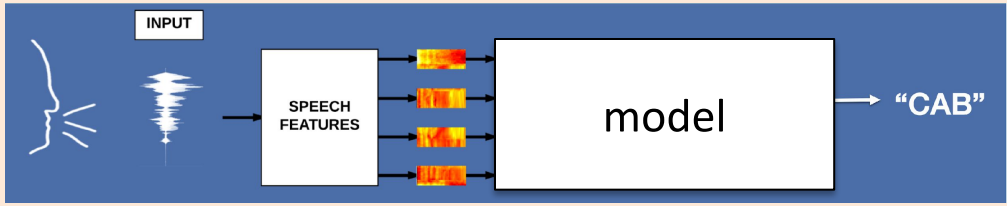
Algorithms for speech and language processing

ASR #2: Language Modeling

Course

I. Generative Models
II. Decoding
III. Learning



X Θ W

Notations

θ: model parameters

$X=x_1 \ldots x_t$: observed speech frames

$W=w_1 .. w_n$: word transcriptions

Decoding: given observed X and param. θ, find the most likely W.

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

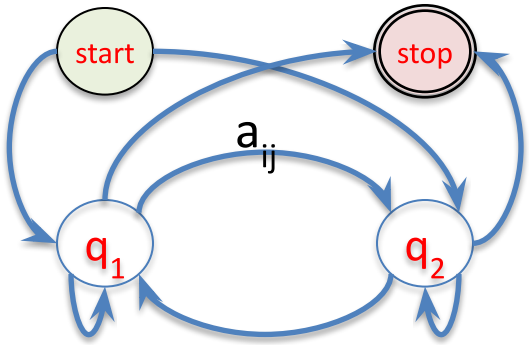
Learning: given an observed X and model, find the most likely θ

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

Part I: Generative Models

- Useful finite state models (recap)
- Model decomposition
  - acoustic model
  - pronunciation model
  - language model
- Putting it all together
- Alternative models (RNNs)

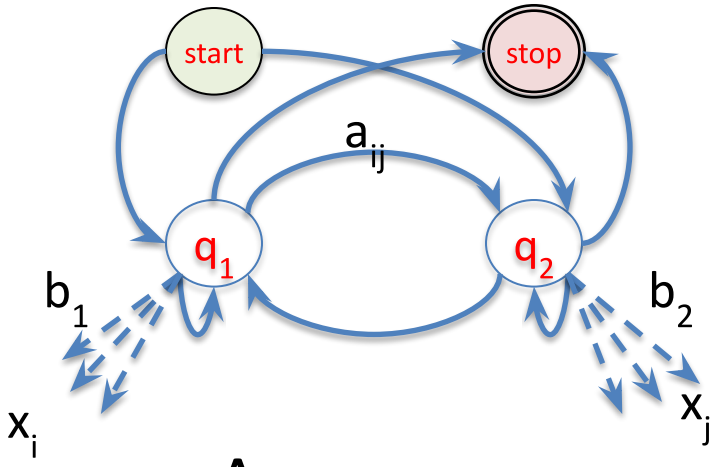
markov chains



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

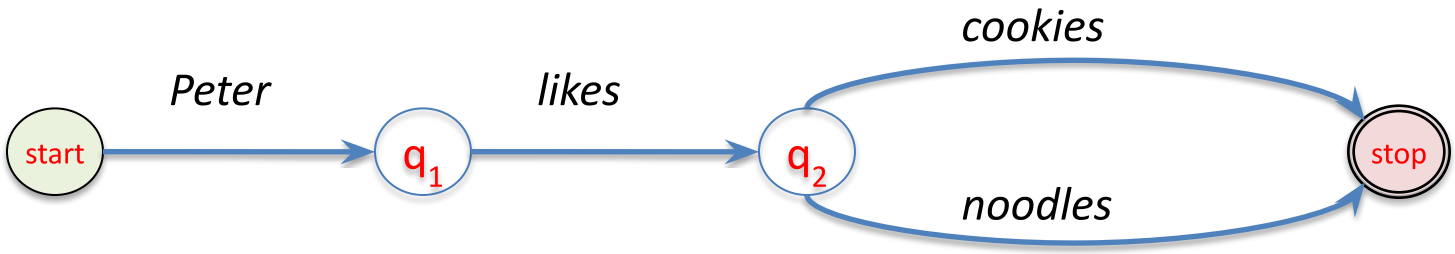
	q_1	q_2
x_1	b_{11}	b_{21}
x_2	b_{12}	a_{22}
x_3	b_{13}	a_{23}

A

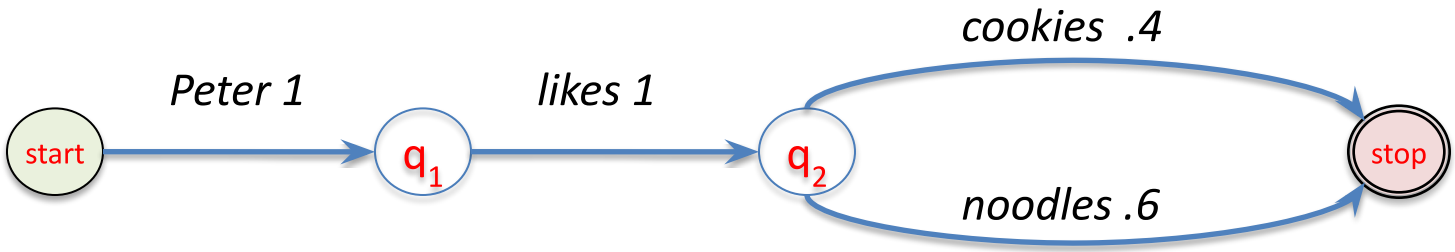
	q_0 (start)	q_1	q_2	q_F (stop)
q_0 (start)	0	a_{01}	a_{02}	0
q_1	0	a_{11}	a_{12}	a_{1F}
q_2	0	a_{21}	a_{22}	a_{2F}
q_F (stop)	0	0	0	0

(weighted) finite state transducers

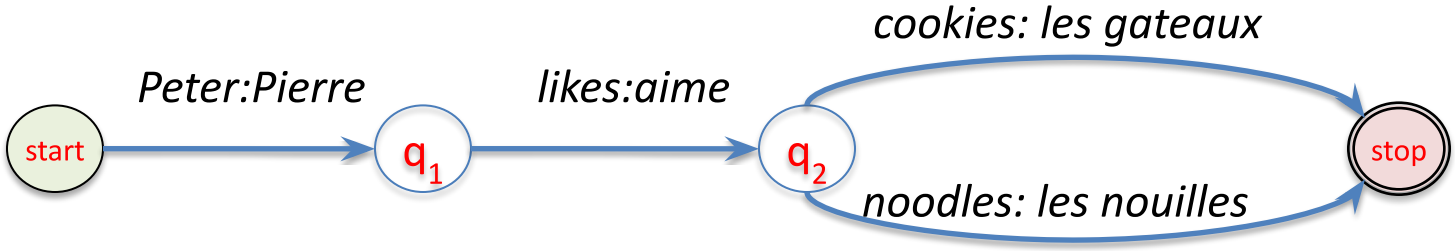
Finite State Acceptor (FSA)



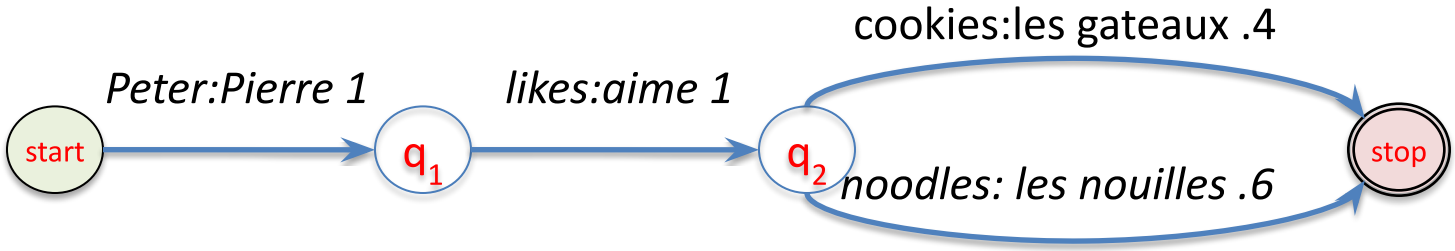
Weighted Finite State Acceptor (WFSA)



Finite State Transducer (FST)

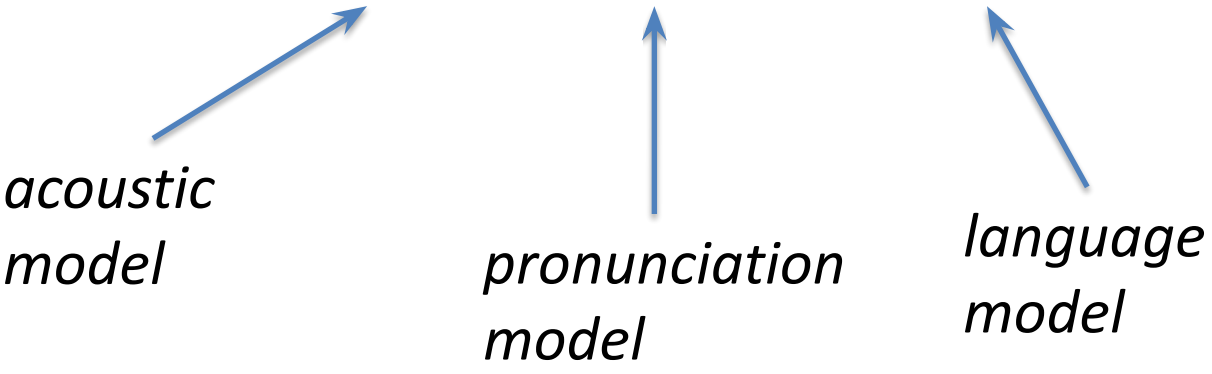


Weighted Finite State Transducer (WFST)



Model decomposition

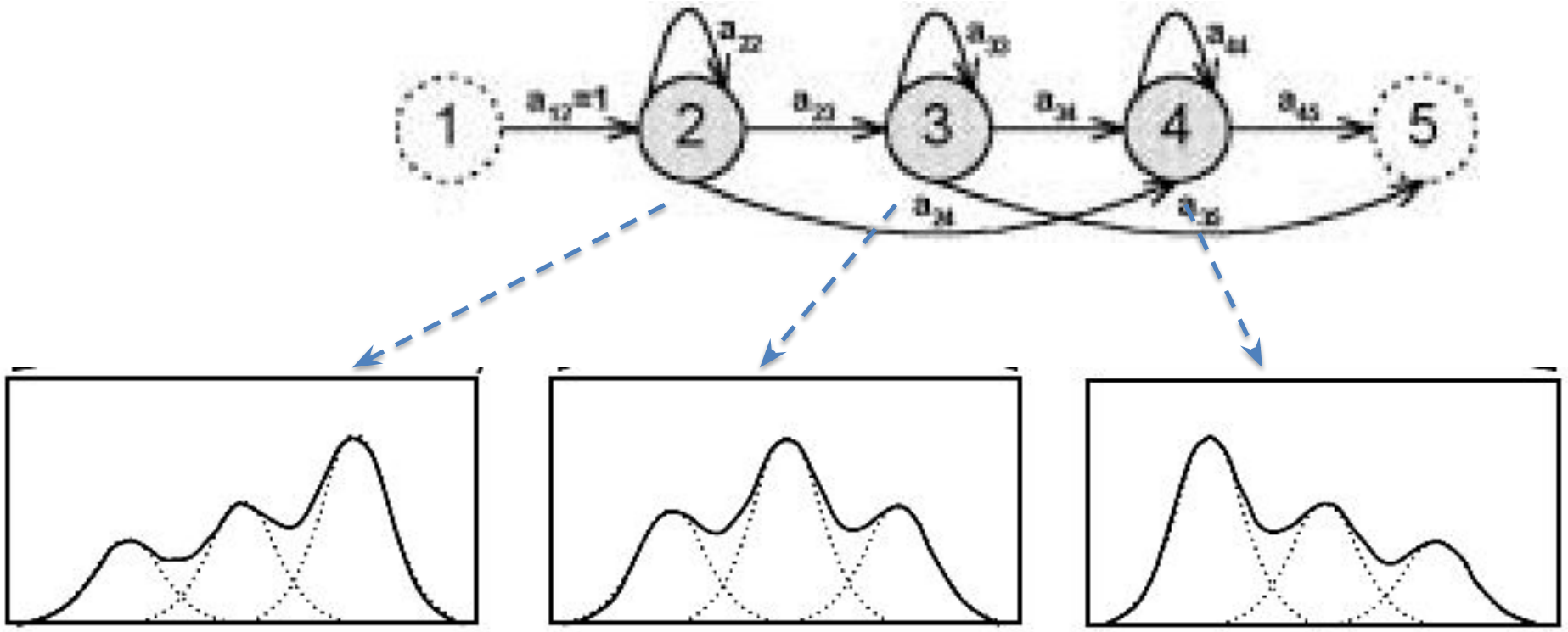
- $W^* = \arg\max_W P_\theta(W|X)$
- $P_\theta(W|X) \ \alpha \ P_\theta(X|W)P_\theta(W)$
- $= P_\theta(X|Q)P_\theta(Q|W)P_\theta(W)$



Acoustic models

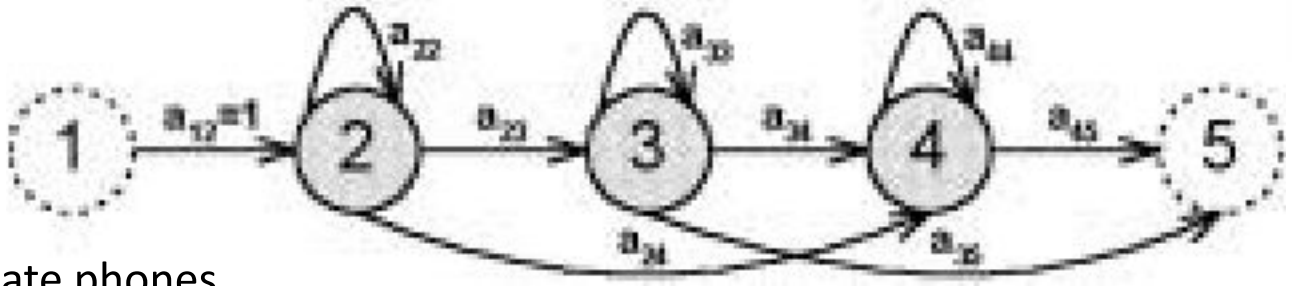
P(X|Q)

- tri-state phones

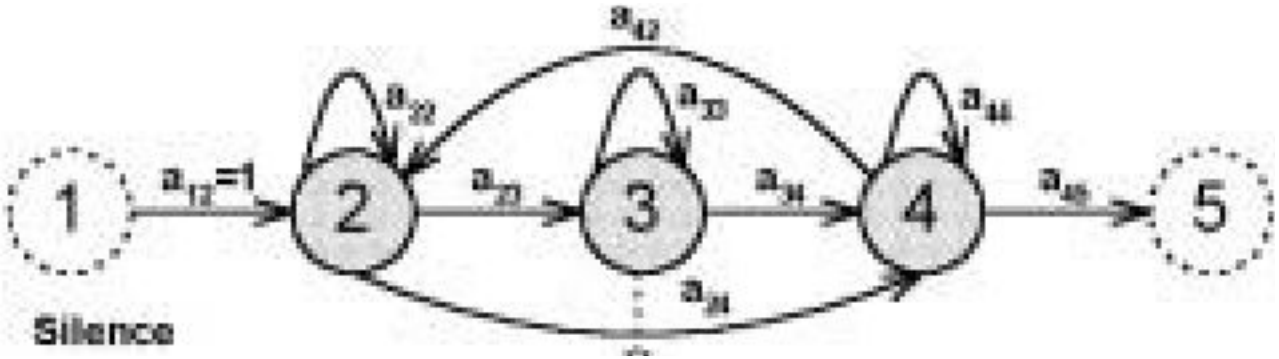


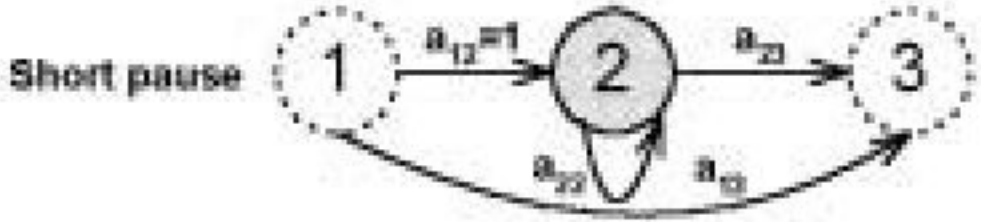
Acoustic models

P(X|Q)



tristate phones





Pronunciation models

P(Q|W)

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

ɨˌlɛˀvə̆n
ɨˌlɛvə̆n
ɨˌɬɛvə̆n
ə'lɛˀvə̆n
ɨˌlɛvə̆n
əlɛvə̆n
ə'ɬɛvn
əɬɛbm
ɨˌ'lɛvn
lɛvə̆n
əɬɛˬˀᵊvə̆n
ɬɛvɱ
əɬɛᵊvə̆n
ɨ'lɛvɱ
…
(total= 220 variants)

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

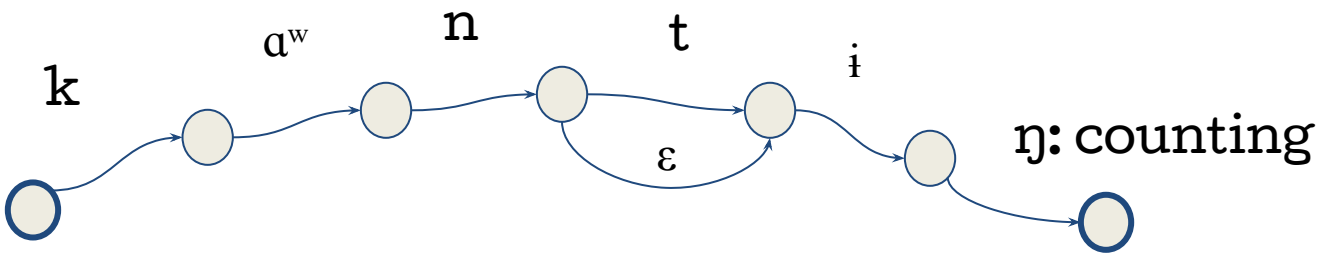
Phonological/Phonetic transformations

What are you doing? ˈwʌtʃəˈduɪn
I can inquire. ˈaɪkŋ̩ˈkwaɪəɹ
Did you eat yet? ˈdʒitjɛʔ
I don’t believe him. aɪˈdoʊ̃bəˈlivɱ
We ought to have come. wiˈɔɾ̪ɣ̩ˈkʌm

Pronunciation models

P(Q|W)

orthography	SAMPA	IPA
counting	kA_wntIN	kɑʷntɨŋ
counting(1)	kA_wnIN	kɑʷnɨŋ
amortization	@mOrt@zeS@n	əmɔrtəzeʃən
amortization(1)	{mOrt@zeS@n	æmɔrtəzeʃən
amortization(2)	@mOrtA_jzeS@n	əmɔrtɑʲzeʃən
..etc		



Problem: what about OOVs (out-of-vocabulary)?

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]

ចាស ⇒ c aː h

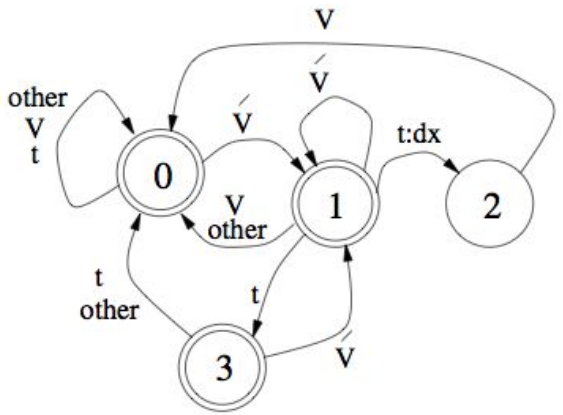
ប្រយ៉ុត្ ⇒ b r ɑː j u t

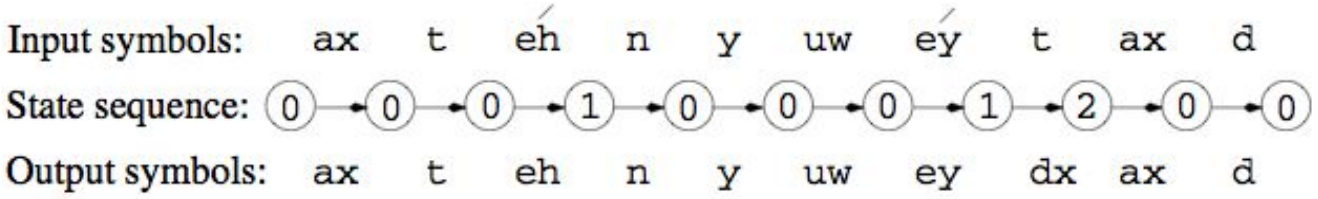
Problem: what about pronunciations that are not listed in the dictionary?

3. phonological rules (hand crafted)

- ex: english flapping

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





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**ial**	*n/a*
Nasal-deletion	[n m ng] → Ø/ ___ [-voice -consonant]	ran**t**	*n/a*
Function words			
h-deletion	h → Ø/ # ___	**he**, **him**	*n/a*
w-deletion	w → Ø/ # ___	**will**, **would**	*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]	cam**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**ts**	*n/a*

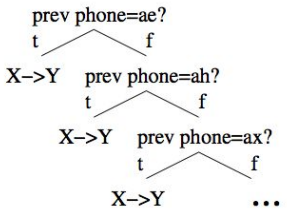
4. phonological rules (data driven)

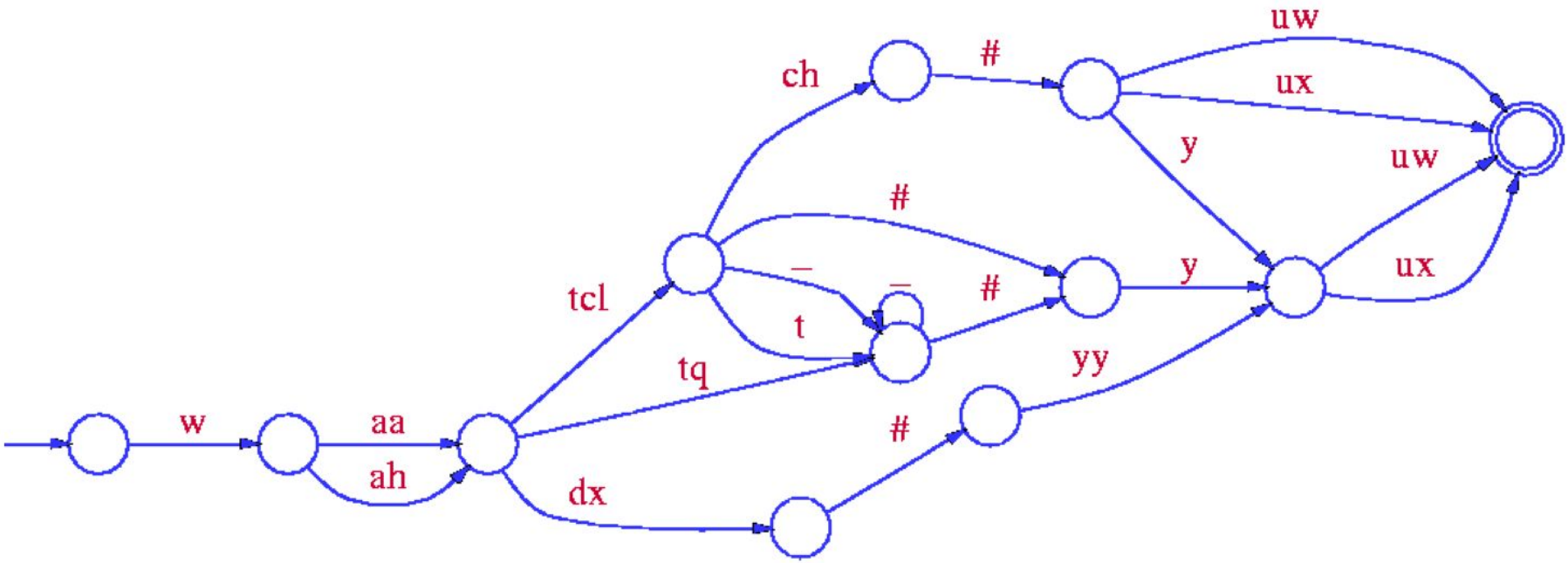
- using a phone recognizer to learn a pronunciation dictionary

a	baseball							game			
ax	b	ey		s	b	ao	l	g	ey	m	**word sequence**
ax	b	eh	ey	s	b	el		g	eh	m	**canonical phone sequence**
ax	b	eh	ey	s	b	el		g	eh	m	**aligned surface phones**
ax	b	eh	ey	s	b	el		g	eh	m	**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.

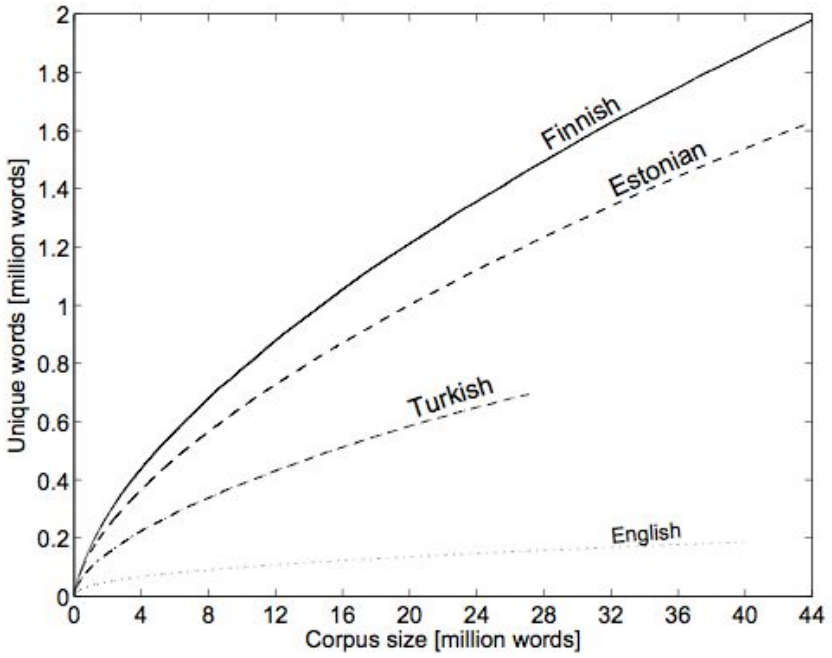
- estimating Ngram models from data
  - counts

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

- 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

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

    clustering

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

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

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

see Jurafsky & Martin 2017

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

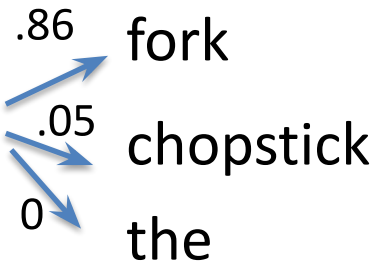
$$PP(W)=P(W)^{-1/N}$$

$$PP(W)=\prod_1^N P(w_i|w_1..w_{i-1})^{-1/N}$$

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

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

He is eating a steak with a knife and a …



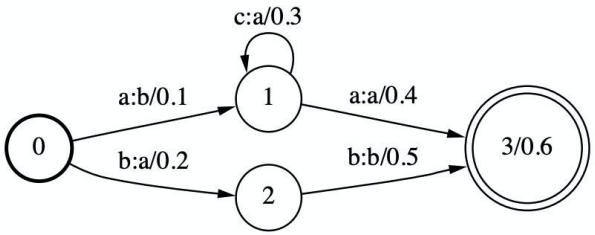
Putting all together

- acoustic model (an wFST): A
- pronunciation model (an wFST): P
- language model (an wFST): G

the combined model (a large wFST!): A□P□G

wFST composition

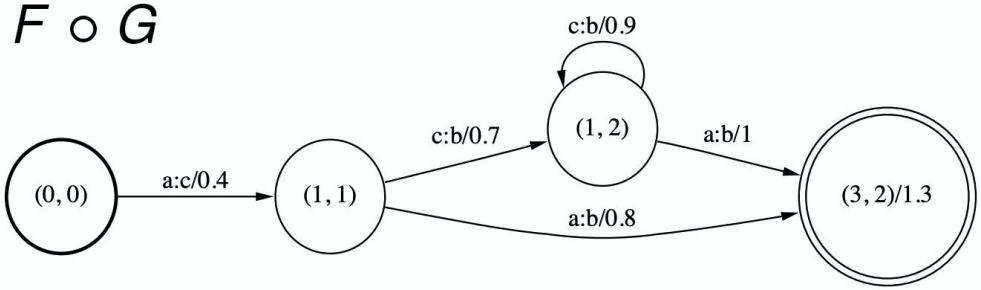
F



a a → b a

a c a → b a a

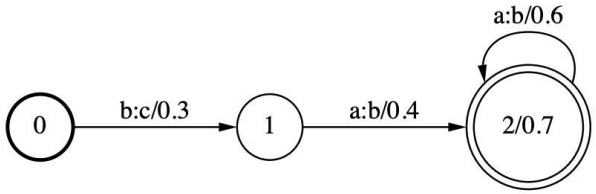
F ∘ *G*



a a → c b

a c a → c b b

G



b a → c b

b a a → c b b

from:
https://jonathan-hui.medium.com/speech-recognition-weighted-finite-state-transducers-wfst-a4ece08a89b7

Putting all together

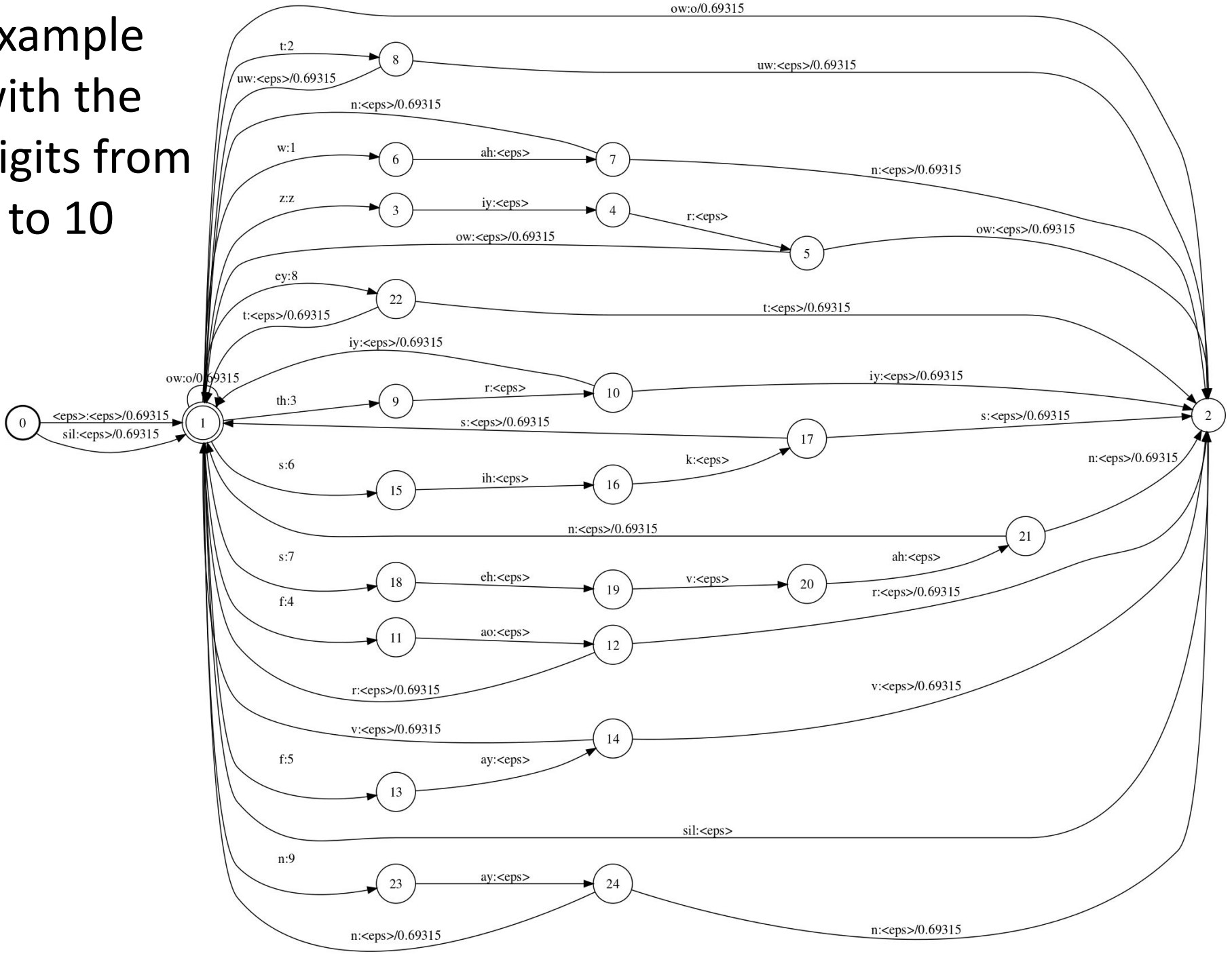
- acoustic model (an wFST): A
- pronunciation model (an wFST): P
- language model (an wFST): G

the combined model (a large wFST!): A□P□G

→ applying the wFST to acoustic input: a large decoding graph.

→ task: finding the optimal path

Example with the digits from 0 to 10



in practice

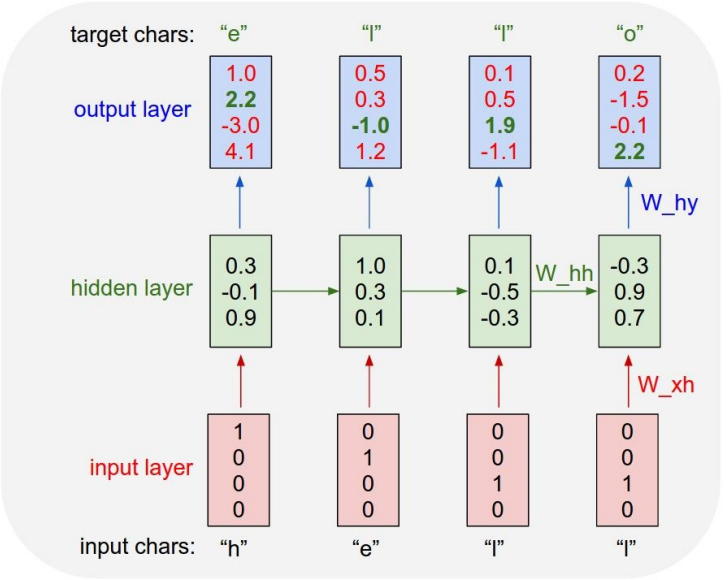
- FST tools
  - openFST (http://www.openfst.org)
  - composing & simplifying FSTs
- decomposing processing in two stages
  - a first decoding lattice, followed by lattice rescoring with a large LM

to know more

- Mohri, M. (1997). Finite-state transducers in language and speech processing. Computational linguistics, 23(2):269–311.
- Mohri, M., Pereira, F., and Riley, M. (2002). Weighted finite-state transducers in speech recognition. Computer Speech & Language, 16(1):69–88.

fast forward: RNN language models

- character language models



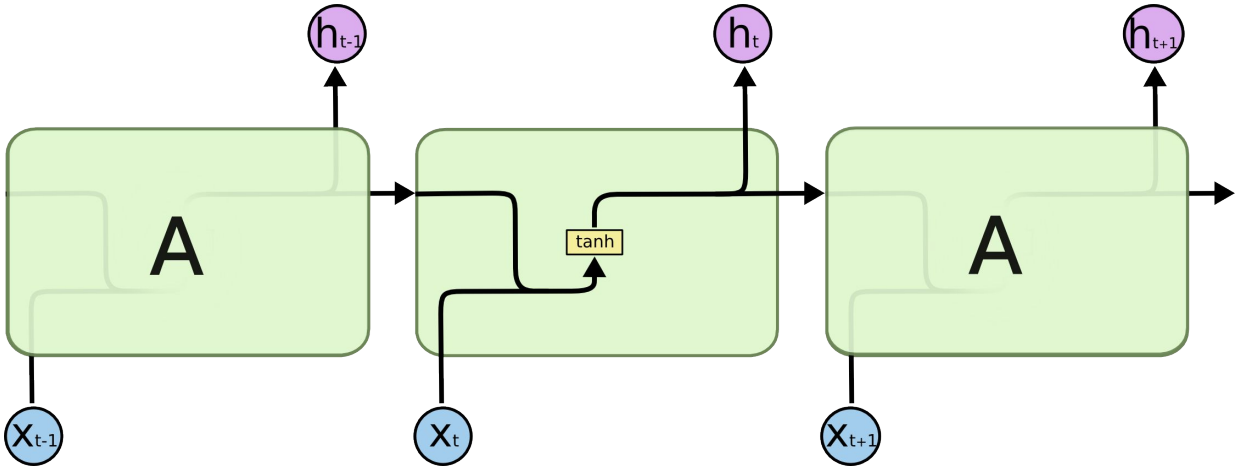
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

- RNNs can approximate P(W) better than ngram models by relying on indefinite amount of past information

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

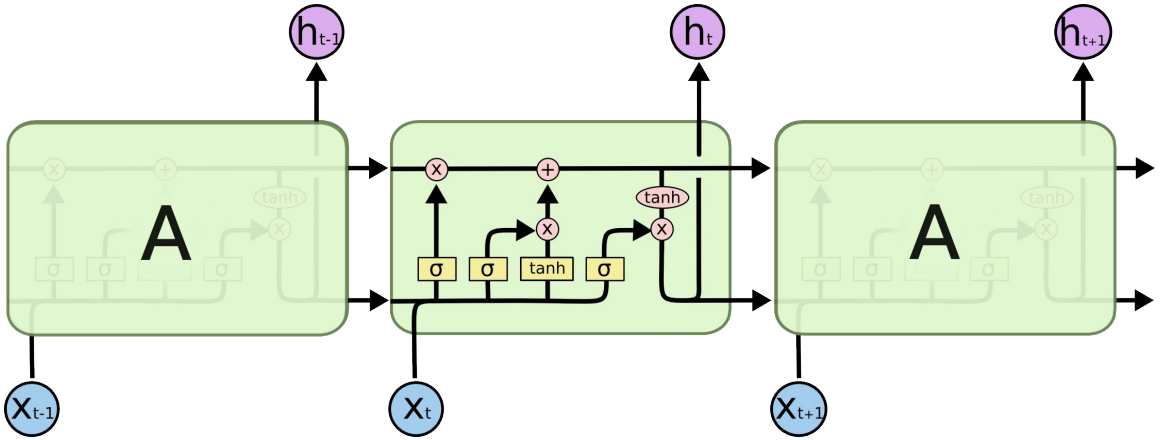
- but, the Markov assumption no longer holds

Vanilla RNN



$$h_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b)$$

LSTM



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

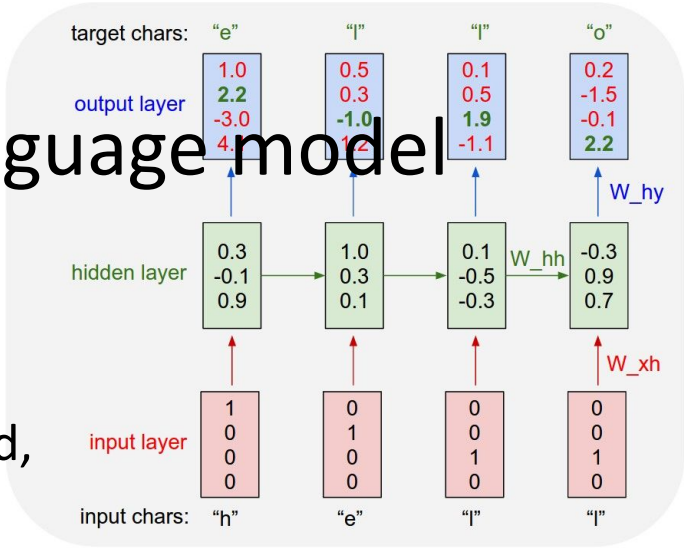
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

- sample output of a character language model



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.

Second Lord:They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

- word RNNs
  - in practice word RNNs work better than char RNNs, but grow very large
  - Typically, the input layer is as large as the nb of words; a linear embedding layer is used to reduce dimensionality
- wordpiece RNNs
  - use Byte Pair Encoding (BPE) to define a lexicon of useful word parts

Part II. decoding

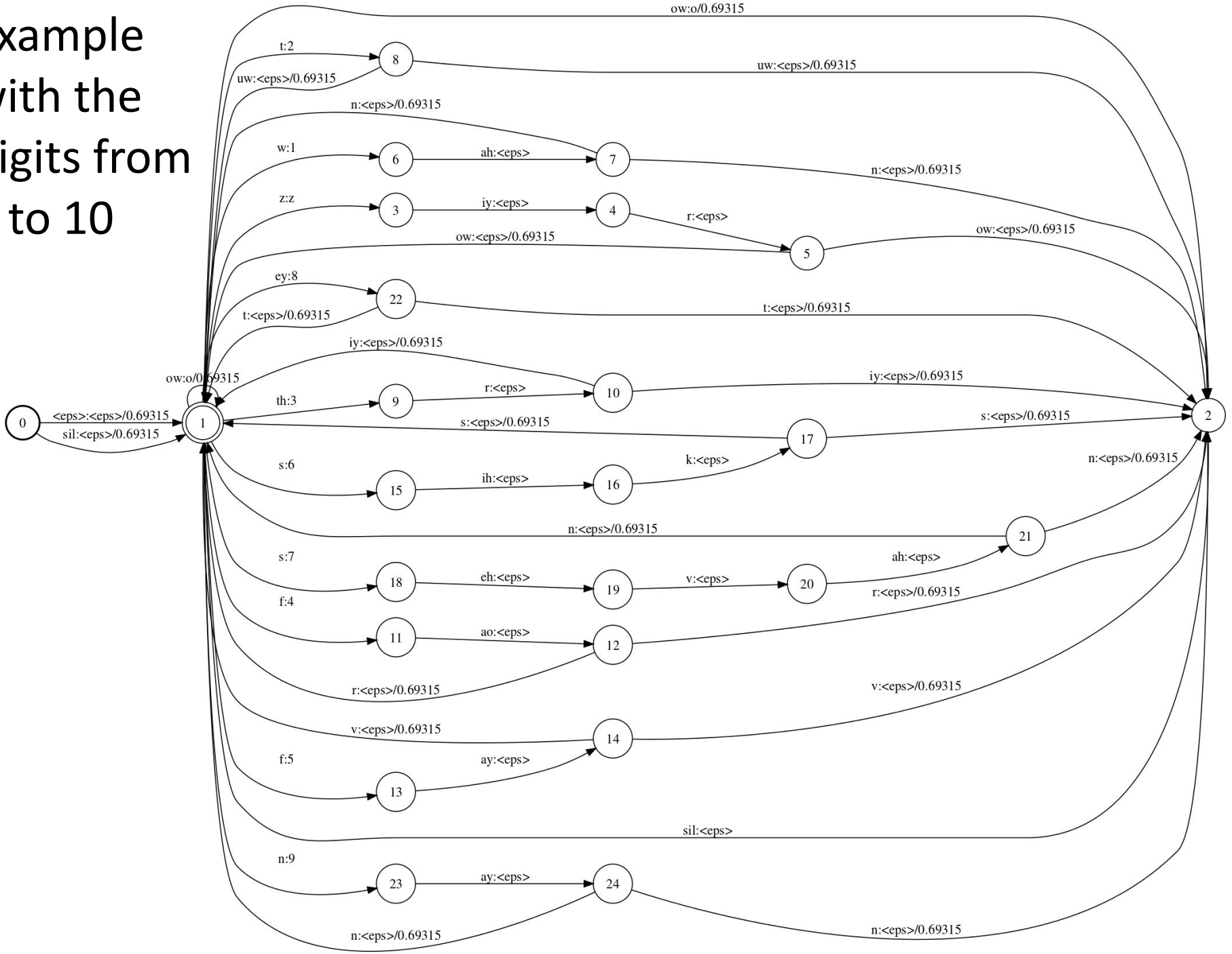
$W^* = \text{argmax}_w \, p_\theta(W|X)$

$W^* = \text{argmax}_w \, p_\theta(X|W).p_\theta(W)$

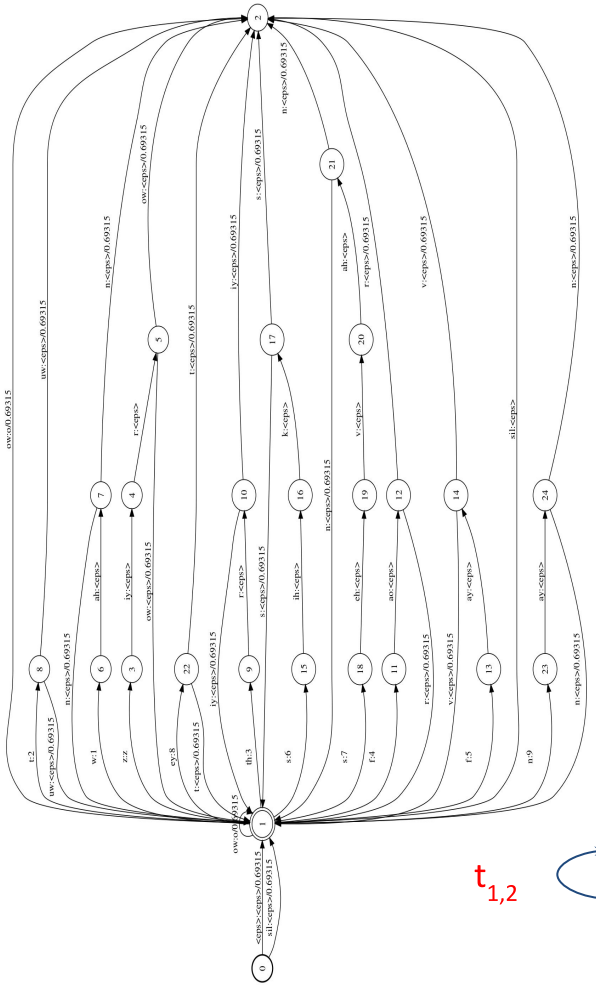
$W^* = \text{argmax}_w \, p_\theta(X|Q).p_\theta(Q|W).p_\theta(W)$

- Heuristics
- Dynamic programming (a quick tour)
- Dynamic time warping
- HMM Decoding

Example with the digits from 0 to 10



K states



N-1
..
2
1
0

$t_{1,2}$

d_{ij}

x_1 x_2 x_3 ….

x_N

N frames

minimizing $\Pi_k d_{ik,jk} \cdot t_{ik-1,\,ik}$

number of possible paths: K^N !!

Search heuristics

- greedy decoding
  - select the best state at each time step
  - efficient but not very good
- beam search
  - select the p best states at each time step
  - still efficient, and slightly better
- Dynamic programming/viterbi
  - efficient and exact solution !

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

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

for n=6?

1. <u>Define subproblems</u>
   – Let D_n be the number of ways to write n as the sum of 1,3,4
2. <u>Find the recurrence</u>
   – 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}

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

3. Set the initial conditions

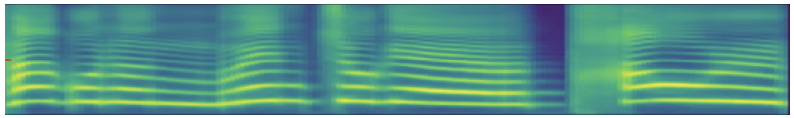
$$D[0] = D[1] = D[2] = 1$$
$$D[3] = 2$$

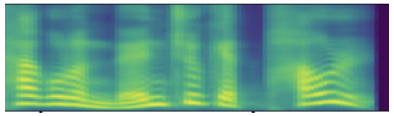
4. Compute the solution

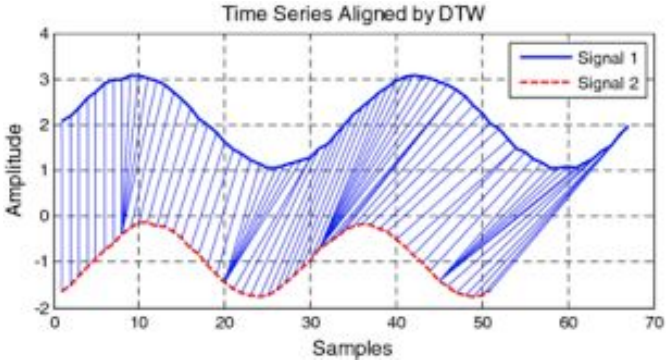
```
for(i = 4; i <= n; i++)
        D[i] = D[i-1] + D[i-3] + D[i-4]
```

Done! (complexity: o(n))

Application: Dynamic Time Warping



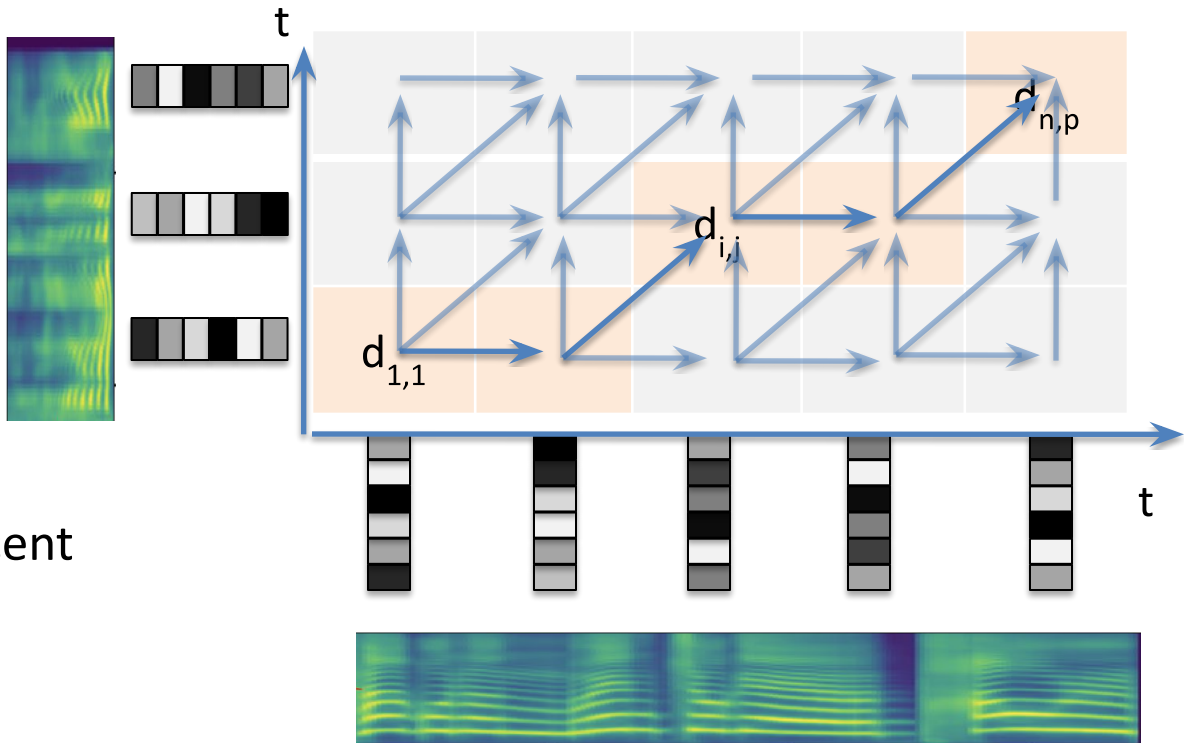




t

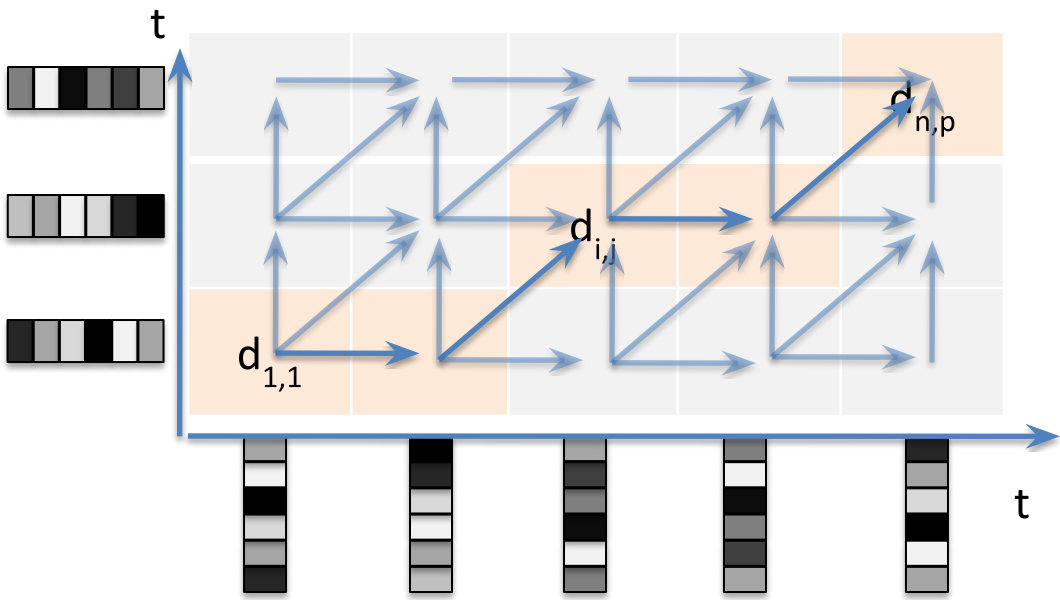
Application: Dynamic Time Warping

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



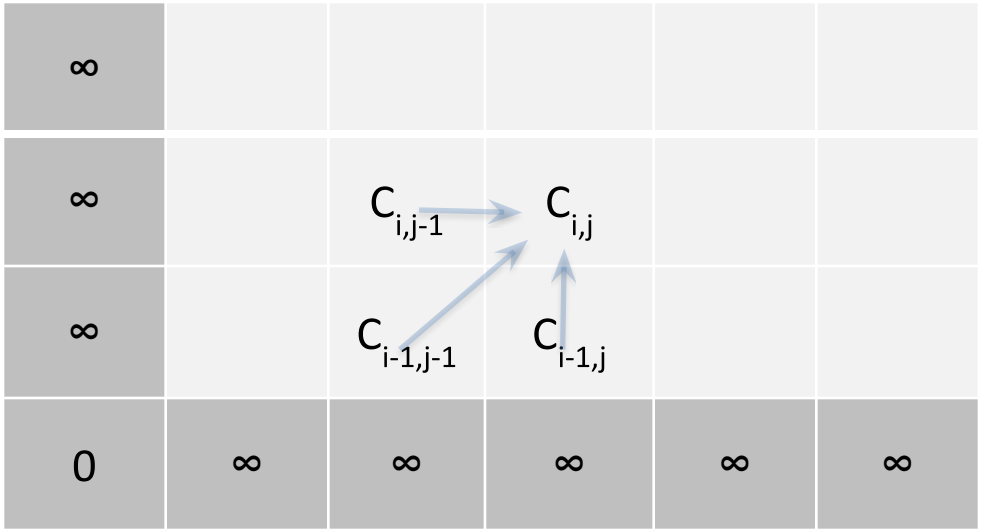
Application: Dynamic Time Warping

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

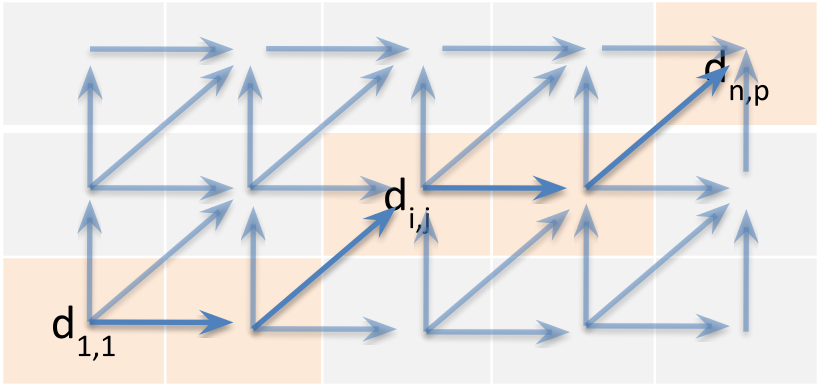


Dynamic programming analysis

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



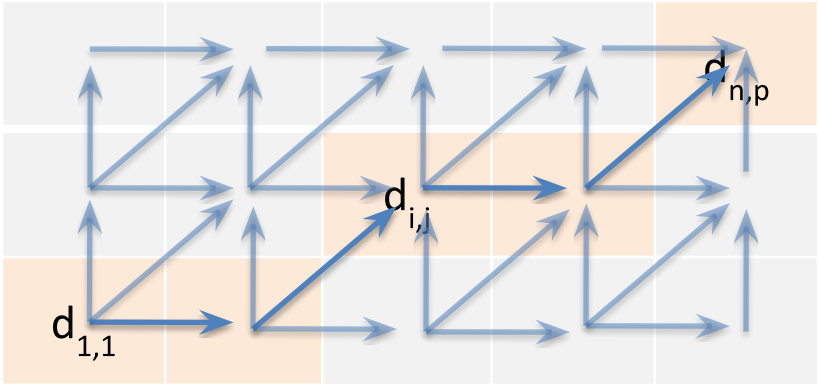
DTW algorithm



- Initialize

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

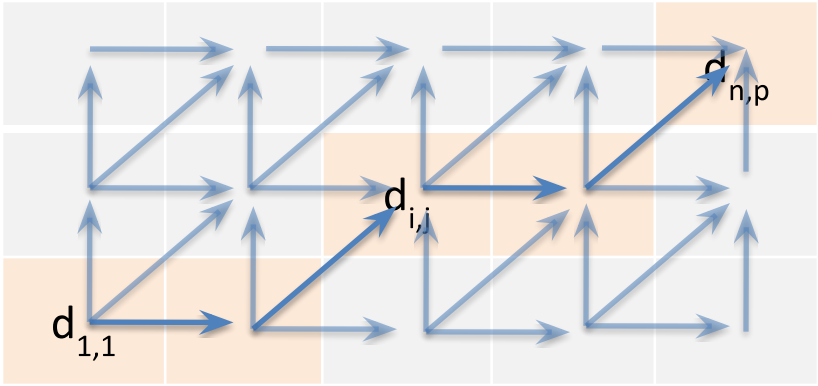
DTW algorithm



- 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	∞	∞	∞	∞	∞

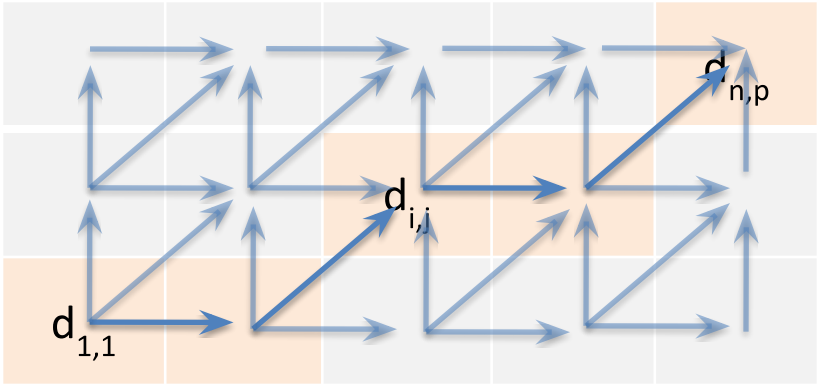
DTW algorithm



- 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	∞	∞	∞	∞	∞

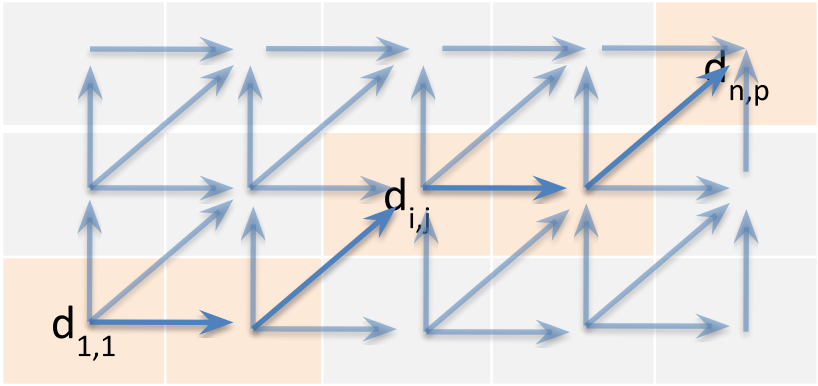
DTW algorithm



- 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	∞	∞	∞	∞	∞

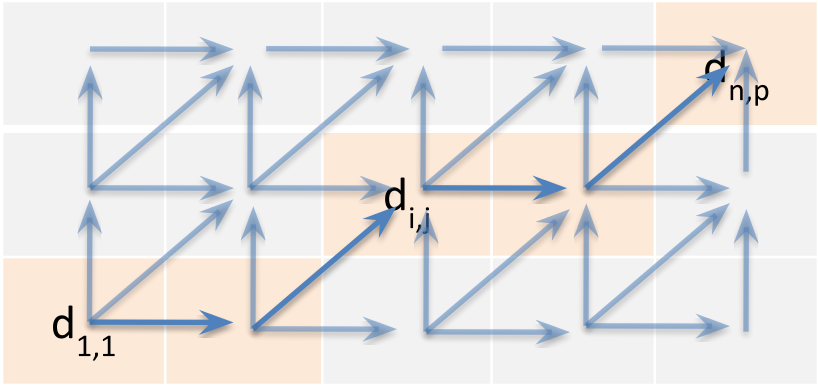
DTW algorithm



- 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

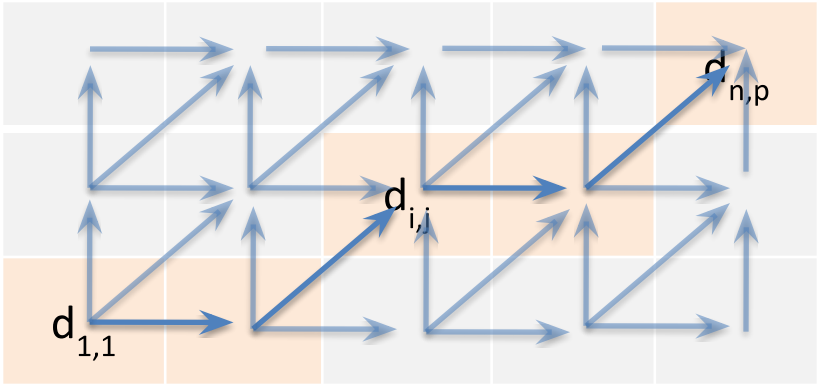


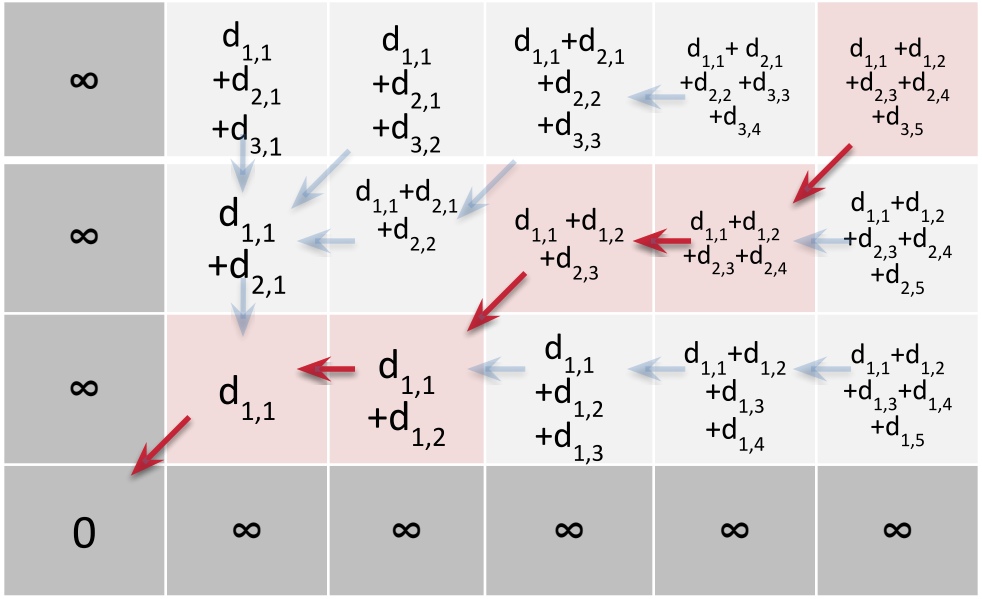
etc…

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

DTW algorithm

- Start from the final point
- Follow the path backward

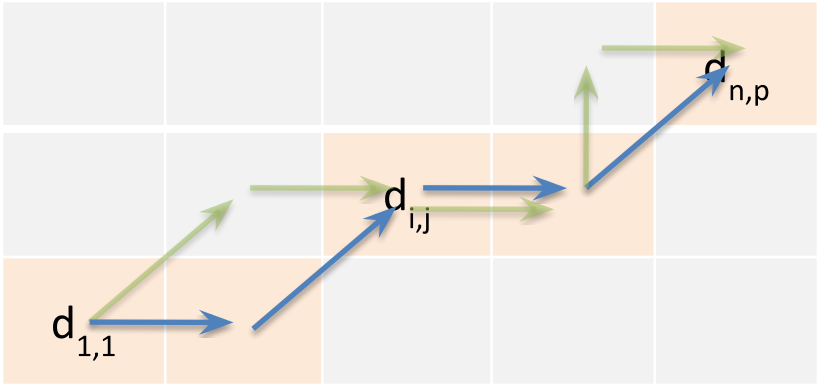


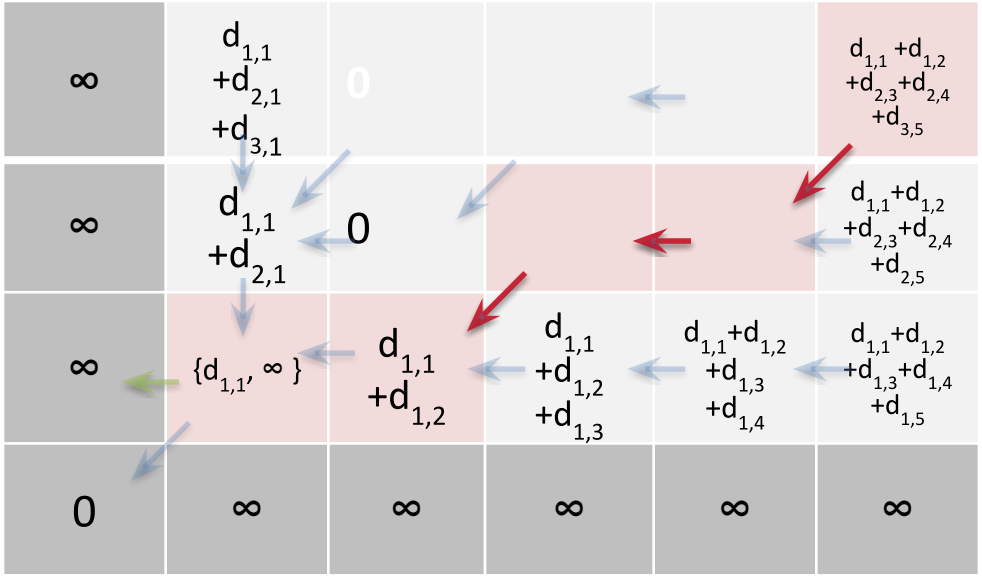


Variation: n-best

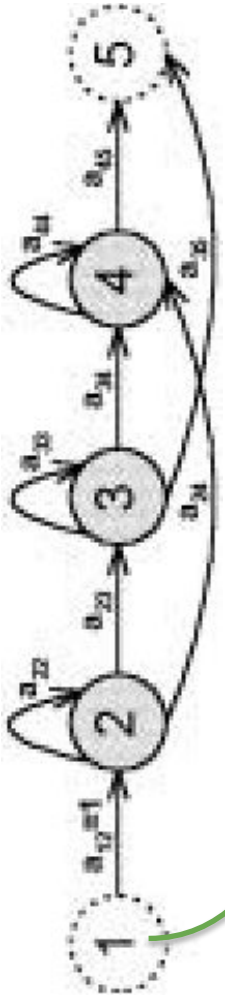
- Change the recurrence formula
  $C_{i,j}=d_{i,j}+best(n,C_{i,j-1} \cup C_{i-1,j-1} \cup C_{i-1,j})$
  where best(n,S)=the n smallests elements in set S.

- Memorize n links backward





HMM decoding (viterbi)



$P(x_1x_2|q_1q_2)=p(q_2|q_1)\ p(x_1|q_2).\ p(q_3|q_2)\ p(x_2|q_2)$

Now:

1. total cost of path is product (of proba) not sum (of distances)
2. transitions have costs, not only states
3. the task is to maximize prob not minimize distance

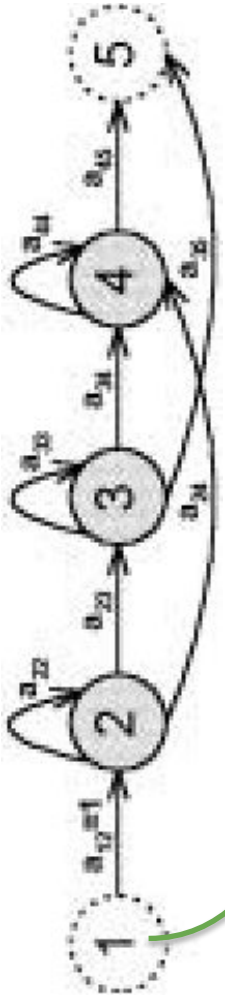
$v_1(2)=a_2$

$a_{21}=p(q_2|q_1)$

$b_2(x_1)=p(x_1|q_2)$

x_1 x_2 x_3 x_4 x_5 t

HMM decoding (viterbi)



$P(x_1x_2|q_1q_2)=p(q_2|q_1)\ p(x_1|q_2).\ p(q_3|q_2)\ p(x_2|q_2)$

$v_1(2)=a_{21}.b_2(x_1)$

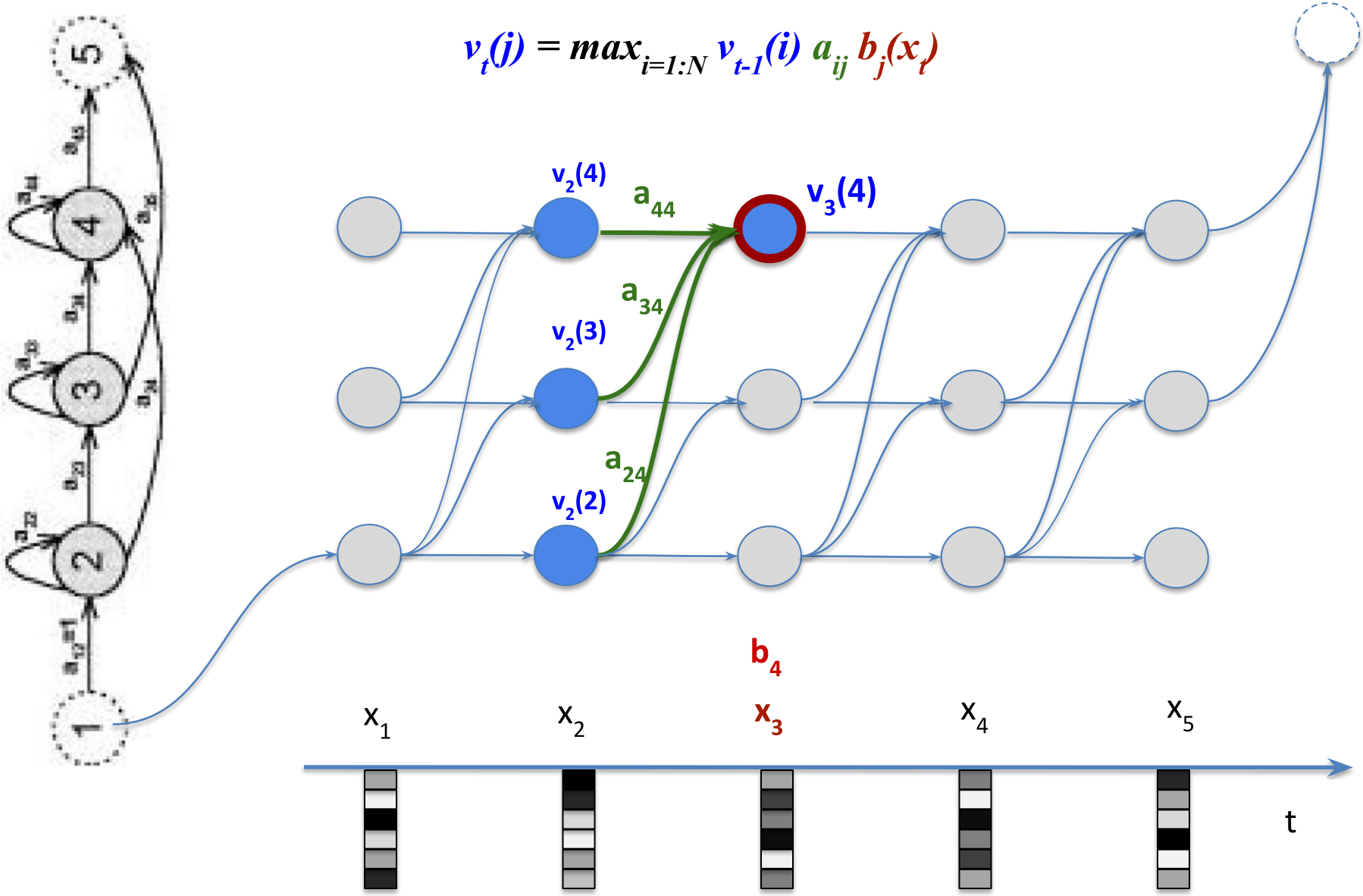
$a_{21}=p(q_2|q_1)$

$b_2(x_1)=p(x_1|q_2)$

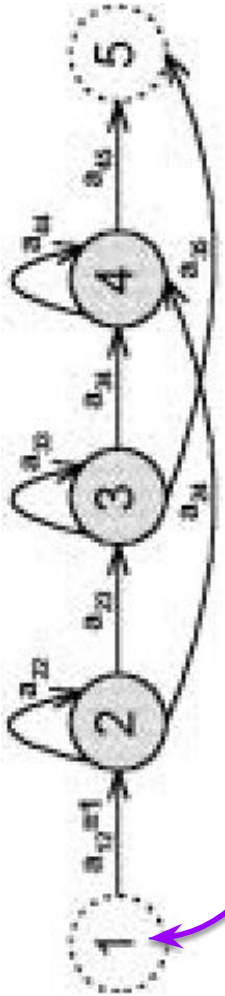
x_1 x_2 x_3 x_4 x_5

t

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

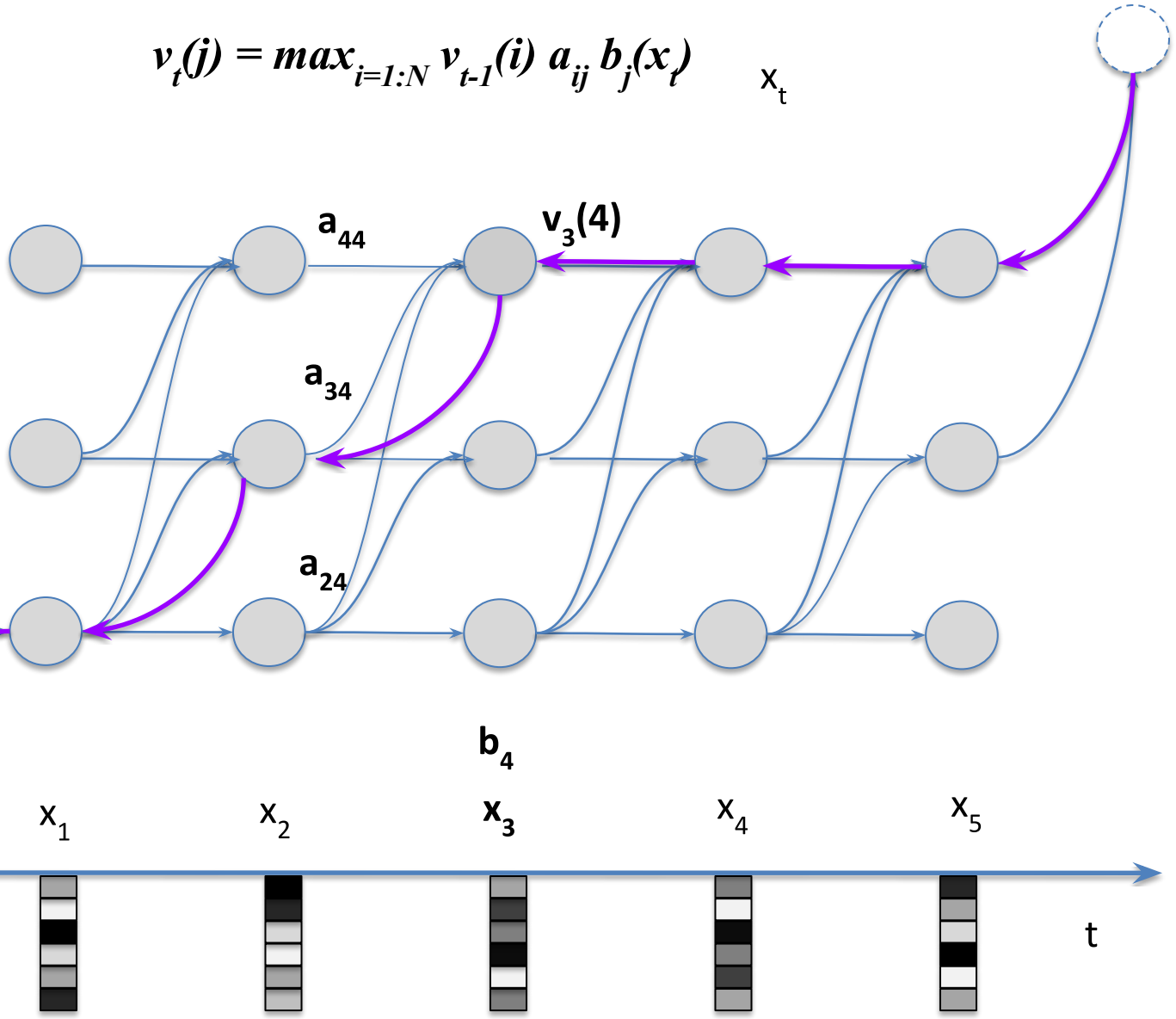


decoding: $Q^* = \text{argmax}_Q P(X|Q)$



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

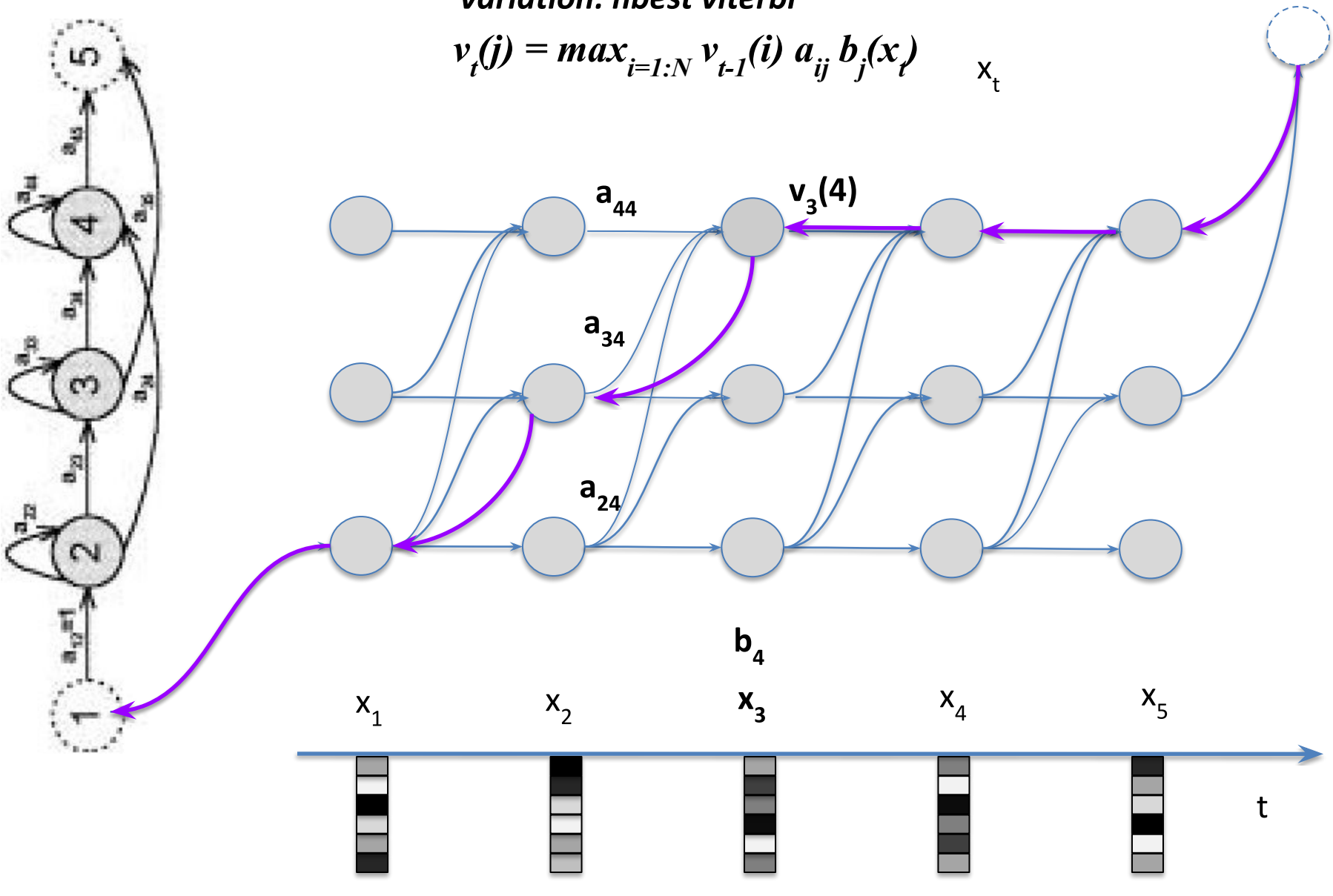
x_t



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

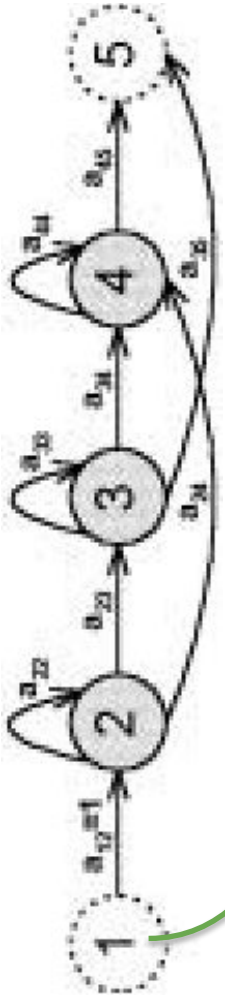
variation: nbest viterbi

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



Learning

Learning



Now:

1. X is given (training corpus)
2. the HMM topology is known
3. $\boldsymbol{\theta}$ (a_{ij} and b_i) is unknown
4. Q (the sequence corresponding to X) is unknown

$a_{21}=p(q_2|q_1)$

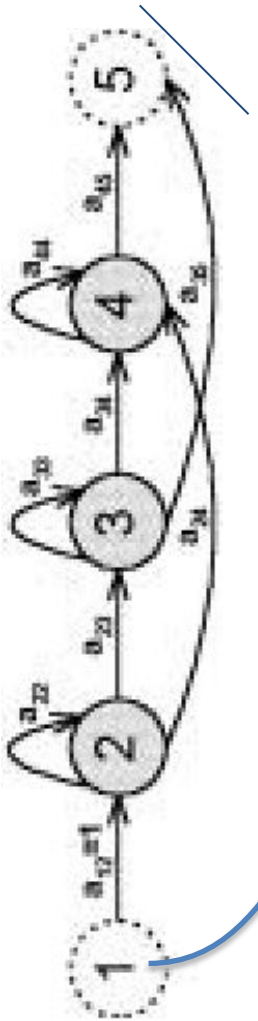
$b_2(x_1)=p(x_1|q_2)$

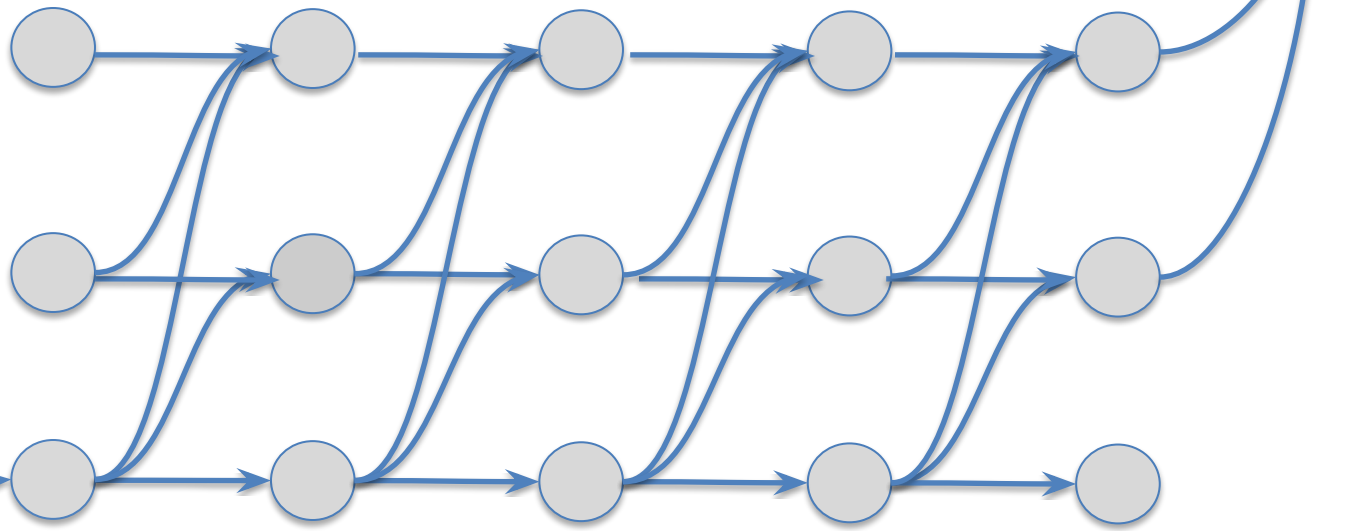
x_1 x_2 x_3 x_4 x_5 t

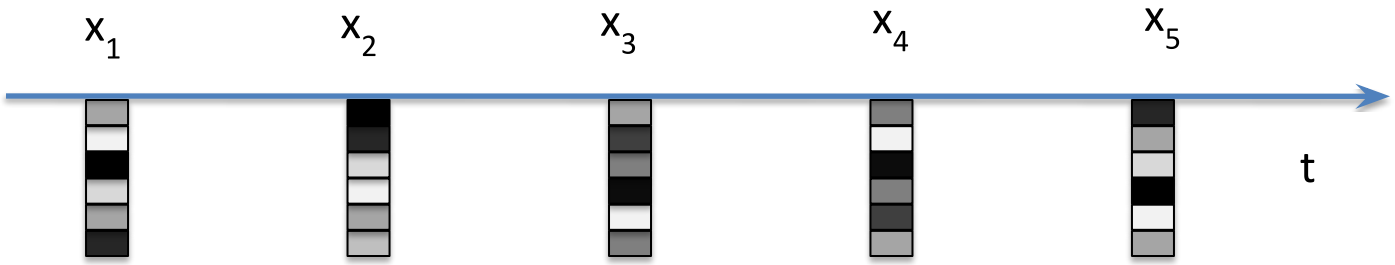
Learning

$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(X)$

$p_{\boldsymbol{\theta}}(X) = \Sigma_Q\, p_{\boldsymbol{\theta}}(X,Q) = \Sigma_Q\, p_{\boldsymbol{\theta}}(X|Q)p(Q)$



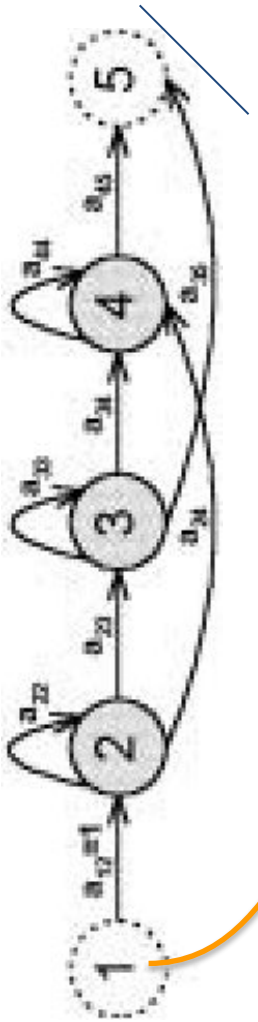


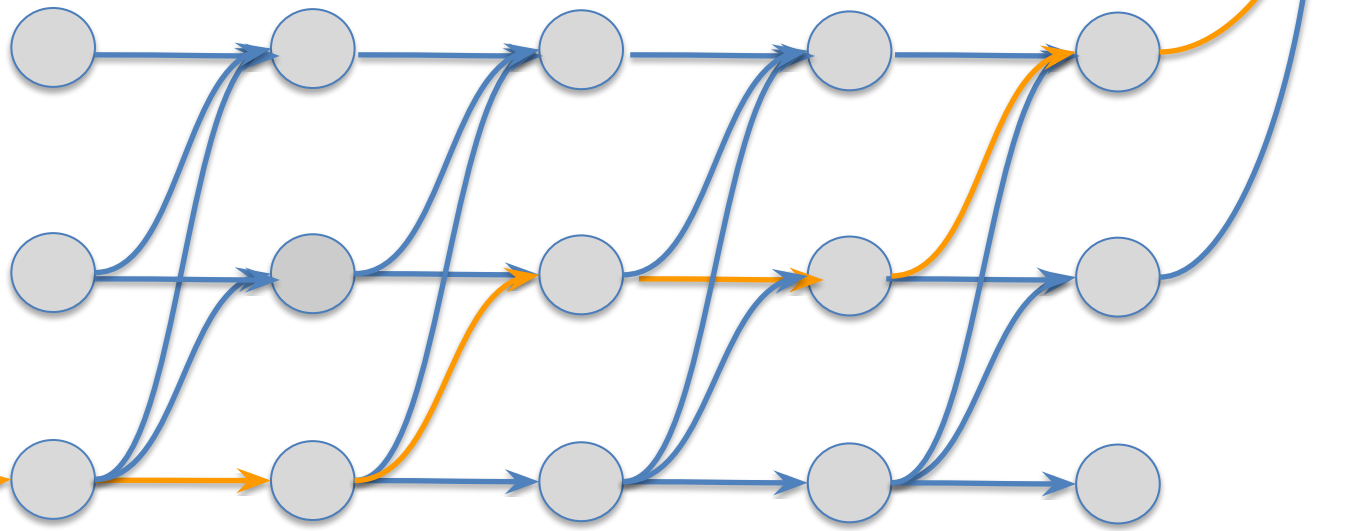


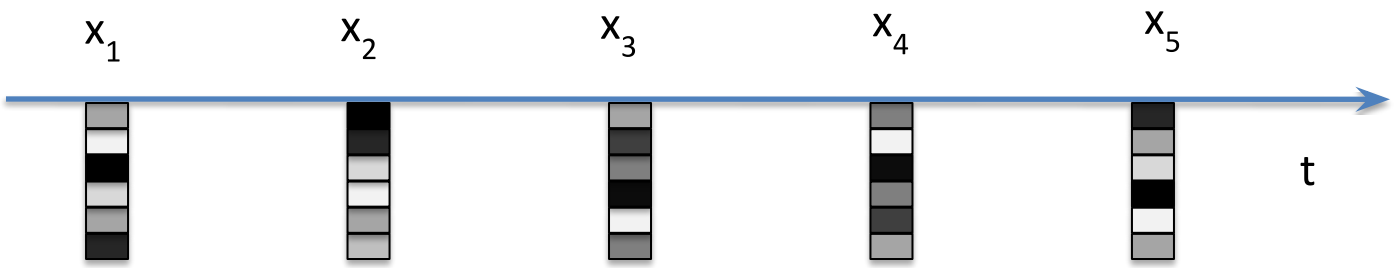
Learning

$\theta^* = \arg\max_\theta \Sigma_Q p_\theta(X|Q)$

$\theta^* \cong \arg\max_\theta p_\theta(X|Q^*)$



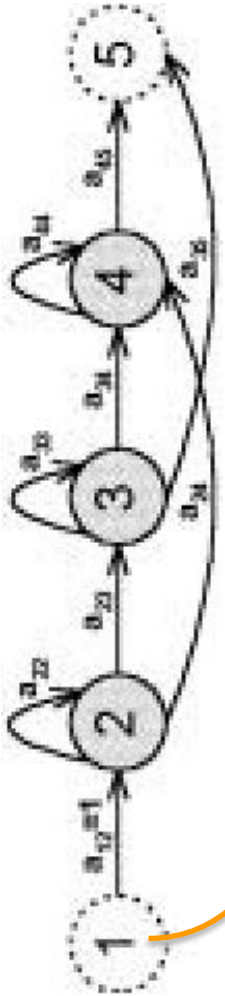


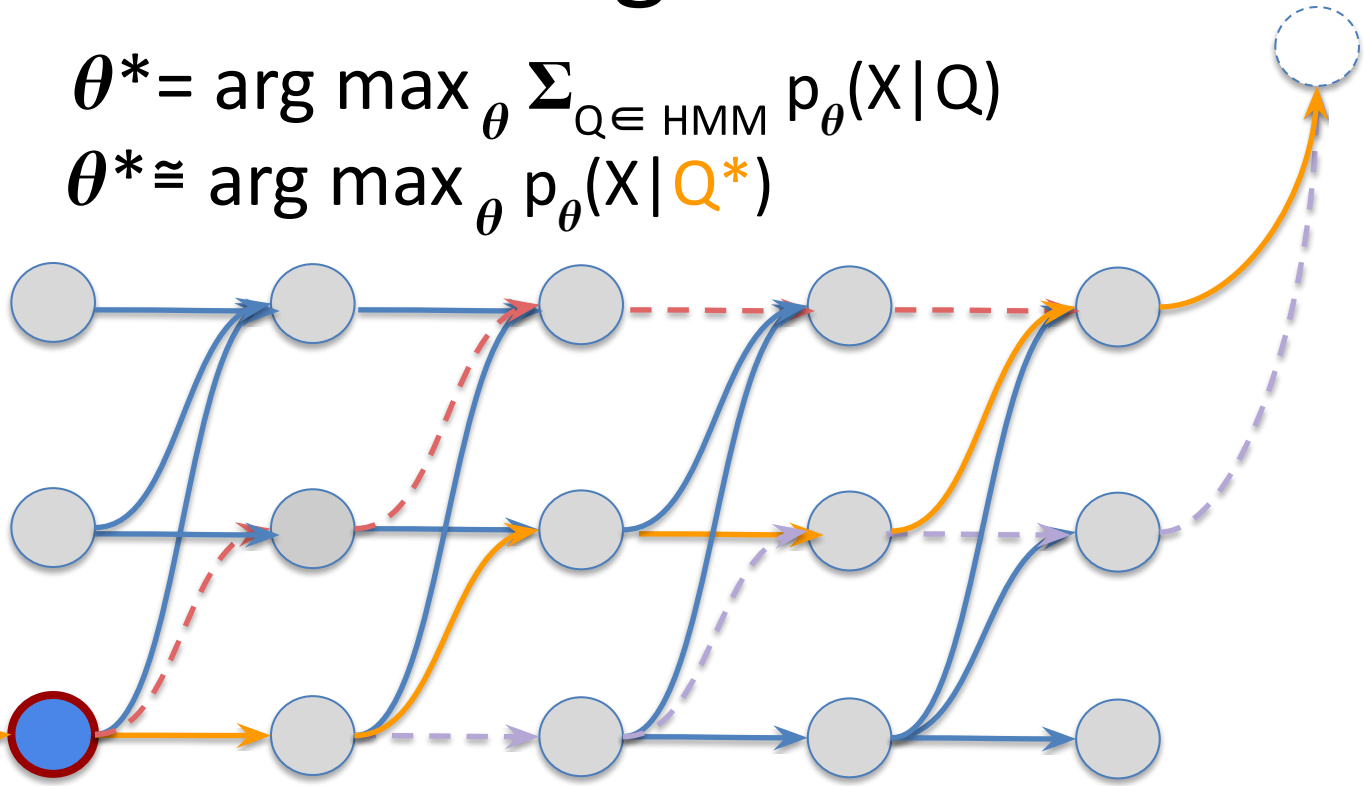


Learning

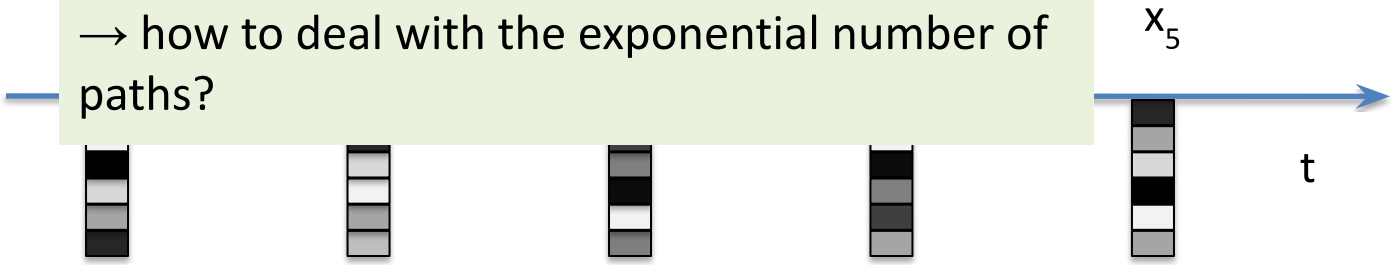
$\theta^* = \arg\max_\theta \Sigma_{Q \in HMM} p_\theta(X|Q)$

$\theta^* \cong \arg\max_\theta p_\theta(X|Q^*)$





→ how to aggregate all the alternative paths?
→ how to deal with the exponential number of paths?



Learning

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

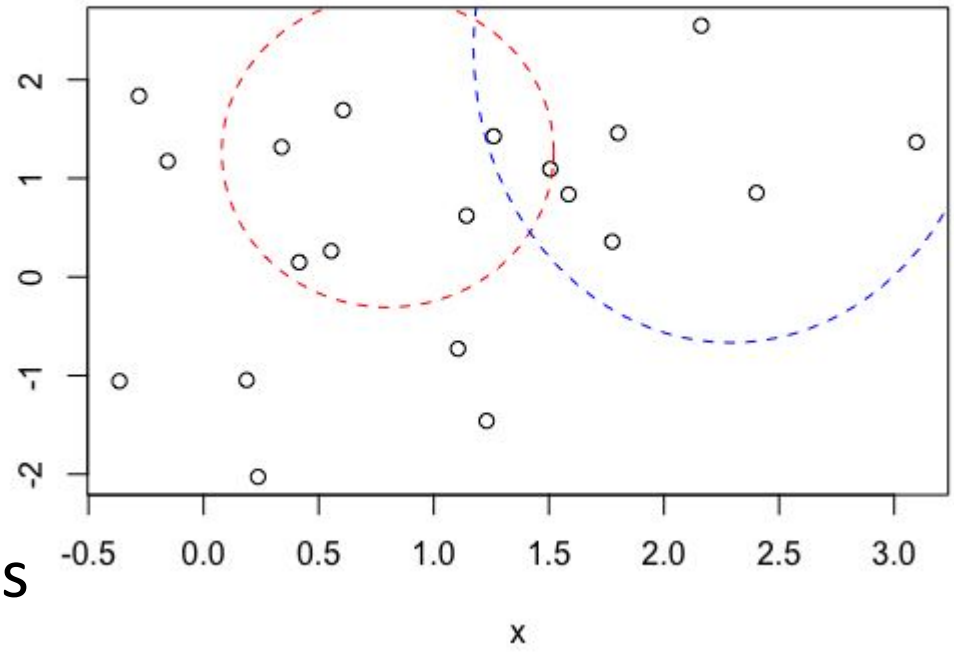
  $\theta^* = \arg\max_\theta p(X|Q,\theta)$
  - Iterate

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 $X=\{x_1..x_N\}$

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

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

$\theta^* = \arg\max_\theta \prod_i p(x_i,q_i|\theta)$

K-means solution:

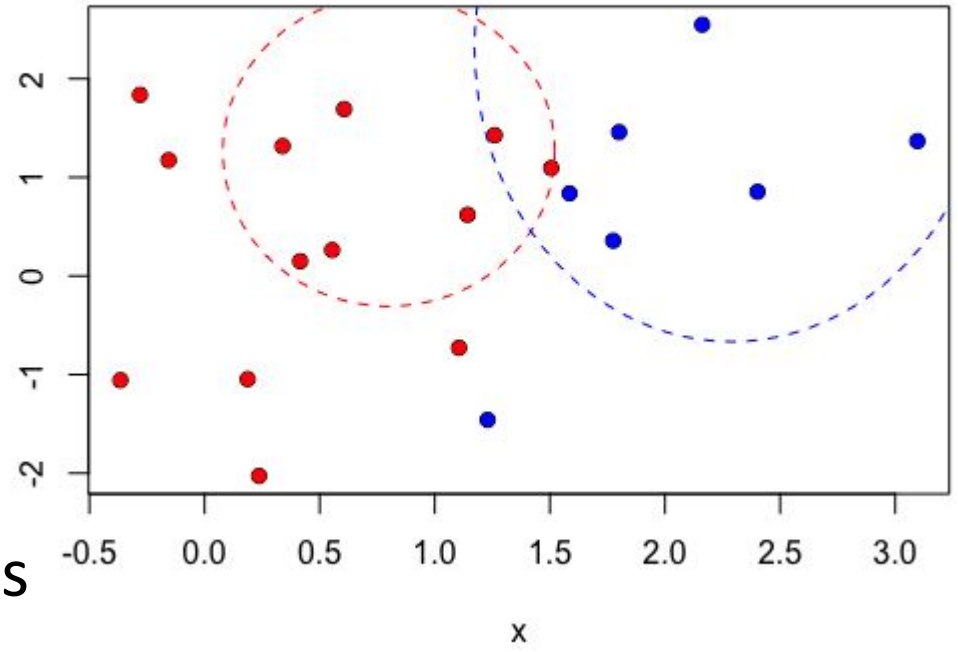
- **initialize random values for θ**
- alternate between
  - finding membership q
  - update the parameter θ

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 $X=\{x_1..x_N\}$

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

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

$\theta^* = \arg\max_\theta \prod_i p(x_i,q_i|\theta)$

K-means solution:

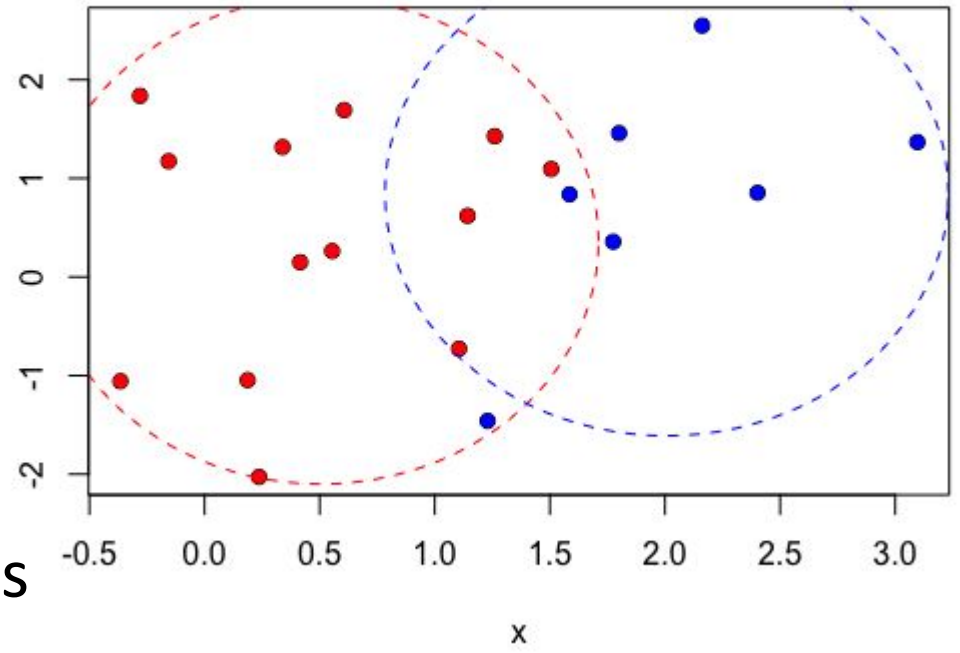
- initialize random values for θ
- alternate between
  - **finding membership q**
  - update the parameter θ

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 $X=\{x_1..x_N\}$

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

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

$\theta^* = \arg\max_\theta \prod_i p(x_i,q_i|\theta)$

K-means solution:

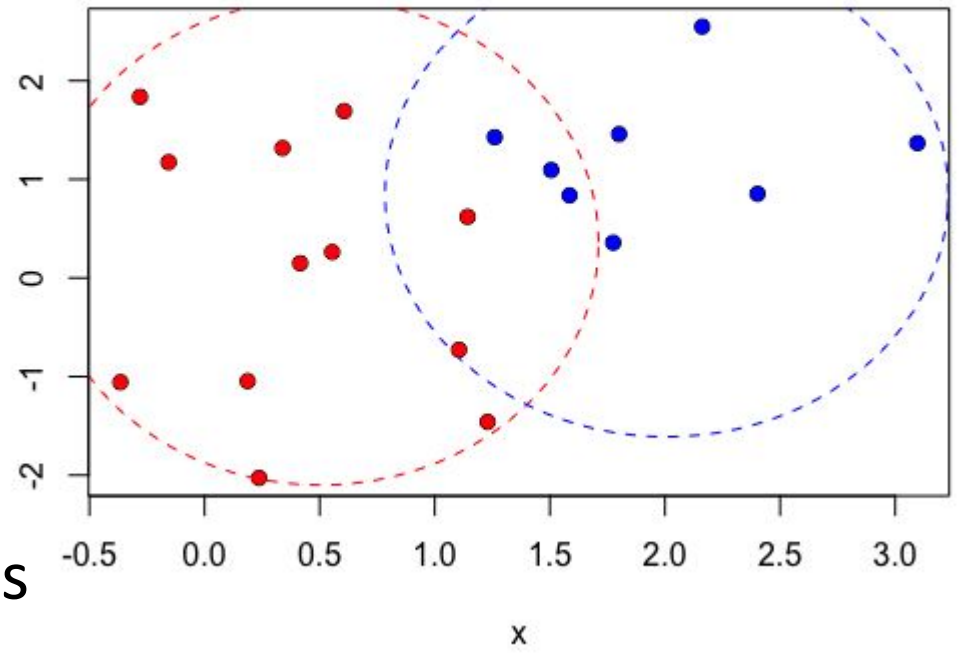
- initialize random values for θ
- alternate between
  - finding membership q
  - **update the parameter θ**

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 $X=\{x_1..x_N\}$

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

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

$\theta^* = \arg\max_\theta \prod_i p(x_i,q_i|\theta)$

K-means solution:

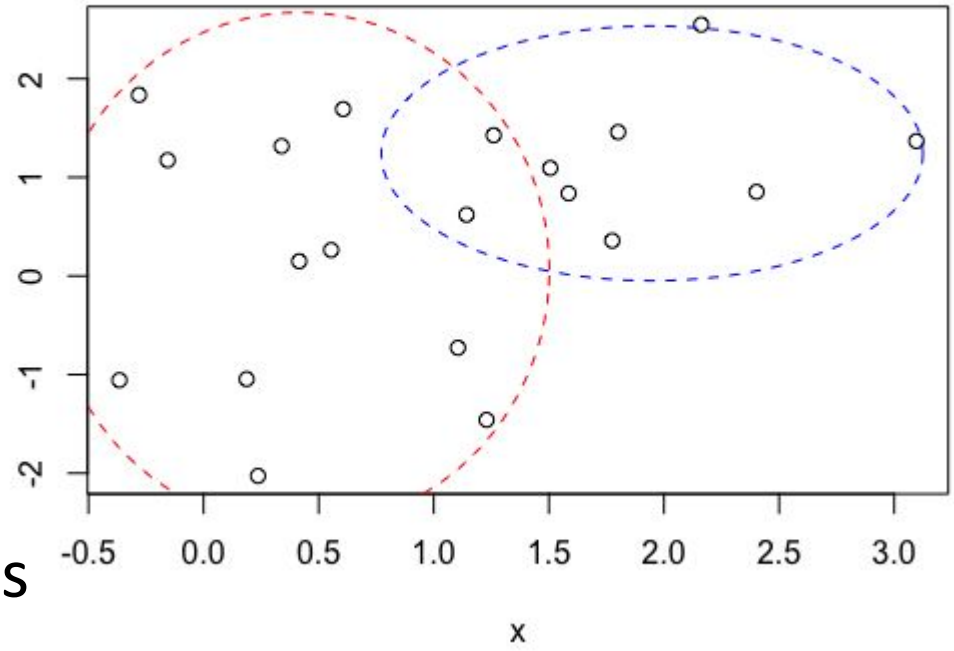
- initialize random values for θ
- alternate between
  - **finding membership q**
  - update the parameter θ

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 $X=\{x_1..x_N\}$

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

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

$\theta^* = \arg\max_\theta \prod_i p(x_i,q_i|\theta)$

K-means solution:

- initialize random values for θ
- alternate between
  - finding membership q
  - update the parameter θ

→ ***guaranteed to converge to a local max***

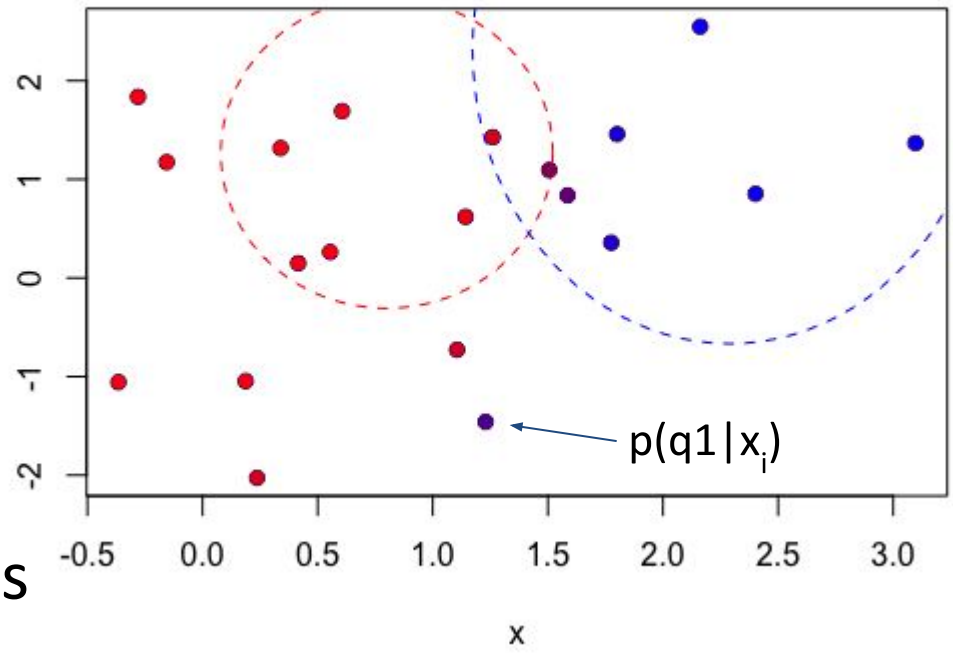
an example: GMMs

- mixture model

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

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

- problem: given observations $X=\{x_1..x_N\}$



$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_i p(x_i|\boldsymbol{\theta})$

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

$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_i p(x_i, q_i|\boldsymbol{\theta})$

The EM solution

- same thing, but instead of a discrete membership, use a graded one: $p(q_j|x_i)$
- now the update is weighed

$$\mu_j = \Sigma_i\, p(q_j|x_i)x_i \,/\, \Sigma_i p(q_j|x_i)$$

→ ***guaranteed to converge to a local max***

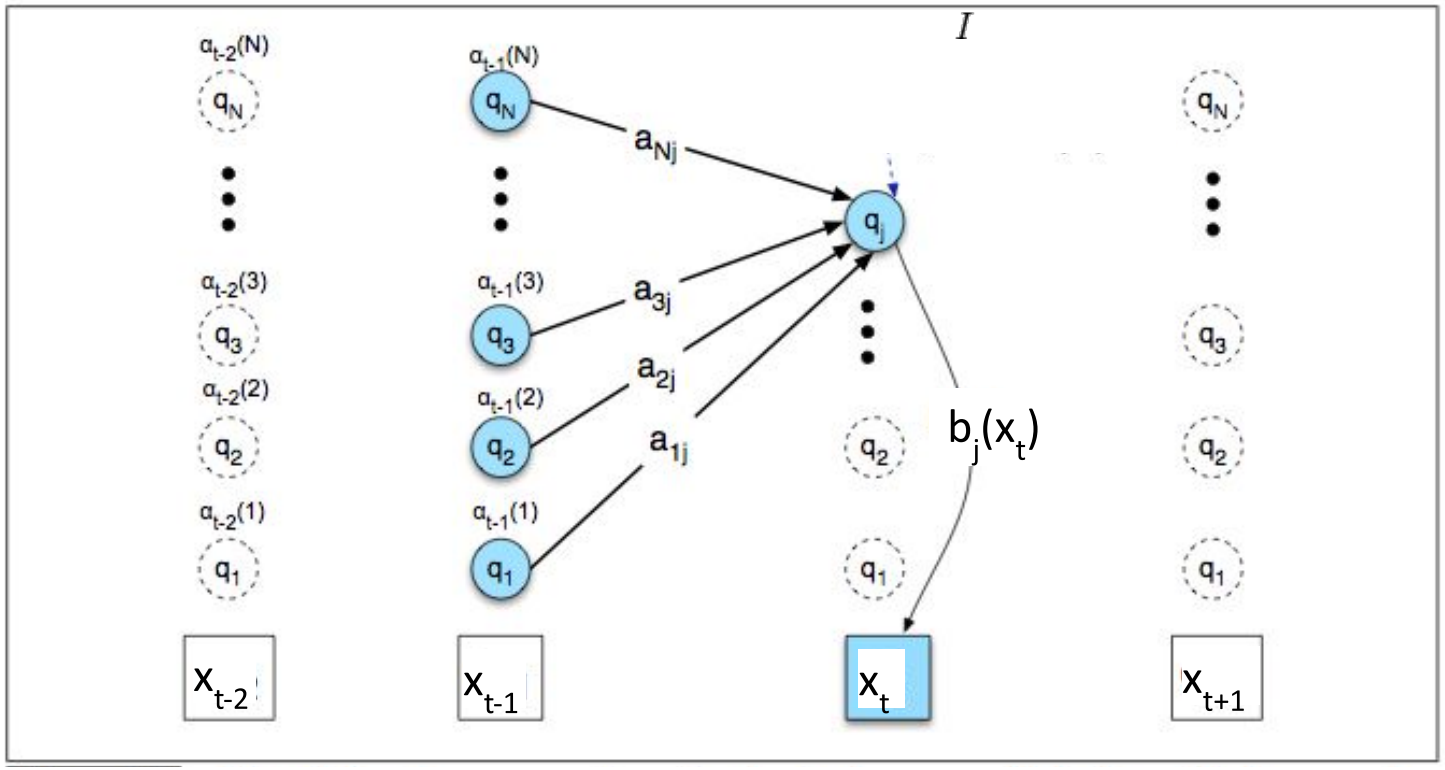
going back to HMMs

- intuition
  - 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*

back to HMMs

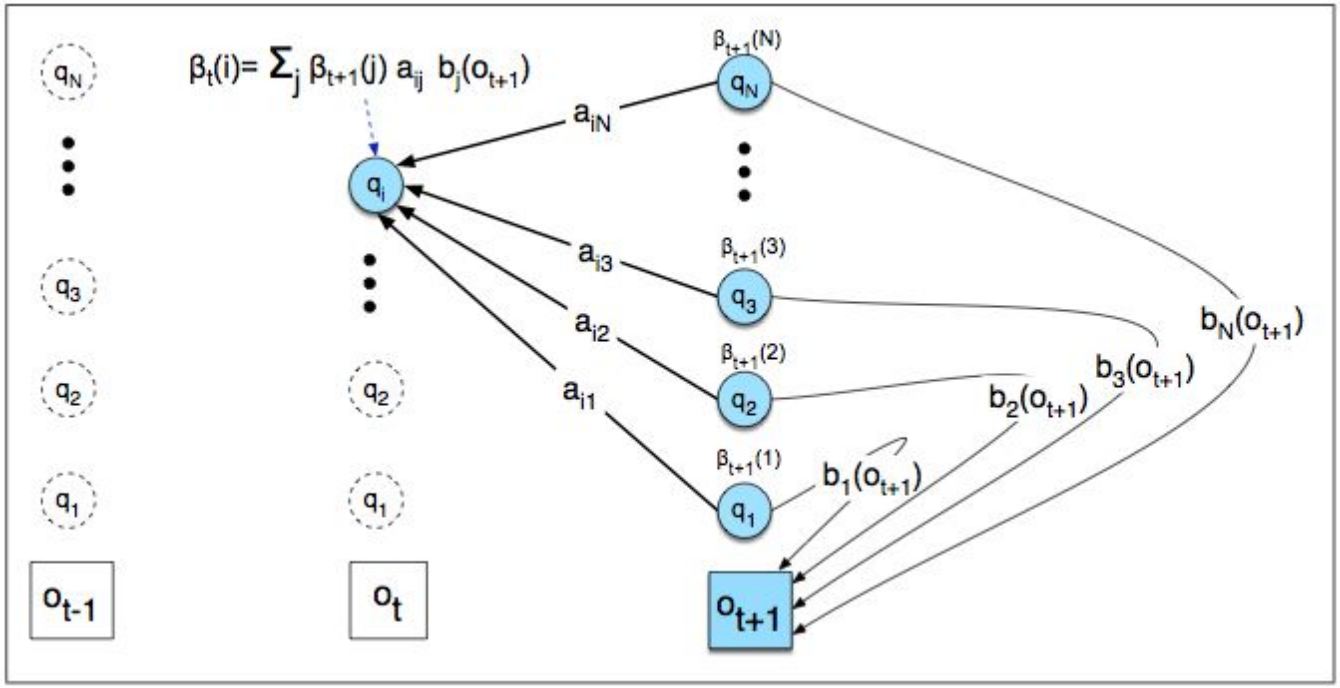
- forward probabilities $\alpha_t(j) = P(x_1, x_2..x_t, q_t = j|\theta)$

$$\alpha_t(j) = \sum_I \alpha_{t-1}(j) a_{ij} b_j(x_t))$$

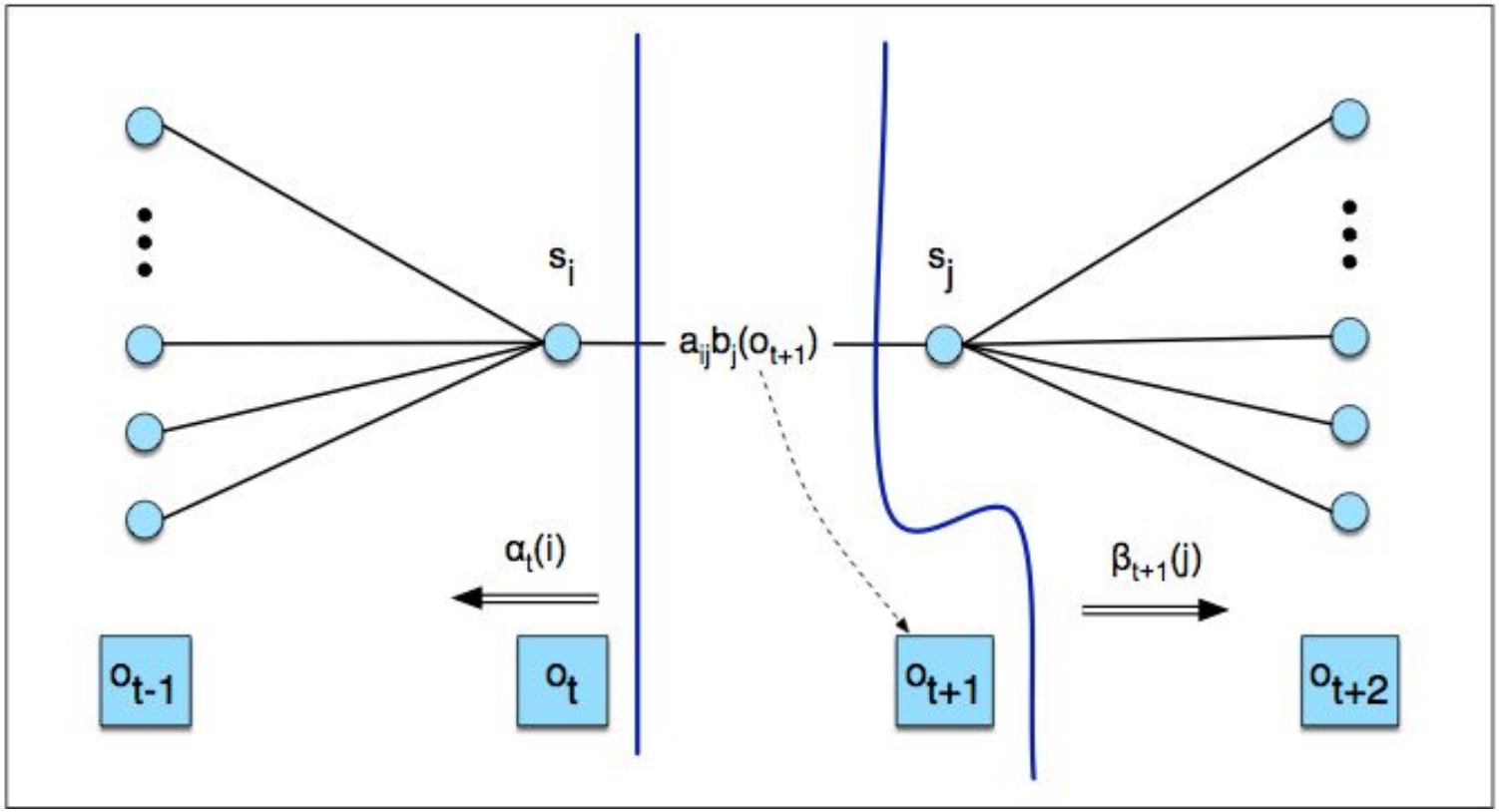


from J&M (2017)

- 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.

Baum-Welch algorithm:

- unsupervised (if you have a very generic HMM topology)
- supervised (if you have the transcription of X and only need to find the alignement)
- in between (if you have a lot of latent variables, eg, the precise lexical variant used for each Wi, etc)

MMI

Optimizing P(X) does not necessarily optimizes the metric of interest (Word Error Rate)

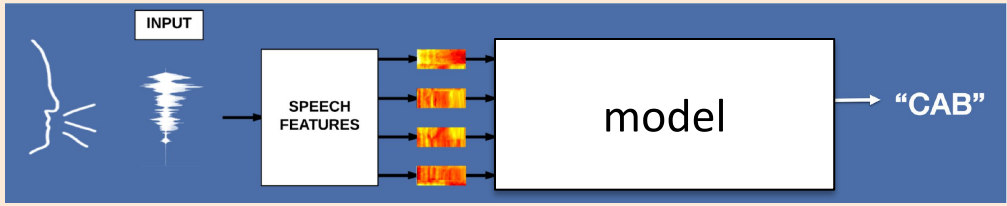
$$\mathrm{MMI}(\boldsymbol{\theta}) = \log P_{\boldsymbol{\theta}}(X|W)P(W) \,/ \sum_{w} P_{\boldsymbol{\theta}}(X|w)P(w)$$

discriminative loss

see for details https://desh2608.github.io/2019-05-21-chain/

Course

I. Generative Models
II. Decoding
III. Learning



X Θ W

Notations

θ: model parameters

$X=x_1 \ldots x_t$: observed speech frames

$W=w_1 .. w_n$: word transcriptions

Decoding: given observed X and param. θ, find the most likely W.

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

Learning: given an observed X and model, find the most likely θ

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