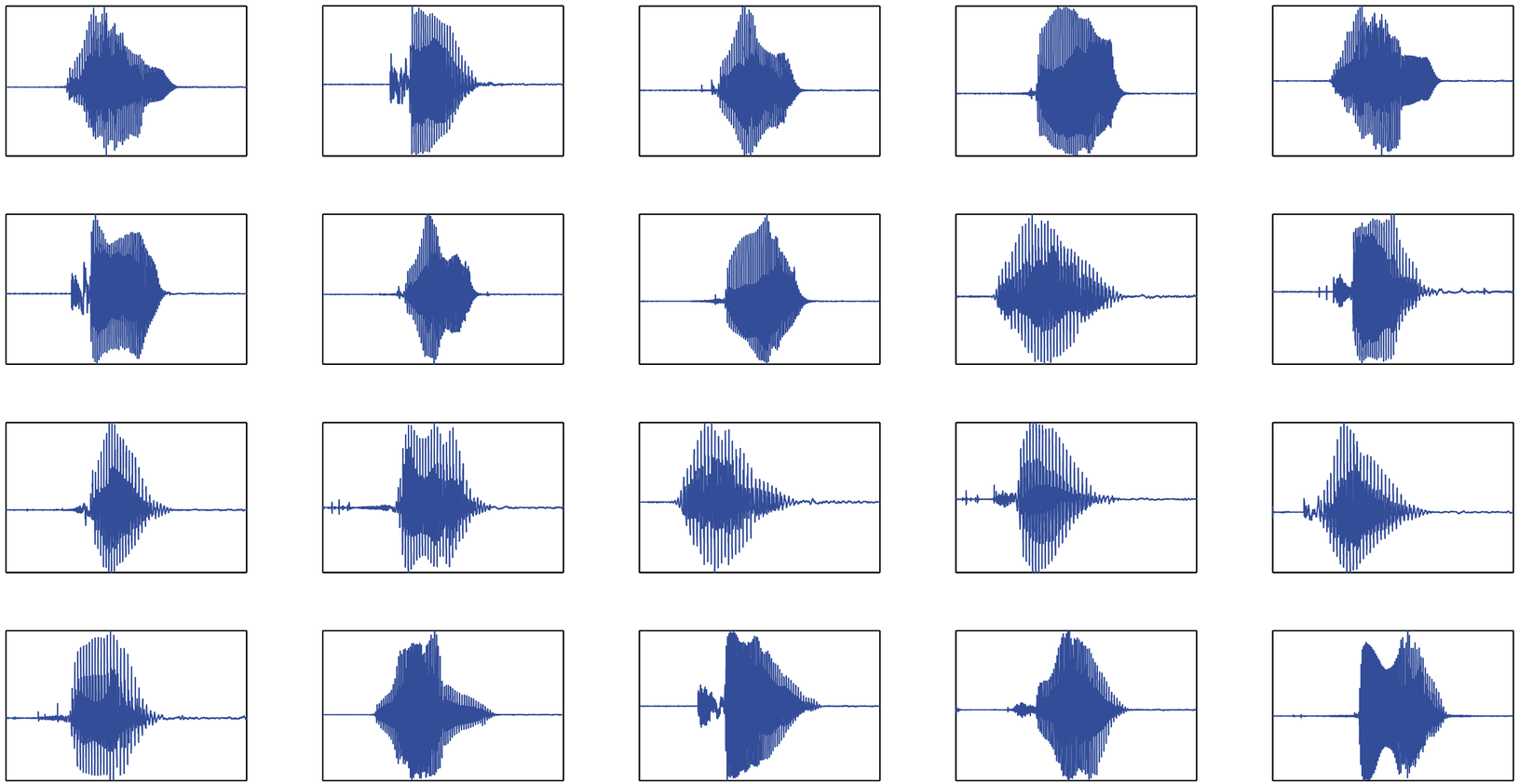
- How do we take time order in account?
  - We could use a Gaussian model as before, but it can't tell the difference between "ten", "net", or "ent"
- How do we deal with timing differences?
  - "three" and "thhhrreeee" should be seen as being the same

A motivating problem to solve

- How to recognize words
  - E.g., yes/no, or spoken digits
- Build reliable features
  - Must be invariant to irrelevant differences
- Build a classifier that can deal with time
  - Invariant to temporal difference in inputs

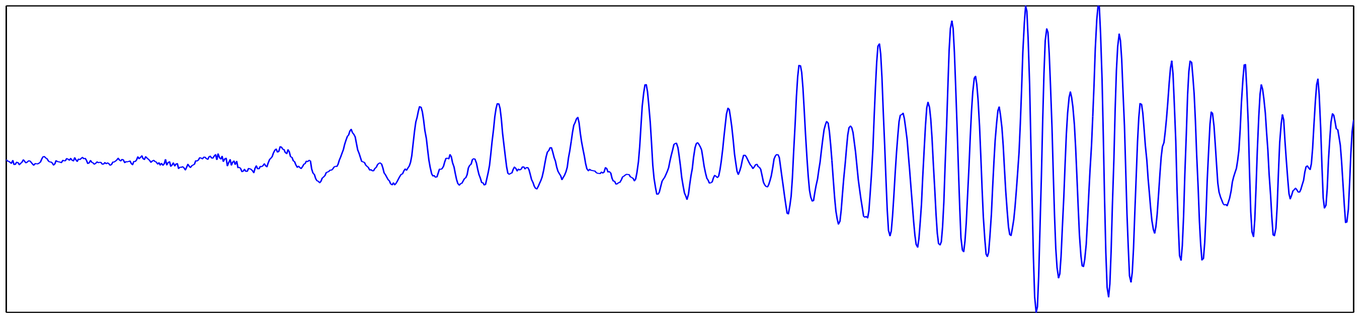
Example data set

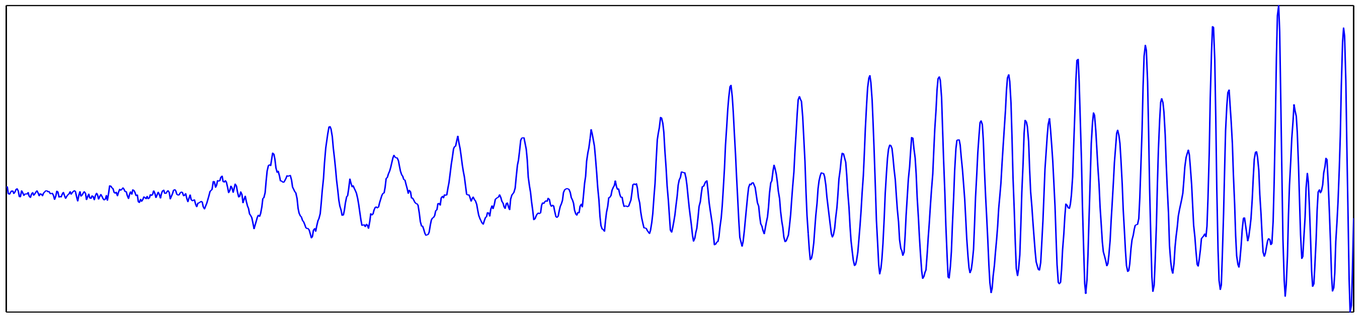
- Spoken digits
  - Which is which? Can we automatically identify them?



Finding a good representation

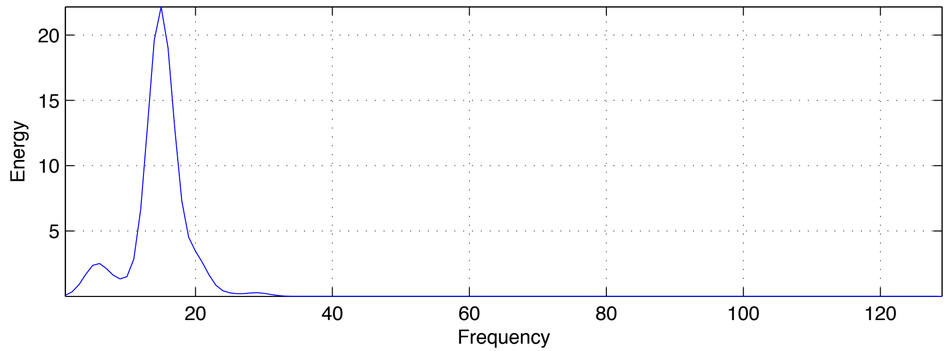
- Small differences are not important
  - E.g., the MSE of these two isn't useful
    - Even though they are the same digit

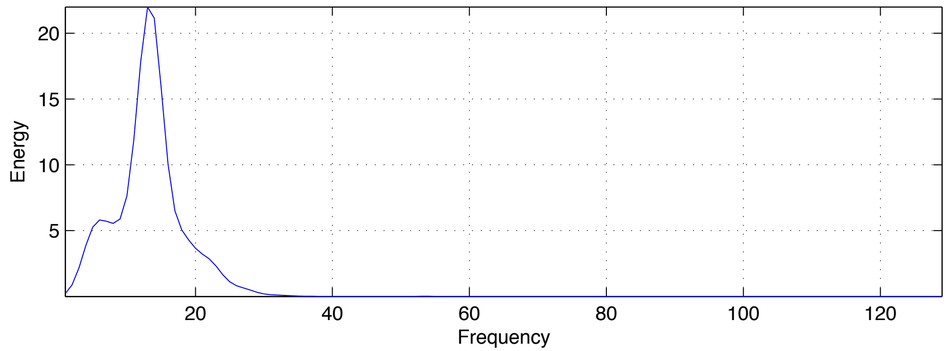




Going to the frequency domain

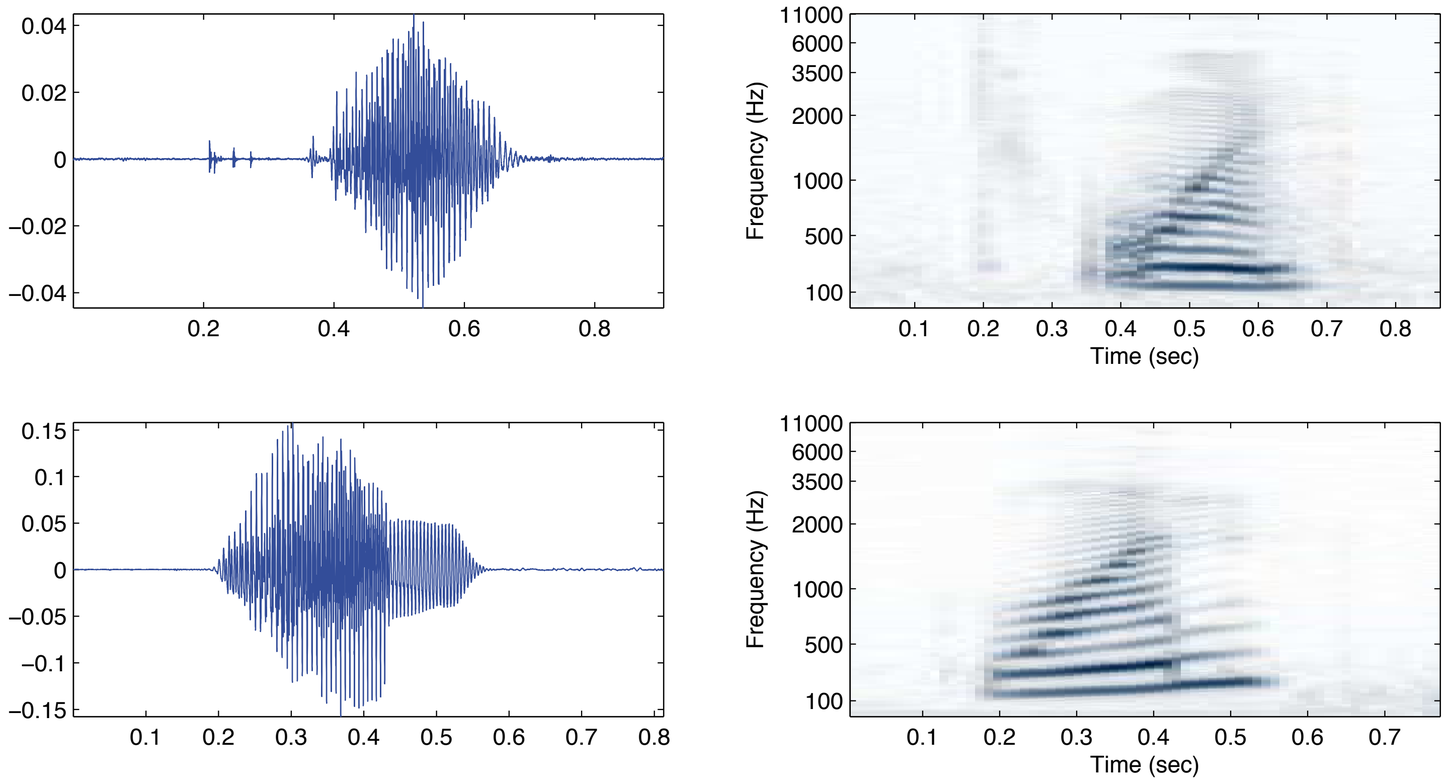
- The magnitude Fourier transform is better
  - The irrelevant waveform differences are hidden
  - But so is the temporal structure





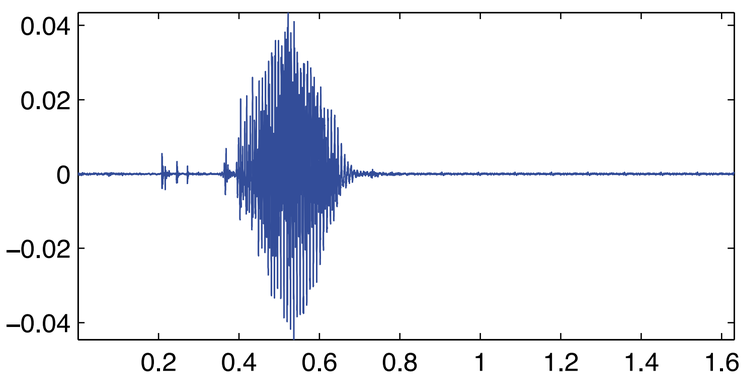
Time/Frequency features

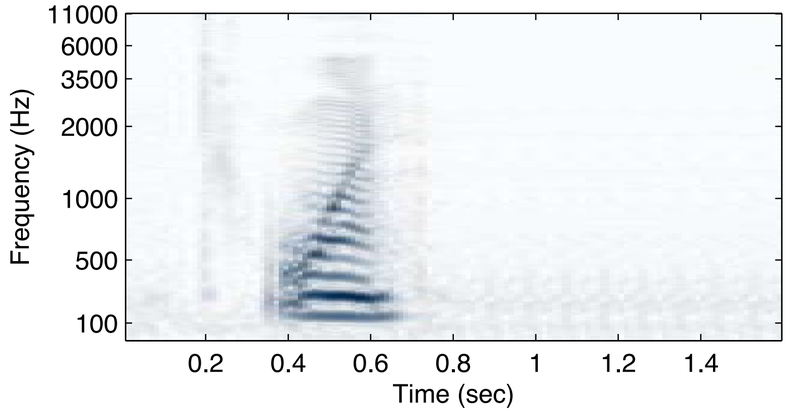
- A more robust representation
  - Recordings look more “similar"

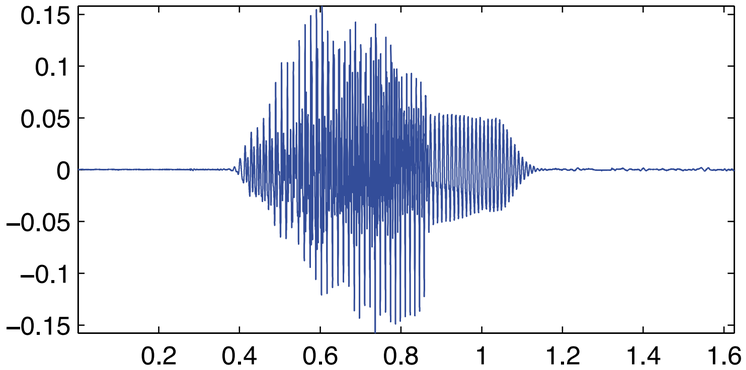


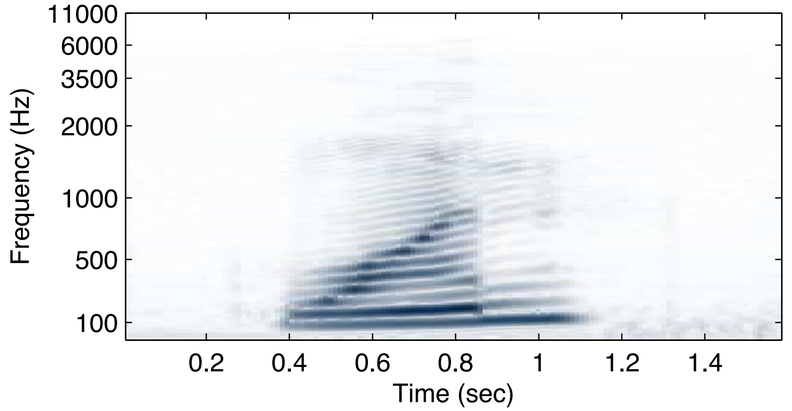
A new problem

- What about relative time differences?
  - There is no 1-to-1 correspondence in the spectral frames



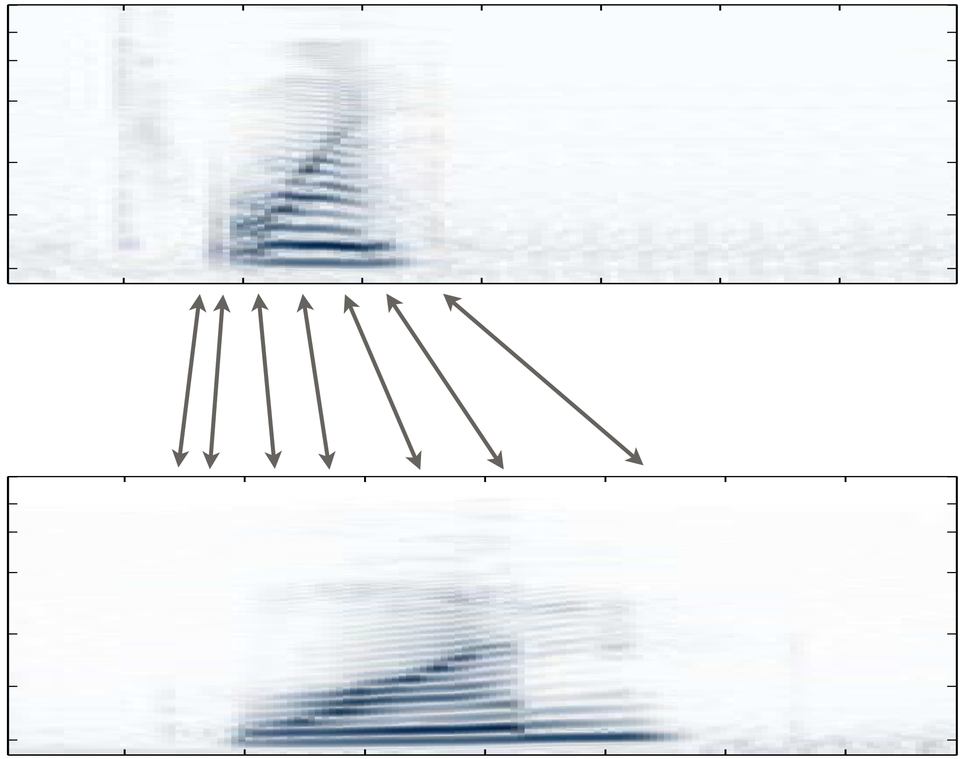






Time warping

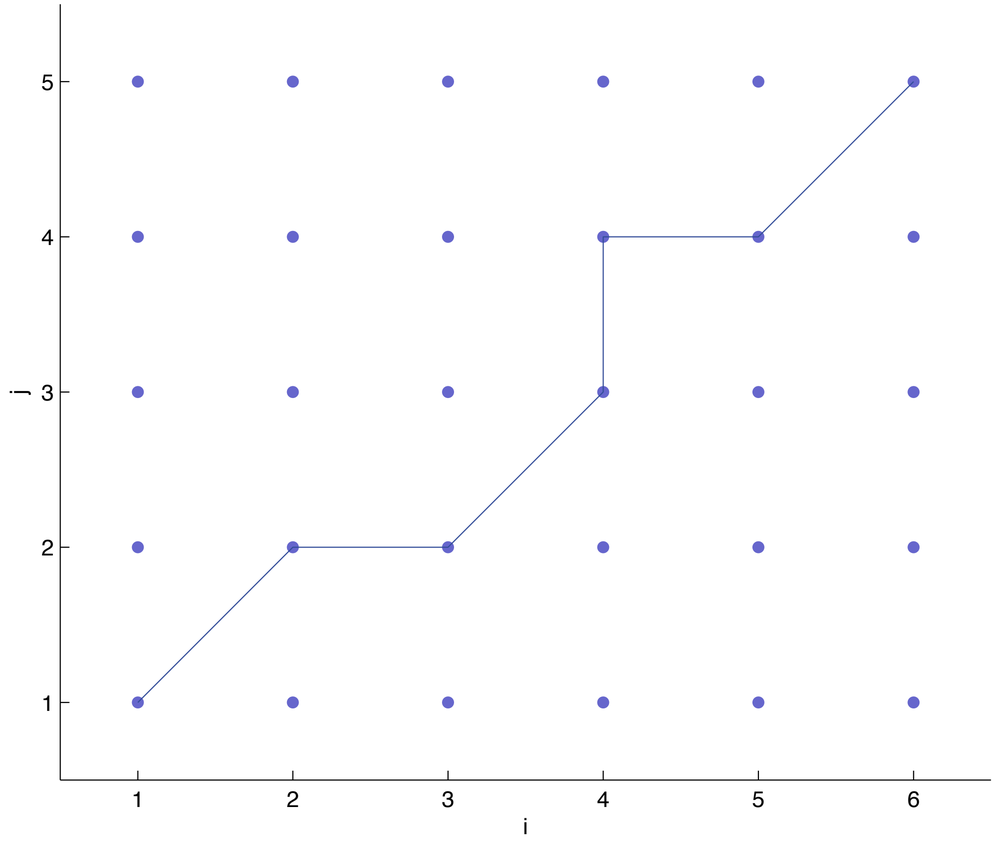
- To compare them we need to time-warp and align
  - How do we find that warp? How do we get overall similarity?



Matching warped sequences

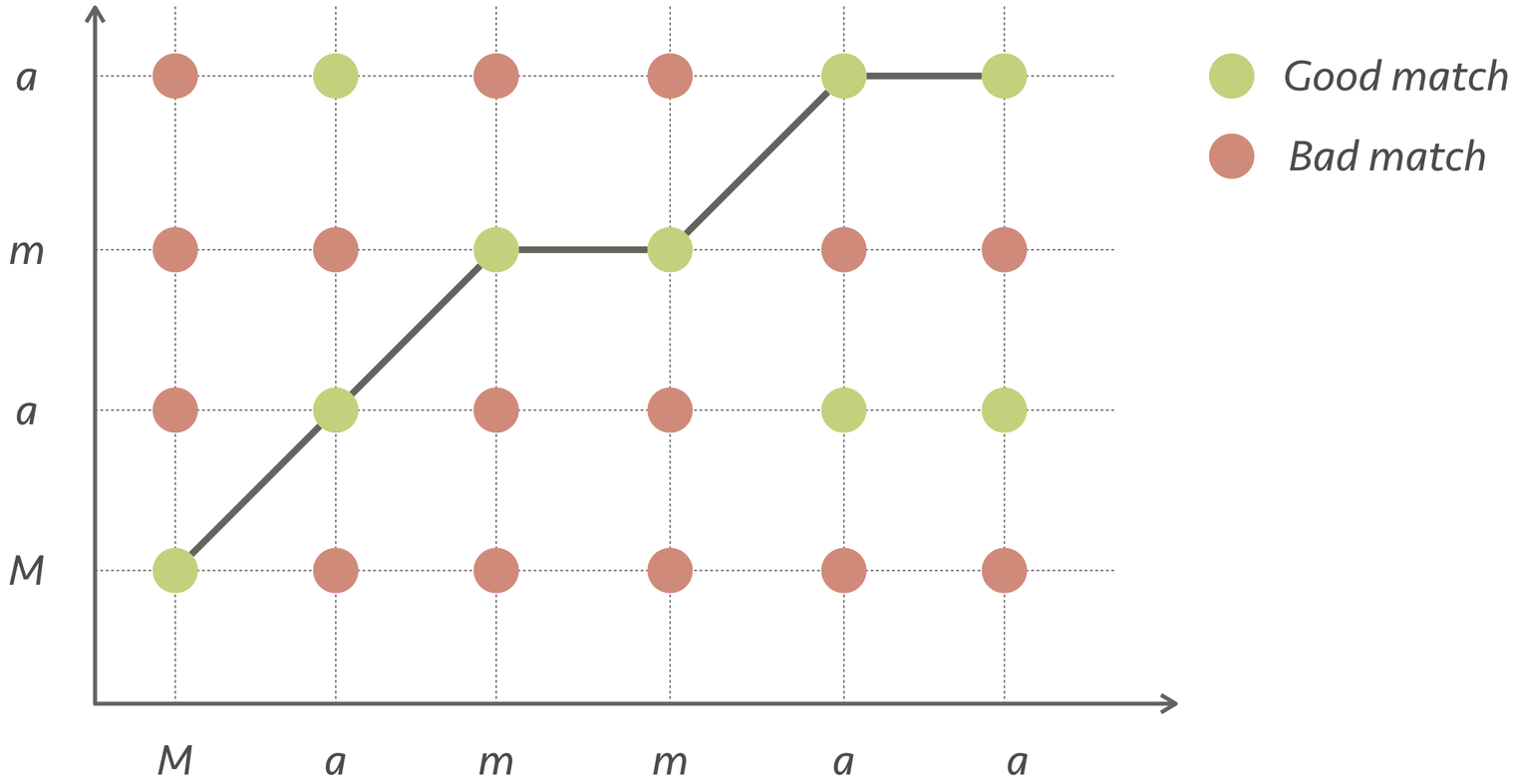
- Represent the warping with a path

$$\mathbf{r}(i), i = 1,2,\ldots,6 \quad \mathbf{t}(j), j = 1,2,\ldots,5$$



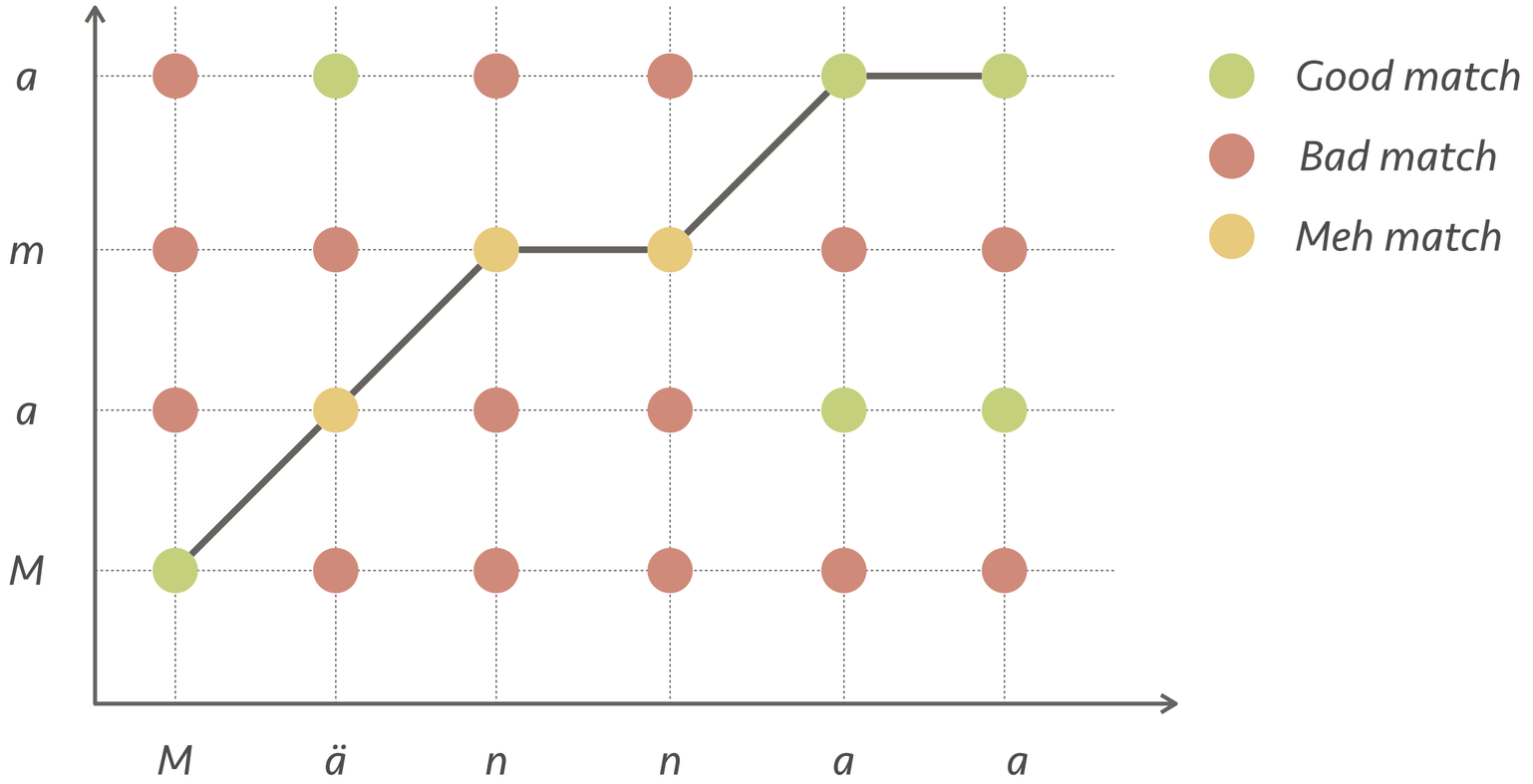
An example with text

- Comparing “Mama” with “Mammaa”



Another example

- Comparing “Mama” with “Männaa”
  - There is still a path, although not as good as before

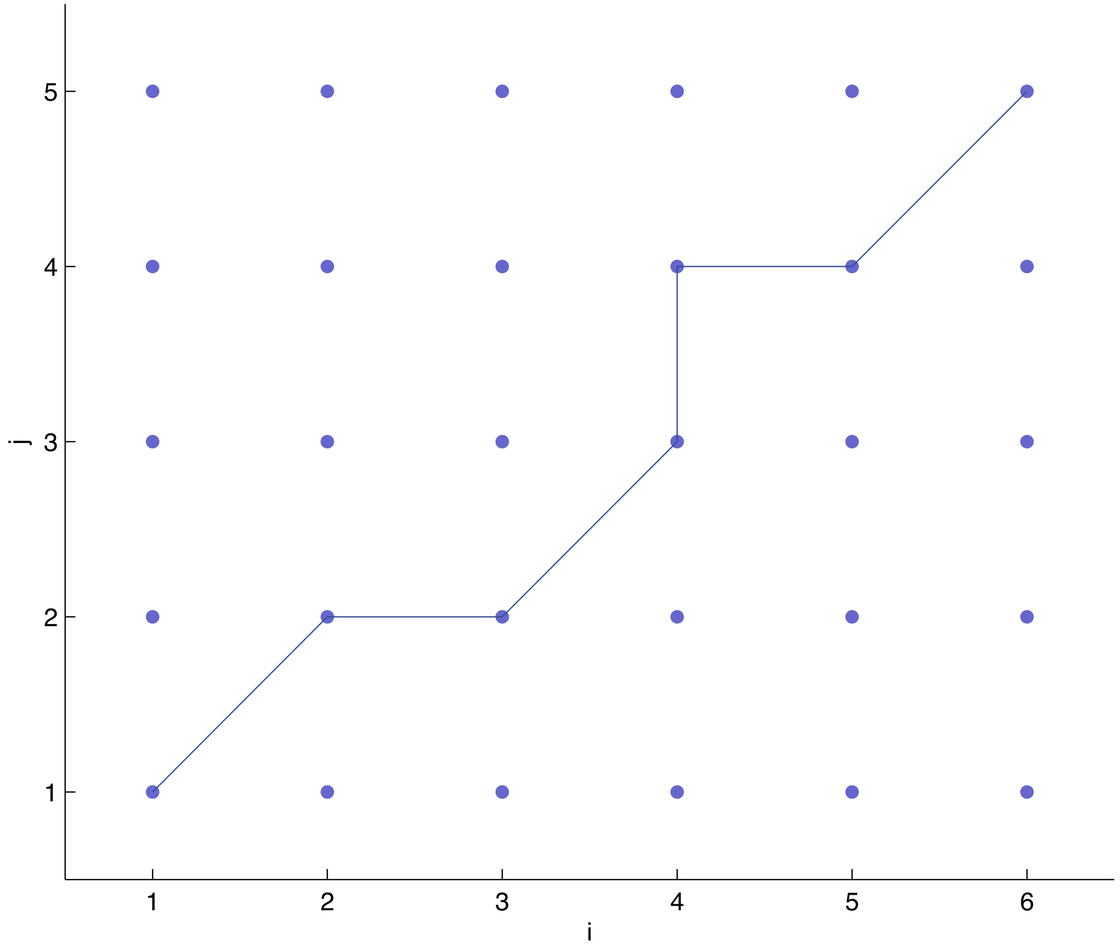


Finding the overall "distance"

- Visiting a node will have a cost:
  - e.g., $d(i,j) = \left\| \mathbf{r}(i) - \mathbf{t}(j) \right\|$
- Overall cost of a path is:

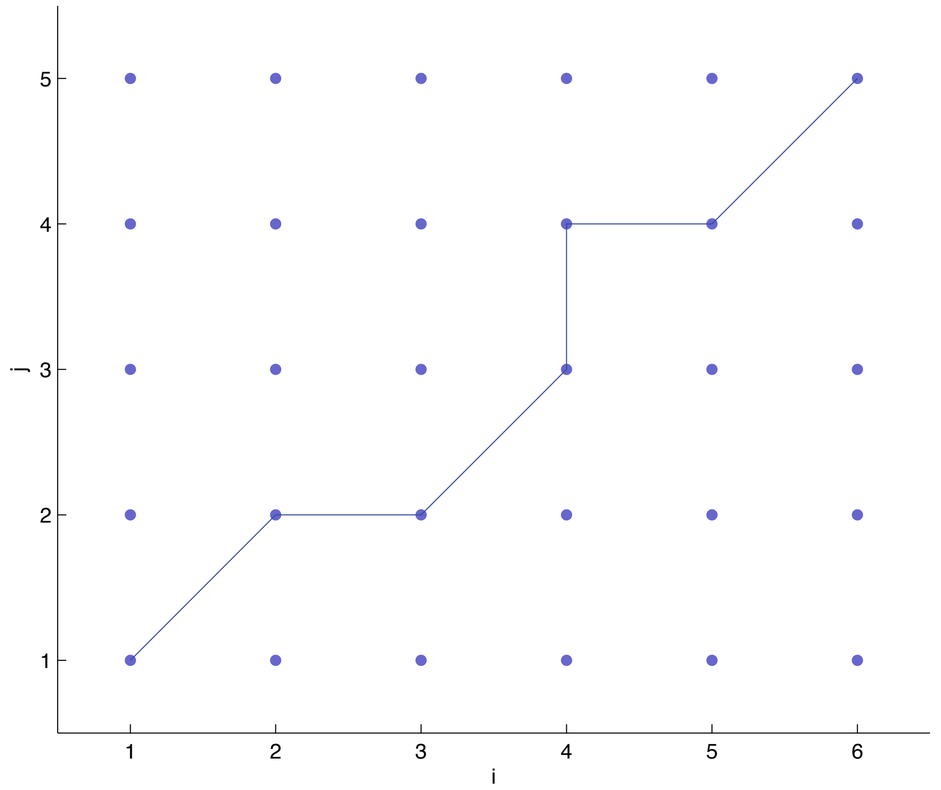
$$D = \sum_k d(i_k, j_k)$$

- The *optimal path*
  - Results in smallest final distance
  - Can be considered as the distance between the two input sequences



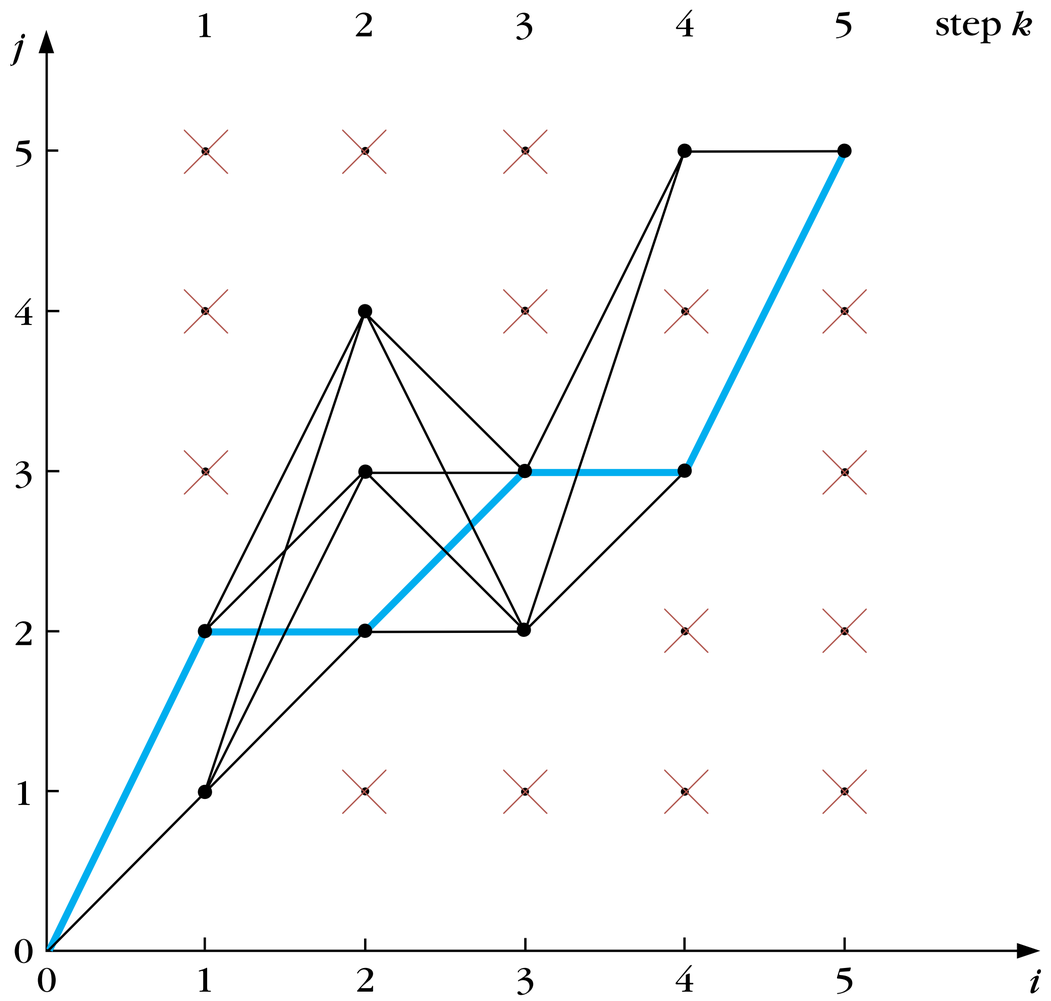
Finding the optimal path

- Huge space of possible paths
  - How do the find the one that gives us the minimal distance?
- We need constraints
  - Limit the search options
- We need some theory!
  - Maybe we can skip a full search



Constraining the search space

- Global constraints
  - Nodes we should not visit
    - ×'d nodes
- Local constraints
  - Allowable transitions
    - Black lines
- Optimal path
  - Blue line



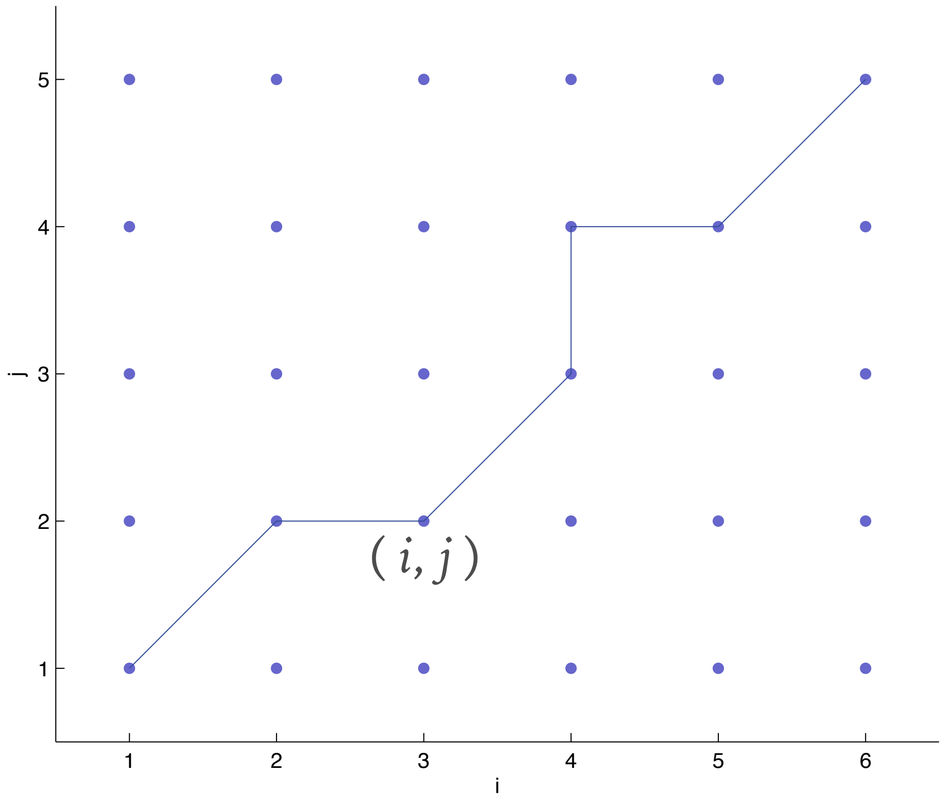
Some theory help

- *Bellman's optimality principle*
  - For an optimal path passing through (i, j):

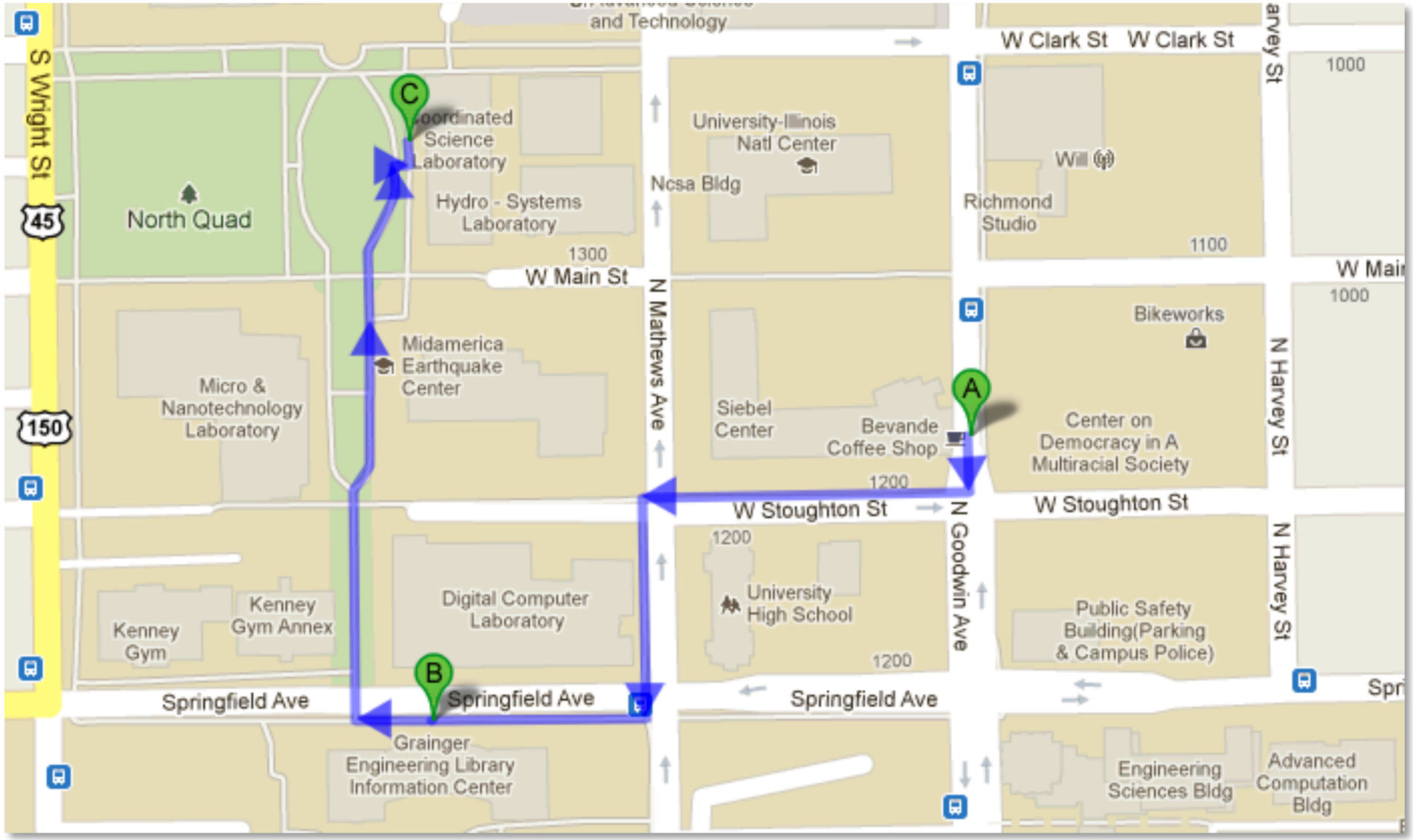
$$(i, j) : (i_0, j_0) \xrightarrow{opt} (i_T, j_T)$$

  - Then:

$$(i_0, j_0) \xrightarrow{opt} (i_T, j_T) = \left\{ (i_0, j_0) \xrightarrow{opt} (i, j), (i, j) \xrightarrow{opt} (i_T, j_T) \right\}$$



Huh?

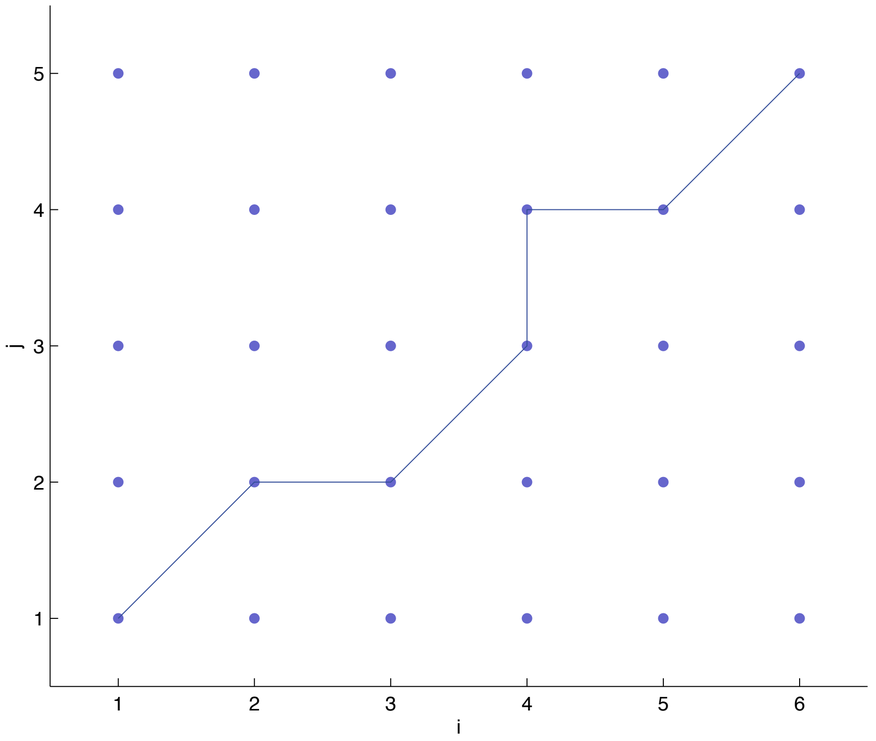

and Technology
W Clark St
W Clark St
S Wright St
45
150
C
Coordinated Science Laboratory
University-Illinois Natl Center
Ncsa Bldg
North Quad
Hydro - Systems Laboratory
1300
W Main St
N Mathews Ave
Midamerica Earthquake Center
Micro & Nanotechnology Laboratory
Siebel Center
Bevande Coffee Shop
A
Richmond Studio
1100
Bikeworks
1000
N Harvey St
Center on Democracy in A Multiracial Society
1200
W Stoughton St
N Goodwin Ave
Kenney Gym
Kenney Gym Annex
Digital Computer Laboratory
University High School
Public Safety Building(Parking & Campus Police)
B
Springfield Ave
Springfield Ave
Springfield Ave
Grainger Engineering Library Information Center
Engineering Sciences Bldg
Advanced Computation Bldg

Finding an optimal path

- Optimal path to (i_k, j_k):

$$D_{\min}(i_k, j_k) = \min_{i_{k-1}, j_{k-1}} D_{\min}(i_{k-1}, j_{k-1}) + d(i_k, j_k \mid i_{k-1}, j_{k-1})$$

- Smaller search
  - min() only searches local transitions
- Vastly faster than otherwise



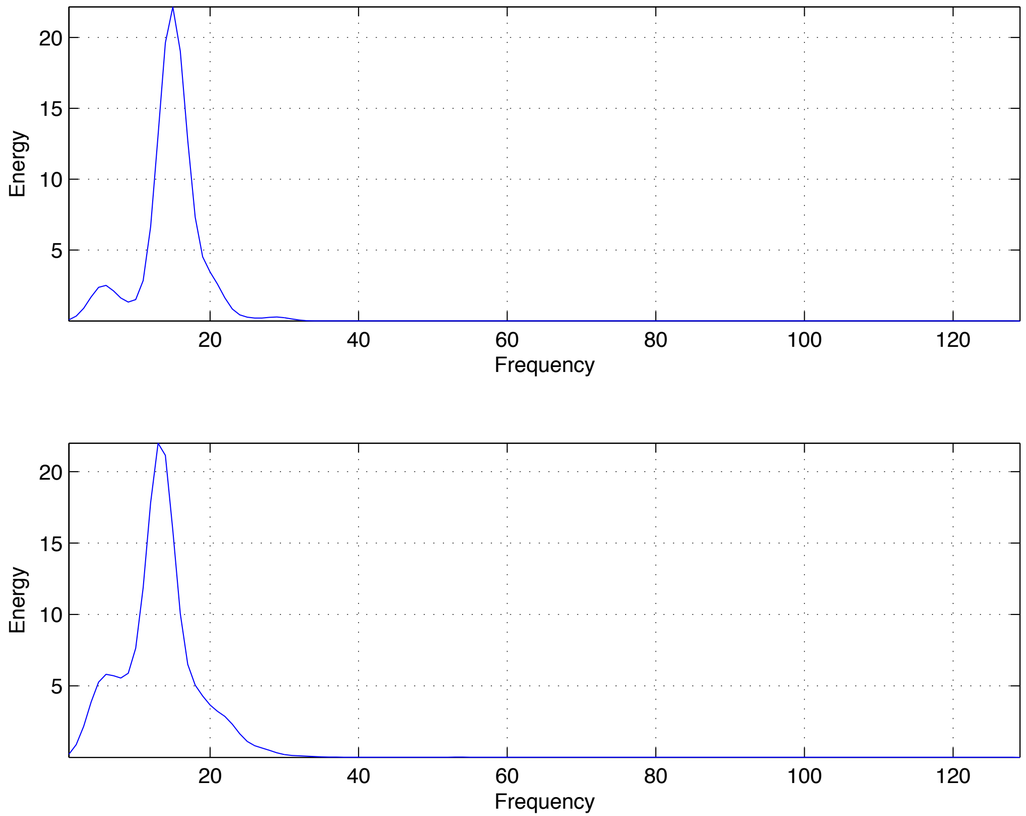
Making this work for speech

- Define a distance function
- Define local constraints
- Define global constraints

Distance function

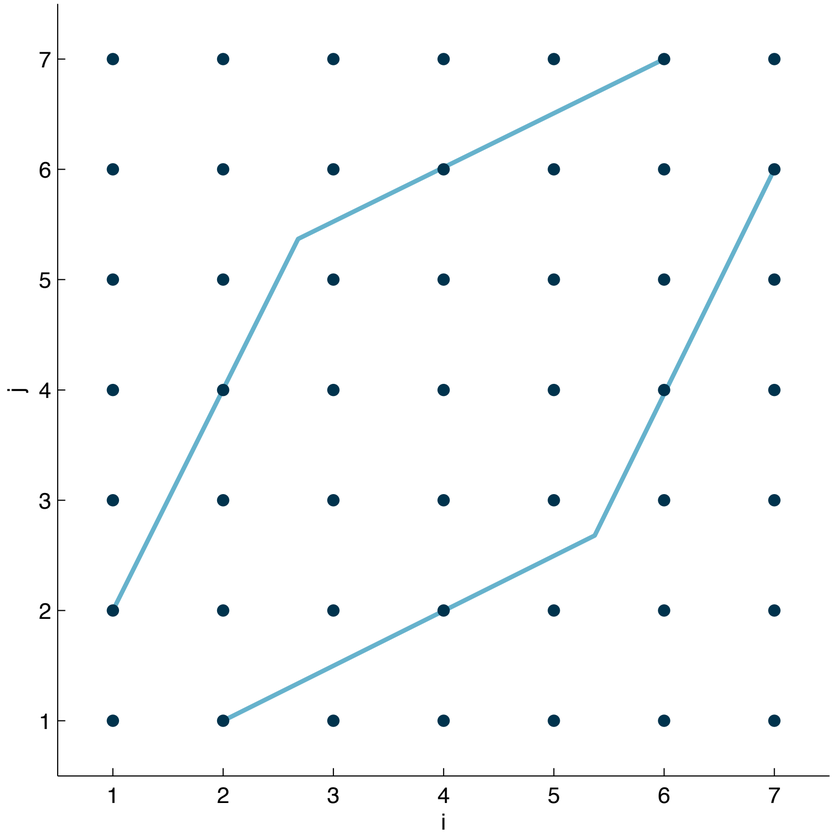
- We can simply use MSE on spectral frames:

$$d(i,j) = \left\| \mathbf{f}_1(i) - \mathbf{f}_2(j) \right\|$$



Global constraints

- Define time ratios that make sense
  - e.g., one sequence cannot be 2x faster than the other



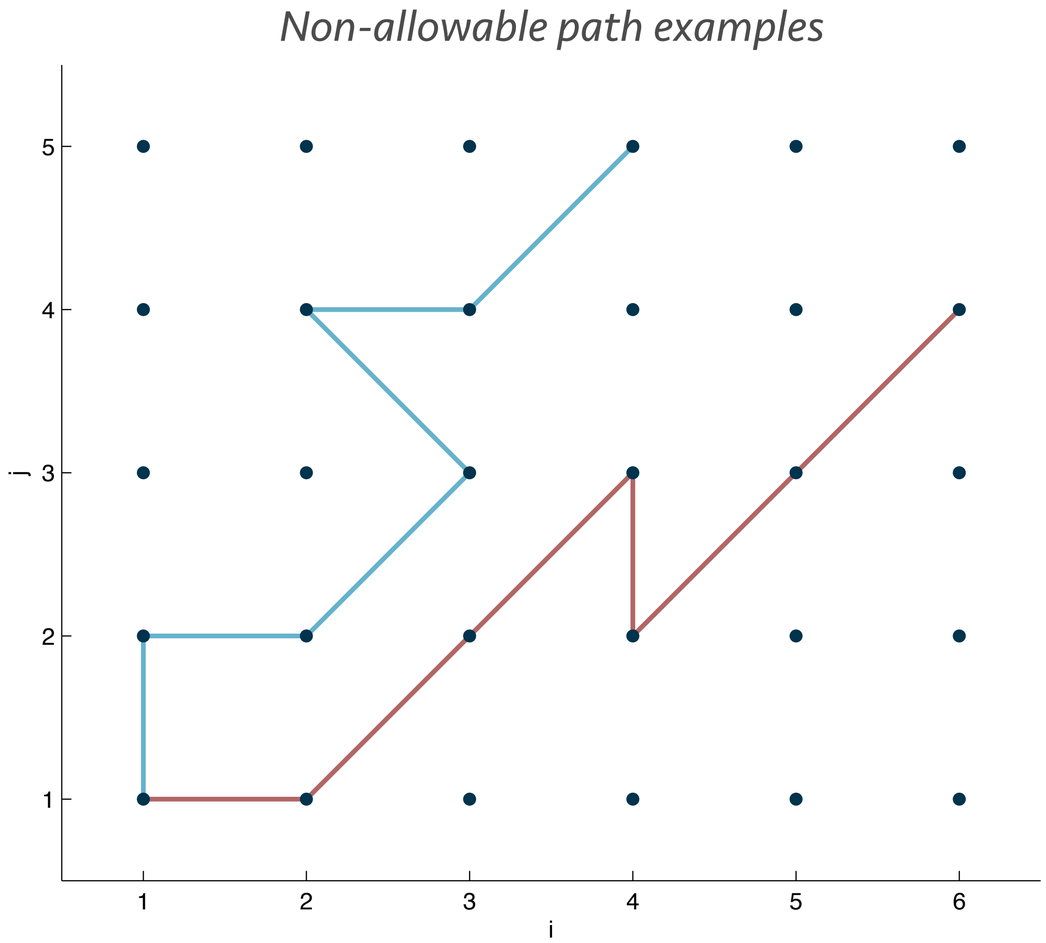
Local constraints

- Monotonicity:

$$i_{k-1} \leq i_k \,, \;\; j_{k-1} \leq j_k$$

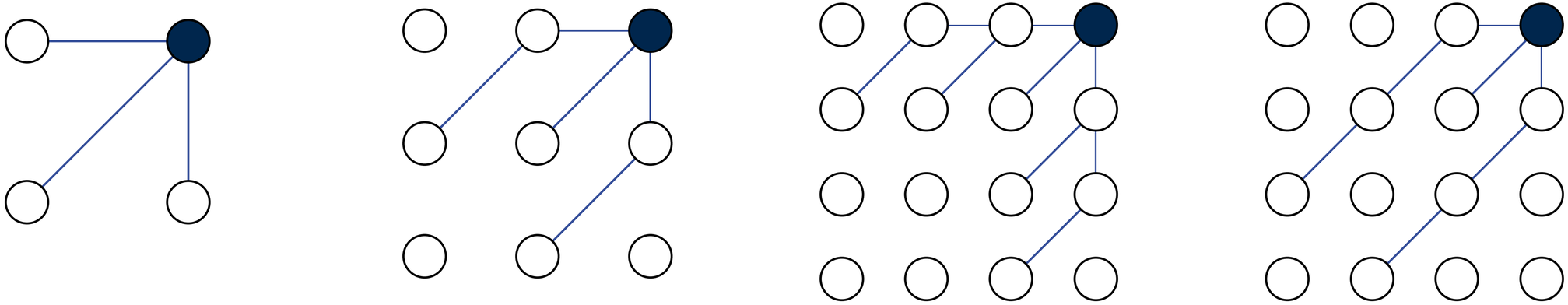
  - Repeat, but do not go back in time

- Enforcing time order
  - Won't match "cat" to "act"



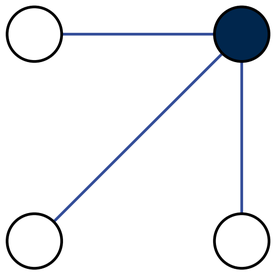
Variations of local constraints

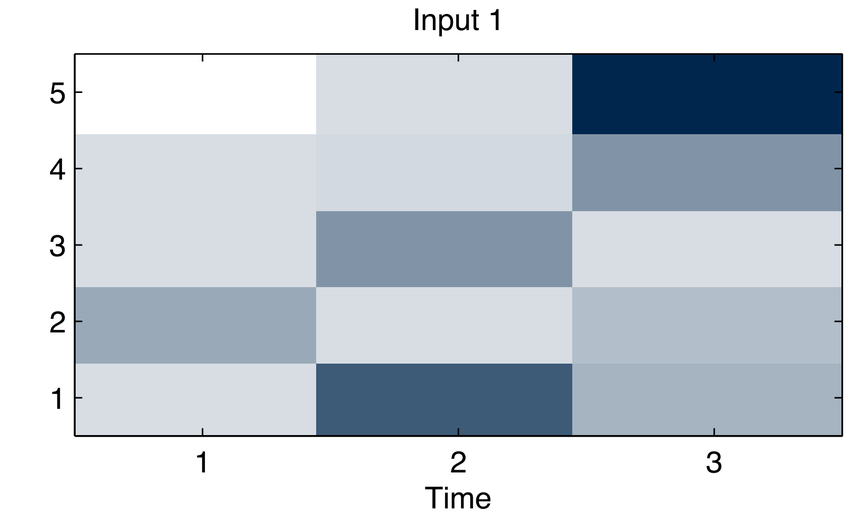
- Acceptable local subpaths
  - Consider only these local movements
  - Depends on application and scope

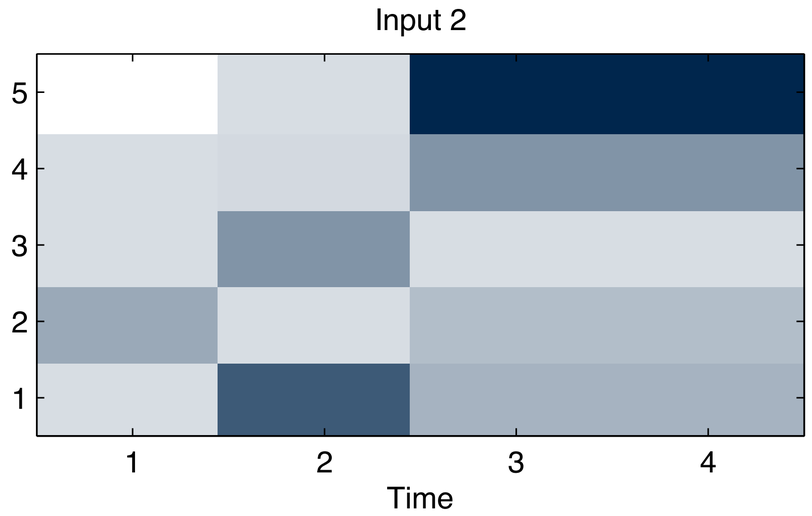


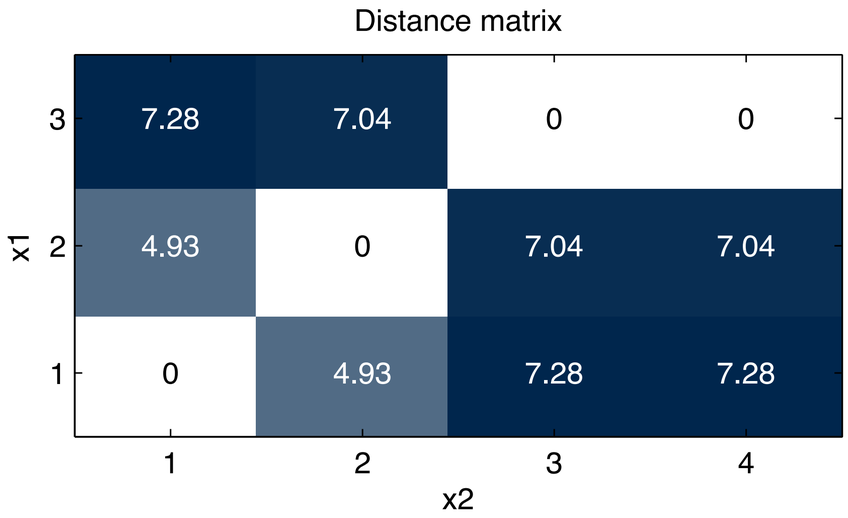
Example toy run

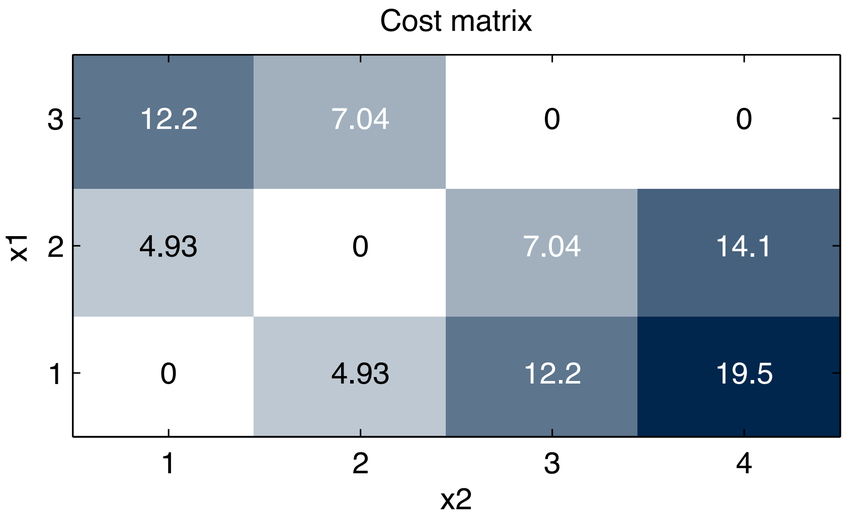
Local constraints



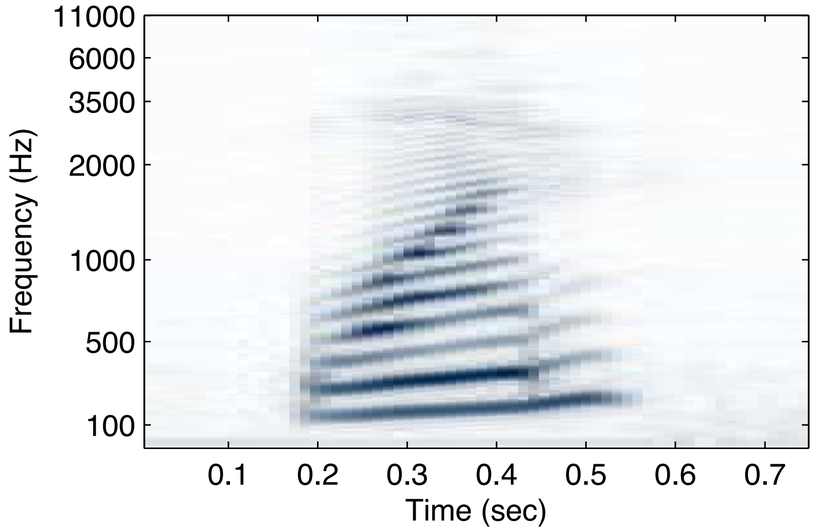


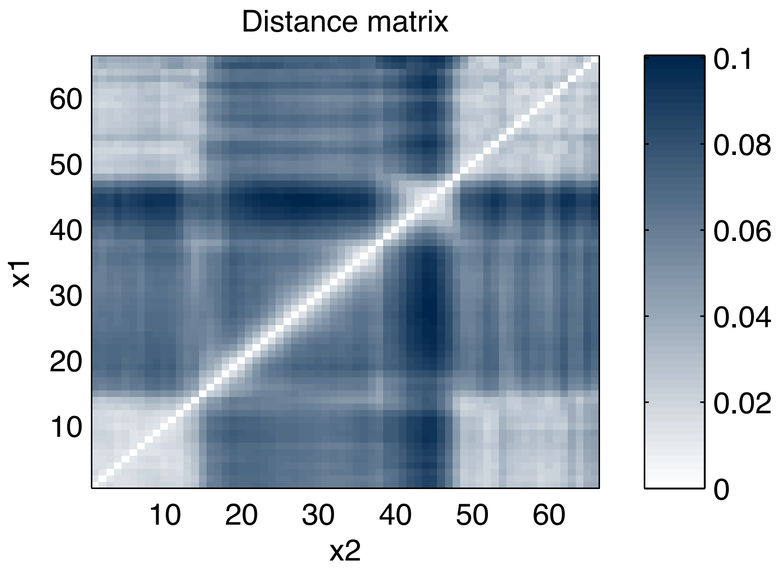


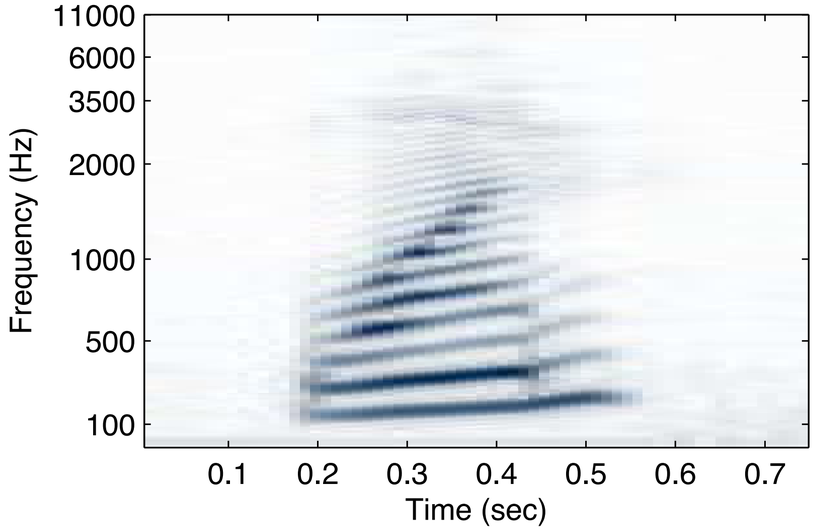


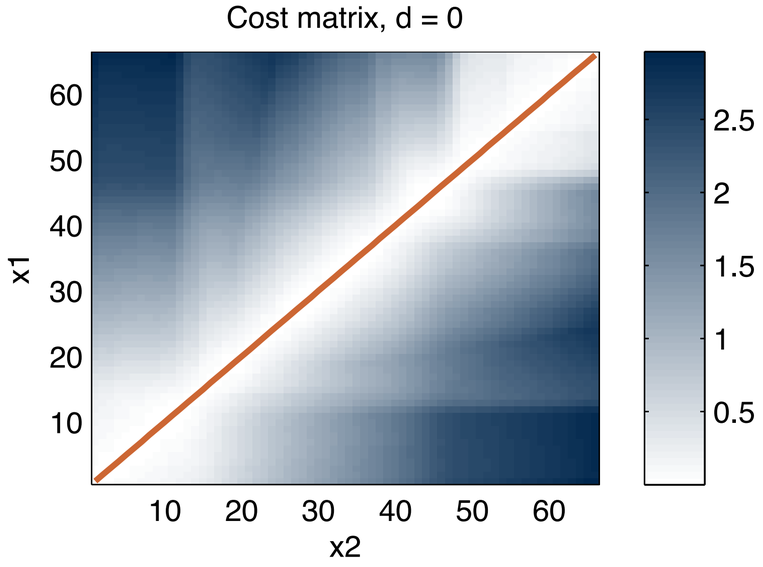


Matching identical utterances

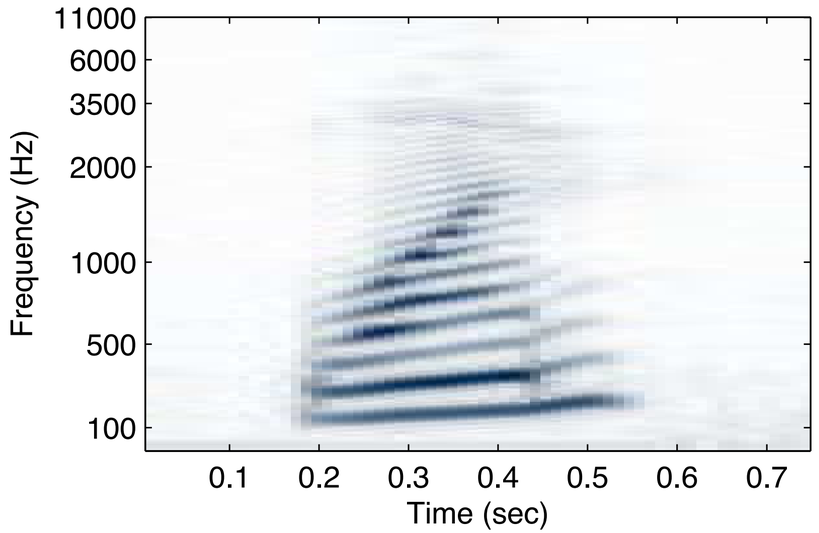


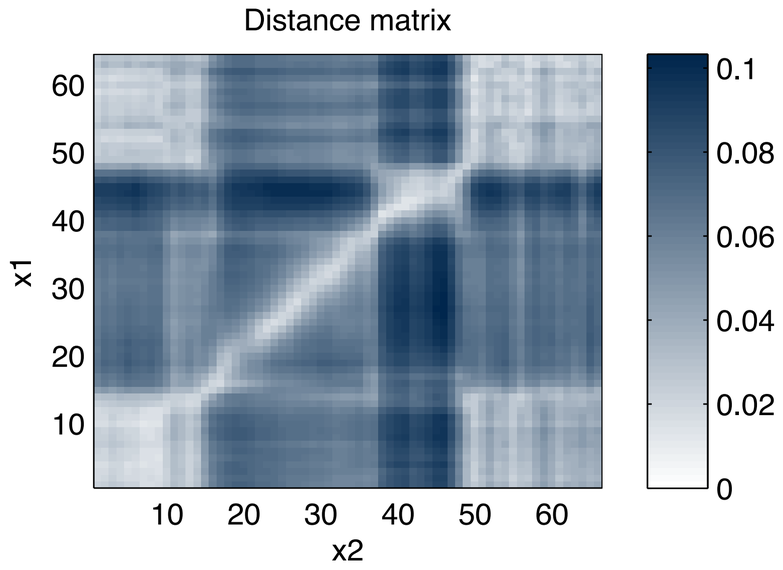


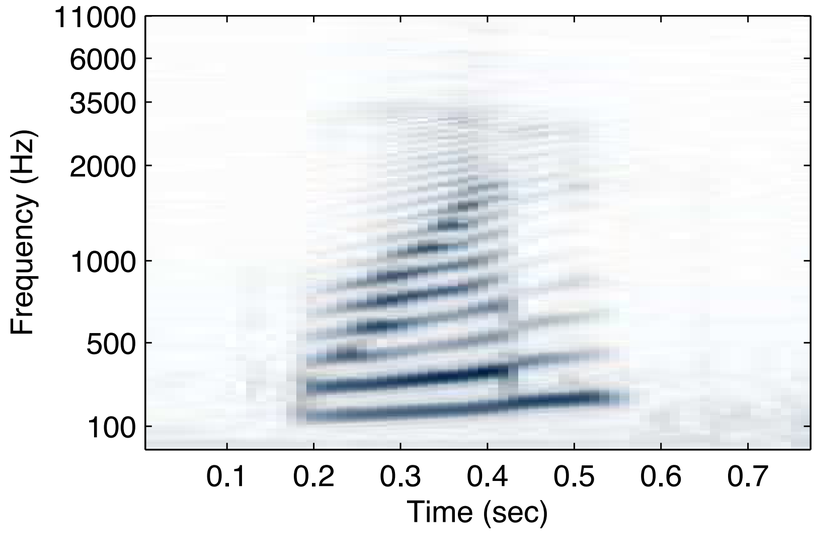


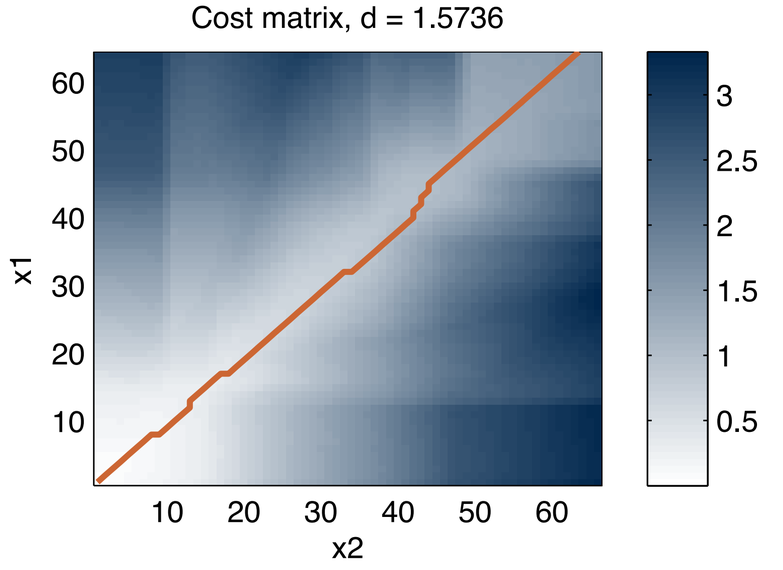


Matching similar utterances

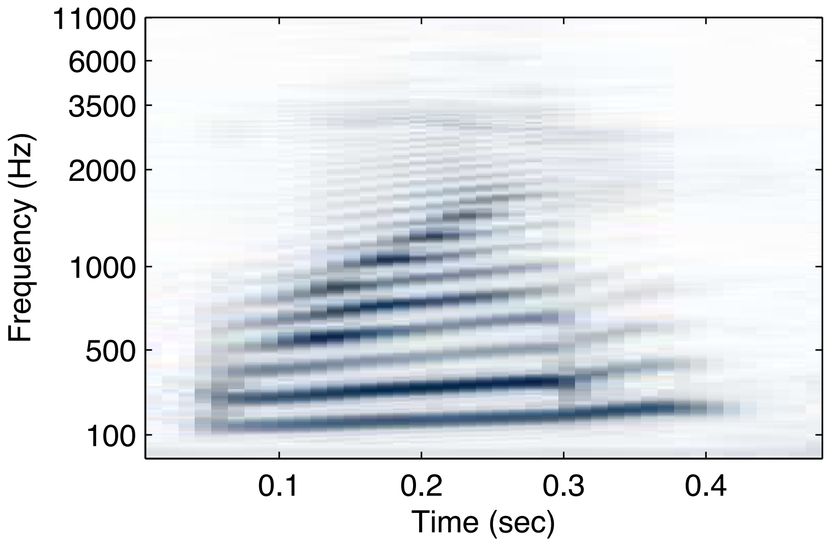


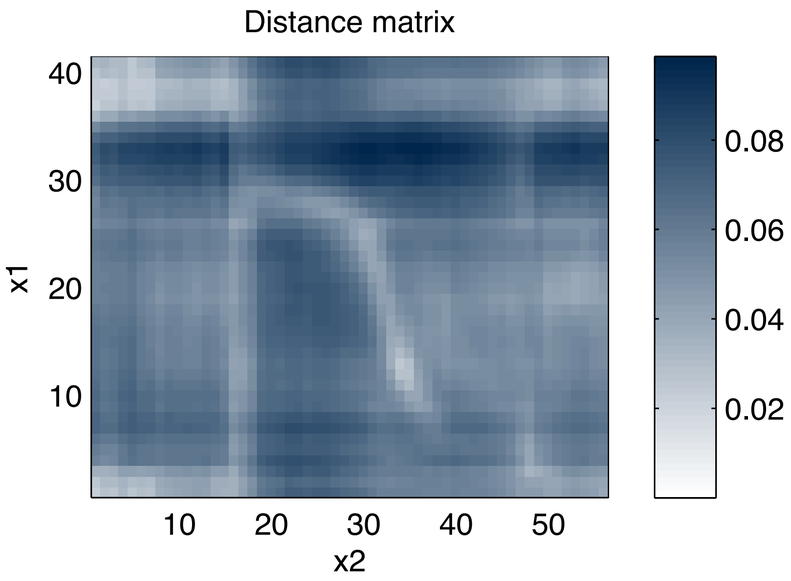


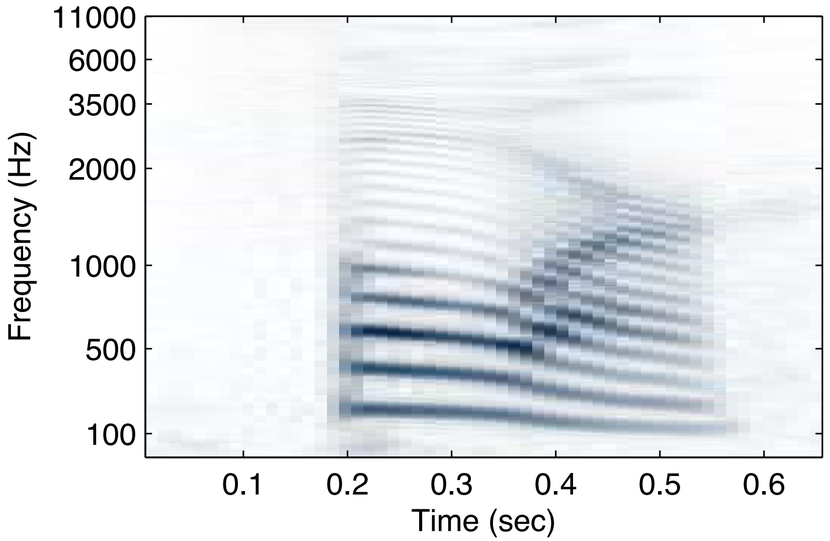


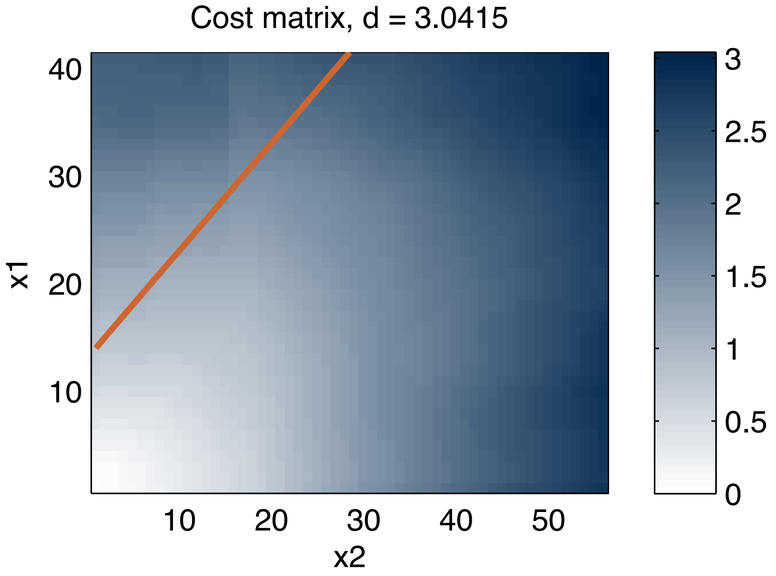


Matching different utterances



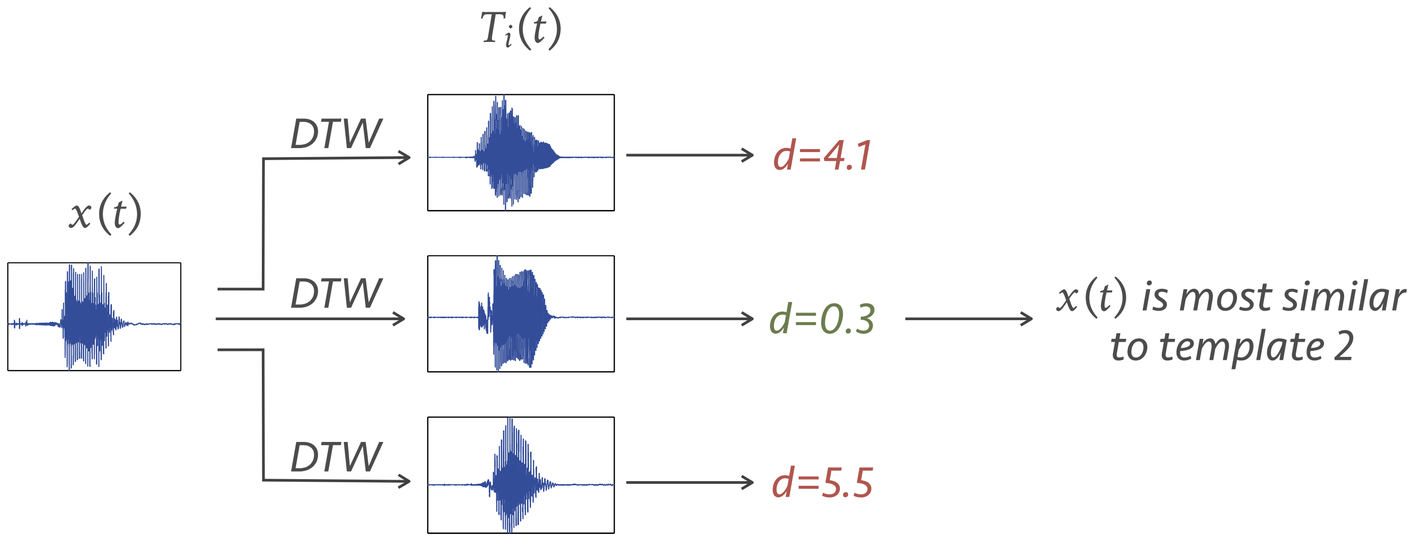




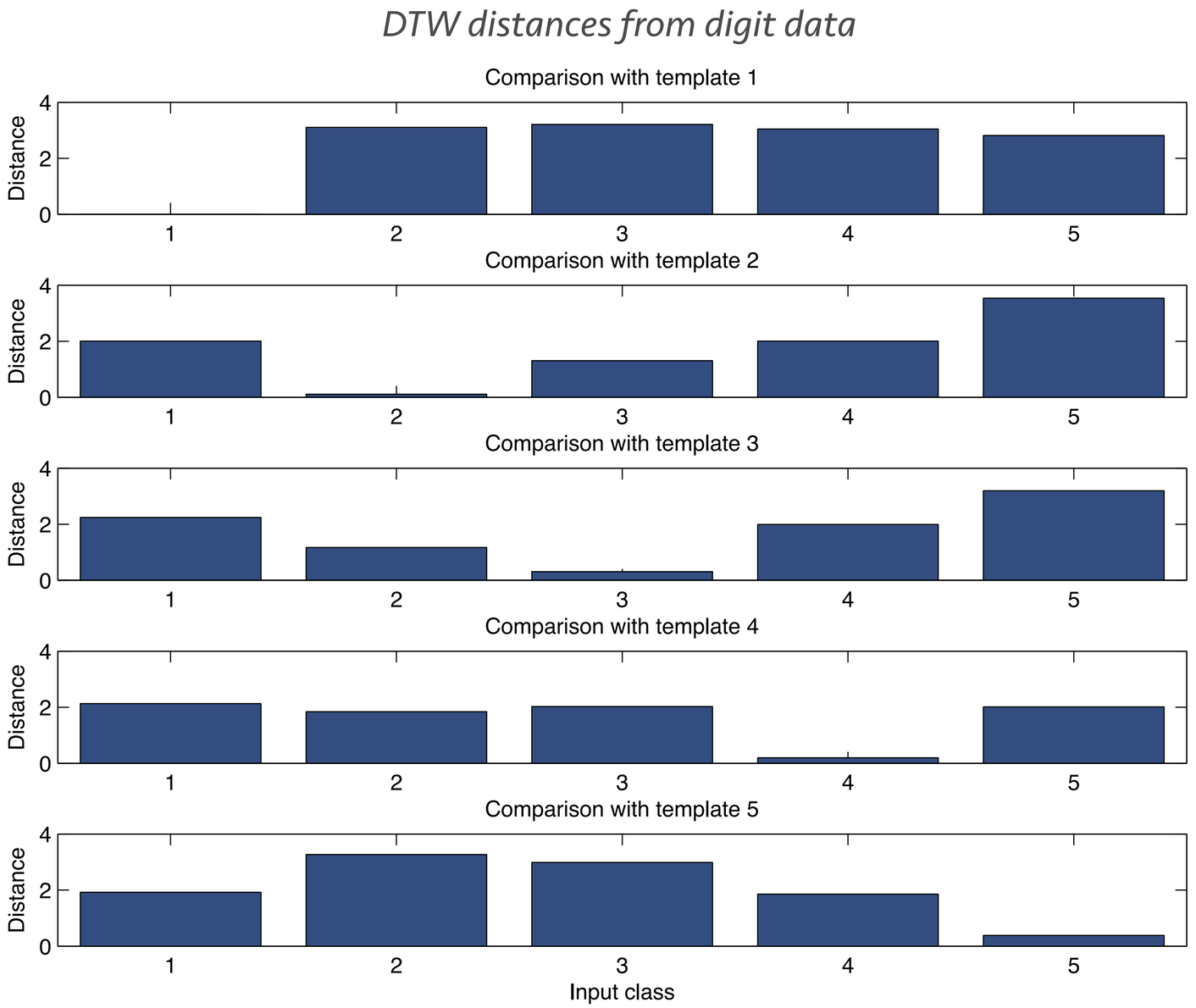


A basic speech recognizer

- Collect templates $T_i(t)$ for each word to detect
  - e.g. yes/no, or digits from 0 to 9
- Compute DTW distance between input and each $T_i(t)$
  - Template with smallest distance is most similar word to input



On our small digit dataset



Another application of DTW

- Simplifying ADR
  - Costs time and money
- Automatically align audio tracks

Good take, lousy audio

Good audio, lousy sync

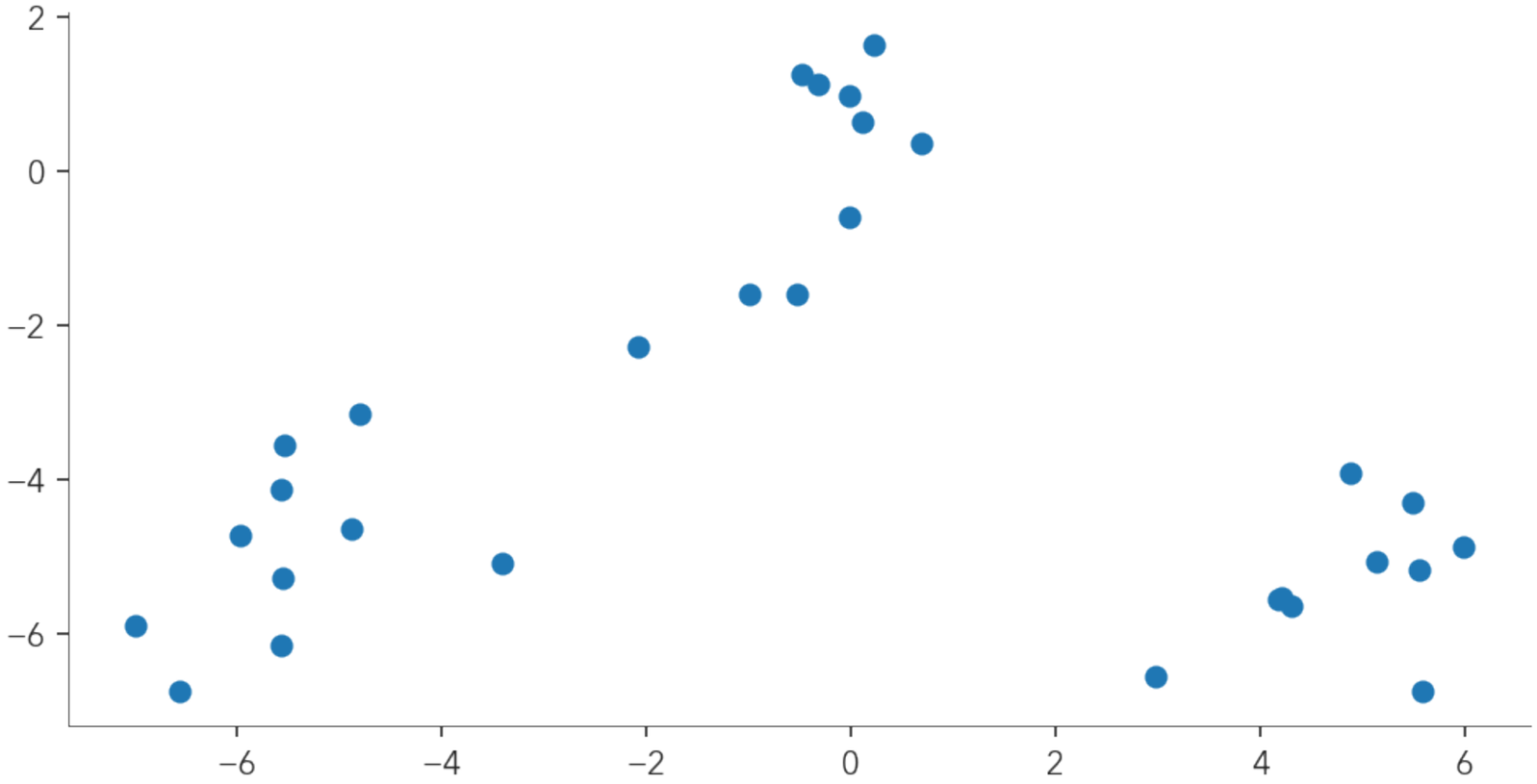
Good take, good audio!

Taking this to the next level

- DTW is very convenient for small problems
  - E.g., a small contacts list on your phone , vocabulary-limited command set, digit recognition, …
- It will not scale to full-blown speech recognition
  - E.g. with thousands of templates
  - For that we need more powerful tools

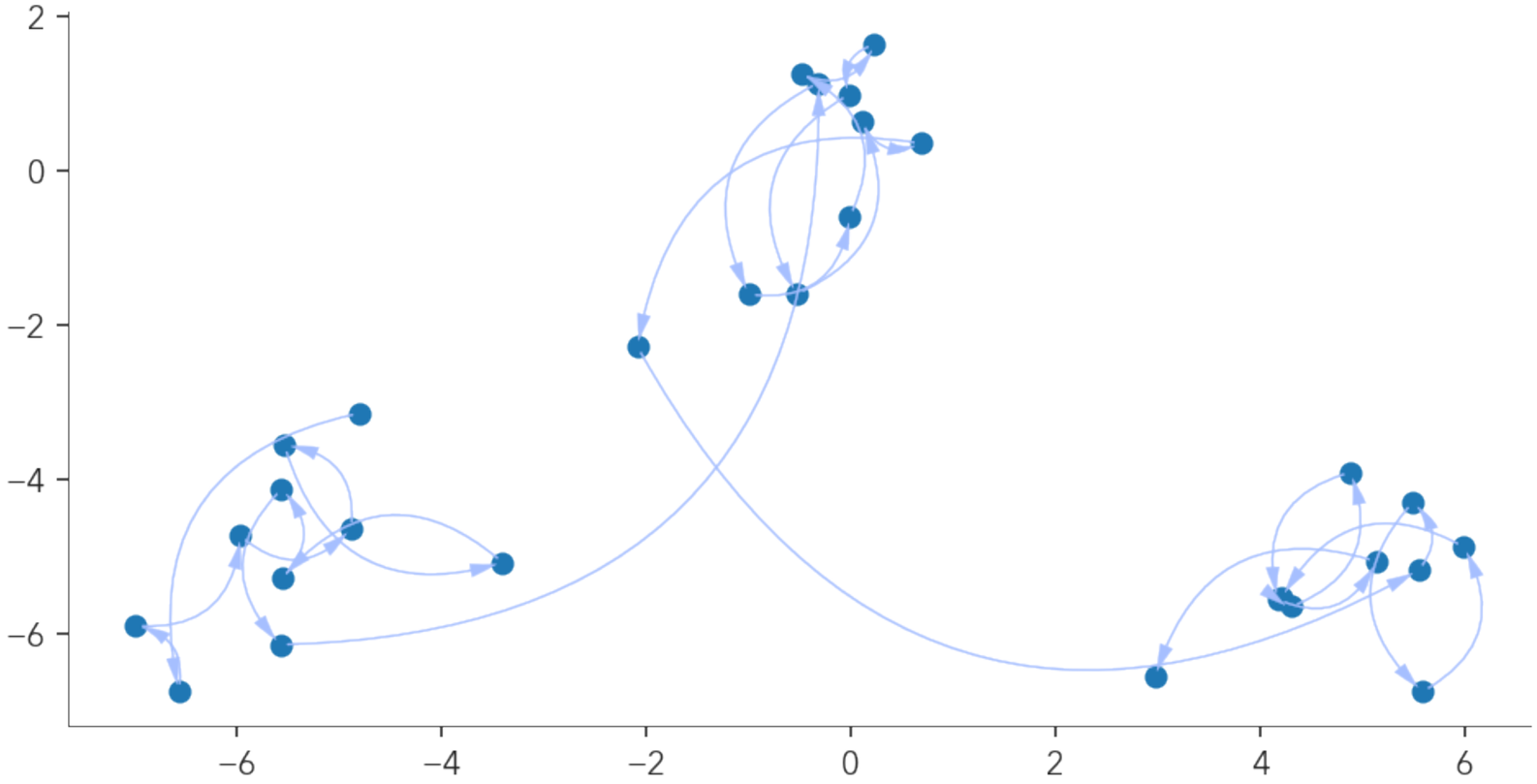
Starting from the GMM

- Gaussian Mixture Models explain data by using multiple Gaussians as opposed to just one



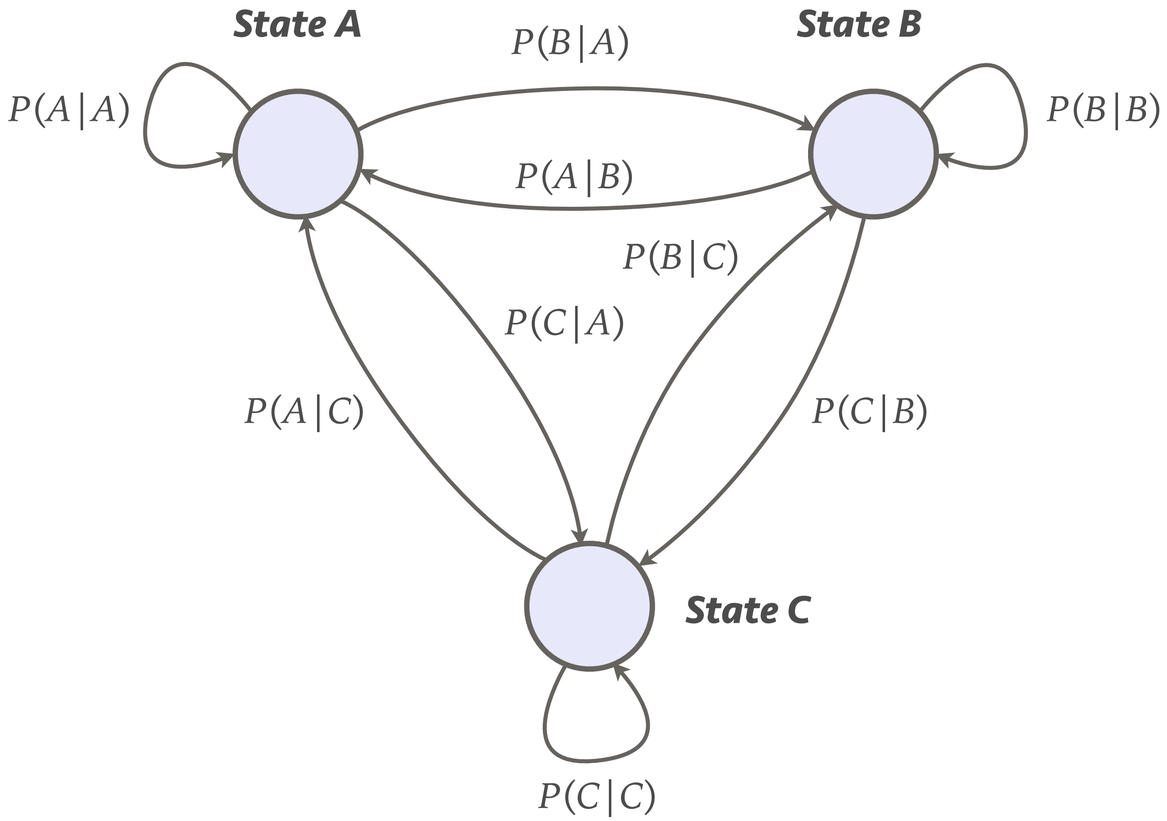
What if there is time order?

- We want to learn the dynamics of the sequence
  - e.g., each cluster can be a phoneme



Representing time

- Hidden Markov Model
  - Gaussian variant for now
- Each state is represented by a Gaussian distribution
- Transition probabilities between all states
  - Only the previous state matters



Initial Probabilities: $P(A), P(B), P(C)$

What can we do with an HMM?

- Learning
  - Given lots of inputs learn the model parameters
    - e.g., learn an HMM for each word you want to recognize
- Evaluation
  - For an input evaluate it's likelihood given the model
    - e.g., given word models find which explains the input best
- Decoding
  - For an input and a model find the optimal state sequence
    - e.g., given the model find what state you are in at any time

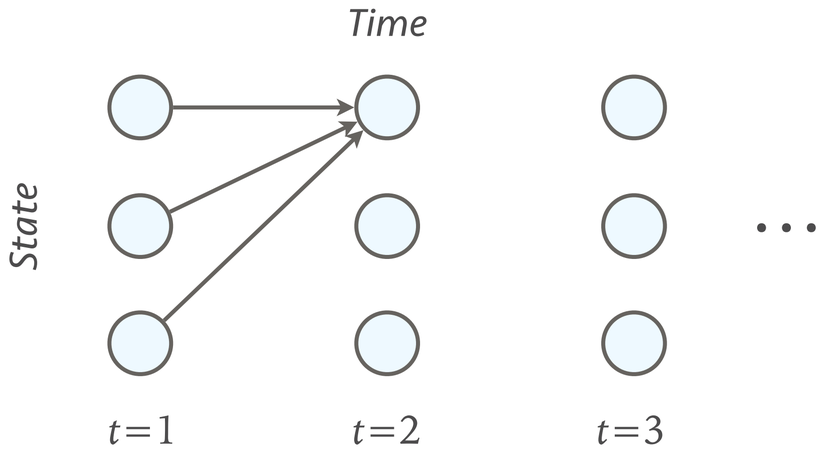
Evaluation

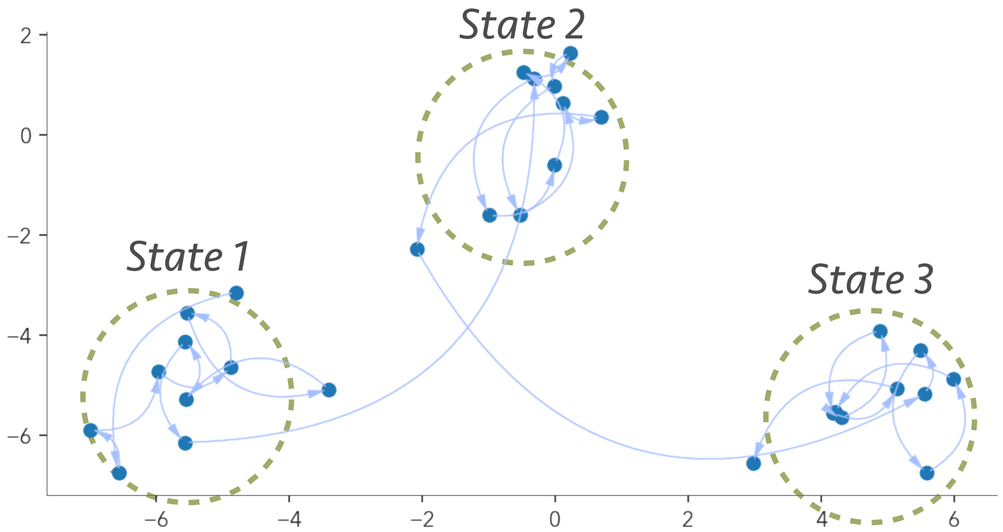
- Propagate likelihoods given the *forward algorithm*:

Probability of initial state *Likelihood of data given state*

$$\alpha_i(1) = P_1(i)P(x_1 \mid i)$$

$$\alpha_i(t+1) = P(x_{t+1} \mid i)\sum_j \alpha_j(t)P(i \mid j)$$





Evaluation

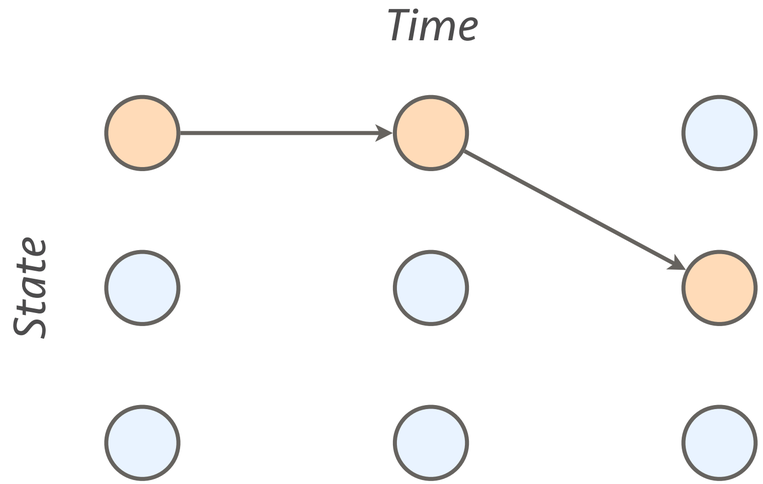
- $\alpha_i(t)$ denotes the probability of being in state i at time t, given the past

- Terminal value of α provides the overall likelihood of the model given input sequence

$$P(\mathbf{x}) = \sum_i \alpha_i(T)$$

Decoding

- The forward pass provides us with state likelihoods
- With the *backwards pass* we can find *most likely* state at any time
  - Like the backwards pass in DTW
  - Most-likely state depends on context!
- The *Viterbi* algorithm



The Viterbi algorithm

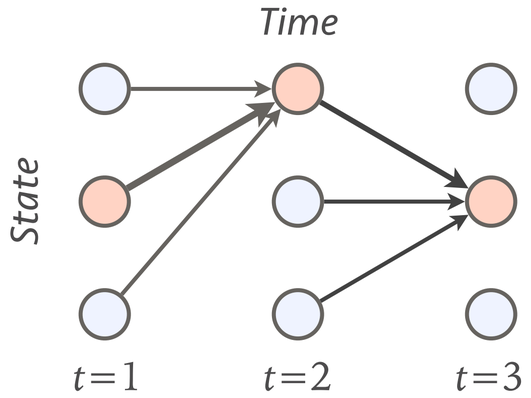
- Similar to forward algorithm, but with *hard* decisions:

$$v_i(1) = P_1(i)P(x_1 \mid i)$$

$$v_i(t+1) = P(x_{t+1} \mid i)\max\Big(P(j \mid i)v_j(t-1)\Big)$$

  - Probability of most likely path up to time $t+1$ that ends in state i
  - For each step remember most likely transition (as with DTW)

- Backwards pass
  - Start with most likely terminal state
  - Work backwards with most likely transitions



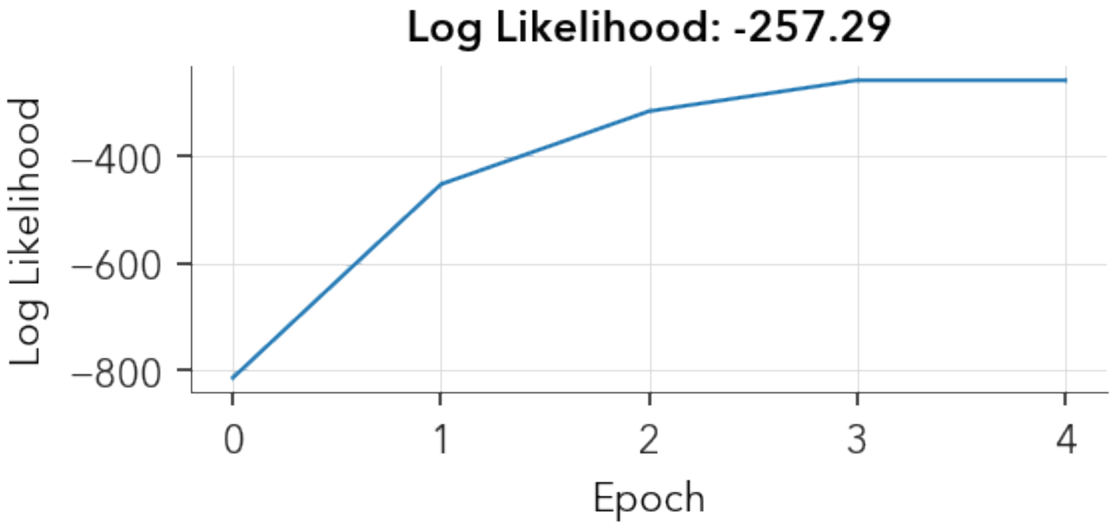
Learning the parameters

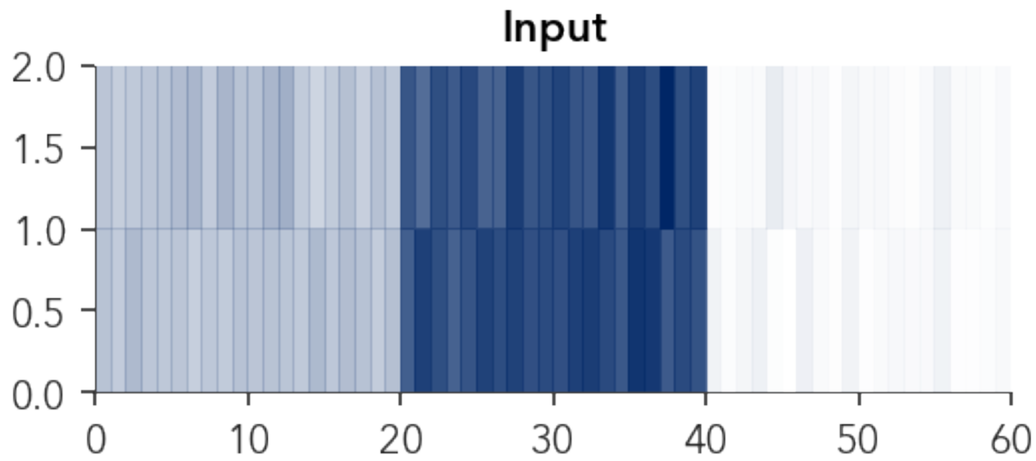
- Simple algorithm: *Viterbi learning*
  - *Init*: set parameters to random values
  - *Loop*: while there is no change
    - Find most likely state sequence
    - Estimate new model parameters using it
      - i.e., estimate state i's Gaussian mean/covariance using all the time frames assigned to state i based on the decoding
      - Transition matrix is estimated by counting state transitions
- Heavy-duty version
  - *Expectation-Maximization*

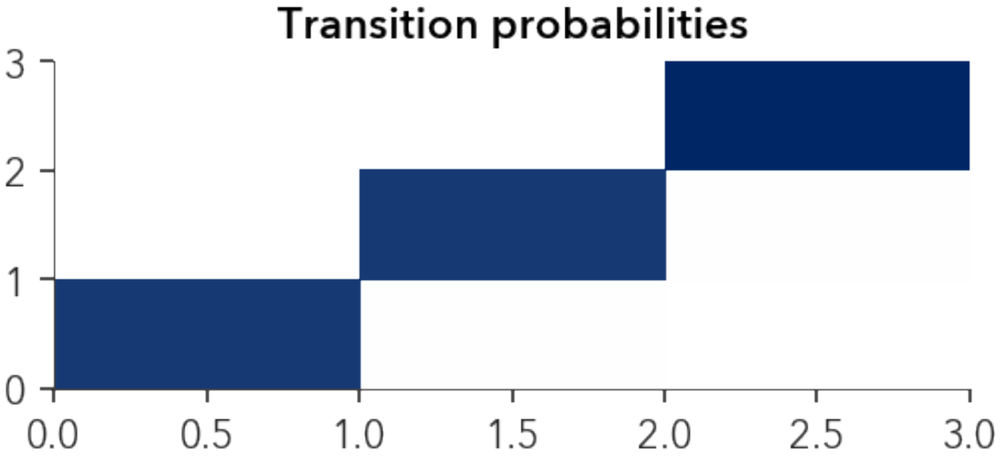
Things to note

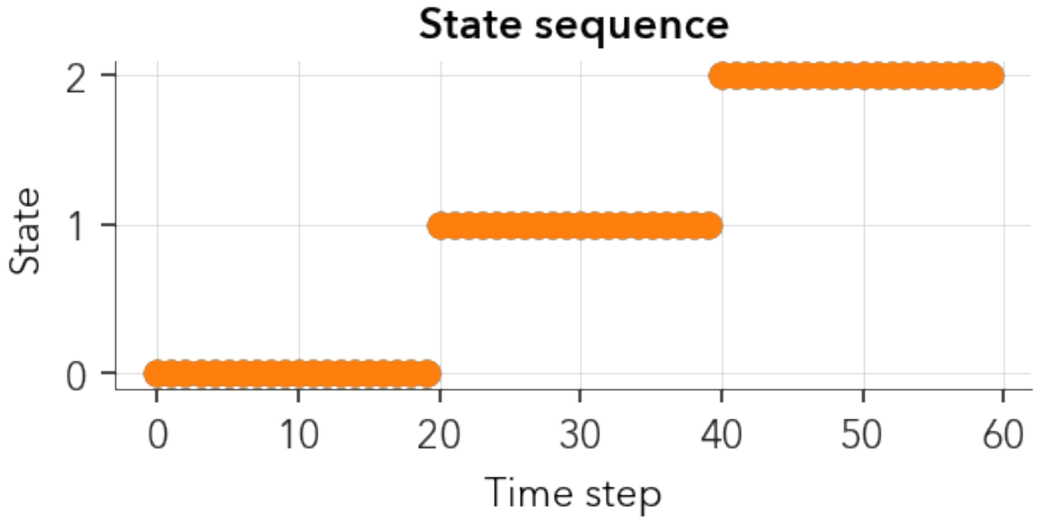
- We can extent HMM learning to work with multiple training sequences
  - e.g., learn an HMM for “one” using 100 recordings of it
- Use log probabilities!
  - What is a state likelihood like after a million time points? Can you compute that with linear probabilities?
- You can use an arbitrary state model
  - e.g., a Gaussian, a neural net, a GMM, …

Learning an HMM



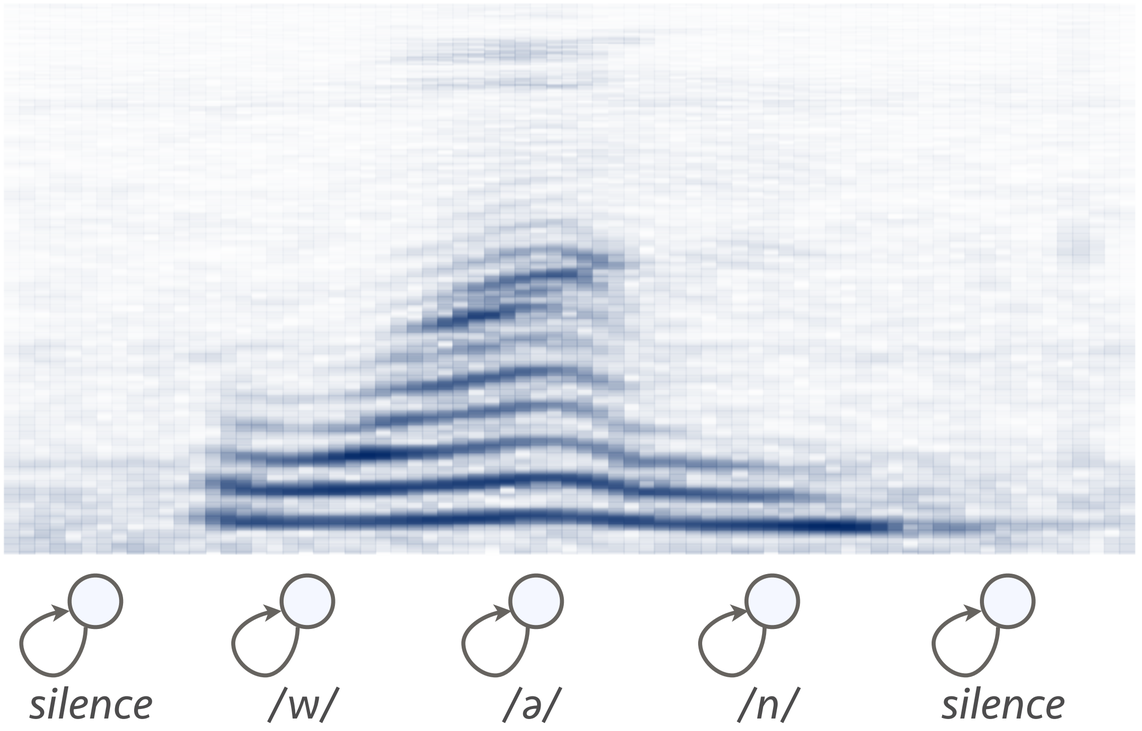




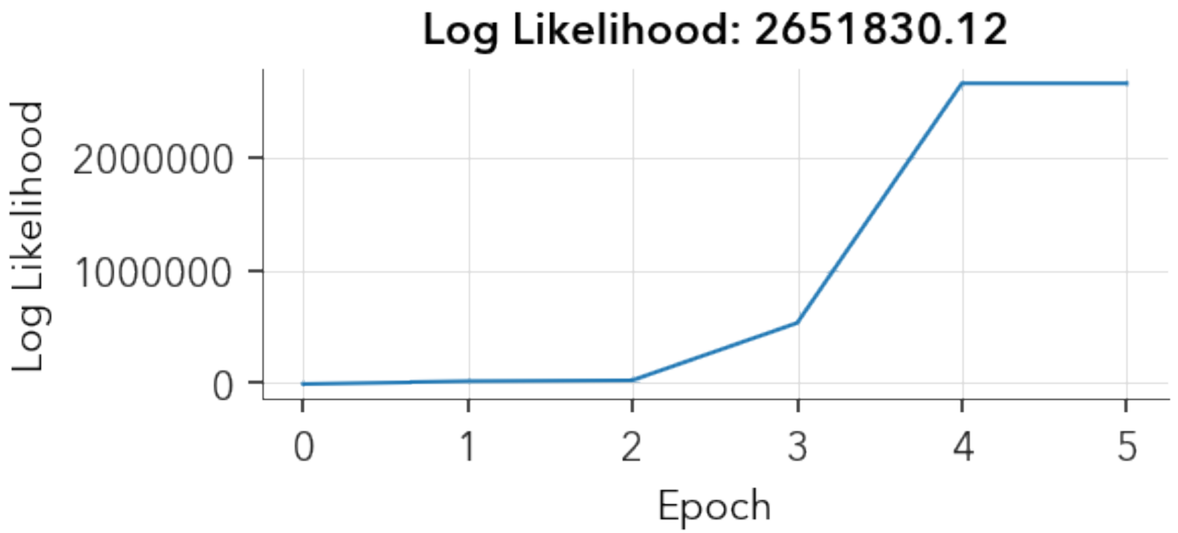


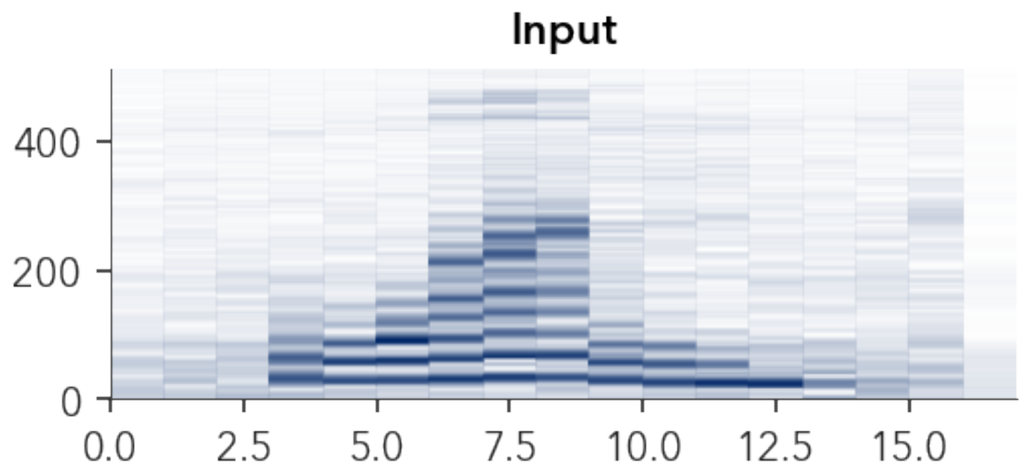
Back to speech

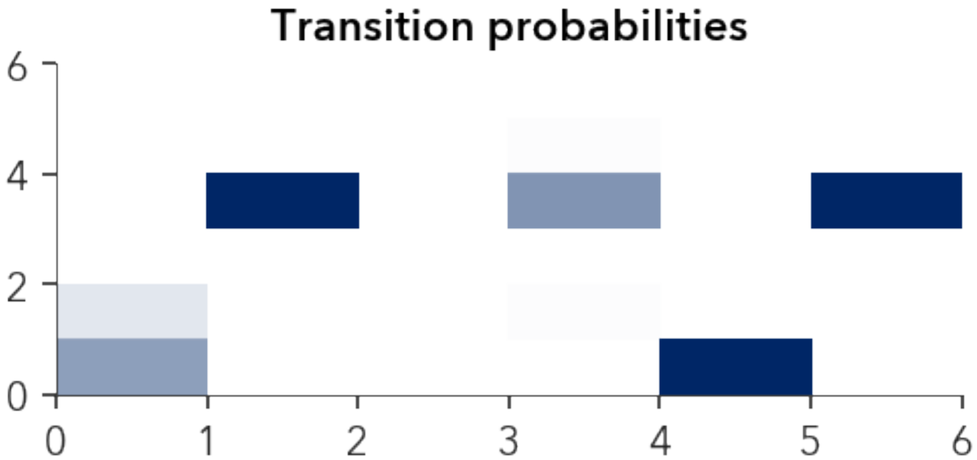
- Learning an HMM for a word
  - Each state should correspond to a static sub-part
  - e.g., a syllable, a phoneme, etc.

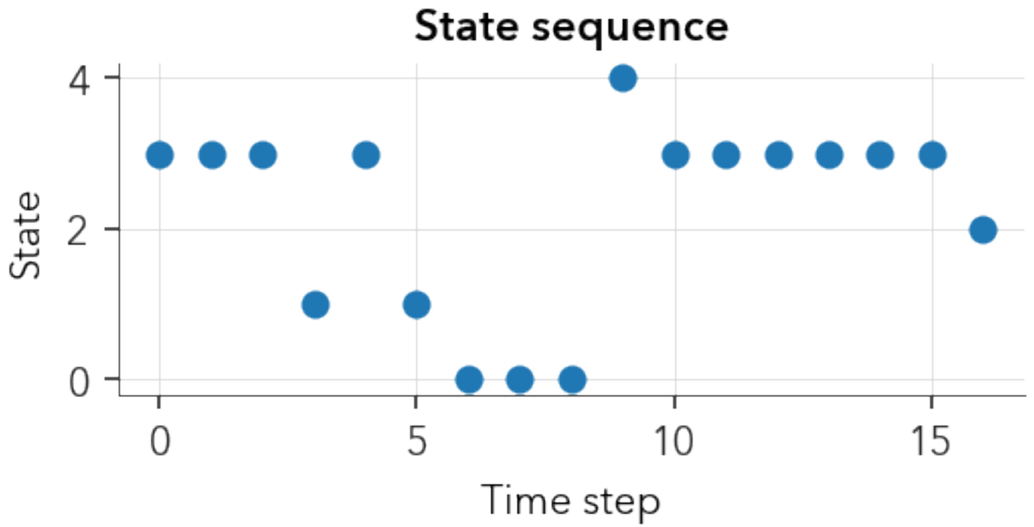


Fully-connected model

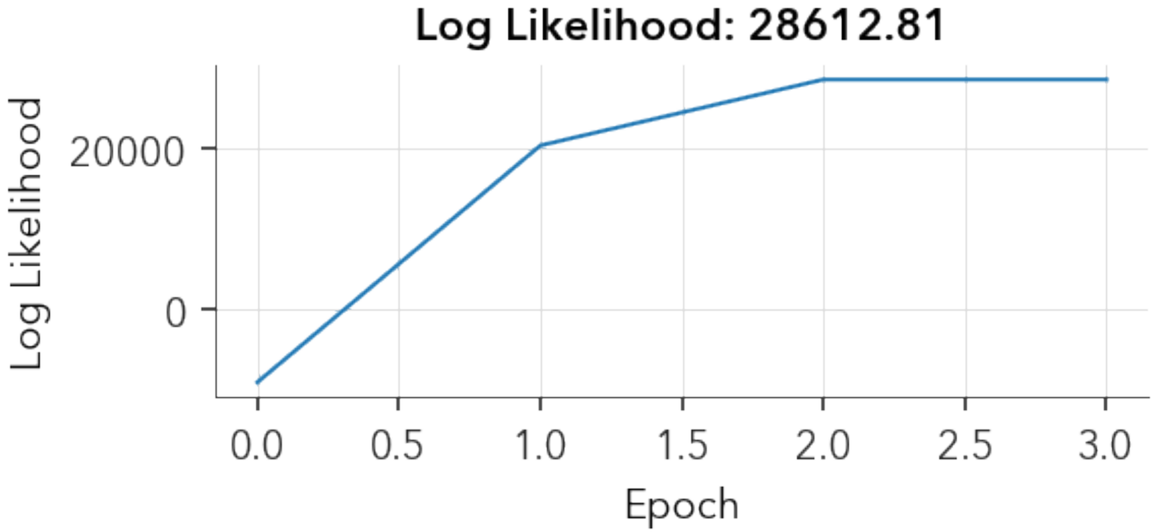


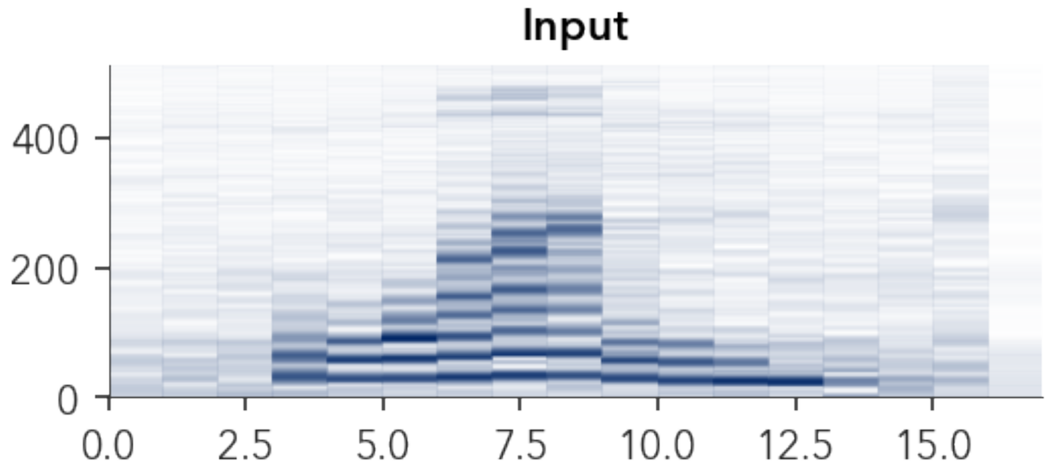


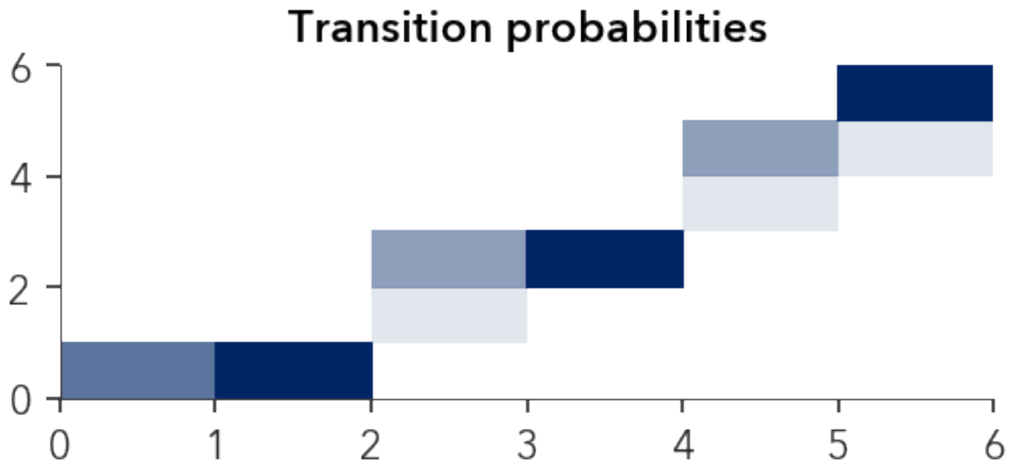


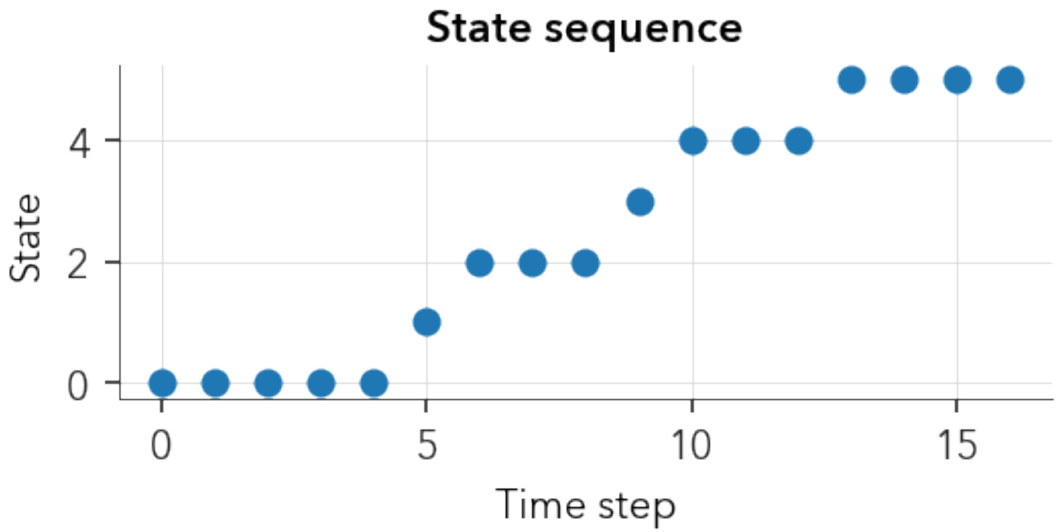


Left-to-right model

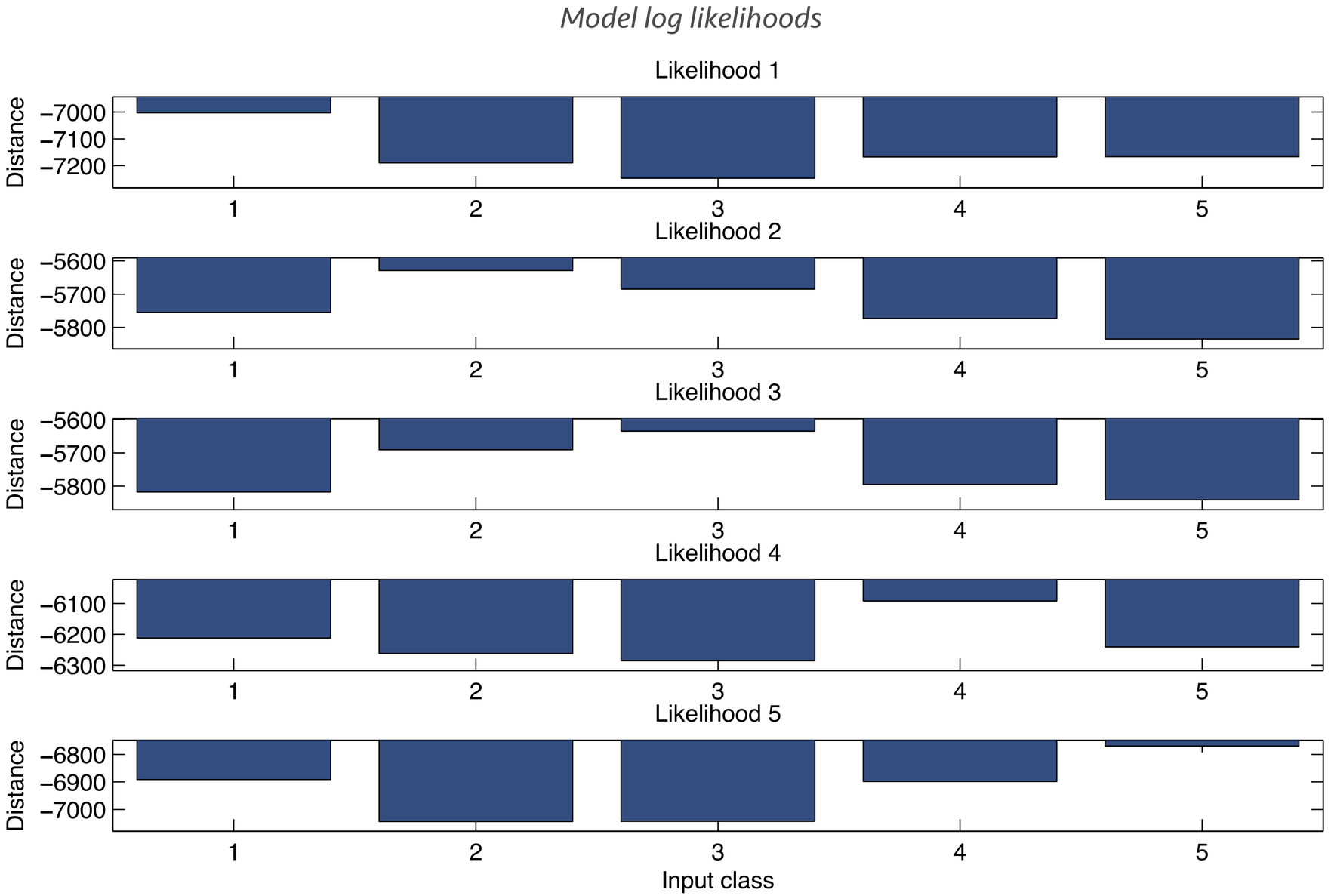






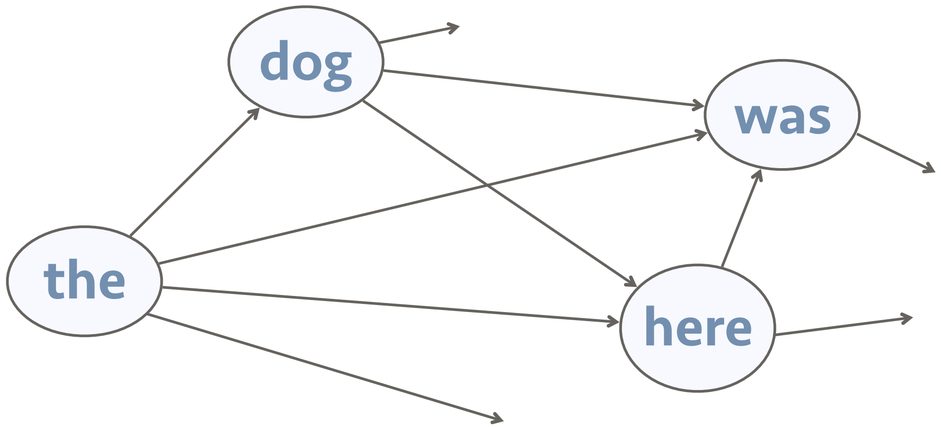


Digit recognition results



Speech recognition in a nutshell

- Small scale recognition
  - Connect word HMMs based on language model



- Large systems are another story …
  - And lately they are solved differently using deep learning

Recap

- Learning time series
- Dynamic Time Warping
  - Align and compare sequences
- Hidden Markov Models
  - Lean sequence models and get likelihoods
- Basic speech recognition setup

Reading material

- DTW for keyword recognition
  - http://www.irit.fr/~Julien.Pinquier/Docs/TP_MABS/res/dtw-sakoe-chiba78.pdf
- A tutorial on HMMs
  - http://www.cs.ubc.ca/~murphyk/Bayes/rabiner.pdf

Next lab

- Designing a simple speech recognizer!