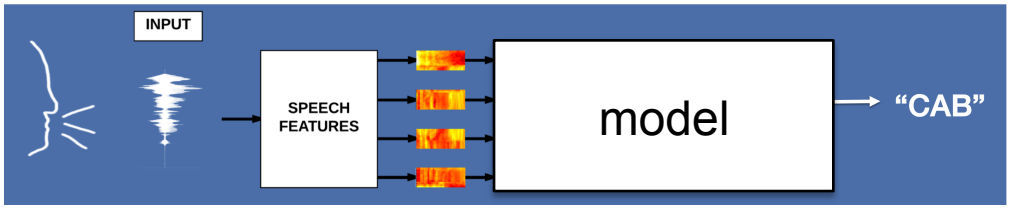
Algorithms for speech and language processing

#3. ASR: Language modeling

The three problems of speech recognition



O Θ W

Notations

- θ: model parameters
- $O=o_1 \ldots o_t$: observed speech frames
- $W=w_1..w_n$: word transcriptions

1. Likelihood: given param θ and observed O, compute p(O|θ).

2. Decoding: given observed O and θ= find the most likely W.

   $W^* = \arg\max_w p(O|W,\theta)$

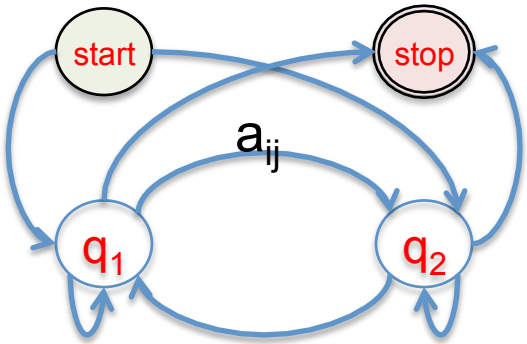
3. Learning: given an observed O and model, find the best θ

   $\theta^* = \arg\max_\theta p(O|\theta)$

Part I: Models

- Useful finite state models
- Model decomposition
  - acoustic model
  - pronunciation model
  - language model

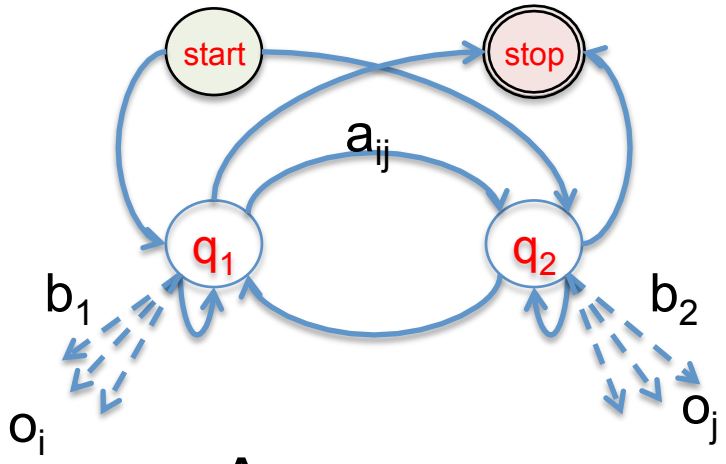
markov chains



A

	q_0 (start)	q_1	q_2	q_F (stop)
q_0 (start)	0	a_{01}	a_{02}	0
q_1	0	a_{11}	a_{12}	a_{1F}
q_2	0	a_{21}	a_{22}	a_{2F}
q_F (stop)	0	0	0	0

hidden markov models (HMMs)



B

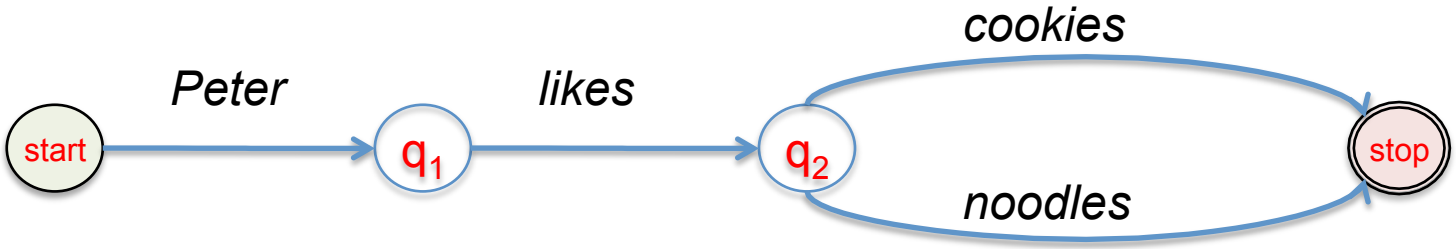
	q_1	q_2
o_1	b_{11}	b_{21}
o_2	b_{12}	a_{22}
o_3	b_{13}	a_{23}

A

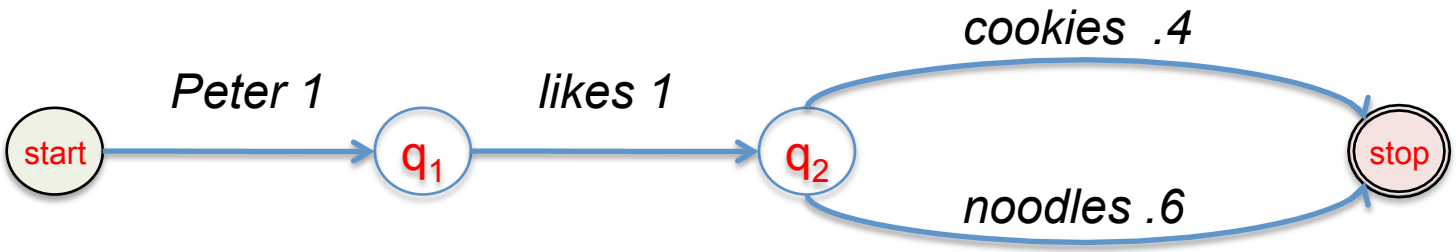
	q_0 (start)	q_1	q_2	q_F (stop)
q_0 (start)	0	a_{01}	a_{02}	0
q_1	0	a_{11}	a_{12}	a_{1F}
q_2	0	a_{21}	a_{22}	a_{2F}
q_F (stop)	0	0	0	0

(weighted) finite state transducers

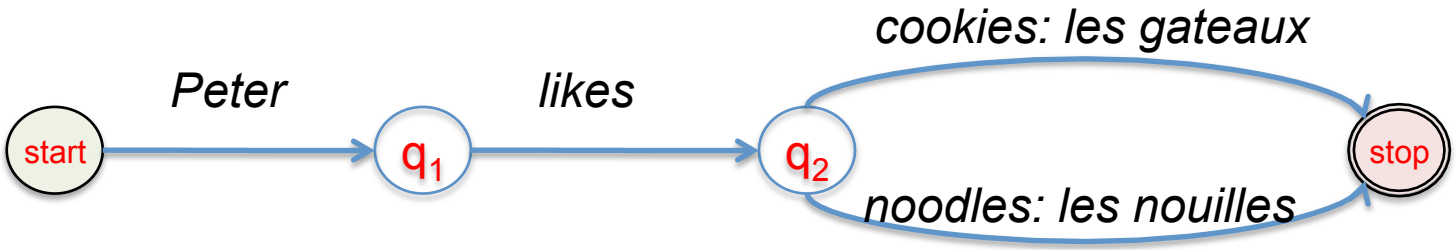
Finite State Acceptor (FSA)



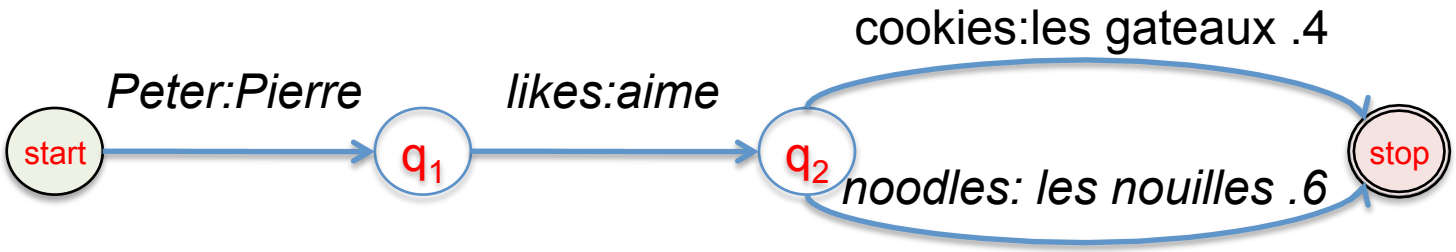
Weighted Finite State Acceptor (WFSA)



Finite State Transducer (FST)

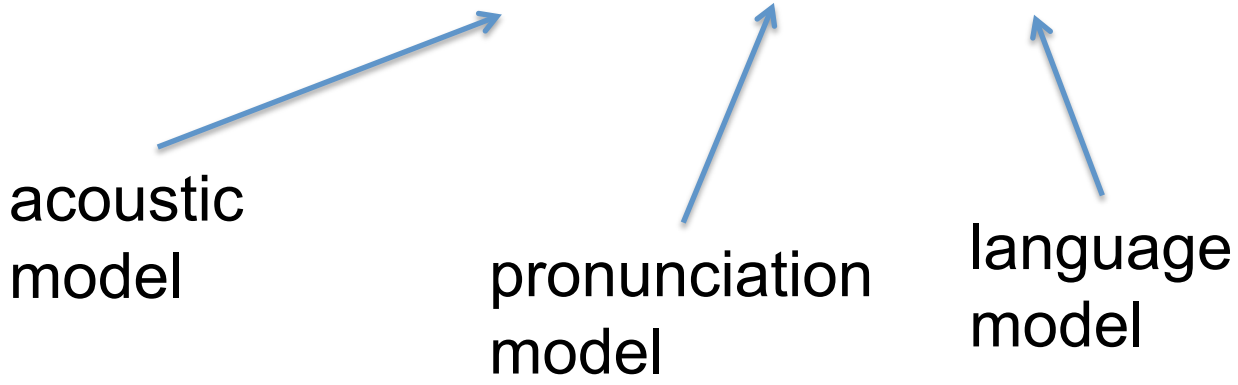


Weighted Finite State Transducer (WFST)



Model decomposition

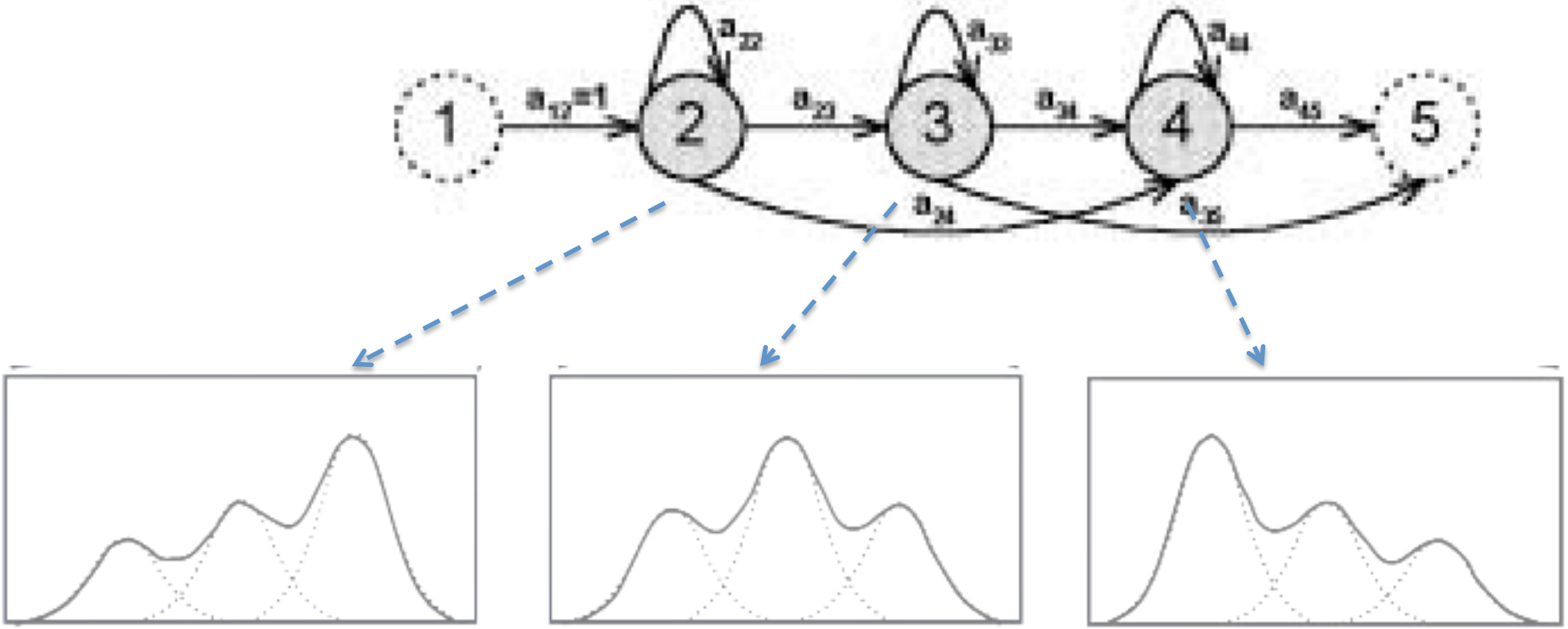
- $W^* = \arg\max_W P(W|O,\Theta)$
- $P(W|O,\Theta) \ \alpha \ P(O|W,\Theta)P(W|\Theta)$
- $= P(O|Q)P(Q|W)P(W)$



Acoustic models

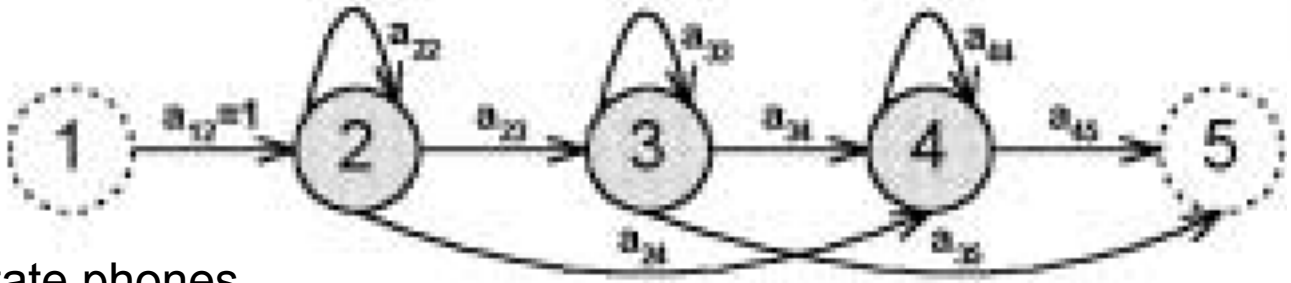
P(O|Q)

- tri-state phones

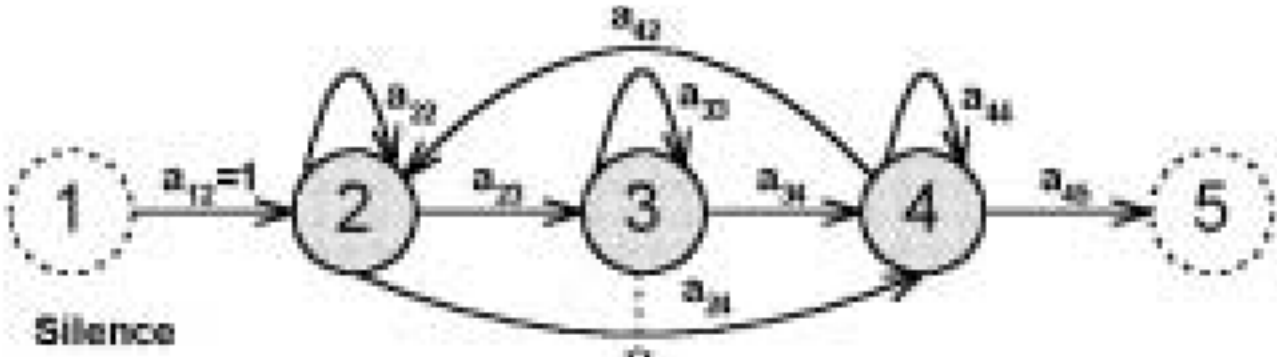


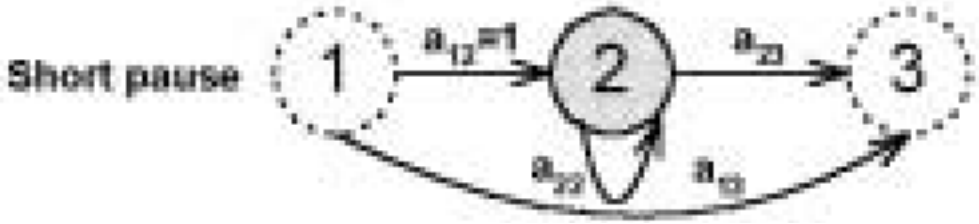
Acoustic models

P(O|Q)



tristate phones





Pronunciation models

P(Q|W)

https://www.youtube.com/watch?v=J3lYLphzAnw

Prononciations of the word ‘eleven’ in the state of Pennsylvanie, USA

ɬˆlɛˀvə̆n
ɬˆlɛvə̆n
ɬˆɬɛvə̆n
əˈlɛˀvə̆n
ɬˆlɛvə̆n
əlɛvə̆n
əˈɬɛvn
əɬɛbm
ɬˆˈlɛvn
lɛvə̆n
əɬɛˬˀᵊvə̆n
ɬɛvɱ
əɬɛᵊvə̆n
ɬˈlɛvɱ
…
(total= 220 variants)

Phonological/Phonetic transformations

What are you doing? ˈwʌtʃəˈduɪn

I can inquire. ˈaɪkŋ̍ˈkwaɪɚ

Did you eat yet? ˈdʒitjɛʔ

I don’t believe him. aɪˈdoʊ̃bəˈlivm̩

We ought to have come. wiˈɔɾ̪ɣ̍ˈkʌm

Pronunciation models

1. Using an existing phonetic dictionary

orthography	SAMPA	IPA
counting	kA_wntlN	kɑʷntɬŋ
counting(1)	kA_wnlN	kɑʷnɬŋ
amortization	@mOrt@zeS@n	əmɔrtəzeʃən
amortization(1)	{mOrt@zeS@n	æmɔrtəzeʃən
amortization(2)	@mOrtA_jzeS@n	əmɔrtɑʲzeʃən
..etc...		

2. Using G2P to extend the dictionary

- eg: g2p-seq2seq (CMU), phonetisaurus
  - align graphemes and phonemes
  - build an ngram model
  - make a WFST from it
  - iterate

"speaking"	=	s	p	ea	k	ing
[spiːkɪŋ]		[s]	[p]	[iː]	[k]	[ɪŋ]

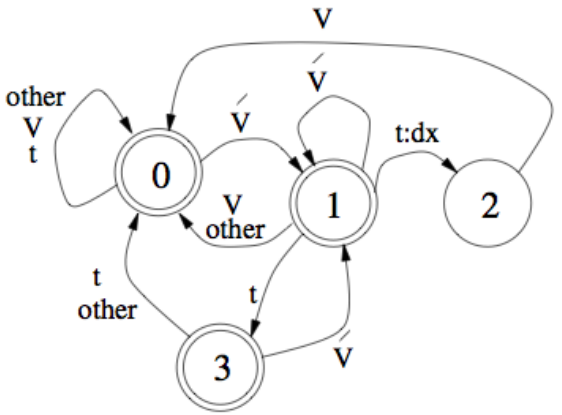
ចាស់ ⇒ c aː h

ប្រយុទ្ធ ⇒ b r ɑː j u t

3. phonological rules (hand crafted)

- ex: english flapping

$$/t/ \rightarrow [dx] \quad / \quad [+\text{vowel}] \_\_ \begin{bmatrix} +\text{vowel} \\ -\text{stress} \end{bmatrix}$$



Input symbols:	ax	t	eh	n	y	uw	ey	t	ax	d
State sequence:	0	0	0	1	0	0	0	1	2	0	0
Output symbols:	ax	t	eh	n	y	uw	ey	dx	ax	d

Name	Rule	Example	Prob
Syllabic Rules*			
Syllabic n	[ax ix] n → en	butt**on**	.35
Syllabic m	[ax ix] m → em	bott**om**	.32
Syllabic l	[ax ix] l → el	bott**le**	.72
Syllabic r	[ax ix] r → axr	butt**er**	.77
Flapping	[tcl dcl] [t d]→ dx /V ___ [ax ix axr]	bu**tt**on	.87
Flapping-r	[tcl dcl] [t d]→ dx /V r ___ [ax ix axr]	bar**t**er	.92
H-voicing	hh → hv / [+voice] ___ [+voice]	**a**head	.92
L-deletion	l → Ø/ ___ y [ax ix axr]	mi**ll**ion	*n/a*
Gliding	iy → y / ___ [ax ix axr]	colon**i**al	*n/a*
Nasal-deletion	[n m ng] → Ø/ ___ [-voice -consonant]	ra**n**t	*n/a*
Function words			
h-deletion	h → Ø/ # ___	**he**, **h**im	*n/a*
w-deletion	w → Ø/ # ___	**w**ill, **w**ould	*n/a*
dh-deletion	dh → Ø/ # ___	**th**is, **th**ose	*n/a*
Dental-deletion	[tcl dcl] [t d] → Ø/ [+vowel] ___ [th dh]	brea**d**th	*n/a*
Final dental-deletion	([tcl dcl]) [t d] → Ø/ [+cons +continuant] ___ #	soft (as)	*n/a*
Slur	ax → Ø/ [+consonant] ___ [r l n] [+vowel]	c**a**m**e**ra	*n/a*
Stressed slur	[+vowel +stress] r → er	w**ar**ts	*n/a*
Pre-stress contraction	ax → Ø/ [+cons] ___ [+cons] [+vowel +stress]	**s**enility	*n/a*
Ruh-reduction	r ax → er / [-word bdry] ___ [-word bdry]	sep**ar**able	*n/a*
Transitional stops			
t-introduction	Ø→ tcl / [+dental +nasal] ___ [+fricative]	prin[**t**]ce	*n/a*
t-deletion	[tcl] → Ø/ [+dental +nasal] ___ [+fricative]	prin**t**s	*n/a*

4. phonological rules (data driven)

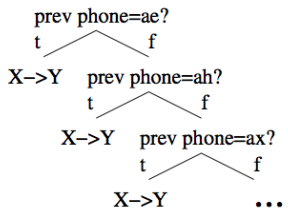
- using a phone recognizer to learn a pronunciation dictionary

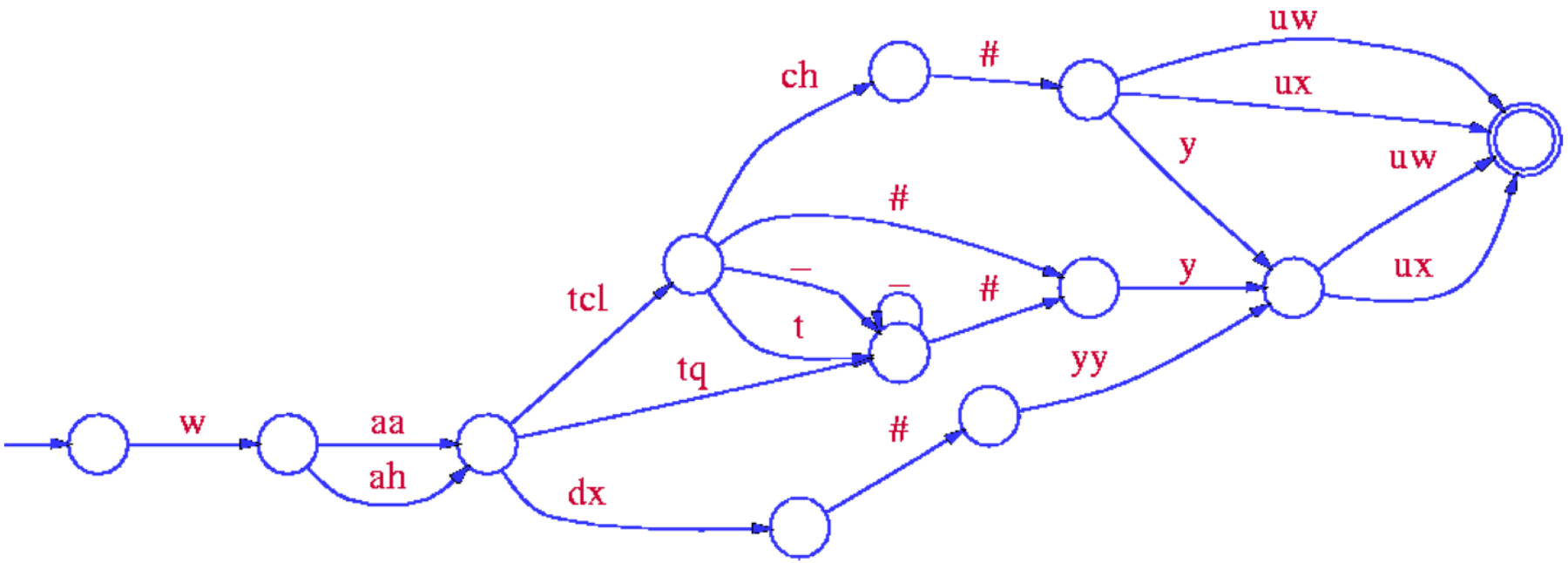
a	baseball						game			
ax	b	ey		s	b	ao	l	g	ey	m
ax	b	eh	ey	s	b	el		g	eh	m
ax	b	eh	ey	s	b	el		g	eh	m

word sequence
canonical phone sequence
aligned surface phones
proposed new pronunciations

- inducing pronunciation rules
  - trigram models
  - decision trees

$$P(S^i) = \prod_{j=1}^{n} (P(s_j^i | c_{j-1}, c_j, c_{j+1}) \quad .$$





“what” **“you”**

Language models

P(W)

$p(w_1..w_n)=p(w_1)p(w_2|w_1) \ldots p(w_n|w_1..w_{n-1})$

- N-gram models
  - truncate $p(w_i|w_1.. w_{i-N+1} .. w_{i-1})$ to $p(w_i|w_{i-N+1}..w_{i-1})$
- 1-gram: $p(w_i)$
- 2-gram: $p(w_i|w_{i-1})$
- 3-gram: $p(w_i|w_{i-2}w_{i-1})$
- etc.

N-1

1 Gram

- To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
- Hill he late speaks; or! a more to leg less first you enter

2 Gram

- Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
- What means, sir. I confess she? then all sorts, he is trim, captain.

3 Gram

- Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done.
- This shall forbid it should be branded, if renown made it empty.

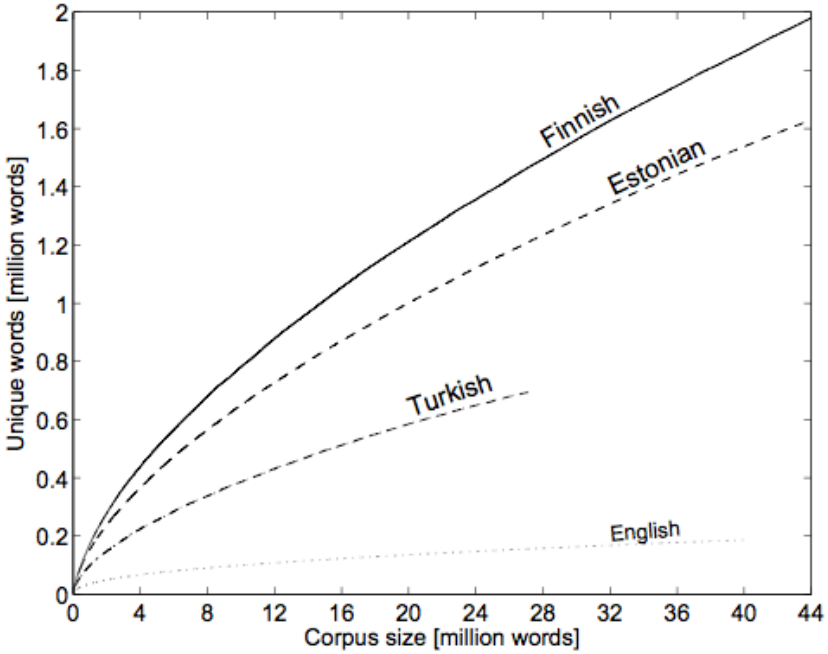
4 Gram

- King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in;
- It cannot be but so.

- estimating Ngram models from data
  - counts

$$P(w_i) = \frac{c_i}{N}$$

  - unknown words (OOVs
    - the lexicon is infinite!
    - a new corpus will always have new words
    - a fast and dirty solution: recoding with <UNK>



M. Creutz et al, “Morph-based speech recognition and modeling OOV words across languages”, ACM Trans Speech and Language Processing, 5(1), art. 3. doi: 0.1145/1322391.1322394

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- unknown ngrams
  - most ngrams have been seen 0 time
  - problem grows exponentially with n
  - true 0s versus sampling
  - solutions:
    - smoothing
    - backoff, interpolation, clustering

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

$$c_i^* = (c_i + 1)\frac{N}{N + V}$$

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &\quad + \lambda_2 P(w_n|w_{n-1}) \\ &\quad + \lambda_3 P(w_n)\end{aligned}$$

see Jurafsky & Martin 2017

- evaluating language models
  - extrinsic: improved WER
  - intrinsic: perplexity

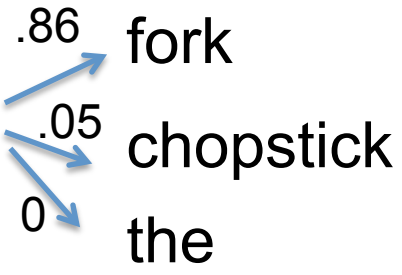
$$PP(W)=P(W)^{-1/N}$$

$$PP(W)= \Pi_1^N P(w_i|w_1..w_{i-1})^{-1/N}$$

$$PP(W)= 2^{H(W)}$$

with $H(W)=-1/N \log_2 P(W)$

He is eating a steak with a knife and a …



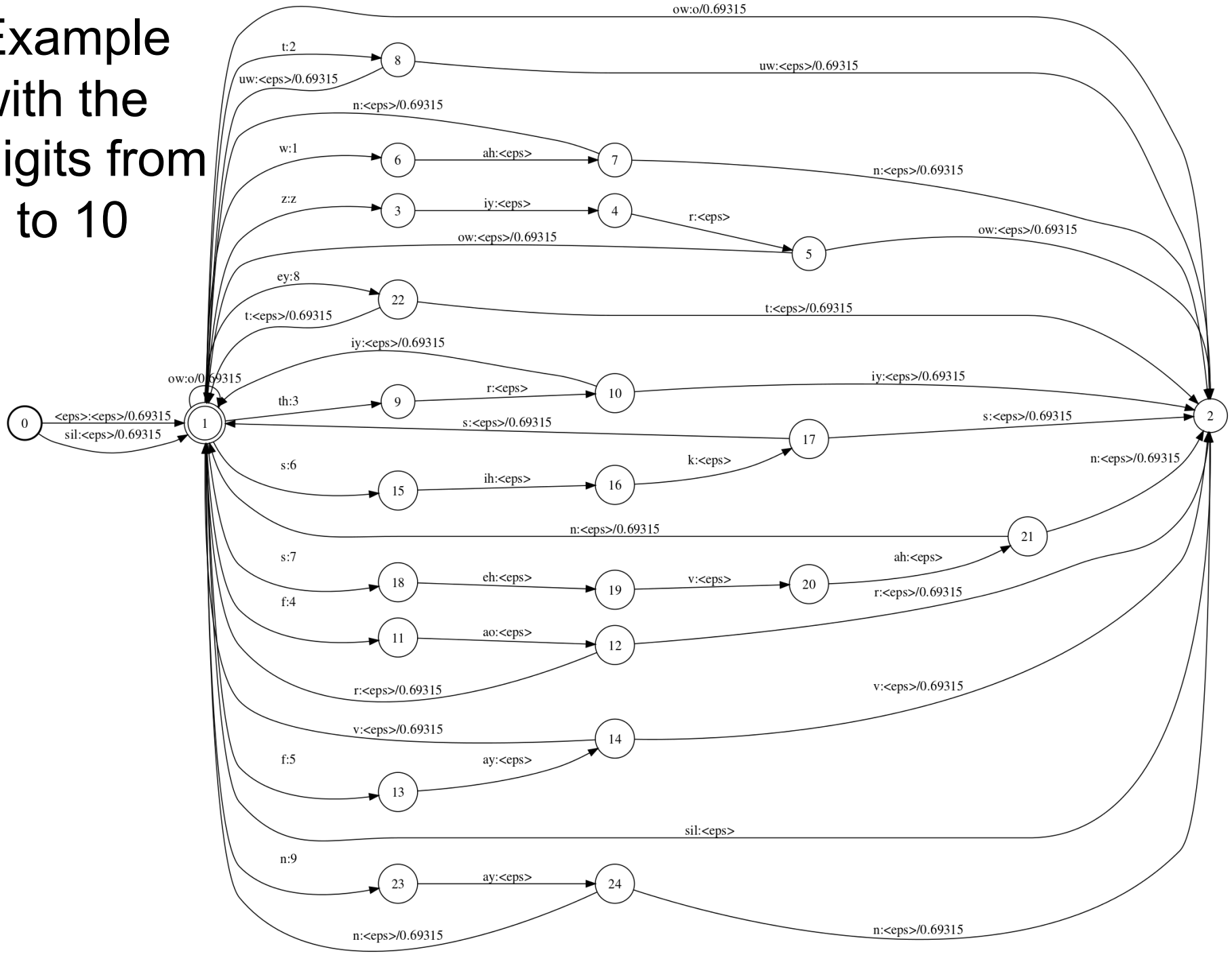
Putting all together

- acoustic model+speech input: a decoding graph
- pronunciation+language model: an fst

→ a gigantic decoding graph

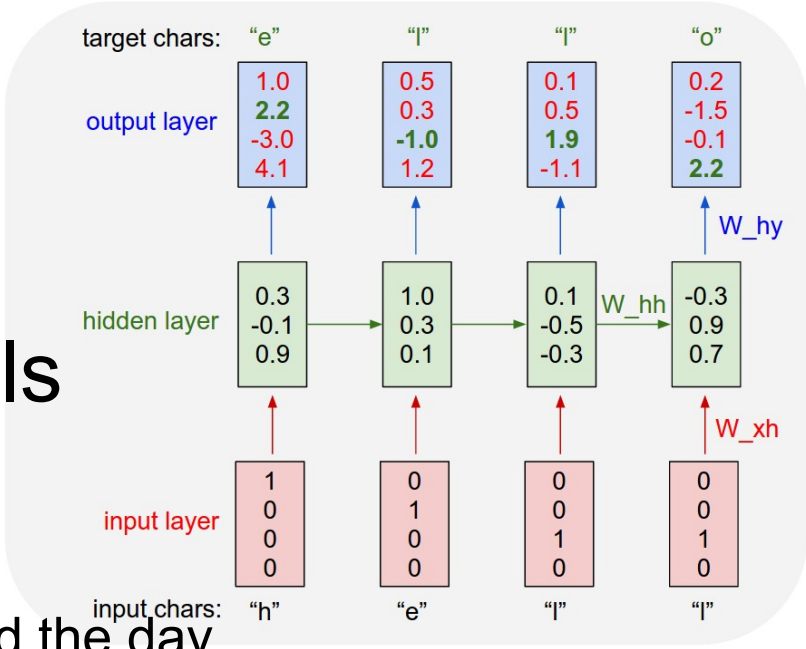
- in practice:
  - fst tools (see practice session)
  - two stage models: a first decoding lattice, followed by lattice rescoring with a large LM

Example with the digits from 0 to 10



an aside: RNN language models

- character language models



PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

- in practice word RNNs work better, but grow very large
- subword RNNs

The decoding problem

- $Q^* = \text{argmax}_Q\ p(O|Q)$
- $Q^* = \text{argmax}_Q\ p(Q|O).p(Q)$

A detour: Dynamic programming

1. define subproblems
2. identify recurence
3. set the initial conditions

From Jaehyun Park, 2015, Dynamic Programming Stanford Univ course materials

- Problem: given n, find the number of different ways to write n as the sum of 1, 3, 4
  - Example: for n = 5, the answer is 6

5 = 1+1+1+1+1

= 1+1+3

= 1+3+1

= 3+1+1

= 1+4

= 4+1

1. Define subproblems
   – Let D_n be the number of ways to write n as the sum of 1,3,4
2. Find the recurrence
   – Consider one possible solution $n=x_1+x_2+\cdots+x_m$
   – If x_m =1,the rest of the terms must sum to n−1
   – Thus, the number of sums that end with x_m = 1 is equal to D_{n-1}
   – if x_m=3, it is equal to D_{n-3}
   – if x_m=4, it is equal to D_{n-4}

   → $D_n = D_{n-1} + D_{n-3} + D_{n-4}$

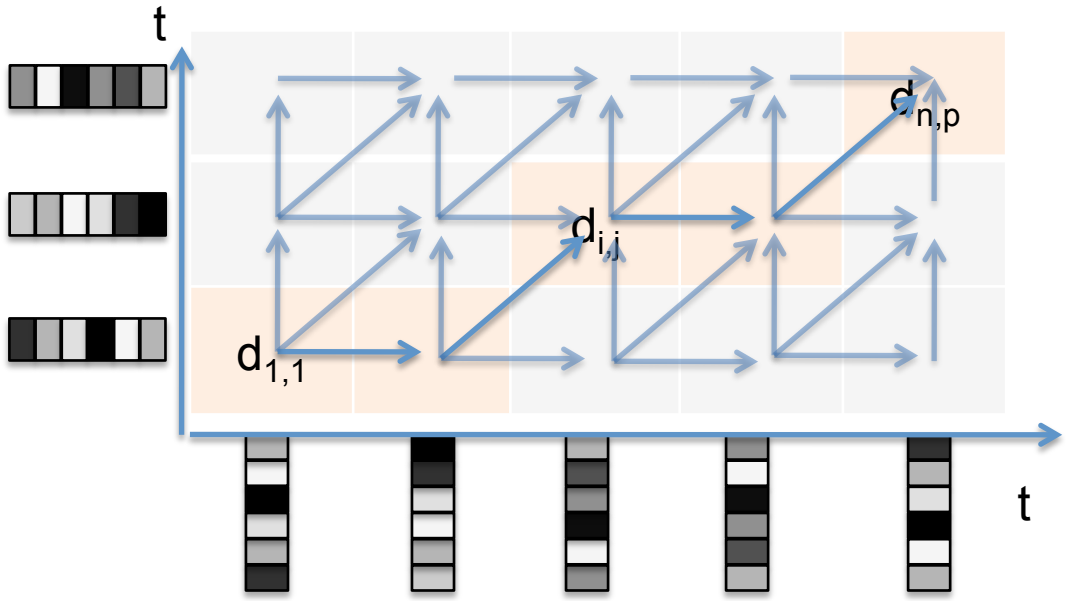
3. Set the initial conditions
   - $D_0 = 1$
   - $D_n = 0$ for all negative n

Done!

```
D[0] = D[1] = D[2] = 1
D[3] = 2
for(i = 4; i <= n; i++)
        D[i] = D[i-1] + D[i-3] + D[i-4]
```

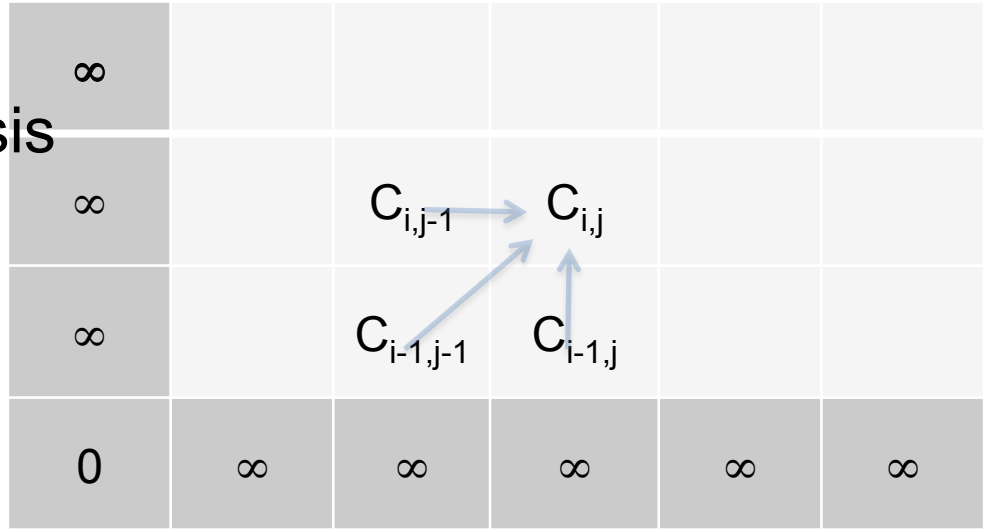
example: Dynamic Time Warping

Find the *path* which minimizes $\Sigma_{path}\ d_{i,j}$
(where the path can only go through increasing and adjacent cells)

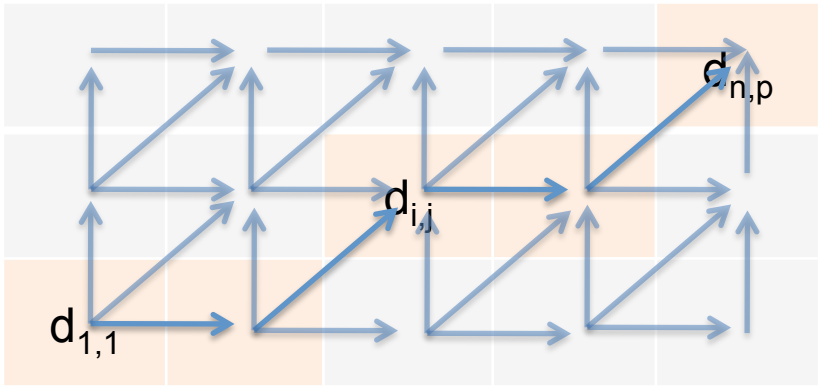


Dynamic programming analysis

subp: C_{ij}
recur: $C_{i,j}=d_{i,j}+\min(C_{i,j-1}, C_{i-1,j-1}, C_{i-1,j})$
init: $C_{0,0}=0$, $C_{i,0}=\infty$, $C_{0,j}=\infty$
$(i>0, j>0)$



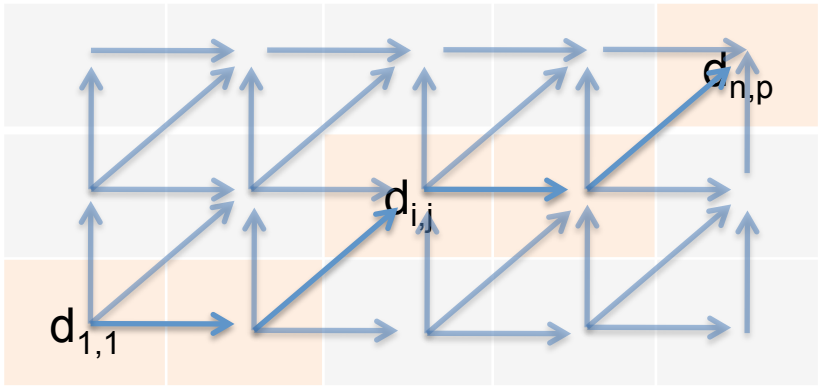
DTW algorithm



- Initialize

∞	0	0	0	0	0
∞	0	0	0	0	0
∞	0	0	0	0	0
0	∞	∞	∞	∞	∞

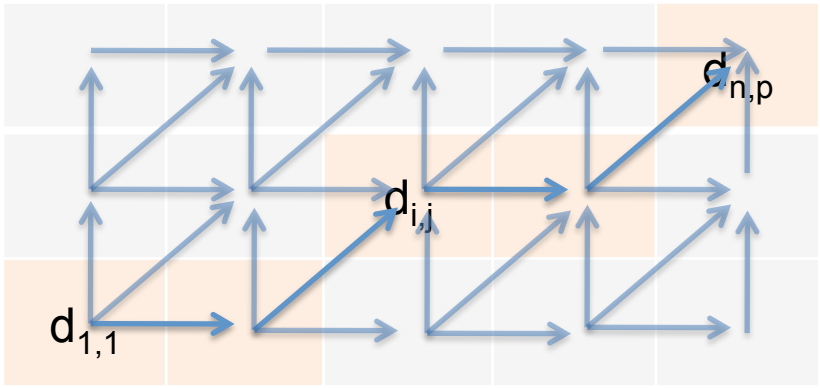
DTW algorithm



- Apply the recurrence formula $C_{i,j}=d_{i,j}+\min(C_{i,j-1}, C_{i-1,j-1}, C_{i-1,j})$
- Leave a pointer to which cell you are coming from

∞	0	0	0	0	0
∞	0	0	0	0	0
∞	$d_{1,1}$	0	0	0	0
0	∞	∞	∞	∞	∞

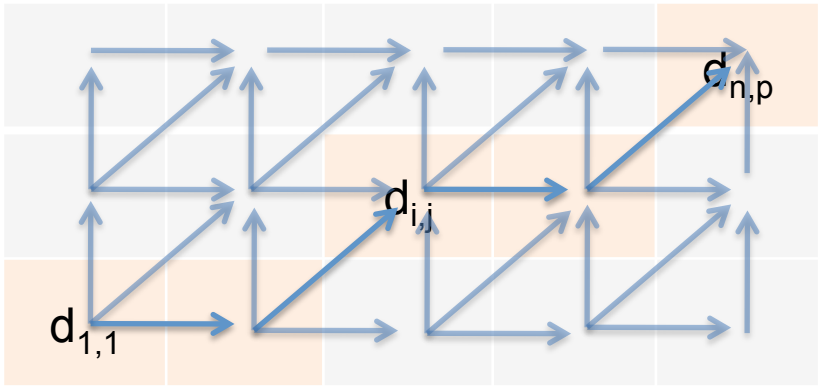
DTW algorithm



- Apply the recurrence formula
  $C_{i,j}=d_{i,j}+\min(C_{i,j-1}, C_{i-1,j-1}, C_{i-1,j})$
- Leave a pointer to which cell you are coming from

∞	0	0	0	0	0
∞	$d_{1,1}$ $+d_{2,1}$	0	0	0	0
∞	$d_{1,1}$	0	0	0	0
0	∞	∞	∞	∞	∞

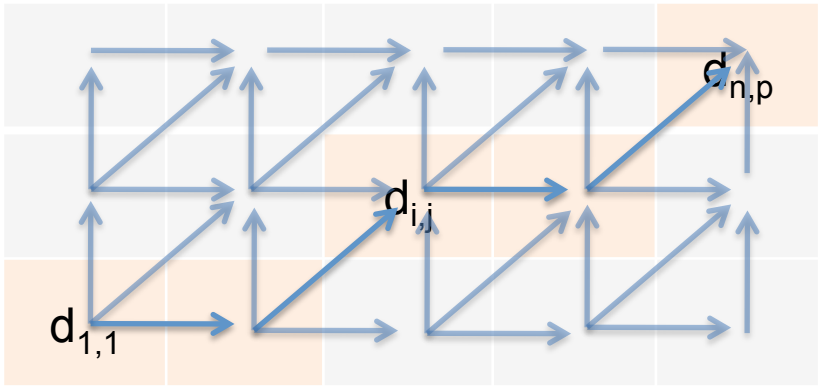
DTW algorithm



- Apply the recurrence formula
  $C_{i,j}=d_{i,j}+\min(C_{i,j-1}, C_{i-1,j-1}, C_{i-1,j})$
- Leave a pointer to which cell you are coming from

∞	$d_{1,1}$ $+d_{2,1}$ $+d_{3,1}$	0	0	0	0
∞	$d_{1,1}$ $+d_{2,1}$	0	0	0	0
∞	$d_{1,1}$	0	0	0	0
0	∞	∞	∞	∞	∞

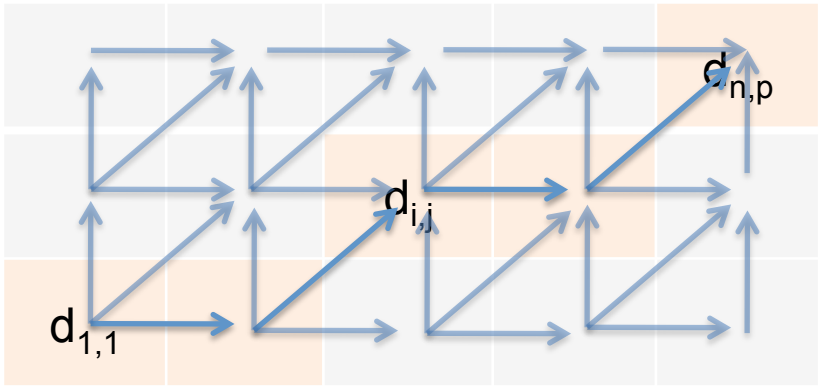
DTW algorithm



- Apply the recurrence formula
  $C_{i,j}=d_{i,j}+\min(C_{i,j-1}, C_{i-1,j-1}, C_{i-1,j})$
- Leave a link to which cell you are coming from

∞	$d_{1,1}$ $+d_{2,1}$ $+d_{3,1}$	0	0	0	0
∞	$d_{1,1}$ $+d_{2,1}$	0	0	0	0
∞	$d_{1,1}$	$d_{1,1}$ $+d_{1,2}$	0	0	0
0	∞	∞	∞	∞	∞

DTW algorithm

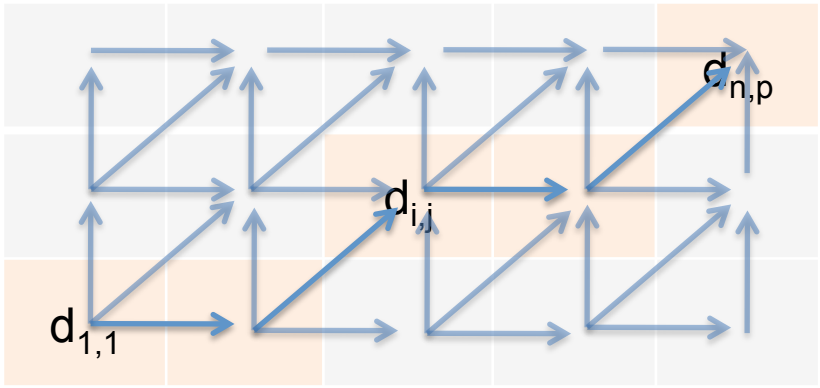


etc…

∞	$d_{1,1}$ $+d_{2,1}$ $+d_{3,1}$	$d_{1,1}$ $+d_{2,1}$ $+d_{3,2}$	$d_{1,1}+d_{2,1}$ $+d_{2,2}$ $+d_{3,3}$	$d_{1,1}+d_{2,1}$ $+d_{2,2}+d_{3,3}$ $+d_{3,4}$	$d_{1,1}+d_{1,2}$ $+d_{2,3}+d_{2,4}$ $+d_{3,5}$
∞	$d_{1,1}$ $+d_{2,1}$	$d_{1,1}+d_{2,1}$ $+d_{2,2}$	$d_{1,1}+d_{1,2}$ $+d_{2,3}$	$d_{1,1}+d_{1,2}$ $+d_{2,3}+d_{2,4}$	$d_{1,1}+d_{1,2}$ $+d_{2,3}+d_{2,4}$ $+d_{2,5}$
∞	$d_{1,1}$	$d_{1,1}$ $+d_{1,2}$	$d_{1,1}$ $+d_{1,2}$ $+d_{1,3}$	$d_{1,1}+d_{1,2}$ $+d_{1,3}$ $+d_{1,4}$	$d_{1,1}+d_{1,2}$ $+d_{1,3}+d_{1,4}$ $+d_{1,5}$
0	∞	∞	∞	∞	∞

DTW algorithm

- Start from the final point
- Follow the path backward

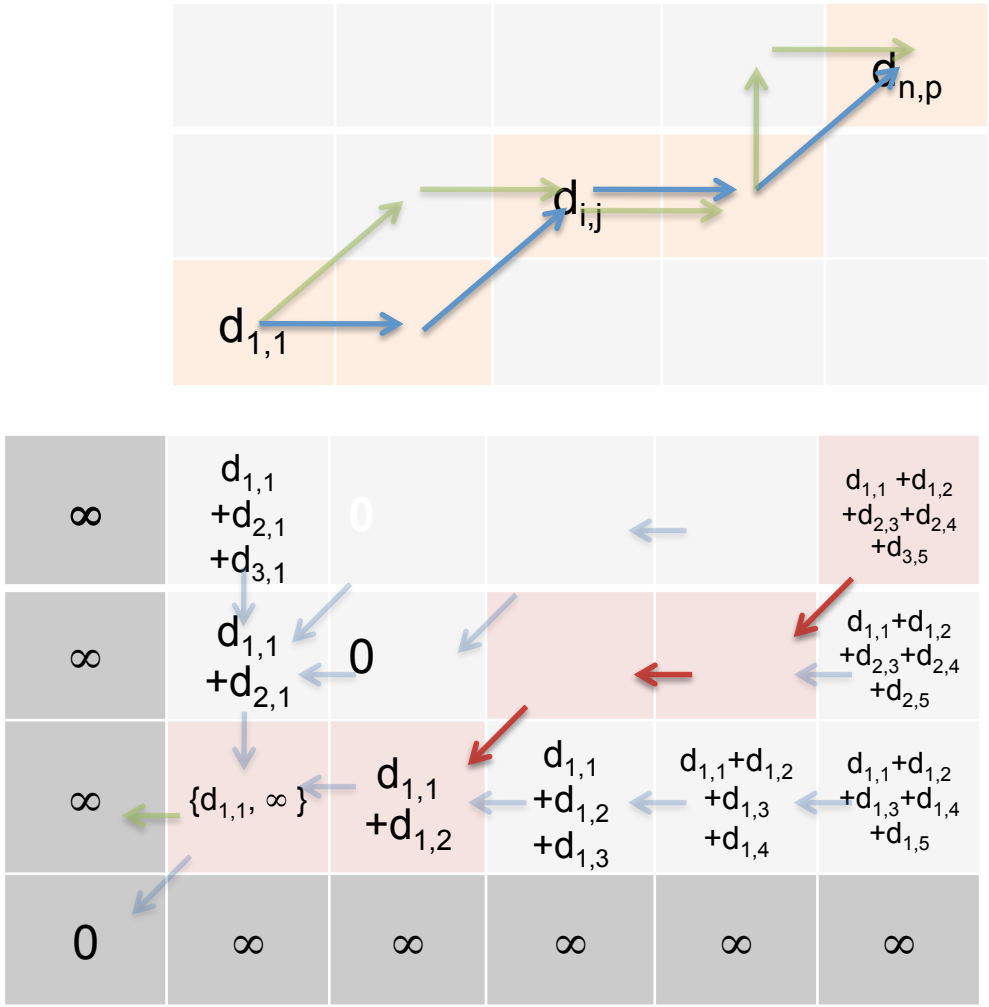


∞	$d_{1,1}$ $+d_{2,1}$ $+d_{3,1}$	$d_{1,1}$ $+d_{2,1}$ $+d_{3,2}$	$d_{1,1}+d_{2,1}$ $+d_{2,2}$ $+d_{3,3}$	$d_{1,1}+d_{2,1}$ $+d_{2,2}+d_{3,3}$ $+d_{3,4}$	$d_{1,1}+d_{1,2}$ $+d_{2,3}+d_{2,4}$ $+d_{3,5}$
∞	$d_{1,1}$ $+d_{2,1}$	$d_{1,1}+d_{2,1}$ $+d_{2,2}$	$d_{1,1}+d_{1,2}$ $+d_{2,3}$	$d_{1,1}+d_{1,2}$ $+d_{2,3}+d_{2,4}$	$d_{1,1}+d_{1,2}$ $+d_{2,3}+d_{2,4}$ $+d_{2,5}$
∞	$d_{1,1}$	$d_{1,1}$ $+d_{1,2}$	$d_{1,1}$ $+d_{1,2}$ $+d_{1,3}$	$d_{1,1}+d_{1,2}$ $+d_{1,3}$ $+d_{1,4}$	$d_{1,1}+d_{1,2}$ $+d_{1,3}+d_{1,4}$ $+d_{1,5}$
0	∞	∞	∞	∞	∞

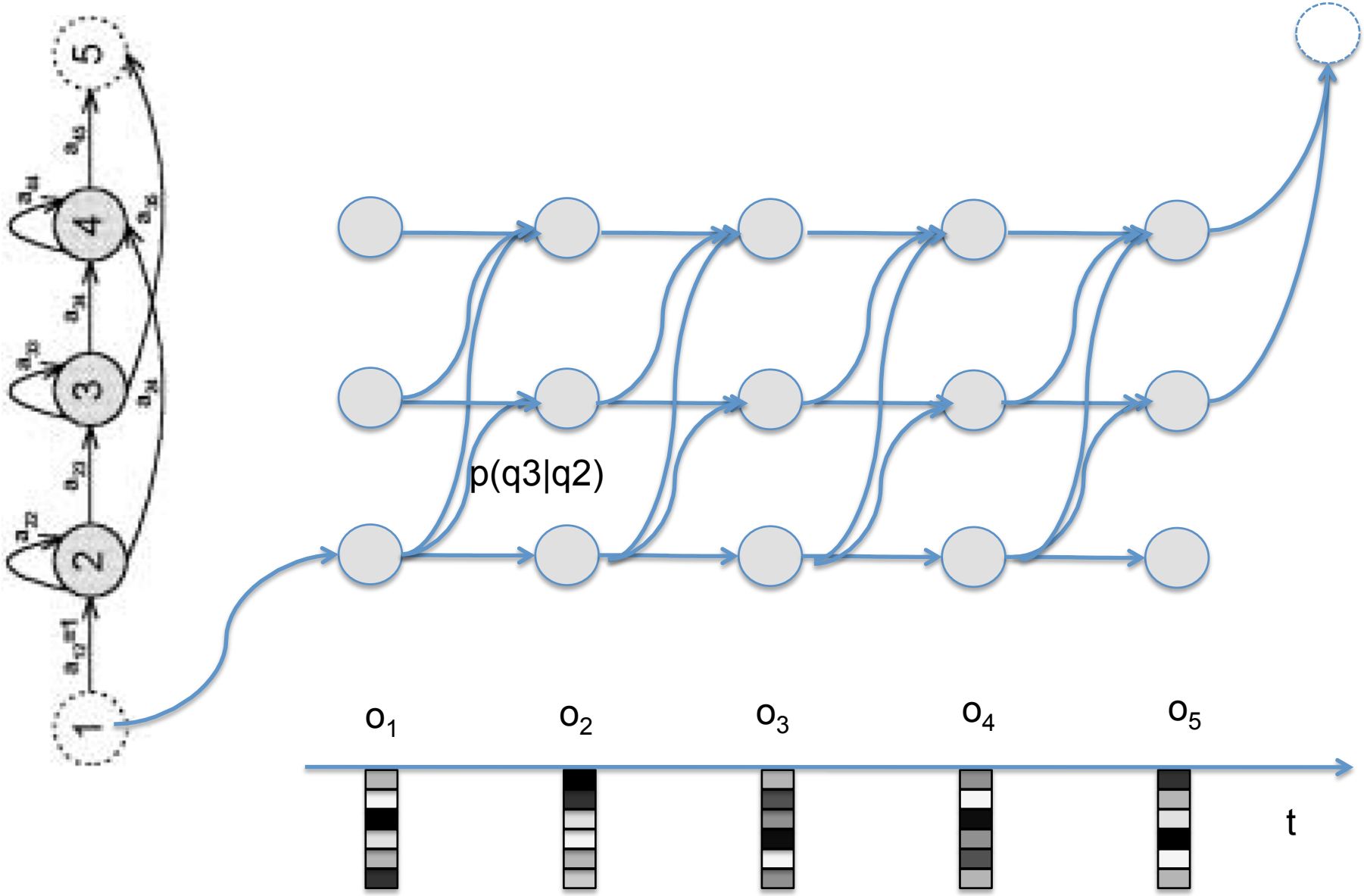
Variation: nbest

- Change the recurrence formula $C_{i,j}=d_{i,j}+best(n,C_{i,j-1} \cup C_{i-1,j-1} \cup C_{i-1,j})$ where best(n,S)=the n smallests elements in set S.

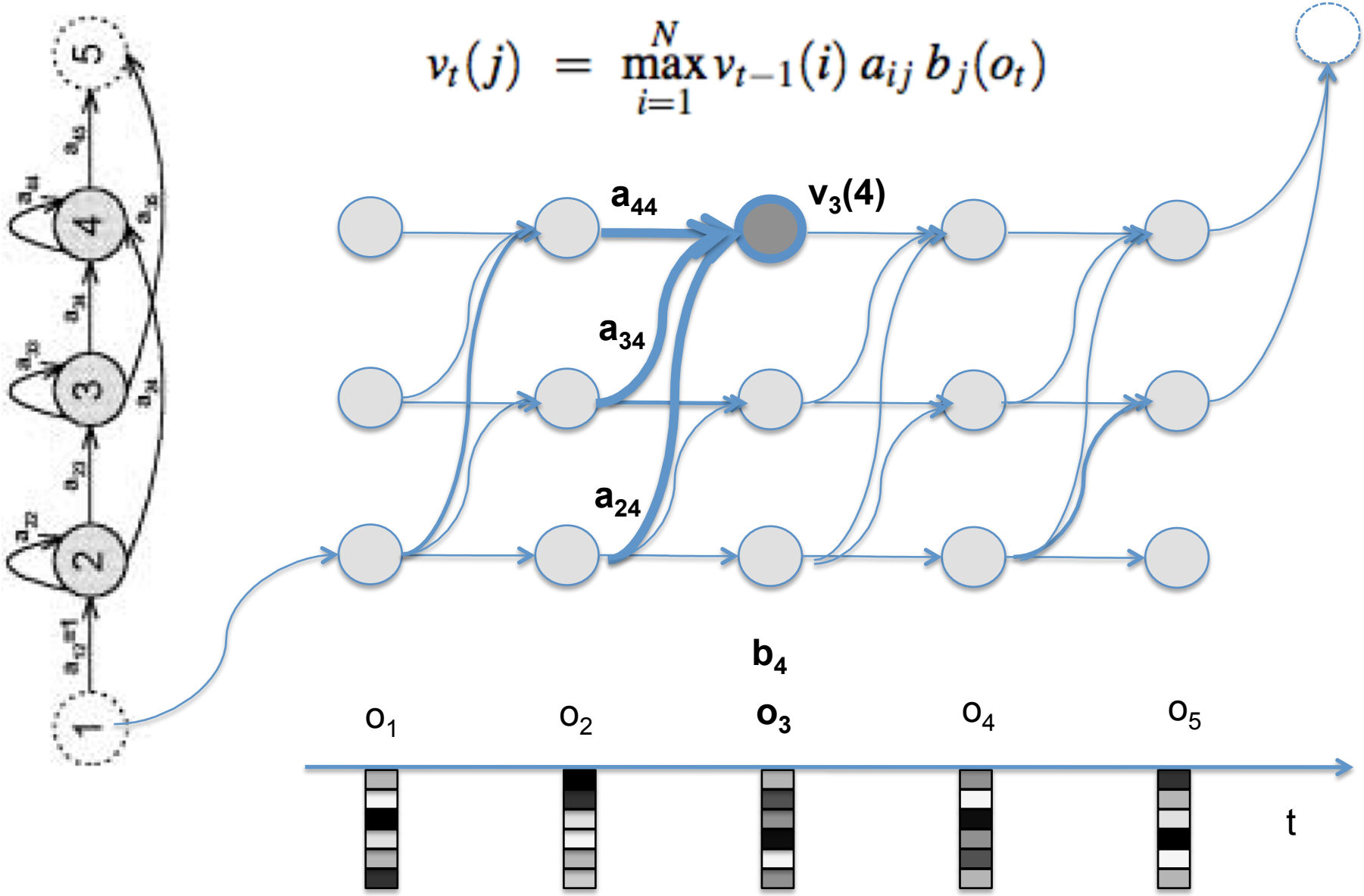
- Memorize n links backward



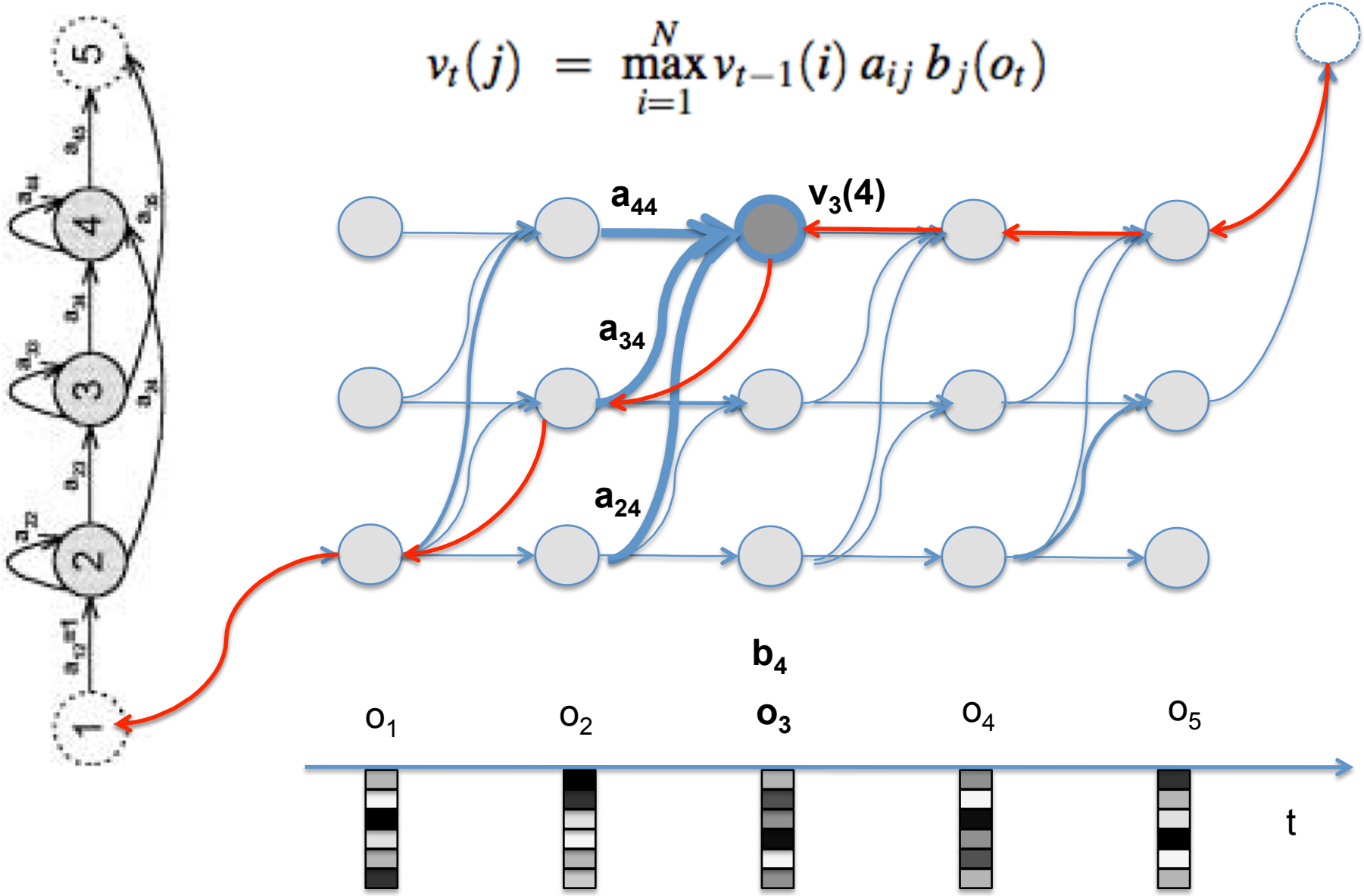
Back to HMMs



decoding: $Q^* = \text{argmax}_Q P(O|Q)$



decoding: $Q^* = \text{argmax}_Q P(O|Q)$

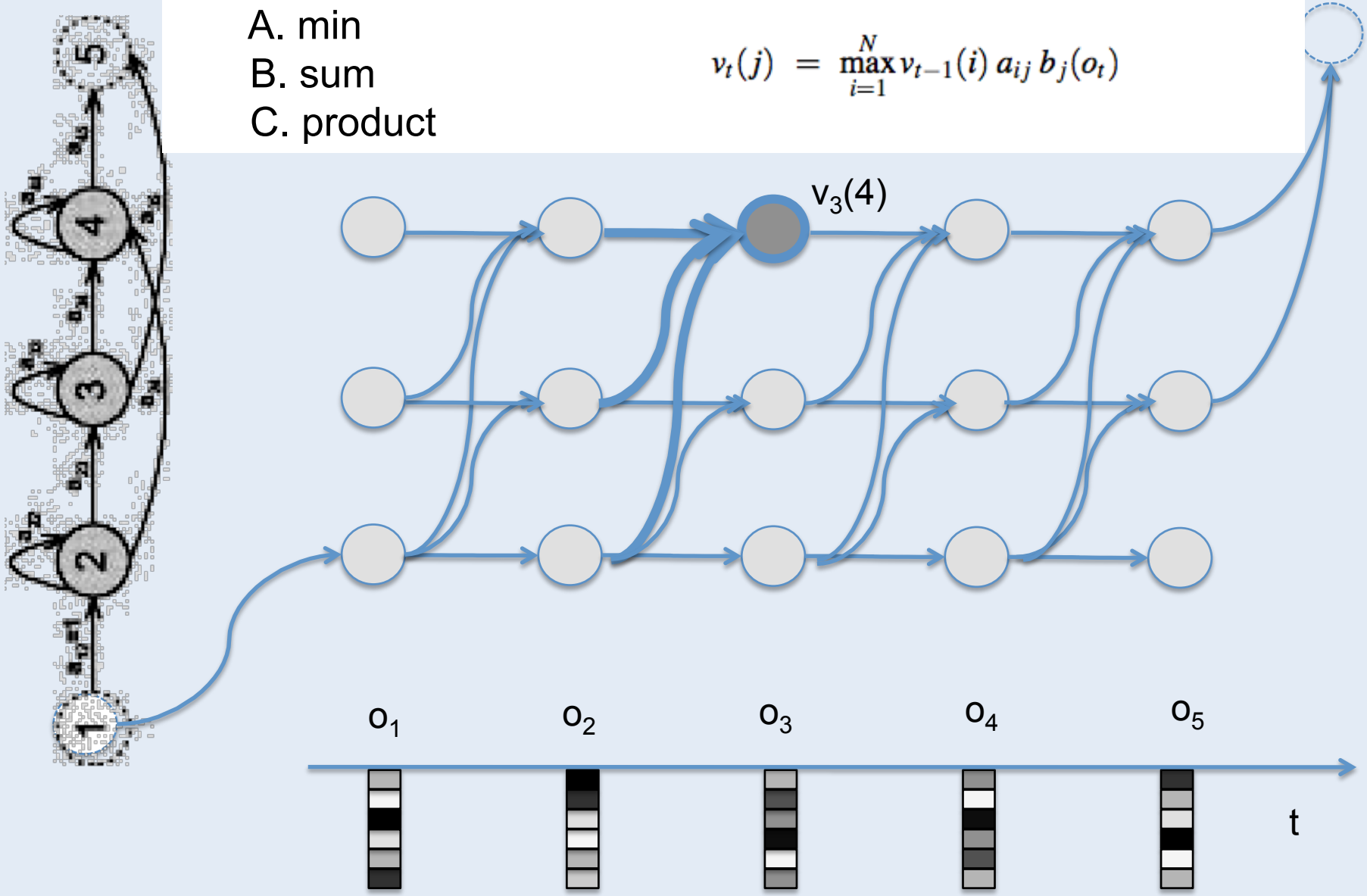


Question 1

To compute the likelihood P(O|Θ), should we replace **max** by:

A. min
B. sum
C. product

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$



Question 2

Is it possible to use dynamic programming to decode with an RNN language model?

A. yes
B. no
C. don’t know

Learning

Learning: Parameter estimation

- $\Theta^* = \arg\max_{\Theta} p(O|\Theta)$
- EM (Baum Welch, forward backward) algorithm:
  - E step: estimate distribution of hidden variable Q (given O and Θ): $p(Q|O,\Theta)$
  - M step: maximize Θ given the estimate of Q:

    $\Theta^* = \arg\max_{\Theta} p(O|Q,\Theta)$
  - Iterate

an example: GMMs

- mixture model

$$p(x) = \sum_{j=1}^{K} w_j \cdot N(x \mid \mu_j, \Sigma_j)$$

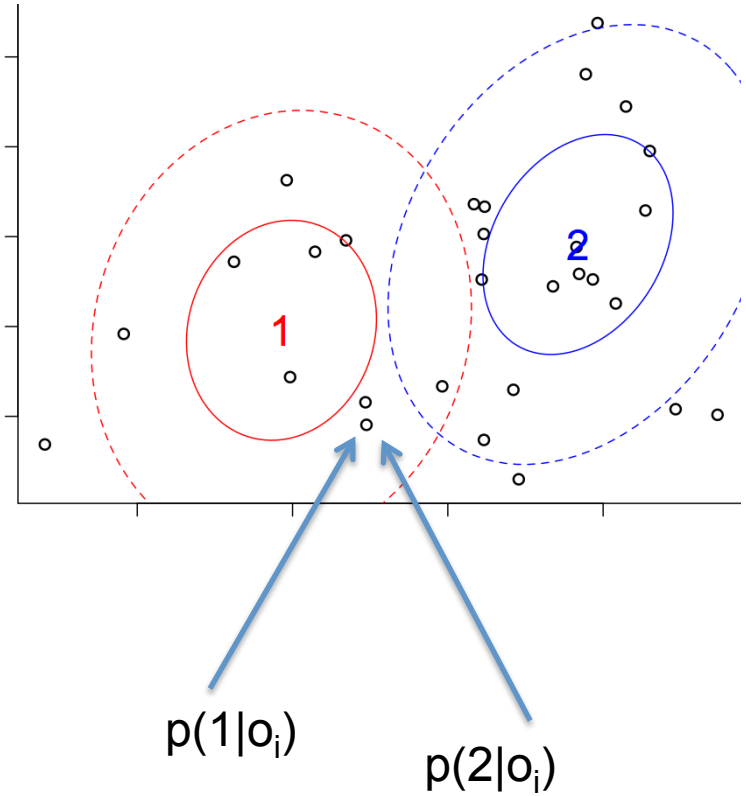
$\theta=\{\mu_i, \Sigma_i\}$

- problem: given observations $O=\{o_1..o_N\}$

$\theta^*=\arg\max_\theta \Pi_i\, p(o_i|\theta)$

- introduce hidden cluster membership $q_i=1..k$

$\theta^*=\arg\max_\theta \Pi_i\, p(o_i,q_i|\theta)$



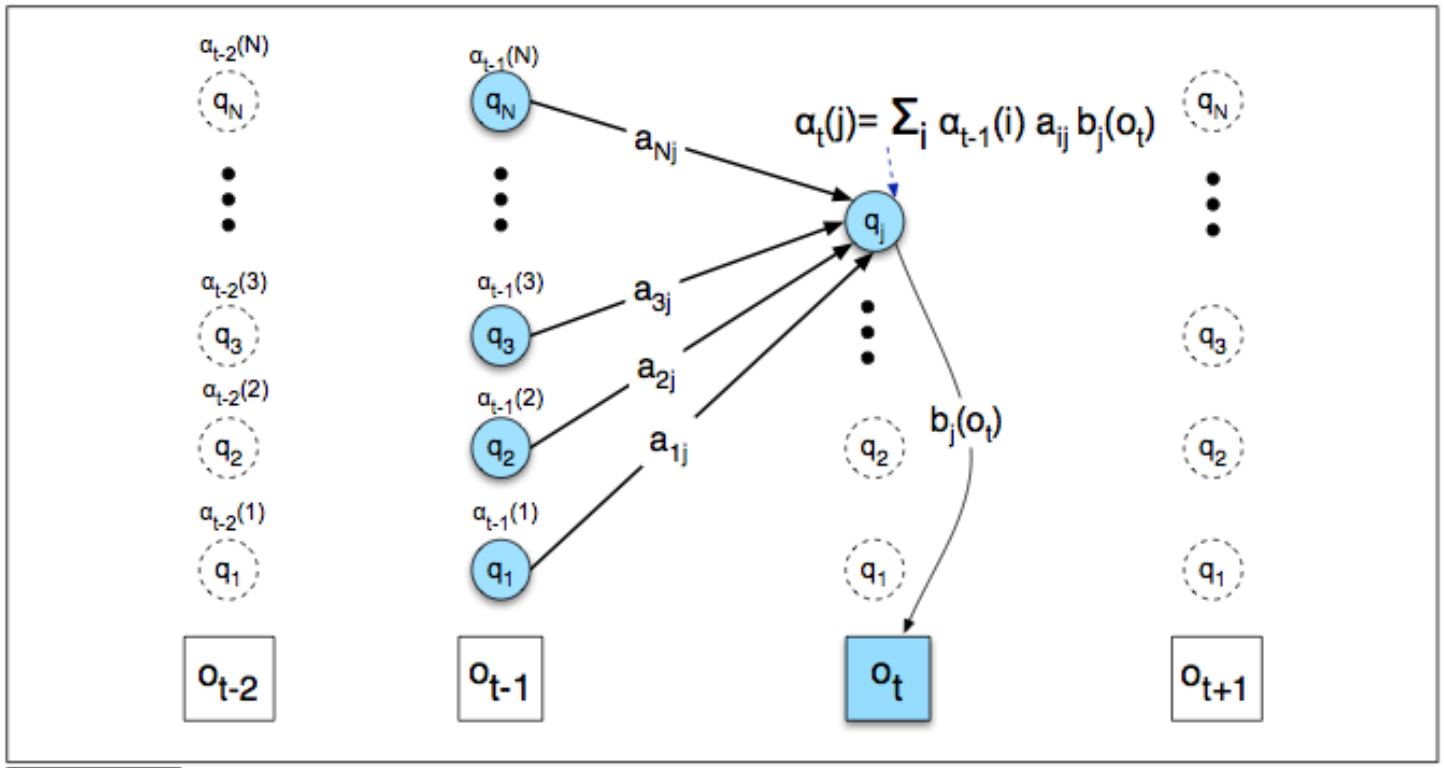
$$\mu_j = \frac{\sum_{i=1}^{N} p(j \mid o_i, \theta^t) o_i}{\sum_{i=1}^{N} p(j \mid o_i, \theta^t)}$$

idem for Σ_j

going back to HMMs

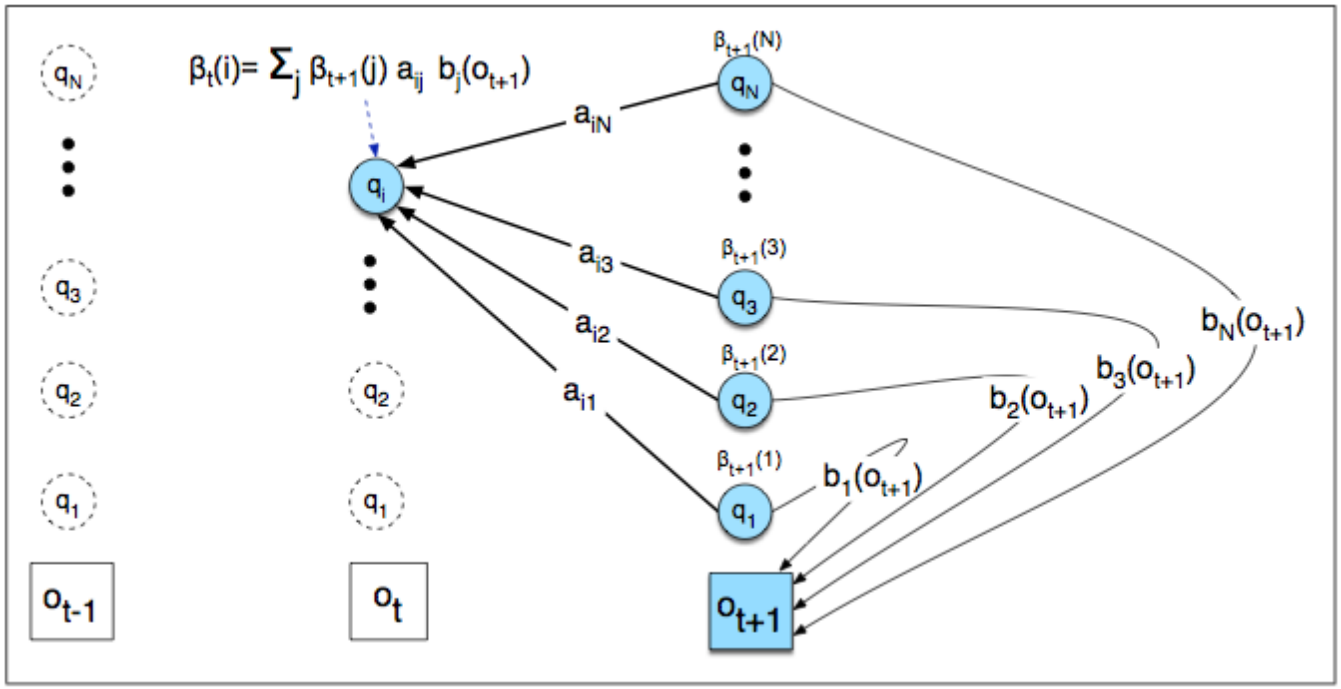
- intuition
  - in a markov chain: $$a_{ij} = \frac{C(i \rightarrow j)}{\sum_{q \in Q} C(i \rightarrow q)}$$
  - in an HMM: iteratively estimate the counts
    - start with an estimate a_{ij} and b_j, use them to decode and get better estimates
    - compute these estimates on all paths, weighted by the probability of the paths

- forward probabilities $\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\Theta)$

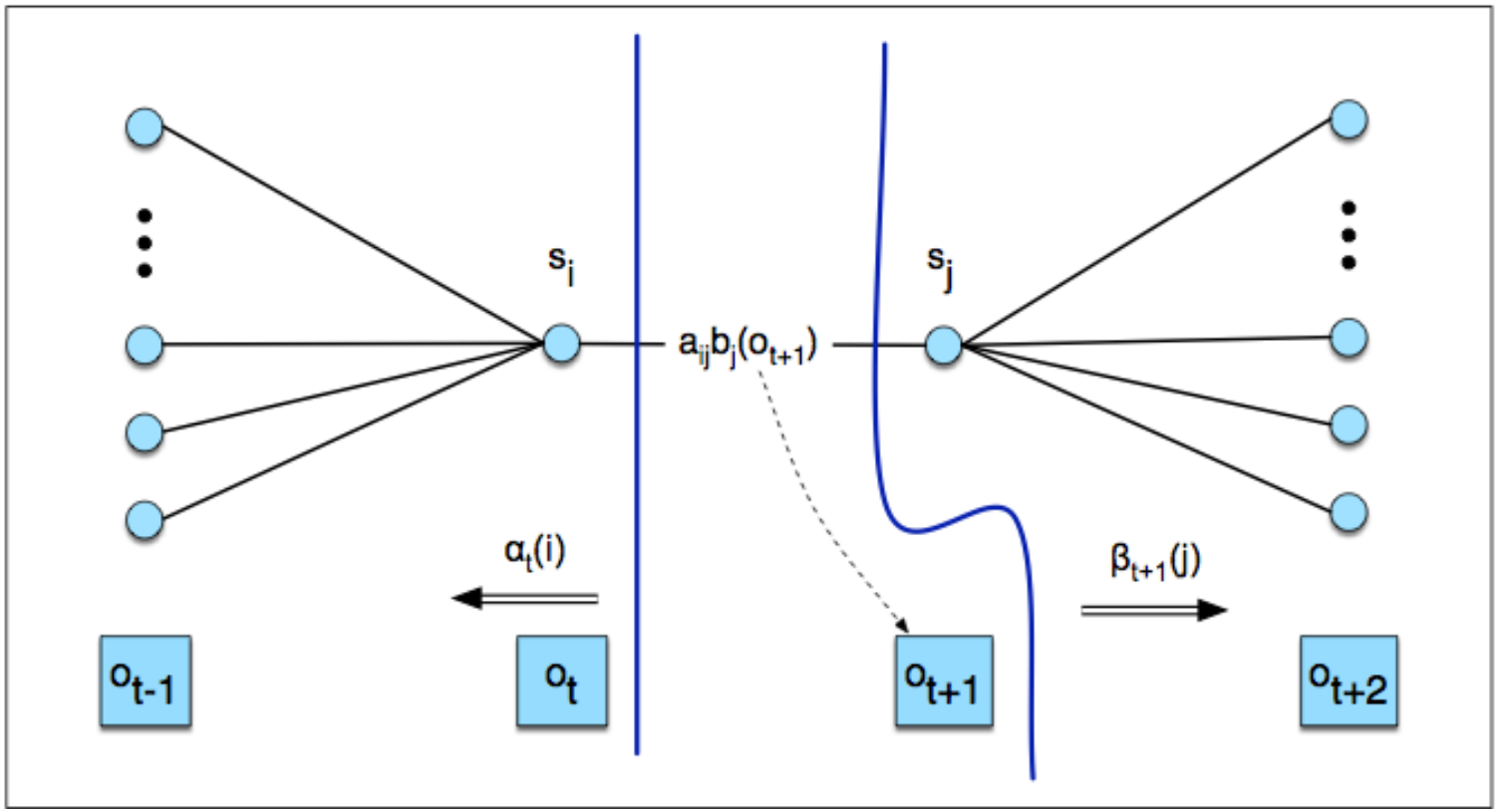


from J&M (2017)

- backward probabilities $\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots o_T | q_t = i, \Theta)$



from J&M (2017)



$$P(q_t = i, q_{t+1} = j, O|\Theta) = \alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j|O,\Theta) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)}$$

from J&M (2017)

Baum-Welch algorithm

- E step

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)} \qquad \gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)}$$

- M Step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)} \qquad \hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

References

- Language models:
  - Jurasky & Martin (2017). Chapter 4. Language modeling with N-grams
- Dynamic programming:
  - Jaehyun Park (2015). Course materials for CS 97SI, Stanford University
- Viterbi decoding, Baum Welch
  - Jurafsky & Martin (2017). Chapter 9. Hidden Markov Models
- to know more
  - Gales & Young (2007). The application of Hidden Markow Models in Speech Recognition. *Foundations and Trends in Signal Processing*, **1(3)**, 195-304.