ANLP

09 - HMMs II (sequences, part II)

David Schlangen
University of Potsdam, MSc Cognitive Systems
Winter 2019 / 2020

POS Tagging

VBD
VBN VBZ
NNP NNS

VB
VBP VBZ
NN NNS CD NN

Fed raises interest rates 0.5 percent

VBD
VBN VBZ
NNP NNS

VB
VBP VBZ
NN NNS CD NN

Fed raises interest rates 0.5 percent

I hereby increase interest rates 0.5%

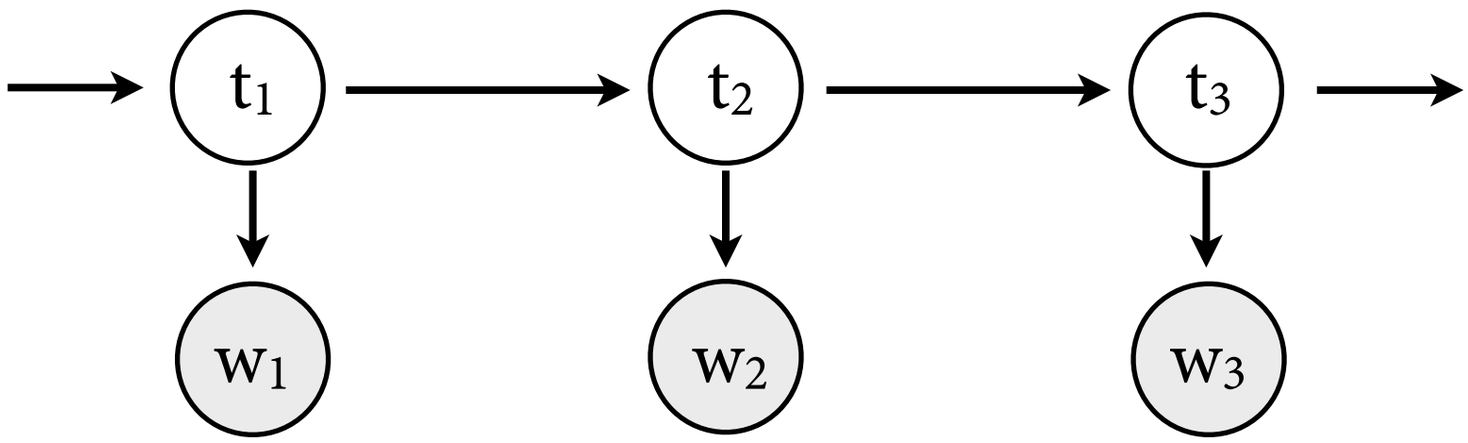
I'm 0.5% interested in the Fed's raises!

- Other paths are also plausible but even more semantically weird…
- What governs the correct choice? Word + context
- Word identity: most words have <=2 tags, many have one (*percent*, *the*)
- Context: nouns start sentences, nouns follow verbs, etc.

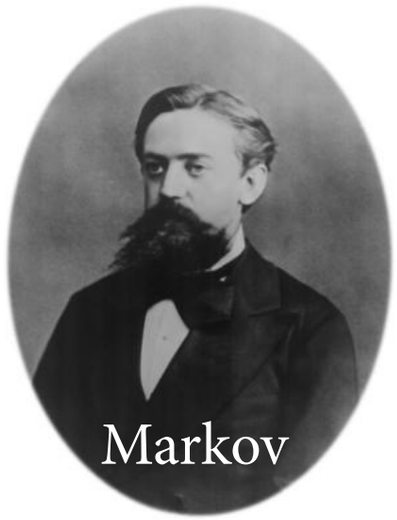
Hidden Markov Models

- Previous generative story: generate words at random from n-gram model $P(w_n \mid w_1,\ldots,w_{n-1})$.

- Replace with new generative story:
  - Language is generated by a two-step process.
  - First, generate sequence of hidden POS tags $t_1, \ldots, t_T$ tag by tag, left to right from bigram model $P(t_i \mid t_{i-1})$.
  - Independently, generate an observable word w_i from each t_i, at random from model $P(w_i \mid t_i)$.



Hidden Markov Models



- A *Hidden Markov Model* is 5-tuple consisting of
  - finite set $Q = \{q_1, \ldots, q_N\}$ of *states* (= POS tags)
  - finite set O of possible *observations* (= words)
  - *transition probabilities* $a_{ij} = P(X_{t+1} = q_j \mid X_t = q_i)$
  - *initial probabilities* $c_i = P(X_1 = q_i)$
  - *emission probabilities* $b_i(o) = P(Y_t = o \mid X_t = q_i)$

$$\sum_{j=1}^{N} a_{ij} = 1$$

$$\sum_{i=1}^{N} c_i = 1$$

$$\sum_{o \in O} b_i(o) = 1$$

- The HMM describes two coupled random processes:
  - event $X_t = q_i$: At time t, HMM is in state q_i.
  - event $Y_t = o$: At time t, HMM emits observation o.

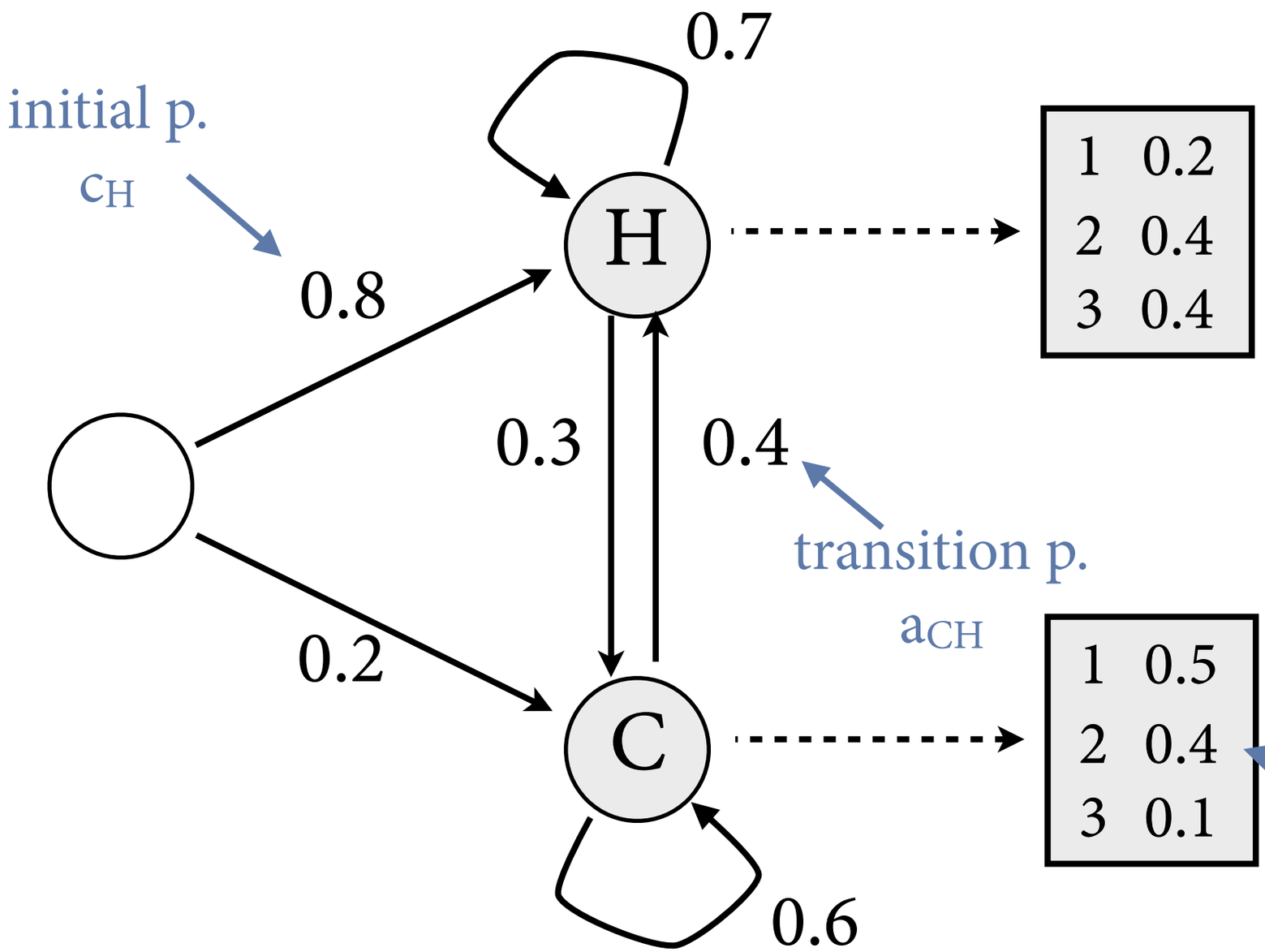
3 Questions

- Q1: Given HMM and sequence of words, what is the probability of sequence? (Language modelling.)

- Q2: Given HMM and sequence of words, what is the most likely state sequence (= sequence of tags) that explains string?

- Q3: Given (un)annotated data, how can we learn parameters of HMM?

Example: Eisner's Ice Cream

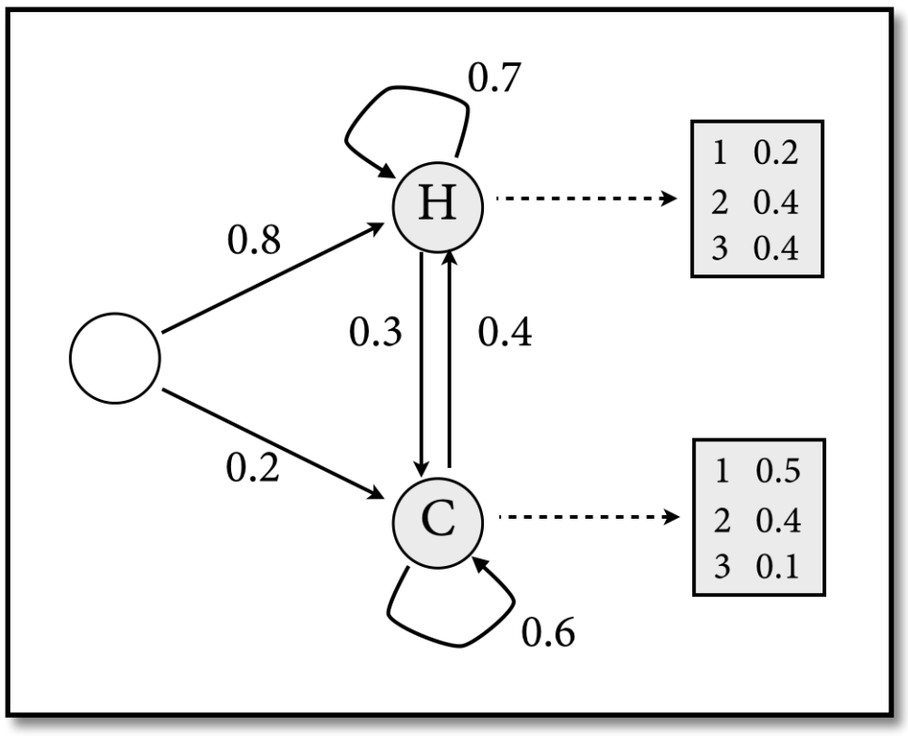


States represent weather on a given day: Hot, Cold
Outputs represent number of ice creams Jason eats that day

Naive approach

- Let's say Jason ate 3, 1, 3 ice creams. What was the most likely weather on these three days?

- Compute max $P(x_1, 3, x_2, 1, x_3, 3)$ by maximizing over all possible state sequences.



$P(x_1, 3, x_2, 1, x_3, 3)$ is max of

P(H,3,H,1,H,3)	0.013
P(H,3,H,1,C,3)	0.001
P(H,3,C,1,H,3)	0.008
...	. . .
P(C,3,C,1,C,3)	0.0004
	= 0.013

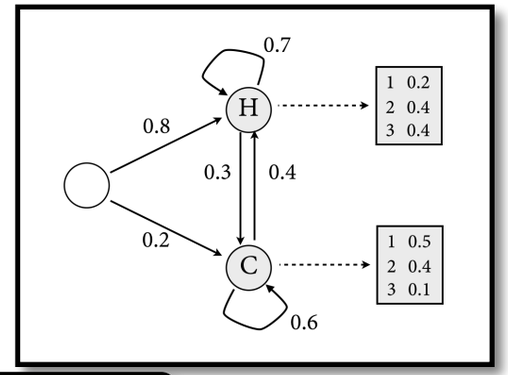
Too expensive

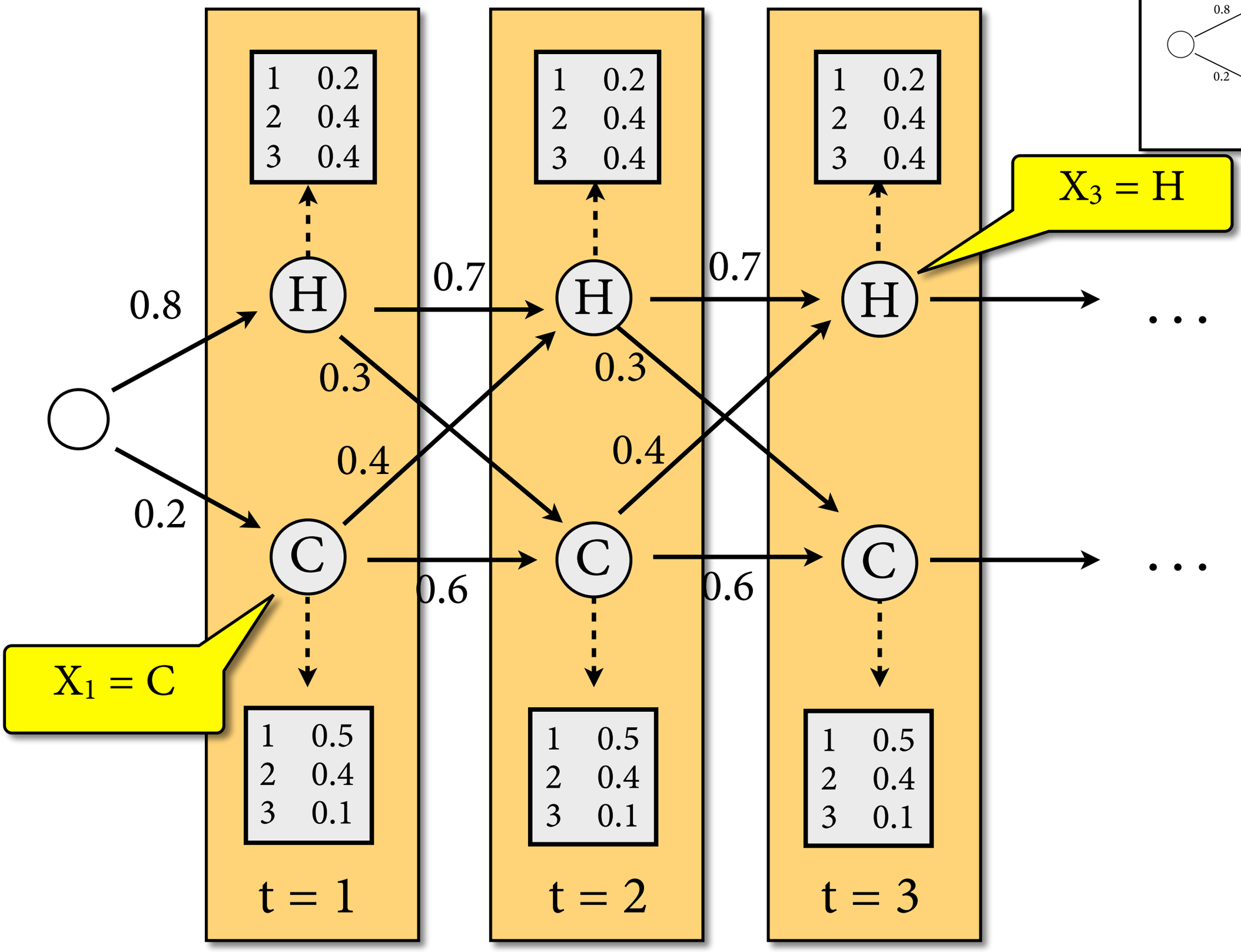
- Naive approach maximizes over exponential set of terms. This is too slow for practical use.

- Visualize this in *trellis:* unfolding of HMM
  - one column for each time point t, represents X_t
  - each column contains a copy of each state of HMM
  - edges from t to t+1 = transitions of HMM

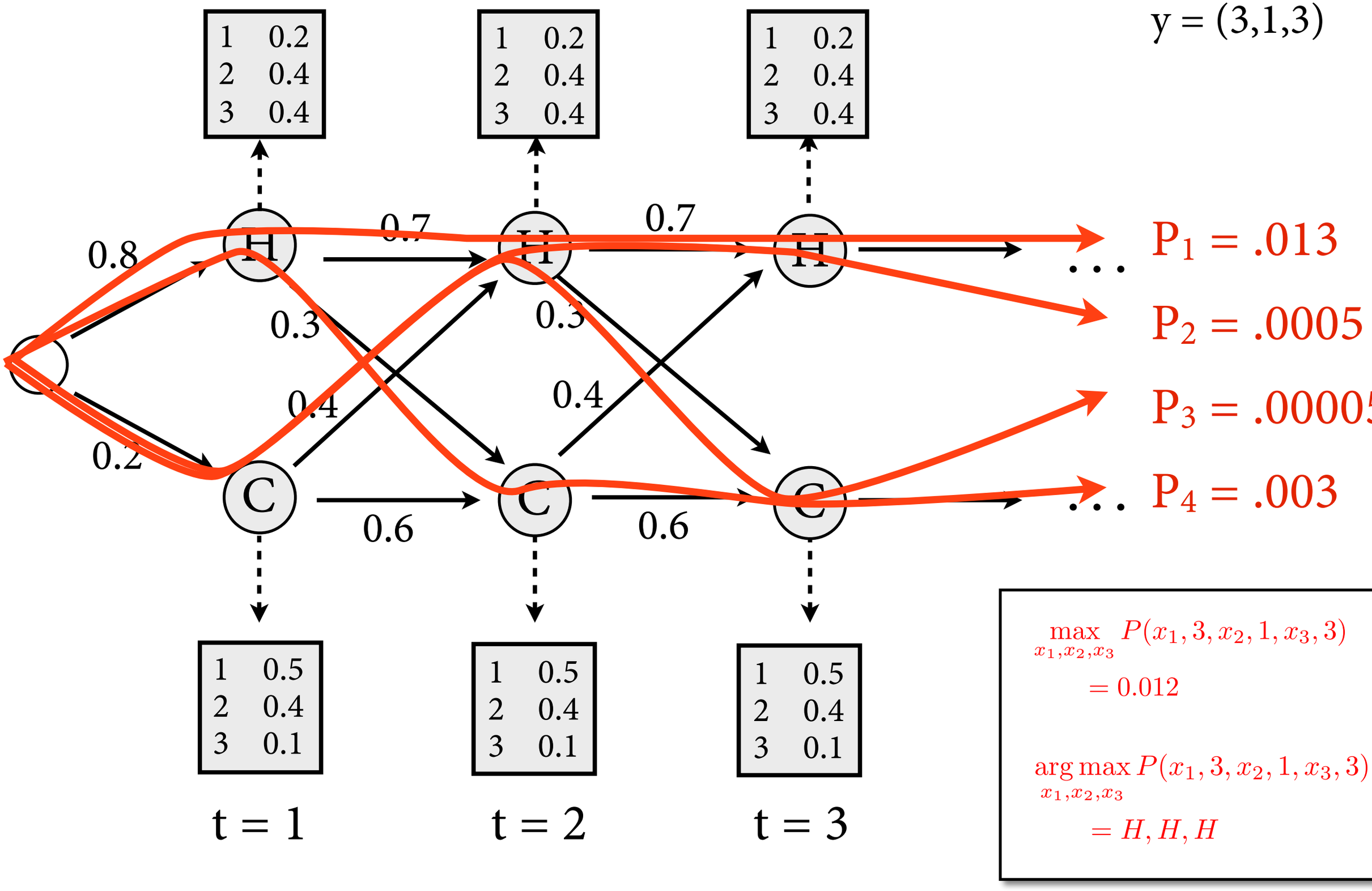
- Each path through trellis represents one state sequence.
  - So computation of max P(x,y) = max over all paths.

Ice cream trellis





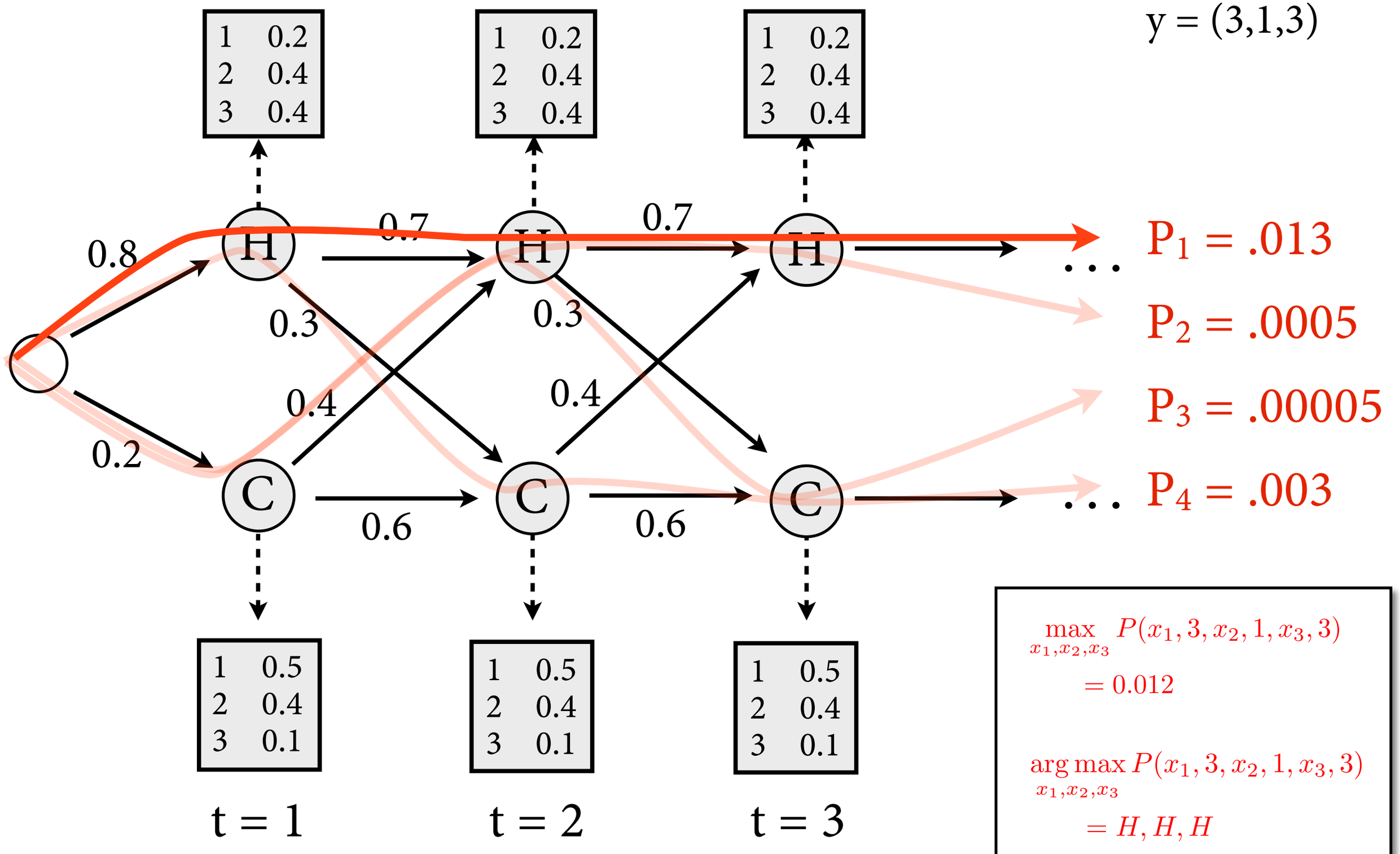
Ice cream trellis



$$\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = 0.012$$

$$\arg\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = H, H, H$$

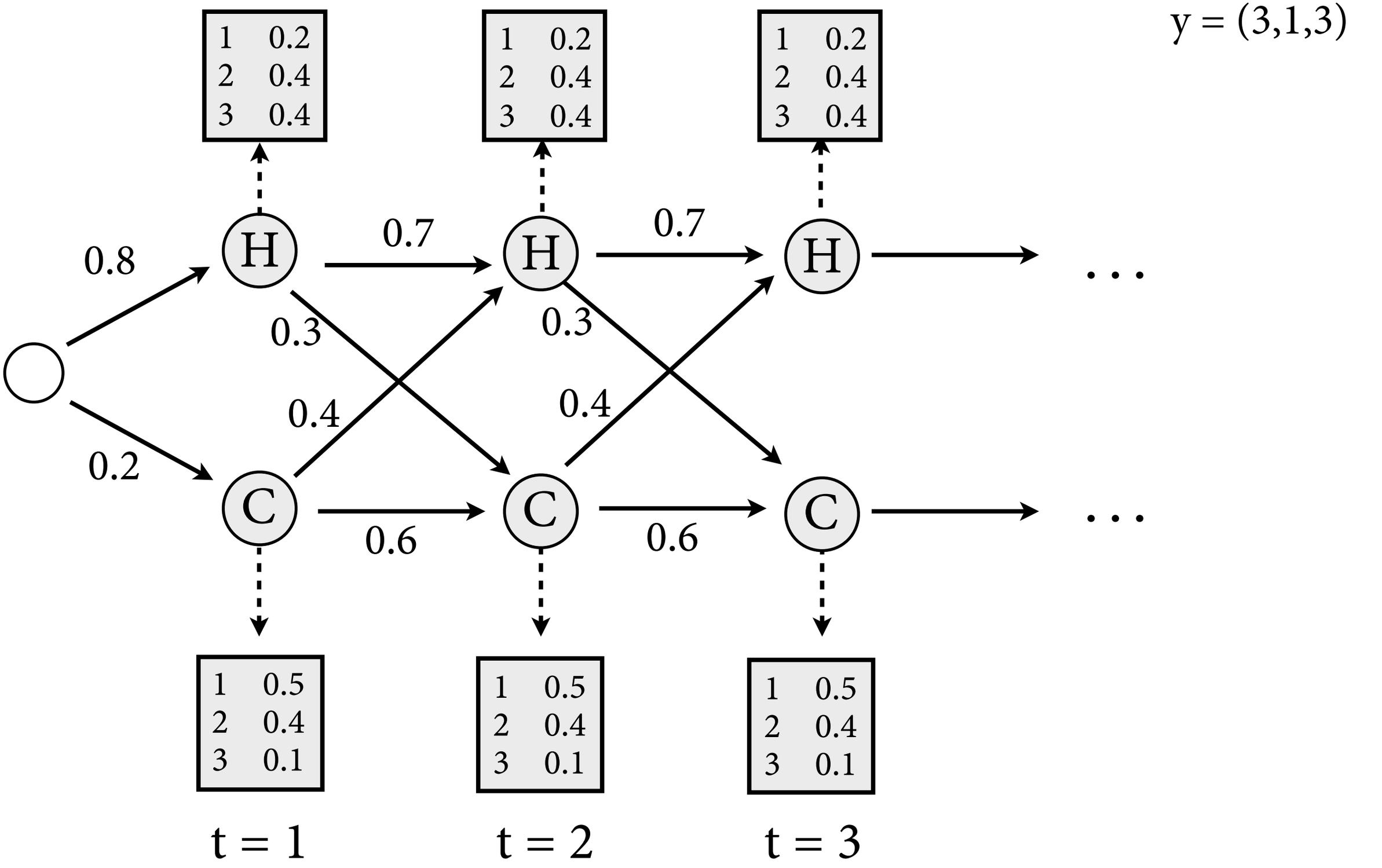
Ice cream trellis



$$\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = 0.012$$

$$\arg\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = H, H, H$$

Ice cream trellis



The Viterbi Algorithm

- Because of statistical independencies, can decompose joint probability:

$$\begin{aligned} P(x_1, y_1, \ldots, x_t, y_t) &= P(y_t \mid x_1, \ldots, x_t, y_1, \ldots, y_{t-1}) \cdot P(x_t \mid x_1, \ldots, x_{t-1}, y_1, \ldots, y_{t-1}) \\ &\quad \cdot P(x_1, y_1, \ldots, x_{t-1}, y_{t-1}) \\ &= P(y_t \mid x_t) \cdot P(x_t \mid x_{t-1}) \cdot P(x_1, y_1, \ldots, x_{t-1}, y_{t-1}) \end{aligned}$$

- Thus, maximum has recursive structure:

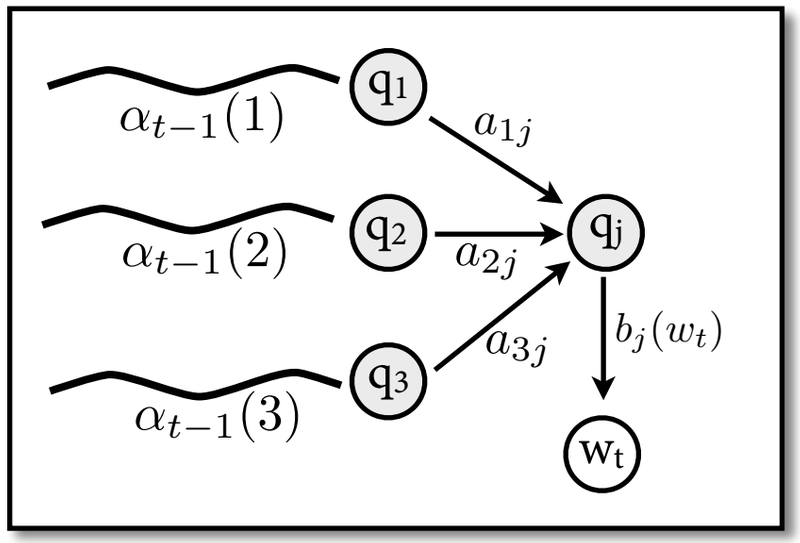
$$\begin{aligned} V_t(j) &= \max_{x_1,\ldots,x_{t-1}} P(y_1, \ldots, y_t, x_1, \ldots, x_{t-1}, X_t = q_j) \\ &= \max_{x_1,\ldots,x_{t-1}} P(y_t \mid X_t = q_j) \cdot P(X_t = q_j \mid x_{t-1}) \cdot P(y_1, \ldots, y_{t-1}, x_1, \ldots, x_{t-1}) \\ &= \max_{x_{t-1}} P(y_t \mid X_t = q_j) \cdot P(X_t = q_j \mid x_{t-1}) \cdot (\max_{x_1,\ldots,x_{t-2}} P(y_1, \ldots, y_{t-1}, x_1, \ldots, x_{t-1})) \\ &= \max_{i} P(y_t \mid X_t = q_j) \cdot P(X_t = q_j \mid X_{t-1} = q_i) \cdot (\max_{x_1,\ldots,x_{t-2}} P(y_1, \ldots, y_{t-1}, x_1, \ldots, X_{t-1} = q_i)) \\ &= \max_{i} P(y_t \mid X_t = q_j) \cdot P(X_t = q_j \mid X_{t-1} = q_i) \cdot V_{t-1}(i) \end{aligned} \quad \boxed{= \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)}$$

The Viterbi Algorithm

$$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t = q_j)$$

- Base case, t = 1:

$$V_1(j) = b_j(y_1) \cdot a_{0j}$$



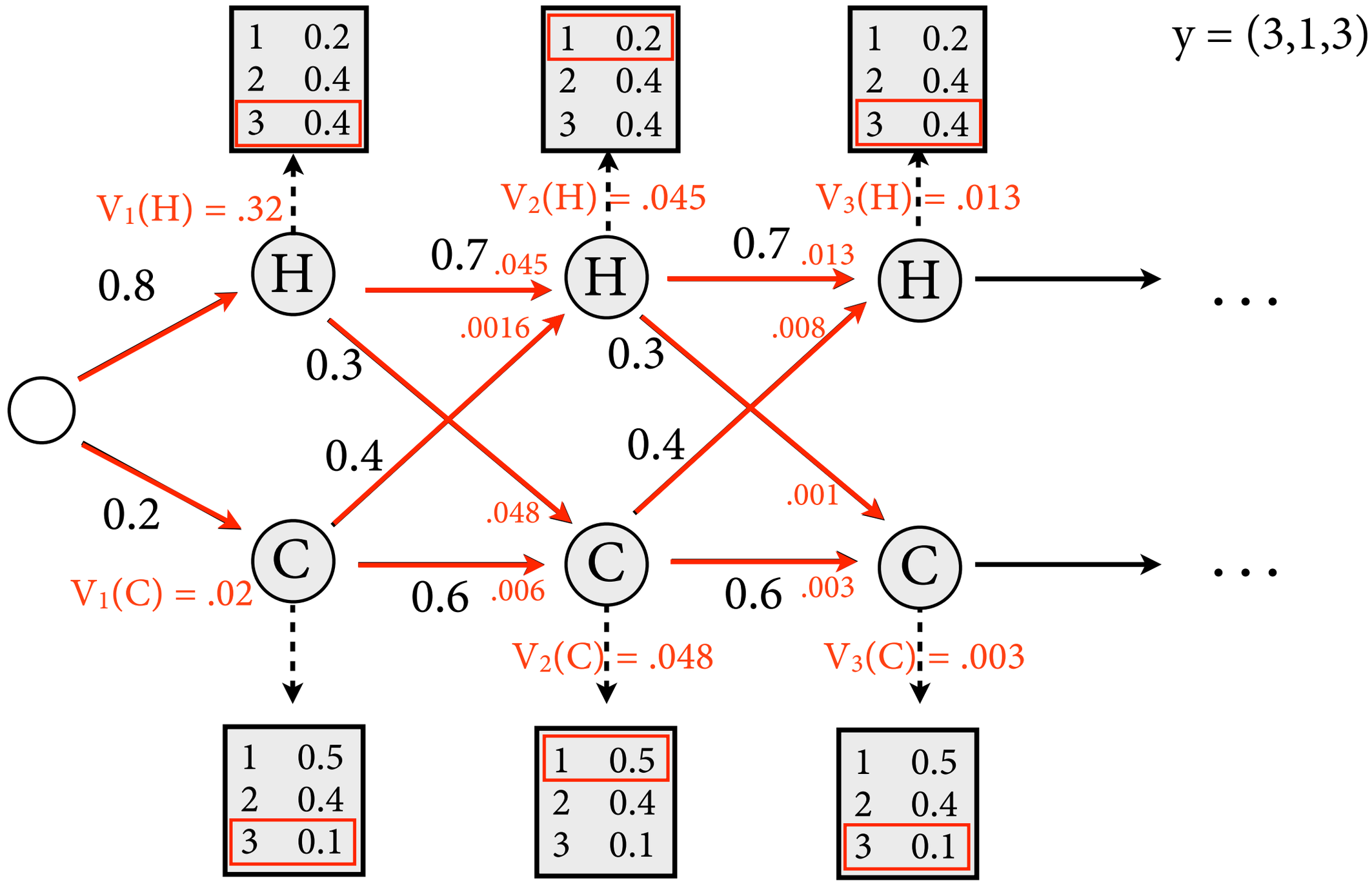
- Inductive case, for t = 2, …, T:

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

- Once we have calculated all V values, we can easily calculate prob of best path:

$$\max_{x_1,\ldots,x_T} P(x_1,y_1,\ldots,x_T,y_T) = \max_{q \in Q} V_T(q)$$

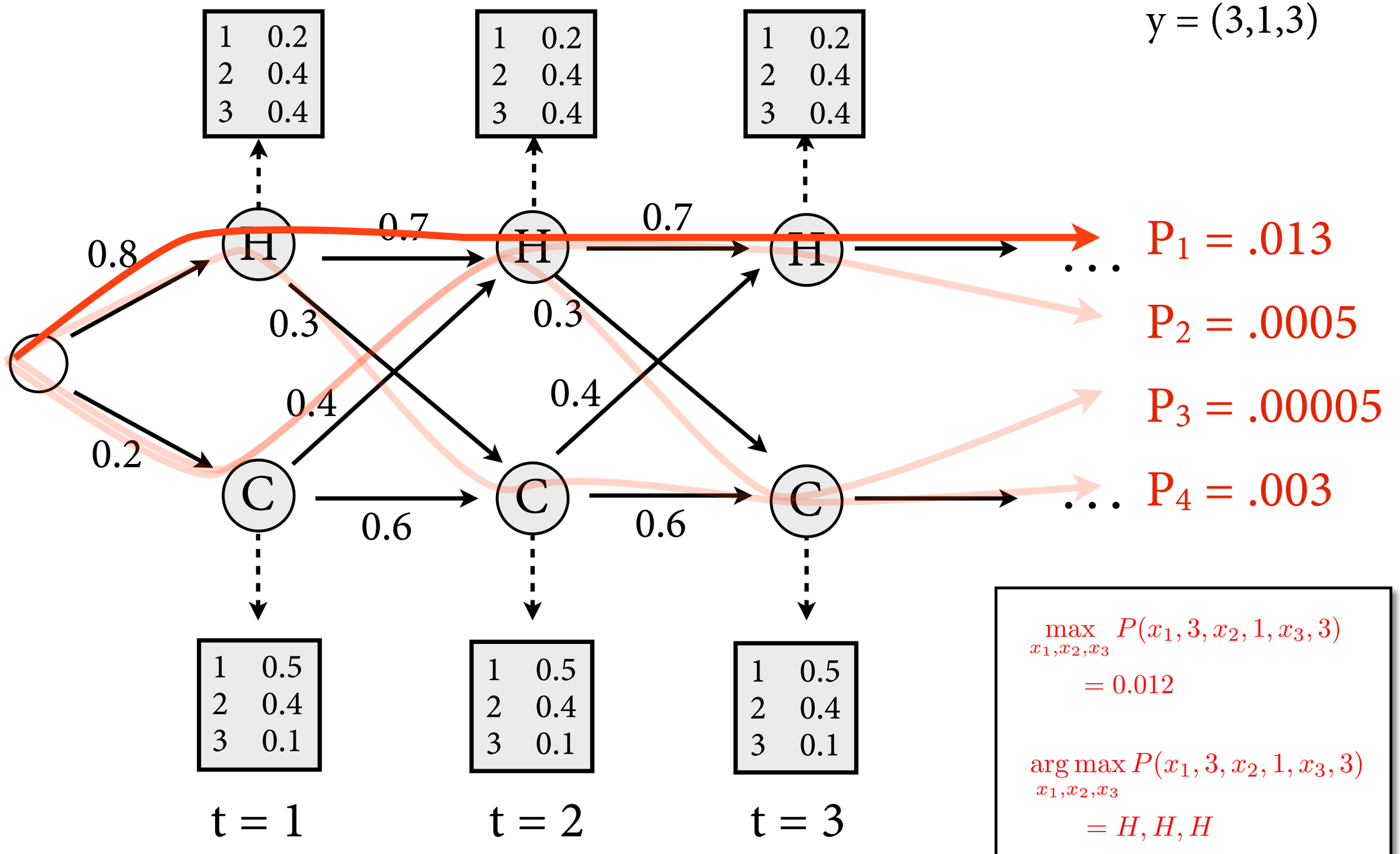
Viterbi Algorithm: Example



$$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t = q_j)$$

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i)\cdot a_{ij}\cdot b_j(y_t)$$

Ice cream trellis



$$\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = 0.012$$

$$\arg\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = H, H, H$$

Backpointers

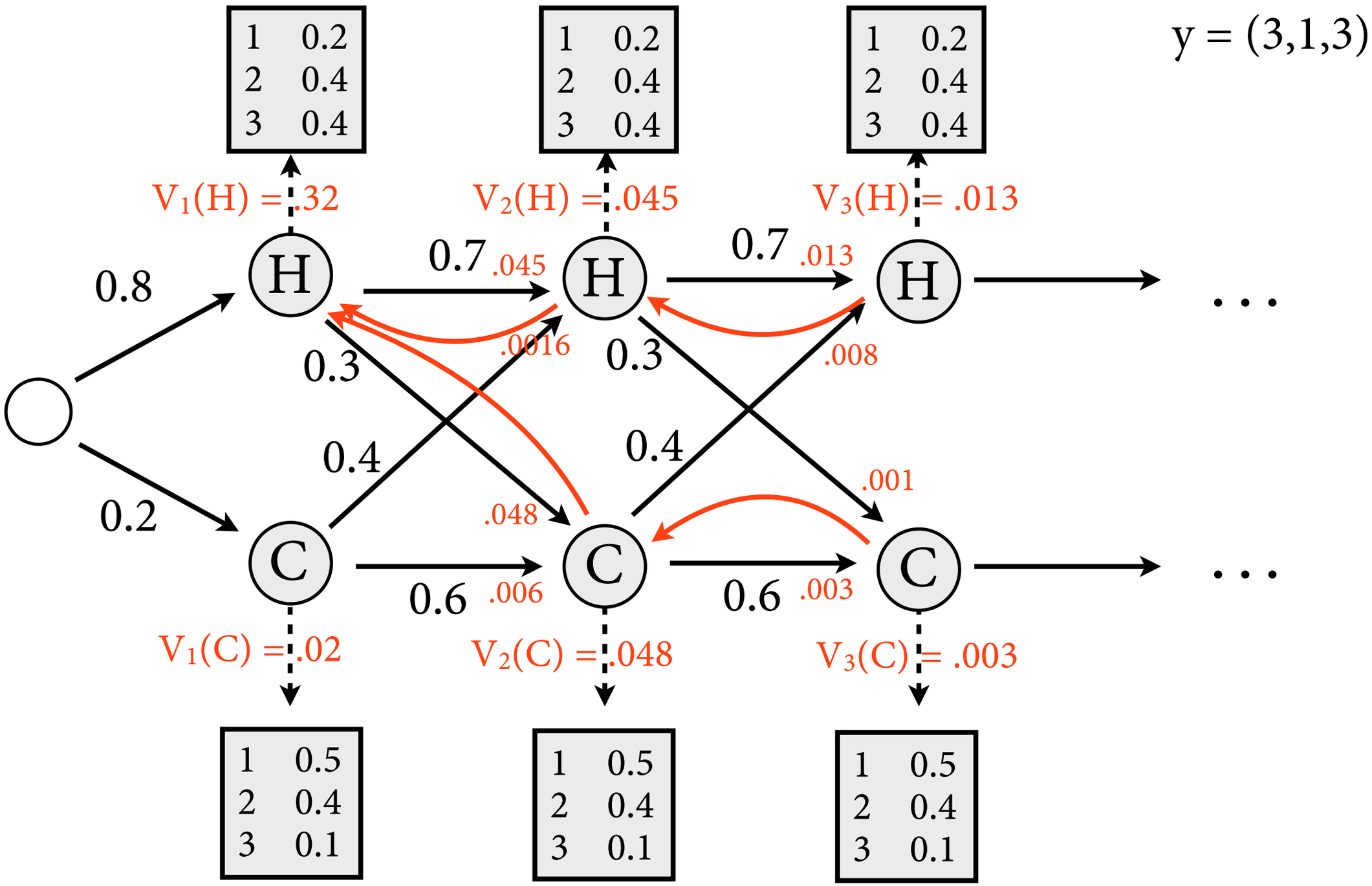
- In the end, we need to reconstruct the sequence of states $x_1, \ldots, x_T$ with max probability.

- For each t, j: remember the value of i for which the maximum was achieved in *backpointer* $bp_t(j)$.

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

- Then just follow backpointers from right to left.

Viterbi Algorithm: Example



$$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t = q_j)$$

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i)\cdot a_{ij}\cdot b_j(y_t)$$

Question 1: Likelihood, P(y)

- How likely is it that Jason Eisner ate 3 ice creams on day 1, 1 ice cream on day 2, 3 ice creams on day 3?

- Want to compute: P(3, 1, 3).

- Same problem as with max:
  - Output 3, 1, 3 can be emitted by many different state sequences.
  - Obtain by marginalization:

$$P(3,1,3) = \sum_{x_1,x_2,x_3 \in Q} P(x_1,3,x_2,1,x_3,3)$$

  - Naive computation is far too slow.

The Forward Algorithm

- Key idea: *Forward probability* $\alpha_t(j)$ that HMM outputs $y_1, \ldots, y_t$ and then ends in $X_t = q_j$.

$$\begin{aligned}\alpha_t(j) &= P(y_1, \ldots, y_t, X_t = q_j) \\ &= \sum_{x_1, \ldots, x_{t-1}} P(y_1, \ldots, y_t, X_1 = x_1, \ldots, X_{t-1} = x_{t-1}, X_t = q_j)\end{aligned}$$

- From this, can compute easily

$$P(y_1, \ldots, y_T) = \sum_{q \in Q} \alpha_T(q)$$

The Forward Algorithm

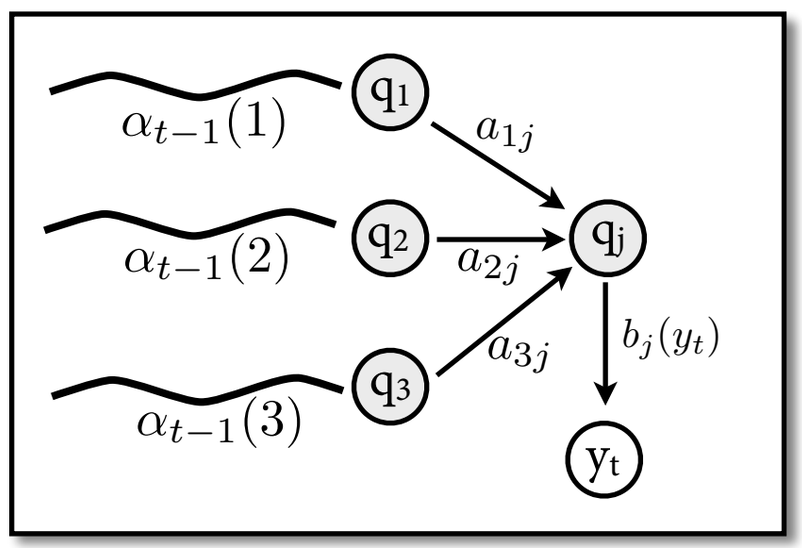
$$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$$

- Base case, t = 1:

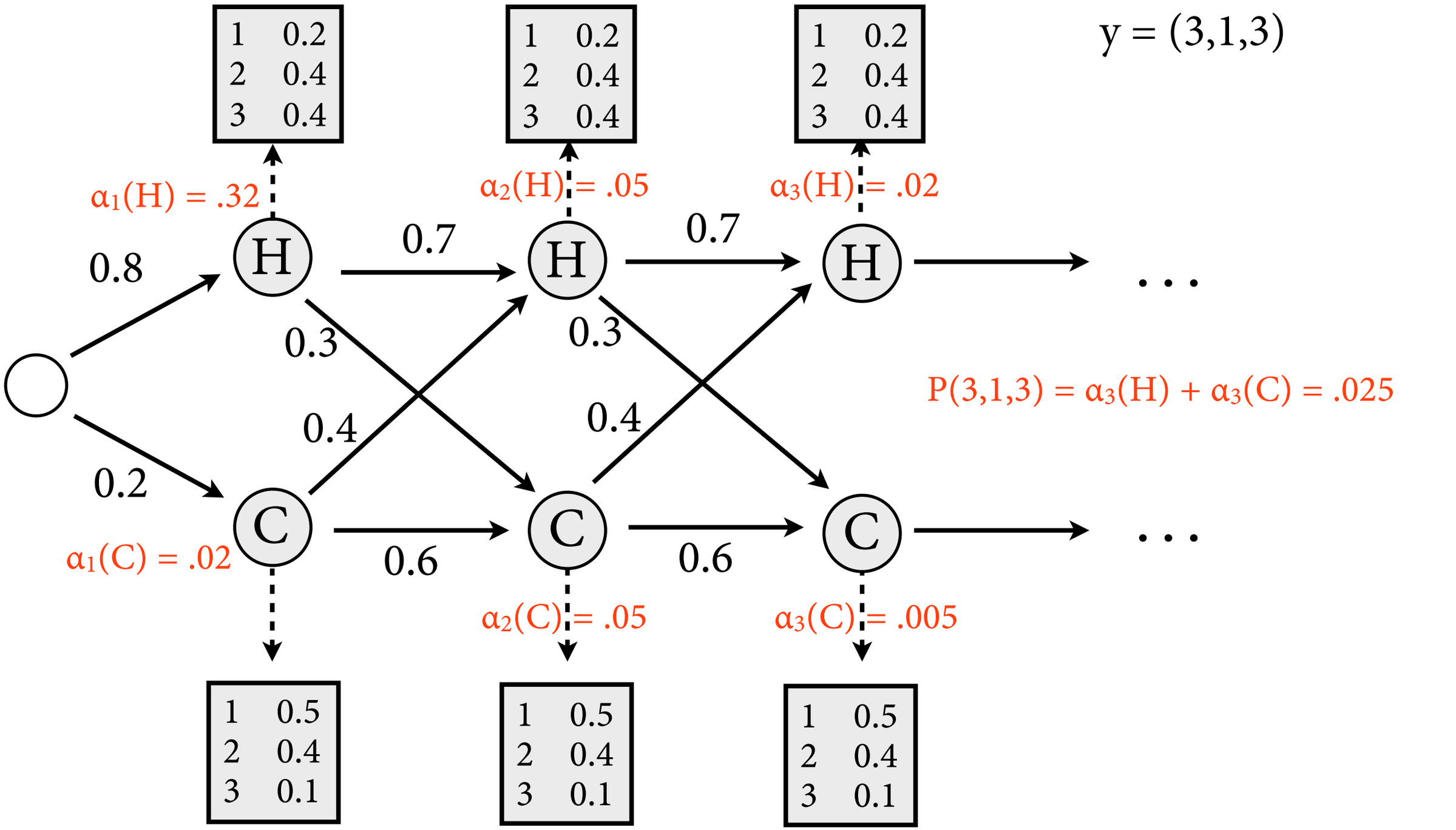
$$\alpha_1(j) = P(y_1, X_1 = q_j) = b_j(y_1) \cdot a_{0j}$$

- Inductive case, compute for t = 2, …, T:

$$\begin{aligned}\alpha_t(j) &= P(y_1, \ldots, y_t, X_t = q_j) \\ &= \sum_{i=1}^{N} P(y_1, \ldots, y_{t-1}, X_{t-1} = q_i) \cdot P(X_t = q_j \mid X_{t-1} = q_i) \cdot P(y_t \mid X_t = q_j) \\ &= \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)\end{aligned}$$



P(3,1,3) with Forward



$$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j) \qquad \alpha_1(j) = b_j(y_1) \cdot a_{0j} \qquad \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

Runtime

- Forward and Viterbi have the same runtime, dominated by inductive step:

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

- Computation of $V_t(j)$ requires iteration over N predecessor states. We do this for N states.

- Total runtime is $O(N^2 \cdot T)$, i.e.
  - linear in sentence length
  - quadratic in size of tag set

Summary

- Hidden Markov Models popular model for POS tagging (and other applications, see later).

- Two coupled random processes:
  - bigram model for hidden states ("Markov Chain")
  - model for producing observable output from each state

- Efficient algorithms for common problems:
  - Likelihood computation: Forward algorithm
  - Best state sequence: Viterbi algorithm.

- Computational advice: work with logs to avoid underflow.

Question 3a: Supervised learning

- Given a set of POS tags and *annotated* training data (w_1,t_1), …, (w_T,t_T), compute parameters for HMM that maximize likelihood of training data.

DT	NN	VBD	NNS	IN	DT	NN
The	representative	put	chairs	on	the	table.

NNP	VBZ	VBN	TO	VB	NR
Secretariat	is	expected	to	race	tomorrow.

Maximum likelihood training

- Estimate bigram model for state sequence:

$$a_{ij} = \frac{C(X_t = q_i, X_{t+1} = q_j)}{C(X_t = q_i)} \qquad c_j = \frac{\#\text{ sentences with } X_1 = q_j}{\#\text{ sentences}}$$

- ML estimate for emission probabilities:

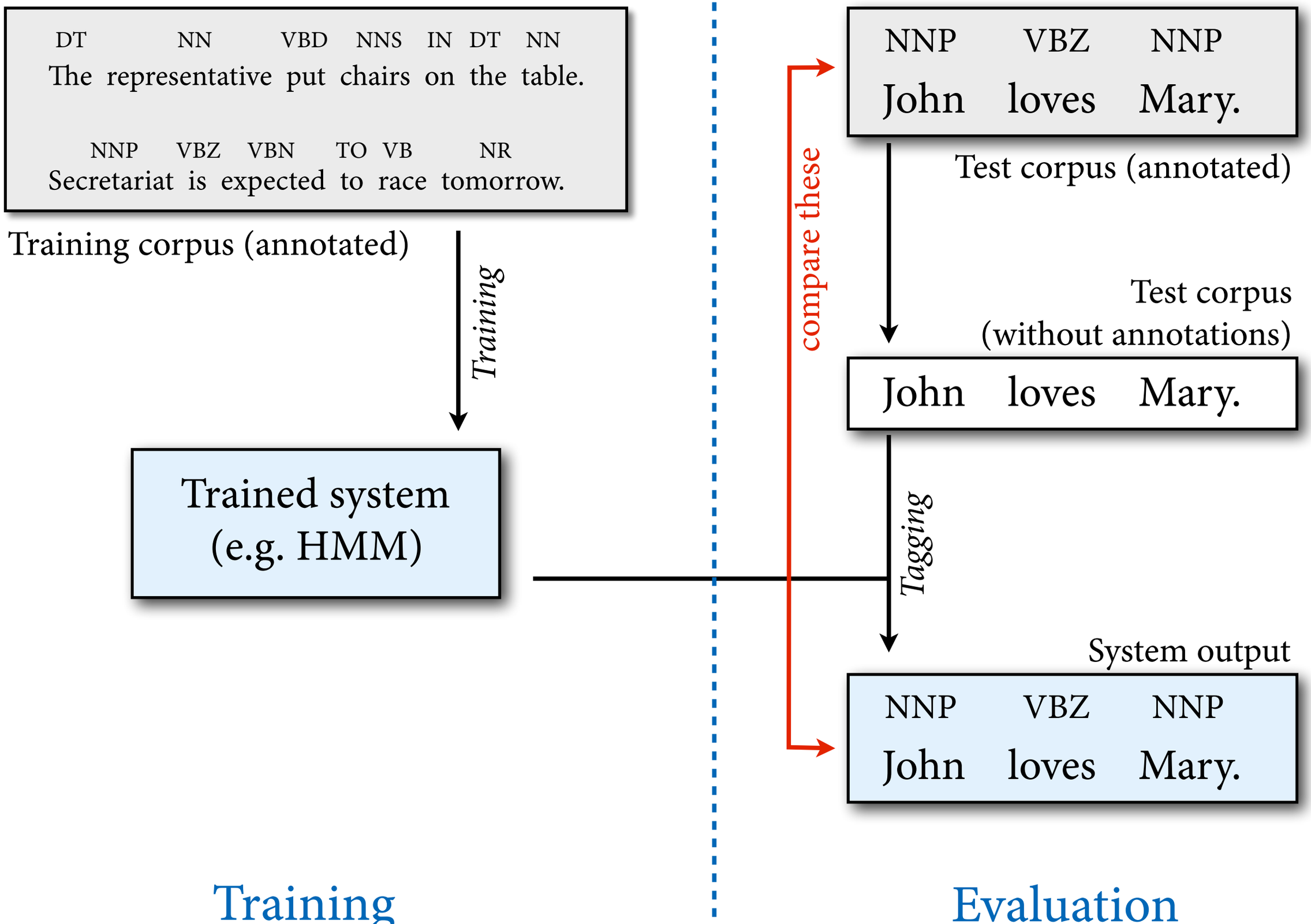
$$b_i(o) = \frac{C(X_t = q_i, Y_t = o)}{C(X_t = q_i)}$$

- Apply smoothing as you would for ordinary n-gram models (increase all counts C by one).

Evaluation

- How do you know how well your tagger works?
- Run it on *test data* and evaluate *accuracy*.
  - Test data: Really important to evaluate on unseen sentences to get a fair picture of how well tagger generalizes.
  - Accuracy: Measure percentage of correctly predicted POS tags.

Evaluation on test data



Question 3b: Unsupervised learning

- Given a set of POS tags and *unannotated* training data $w_1, \ldots, w_T$, compute parameters for HMM that maximize likelihood of training data.

- Relevant because annotated data is expensive to obtain, but raw text is really cheap.

The representative put chairs on the table.

Secretariat is expected to race today.

doesn't work very well for POS tagging
can be done using Expectation/Maximization
will skip that here

Trigram Taggers

NNP VBZ NN NNS CD NN

Fed raises interest rates 0.5 percent

- Trigram model: y_1 = (<S>, NNP), y_2 = (NNP, VBZ), …
- P((VBZ, NN) | (NNP, VBZ)) — more context! Noun-verb-noun S-V-O
- Tradeoff between model capacity and data size — trigrams are a “sweet spot” for POS tagging

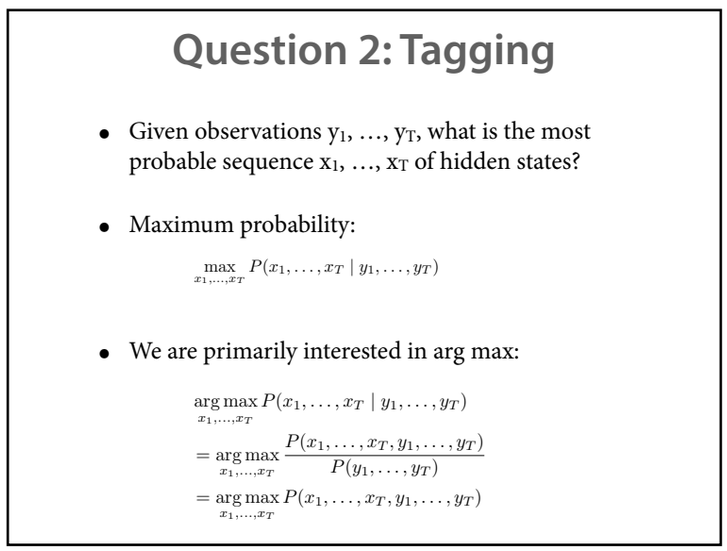
HMM POS Tagging

- Baseline: assign each word its most frequent tag: ~90% accuracy
- Trigram HMM: ~95% accuracy / 55% on unknown words
- TnT tagger (Brants 1998, tuned HMM): 96.2% accuracy / 86.0% on unks
- State-of-the-art (BiLSTM-CRFs): 97.5% / 89%+

Slide credit: Dan Klein

slide credits

slides that look like this



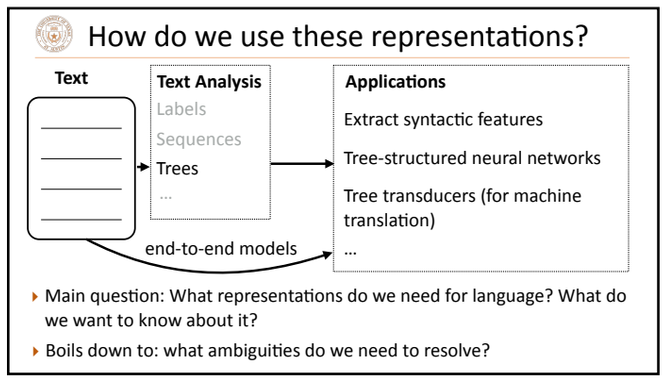
Question 2: Tagging

- Given observations $y_1, \ldots, y_T$, what is the most probable sequence $x_1, \ldots, x_T$ of hidden states?
- Maximum probability:

$$\max_{x_1,\ldots,x_T} P(x_1, \ldots, x_T \mid y_1, \ldots, y_T)$$

- We are primarily interested in arg max:

$$\begin{aligned}
&\arg\max_{x_1,\ldots,x_T} P(x_1, \ldots, x_T \mid y_1, \ldots, y_T) \\
&= \arg\max_{x_1,\ldots,x_T} \frac{P(x_1, \ldots, x_T, y_1, \ldots, y_T)}{P(y_1, \ldots, y_T)} \\
&= \arg\max_{x_1,\ldots,x_T} P(x_1, \ldots, x_T, y_1, \ldots, y_T)
\end{aligned}$$



come from

earlier editions of this class (ANLP), given by Alexander Koller

CS388 given by Greg Durrett at U Texas, Austin

and their use is gratefully acknowledged. I try to make any modifications obvious, but if there are errors on a slide, assume that I added them.