Hidden Markov models

CSC401/2511 – Natural Language Computing – Winter 2023
Lecture 8
University of Toronto

Logistics (Mar 1, 2023)

- **A2 released** on Feb 11, due Mar 10
- Please do not share assignment codes after you are done
- A2 tutorials' planned schedule:
  - ~~Feb 17: A2 tutorial – 1: Intro to PyTorch (ft. Gavin Guan).~~
  - Mar 3: A2 tutorial – 2: Machine translation (ft. Frank Niu)
  - Mar 10: A2 – Q/A and OH (*submission due at mid-night*)

- Office hours: Wed 12.30 am – 1.30 pm (zoom, note the channel)
- Final exam: planned in-person
- Lecture feedback form:
  - Anonymous
  - Please share any thoughts/suggestions
- **Questions?**

Scan Me

Observable Markov model

- We've seen this type of model:
  - e.g., consider the 7-word vocabulary:

    $$\{ship, pass, camp, frock, soccer, mother, tops\}$$

  - What is the probability of the **sequence**

    $$ship, ship, pass, ship, tops \qquad ?$$

  - Assuming a **bigram** model (i.e., **1st-order Markov**),

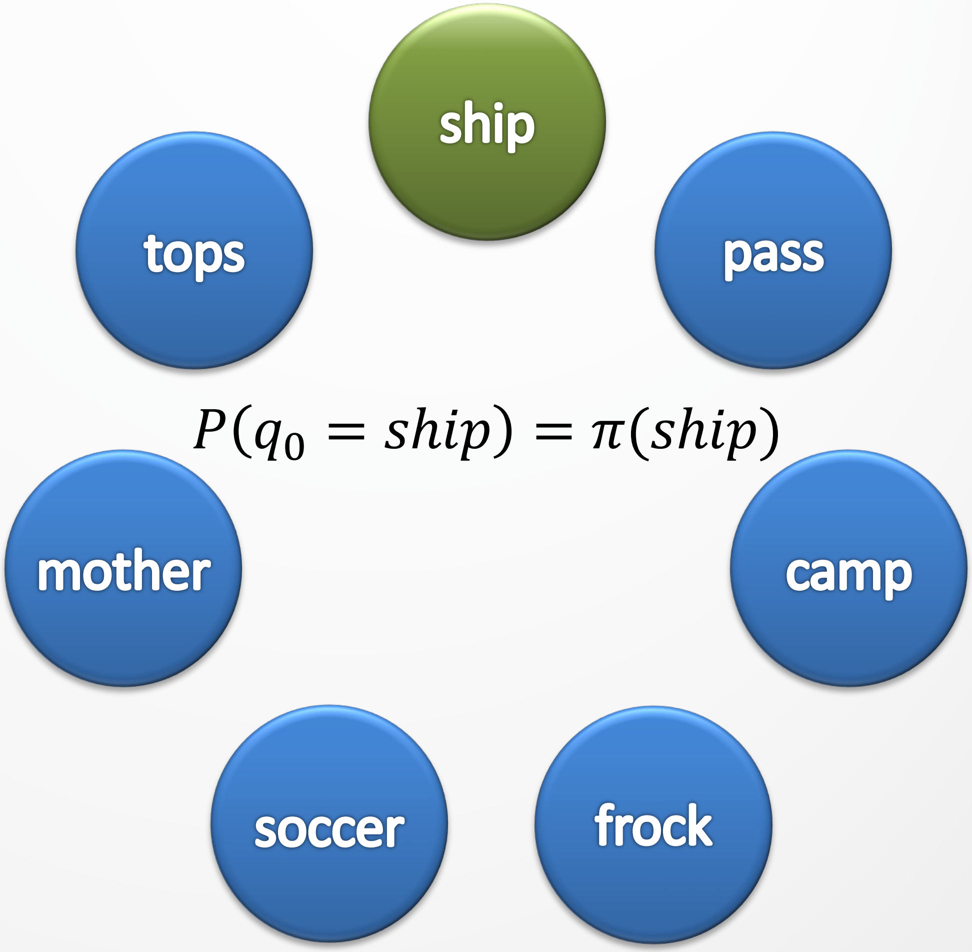
    $$P(ship|\text{<s>})P(ship|ship)P(pass|ship) \cdot P(ship|pass)P(tops|ship)$$

Observable Markov model

- This can be conceptualized graphically.
- We start with N **states**, $s_1, s_2, \dots, s_N$ that represent unique observations in the world.
- Here, $N = 7$ and each state represents one of the words we can observe.

Observable Markov model

- We have discrete **timesteps**, $t = 0, t = 1, \ldots$
- On the t^{th} timestep the system is in exactly one of the available states, $\boldsymbol{q_t}$.
  - $q_t \in \{s_1, s_2, \ldots, s_N\}$
- We could start in any state. The probability of starting with a particular state s is $P(q_0 = s) = \boldsymbol{\pi(s)}$



ship

tops

pass

$P(q_0 = ship) = \pi(ship)$

mother

camp

soccer

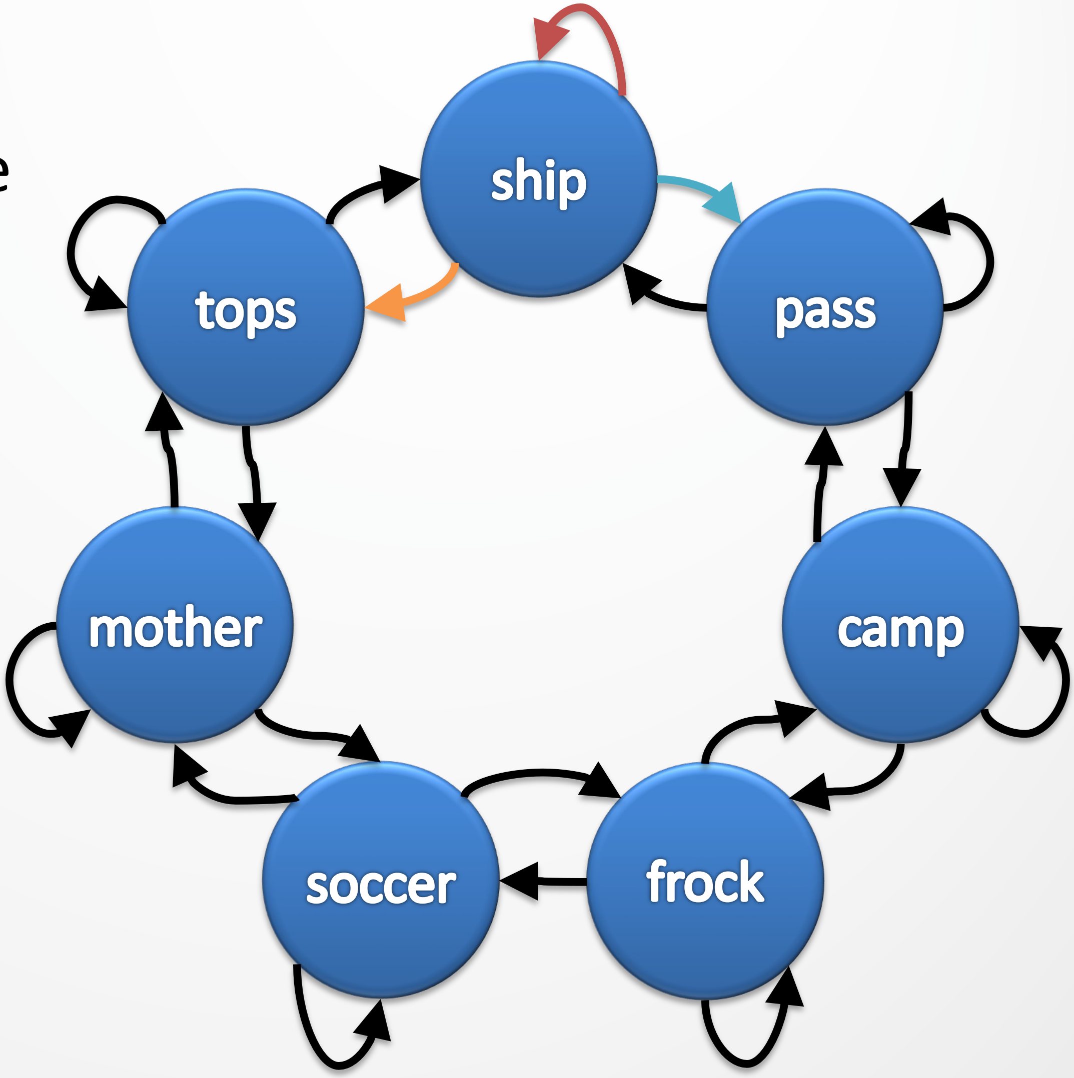
frock

Observable Markov model

- At each step we must move to a state with some probability.

- Here, an arrow from q_t to q_{t+1} represents $P(q_{t+1}|q_t)$

- $P(ship|ship)$
- $P(tops|ship)$
- $P(pass|ship)$
- $P(frock|ship) = 0$



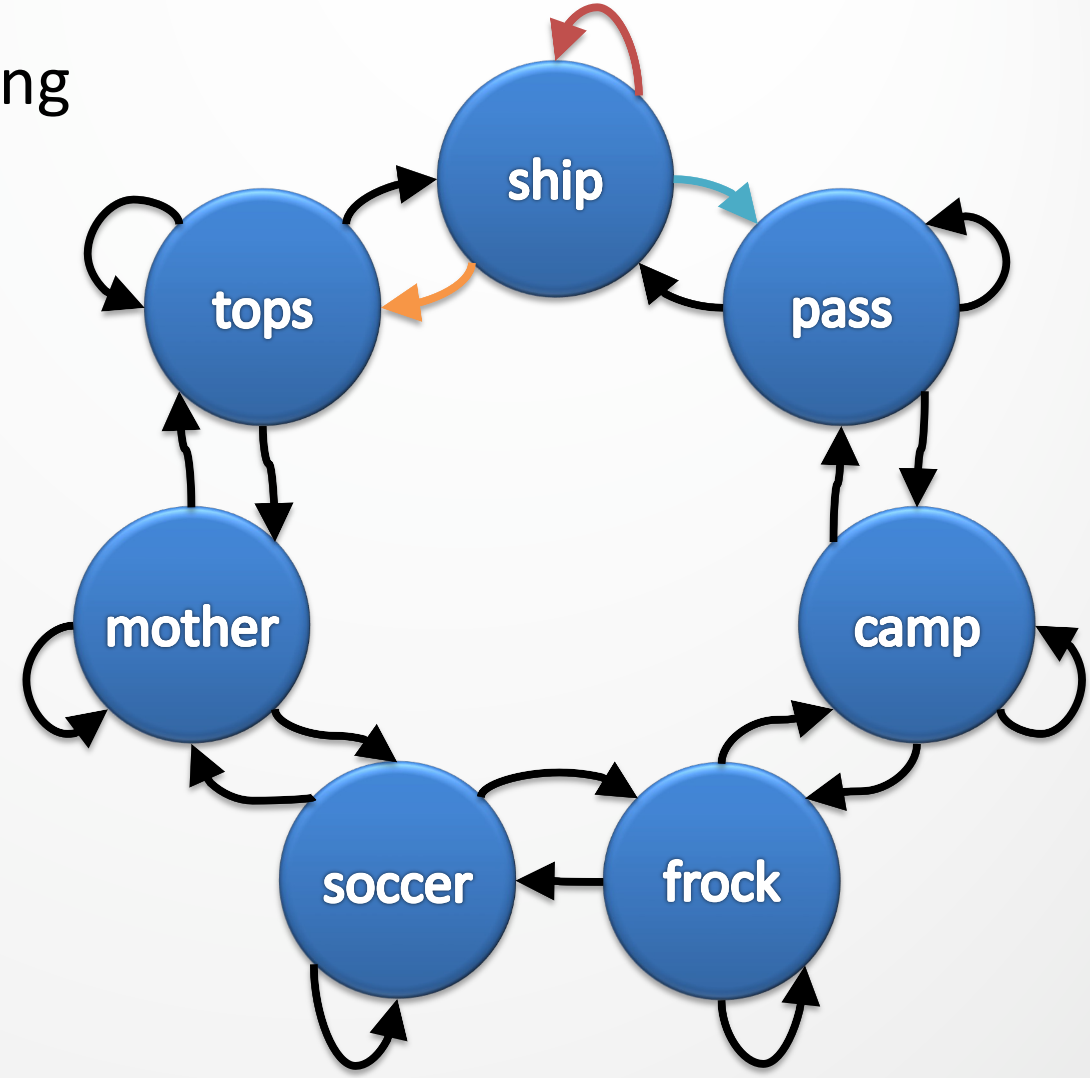
Observable Markov model

- Probabilities on all outgoing arcs must sum to 1.

- $P(ship|ship) +$
  $P(tops|ship) +$
  $P(pass|ship) = 1$

- $P(ship|tops) +$
  $P(tops|tops) +$
  $P(mother|tops) = 1$

- …



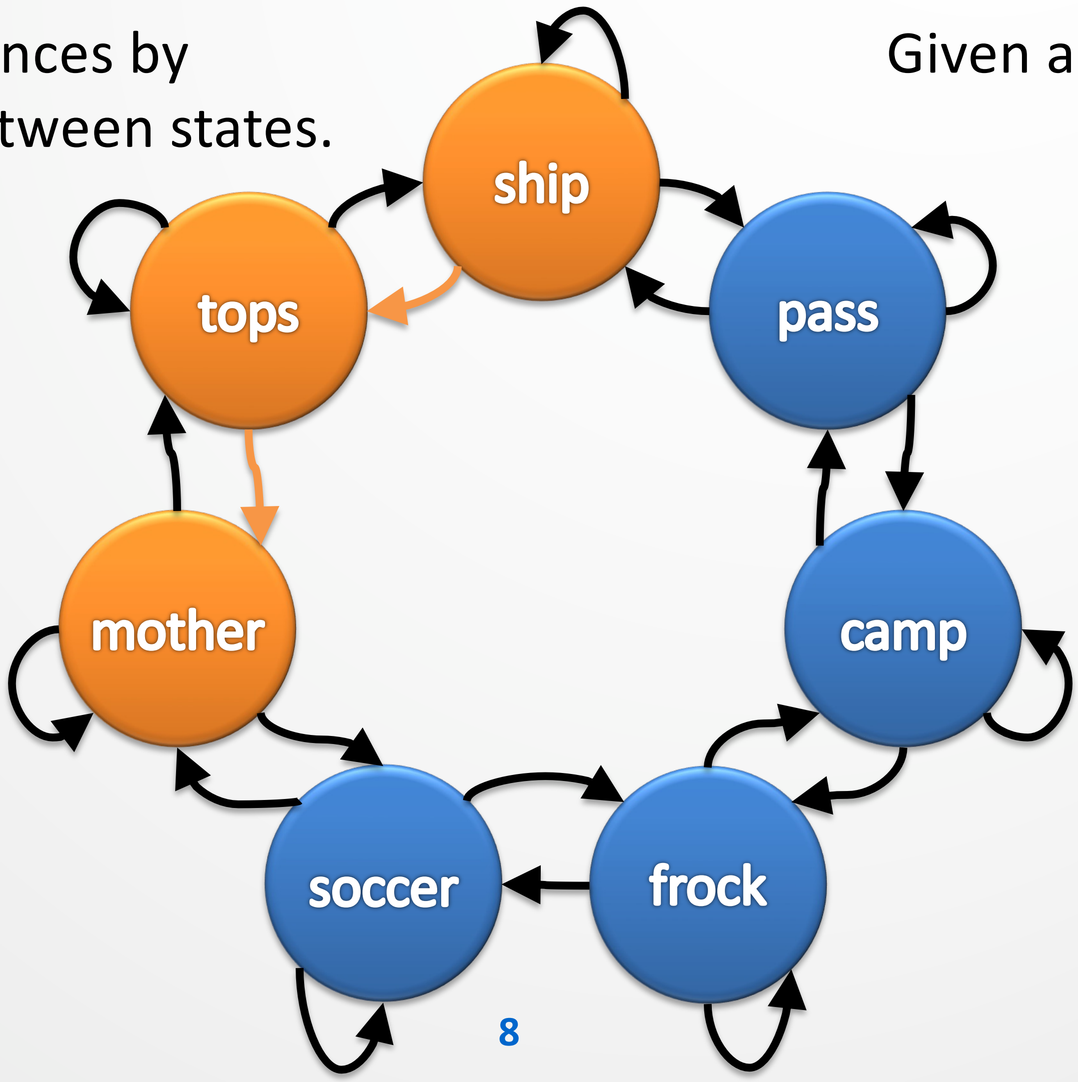
Using the graph

Random walk

Generate sequences by transitioning between states.

Observation likelihood

Given a path, *build* its probability.



A multivariate system

- What if the probabilities of observing words depended ***only*** on some ***other*** variable, like ***mood***?

word	P(word)
ship	0.1
pass	0.05
camp	0.05
frock	0.6
soccer	0.05
mother	0.1
tops	0.05

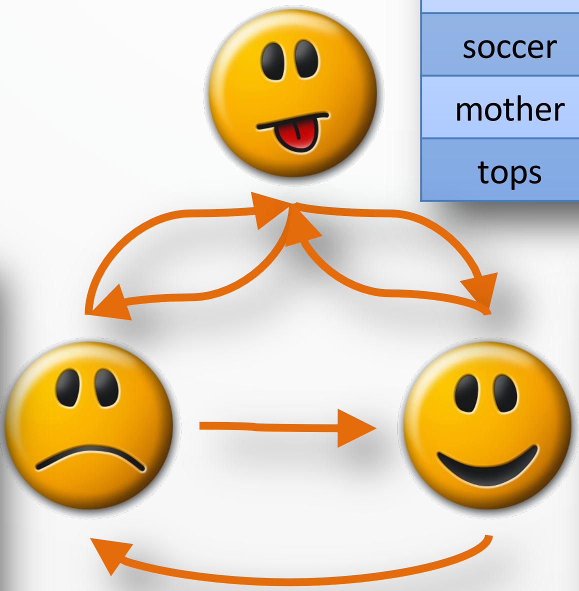
word	P(word)
ship	0.25
pass	0.25
camp	0.05
frock	0.3
soccer	0.05
mother	0.09
tops	0.01

word	P(word)
ship	0.3
pass	0
camp	0
frock	0.2
soccer	0.05
mother	0.05
tops	0.4

A multivariate system

- What if that variable **changes** over time?
  - e.g., I'm **happy** one second and **disgusted** the next.
- Here, **state** ≡ mood

  observation ≡ word.



word	P(word)
ship	0.1
pass	0.05
camp	0.05
frock	0.6
soccer	0.05
mother	0.1
tops	0.05

word	P(word)
ship	0.25
pass	0.25
camp	0.05
frock	0.3
soccer	0.05
mother	0.09
tops	0.01

word	P(word)
ship	0.3
pass	0
camp	0
frock	0.2
soccer	0.05
mother	0.05
tops	0.4

Observable multivariate systems

- Imagine you have access to my emotional state somehow.
- All your data are completely **observable** at every time step.
- E.g.,

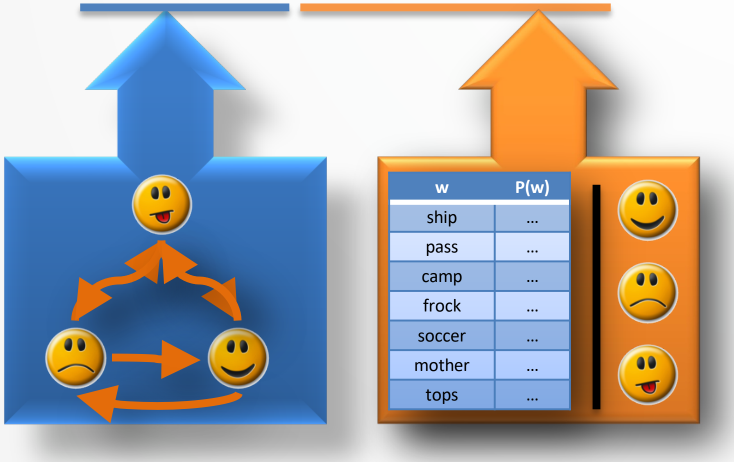
t	*0*	*1*	*2*	…
state	🙂	🙂	😛	…
word	*mother*	*frock*	*soccer*	…

$$\equiv$$

$\langle mother, frock, soccer \rangle$, ⟨🙂,🙂,😛⟩

Observable multivariate systems

- What is the probability of a sequence of words and states?
  - $P(w_{0:t}, q_{0:t}) = \underline{P(q_{0:t})}\,\underline{P(w_{0:t}|q_{0:t})} \approx \prod_{i=0}^{t} P(q_i|q_{i-1})P(w_i|q_i)$



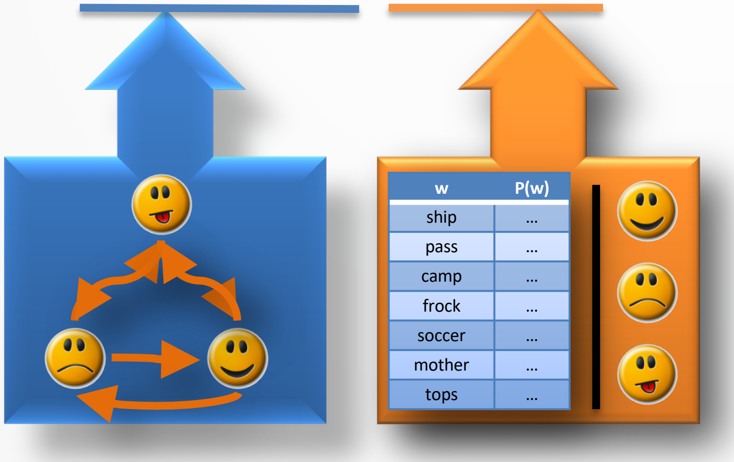
w	P(w)
ship	…
pass	…
camp	…
frock	…
soccer	…
mother	…
tops	…

- e.g.,

$$P(\langle ship, pass\rangle, \langle \text{🙂}, \text{😛}\rangle) = \underline{P(q_0 = \text{🙂})}\,\underline{P(ship|\text{🙂})}\,\underline{P(\text{😛}|\text{🙂})}\,\underline{P(pass|\text{😛})}$$

Observable multivariate systems

- **Q**: How do you **learn** these probabilities?
  - $P(w_{0:t}, q_{0:t}) \approx \prod_{i=0}^{t} \underline{P(q_i|q_{i-1})}\,\underline{P(w_i|q_i)}$



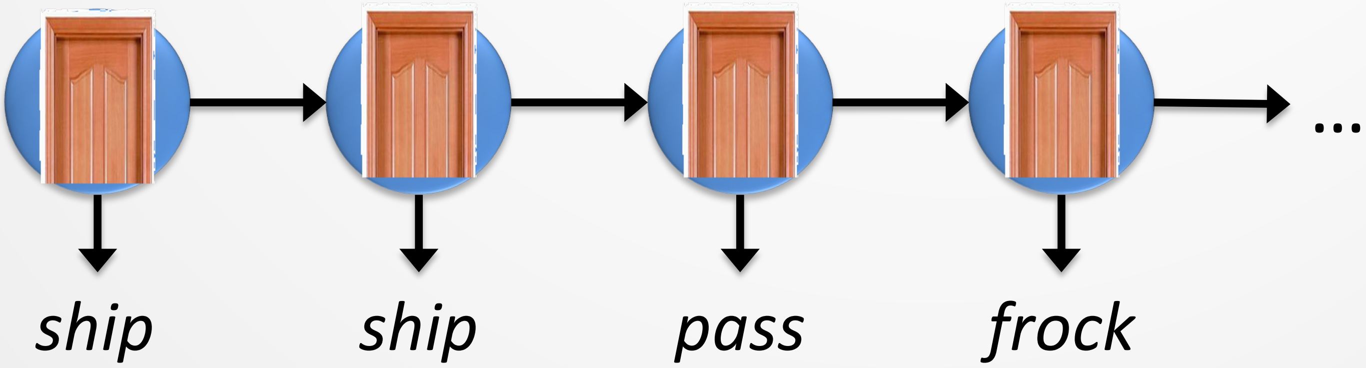
w	P(w)
ship	…
pass	…
camp	…
frock	…
soccer	…
mother	…
tops	…

- **A**: When all data are observed, basically the same as before.
  - $P(q_i|q_{i-1}) = \frac{P(q_{i-1}q_i)}{P(q_{i-1})}$ is learned with MLE from training data.
  - $P(w_i|q_i) = \frac{P(w_i, q_i)}{P(q_i)}$ is also learned with MLE from training data.

Hidden variables

- Q: What if you **don't** know the **states** during *testing*?
  - e.g., compute $P(\langle ship, ship, pass, frock \rangle)$

- Q: What if you **don't** know the **states** during *training*?



Examples of hidden phenomena

- We want to represent **surface** (i.e., **observable**) phenomena as the **output** of **hidden** underlying systems.
  - e.g.,
    - **Words** are the outputs of hidden **parts-of-speech**,
    - **French phrases** are the outputs of hidden **English phrases**,
    - **Speech sounds** are the outputs of hidden **phonemes**.

  - in other fields,
    - **Encrypted symbols** are the outputs of hidden **messages**,
    - **Genes** are the outputs of hidden **functional relationships**,
    - **Weather** is the output of hidden **climate conditions**,
    - **Stock prices** are the outputs of hidden **market conditions**,
    - …

Definition of an HMM

- A **hidden Markov model** (HMM) is specified by the 5-tuple $\{S, W, \Pi, A, B\}$:
  - $S = \{s_1, \dots, s_N\}$: set of **states** (e.g., moods)
  - $W = \{w_1, \dots, w_K\}$: output **alphabet** (e.g., words)

$$\theta\left\{\begin{array}{ll} \bullet\ \Pi = \{\pi_1, \dots, \pi_N\} & \text{: \textbf{initial} state probabilities} \\ \bullet\ A = \{a_{ij}\}, i, j \in S & \text{: state \textbf{transition} probabilities} \\ \bullet\ B = b_i(w), i \in S, w \in W & \text{: state \textbf{output} probabilities} \end{array}\right.$$

yielding

- $Q = \{q_0, \dots, q_{T-1}\}, q_i \in S$: state sequence
- $\mathcal{O} = \{\sigma_0, \dots, \sigma_{T-1}\}, \sigma_i \in W$: output sequence

A hidden Markov production process

- An HMM is a **representation** of a process in the world.
  - We can *synthesize* data, as in Shannon's game.
- This is how an HMM **generates** new sequences:

```
• $t := 0$
• Start in state $q_0 = s_i$ with probability $\pi_i$
• Emit observation symbol $\sigma_0 = w_k$ with probability $b_i(\sigma_0)$
• While (not forever)
    • Go from state $q_t = s_i$ to state $q_{t+1} = s_j$ with probability $a_{ij}$
    • Emit observation symbol $\sigma_{t+1} = w_k$ with probability $b_j(\sigma_{t+1})$
    • $t := t + 1$
```

Fundamental tasks for HMMs

1. Given a **model** with particular parameters $\theta = \langle \Pi, A, B \rangle$, how do we efficiently compute the likelihood of a *particular* **observation sequence**, $P(\mathcal{O}; \theta)$?

We previously computed the probabilities of word sequences using *N*-grams.

The probability of a particular sequence is usually useful as a means to some other end.

Fundamental tasks for HMMs

2. Given an **observation sequence** $\mathcal{O}$ and a **model** θ, how do we choose a **state sequence** $Q = \{q_0, \dots, q_{T-1}\}$ that *best explains* the observations?

This is the task of **inference** – i.e., guessing at the best explanation of unknown (‘latent’) variables given our model.

This is often an important part of **classification**.

Fundamental tasks for HMMs

3. Given a large **observation sequence** $\mathcal{O}$, how do we choose the *best parameters* $\theta = \langle \Pi, A, B \rangle$ that explain the data $\mathcal{O}$?

This is the task of **training**.

As before, we want our parameters to be set so that the available training data is maximally likely,
But doing so will involve guessing unseen information.

Task 1: Computing $P(\mathcal{O};\theta)$

- We've seen the probability of a joint sequence of observations and states:

$$P(\mathcal{O},Q;\theta) = P(\mathcal{O}|Q;\theta)P(Q;\theta)$$
$$= \pi_{q_0} b_{q_0}(\sigma_0) a_{q_0 q_1} b_{q_1}(\sigma_1) a_{q_1 q_2} b_{q_2}(\sigma_2) \ldots$$

- To get the probability of our observations **without** seeing the state, we must **sum over all possible state sequences**:

$$P(\mathcal{O};\theta) = \sum_Q P(\mathcal{O},Q;\theta) = \sum_Q P(\mathcal{O}|Q;\theta)P(Q;\theta).$$

Computing $P(\mathcal{O};\theta)$ naïvely

- To get the total probability of our observations, we could *directly* **sum over all possible state sequences**:

$$P(\mathcal{O};\theta) = \sum_Q P(\mathcal{O}|Q;\theta)P(Q;\theta).$$

- For observations of length T, each state sequence involves $2T$ multiplications (1 for each state transition, 1 for each observation, 1 for the start state, minus 1).
- There are up to N^T possible state sequences of length T given N states.

$$\therefore \sim(1 + T + T - 1) \cdot N^T \text{ multiplications}$$ ☹

Computing $P(\mathcal{O};\theta)$ cleverly

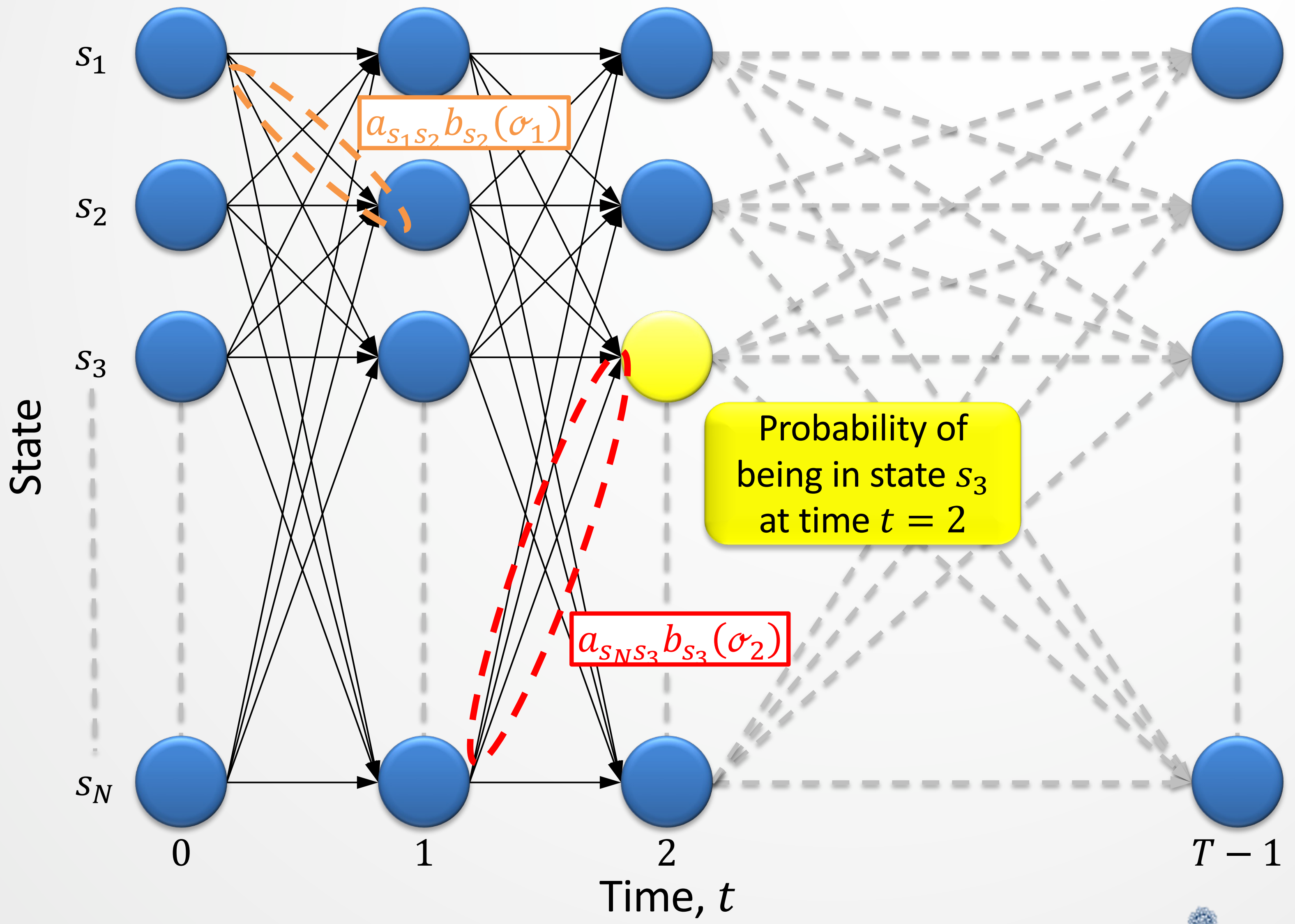
- To avoid this complexity, we use **dynamic programming**; we **remember**, rather than **recompute**, partial results.

- We make a **trellis** which is an array of **states vs. time**.
  - The element at (i,t) is $\boldsymbol{\alpha_i(t)}$

the probability of being in state i at time t *after seeing all observations to that point*:

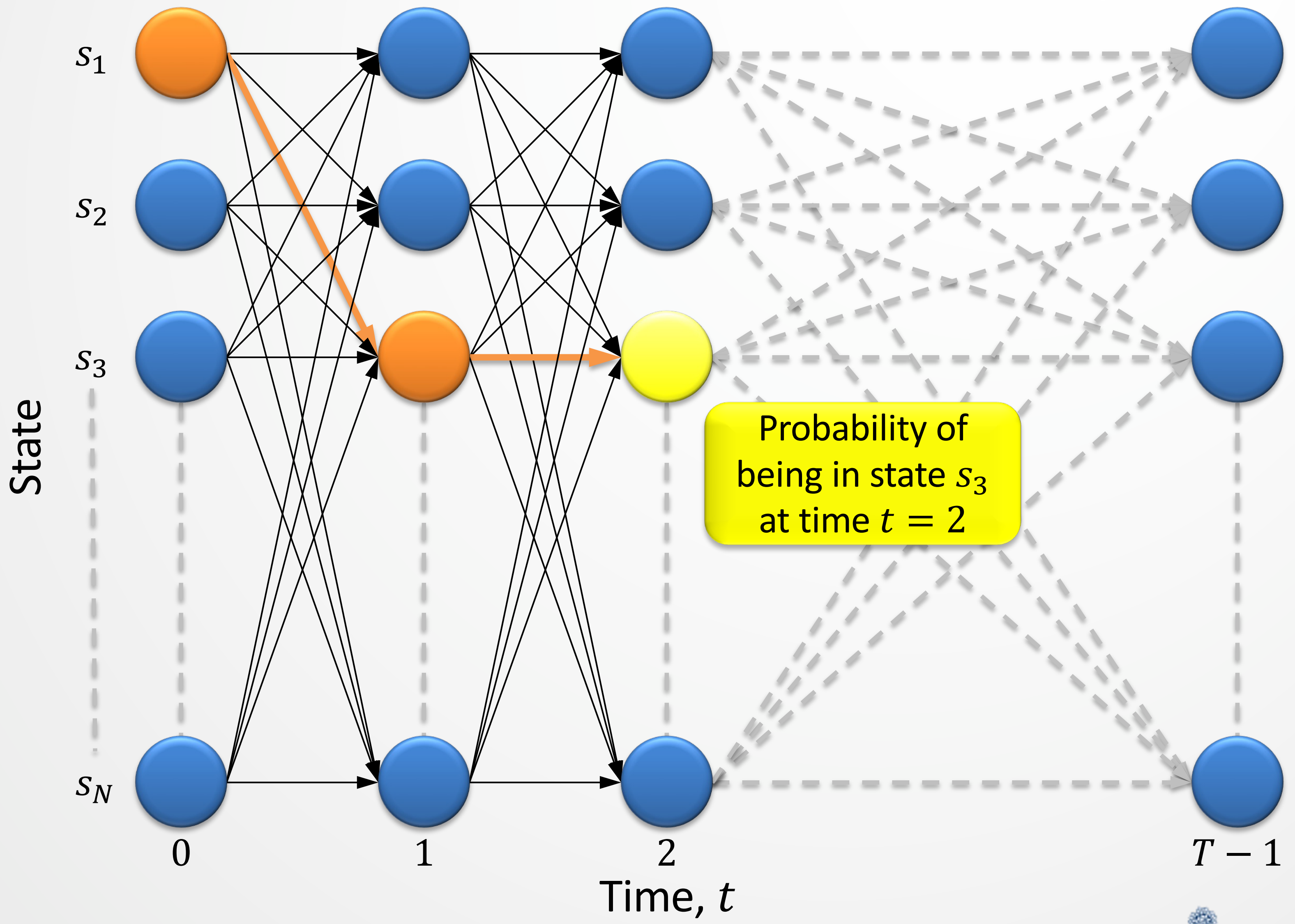
$P(\mathcal{O}_{o:t}, q_t = s_i; \theta)$

Trellis



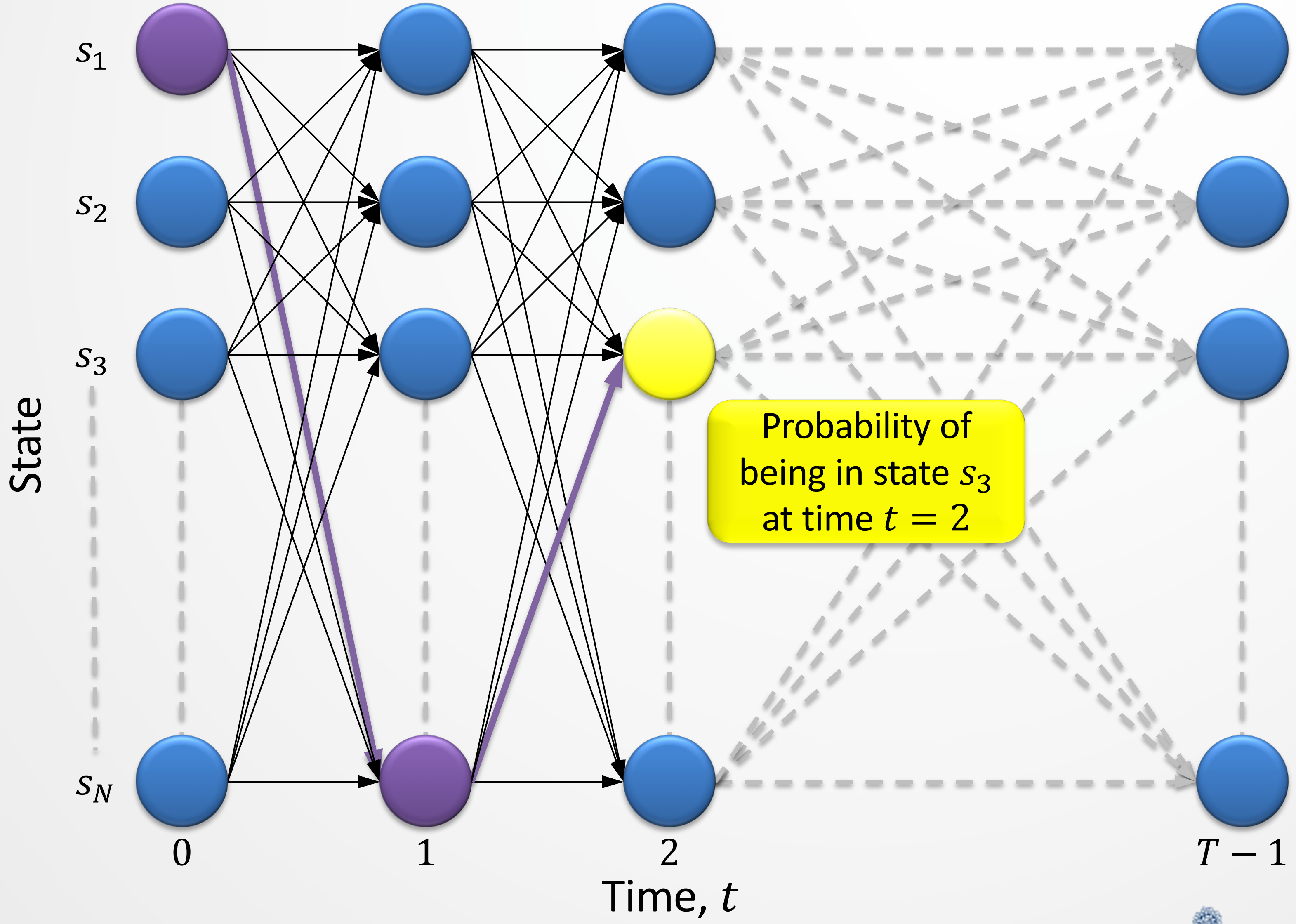
Probability of being in state s_3 at time $t = 2$

Alternative paths through the trellis



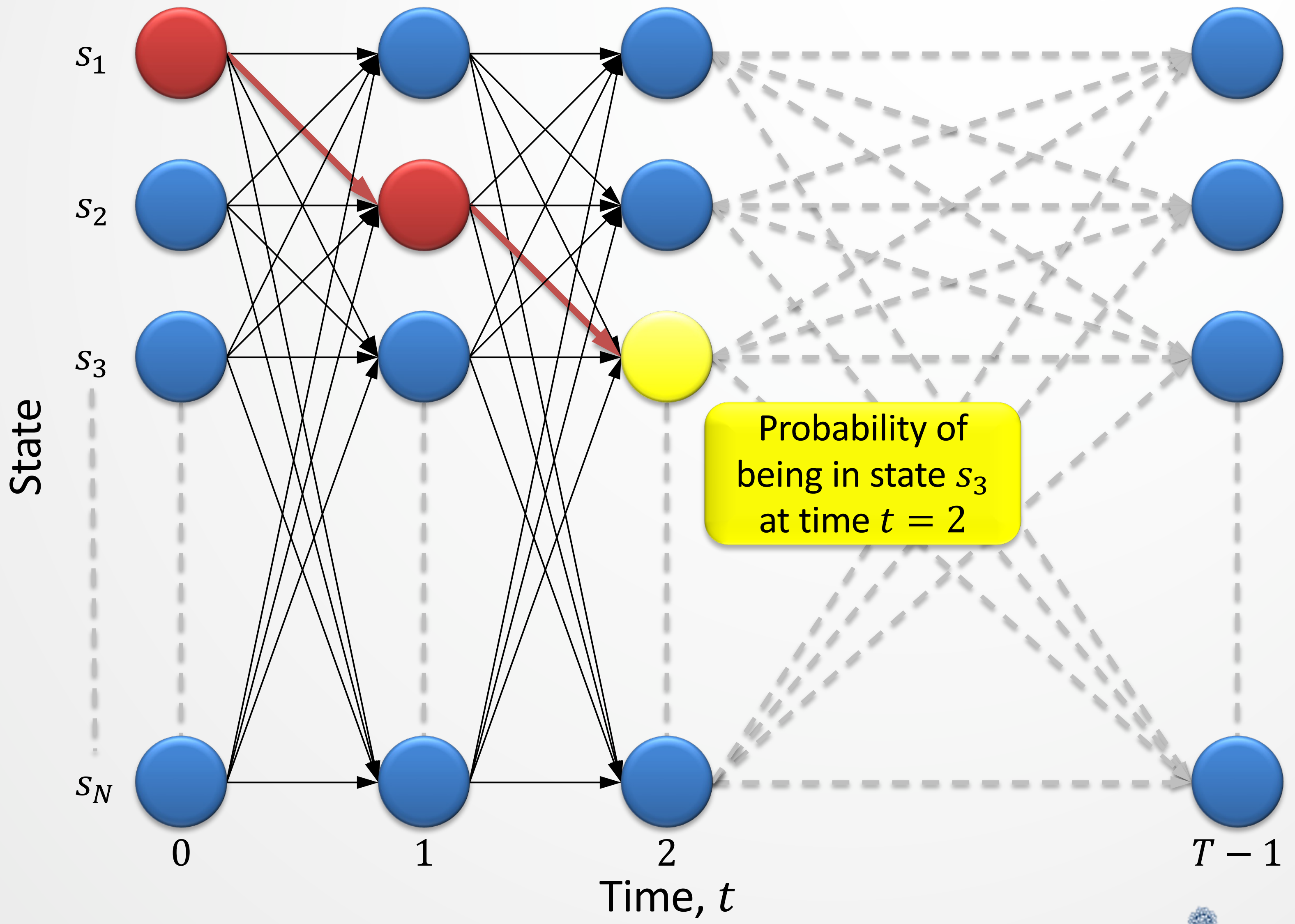
Probability of being in state s_3 at time $t = 2$

Alternative paths through the trellis



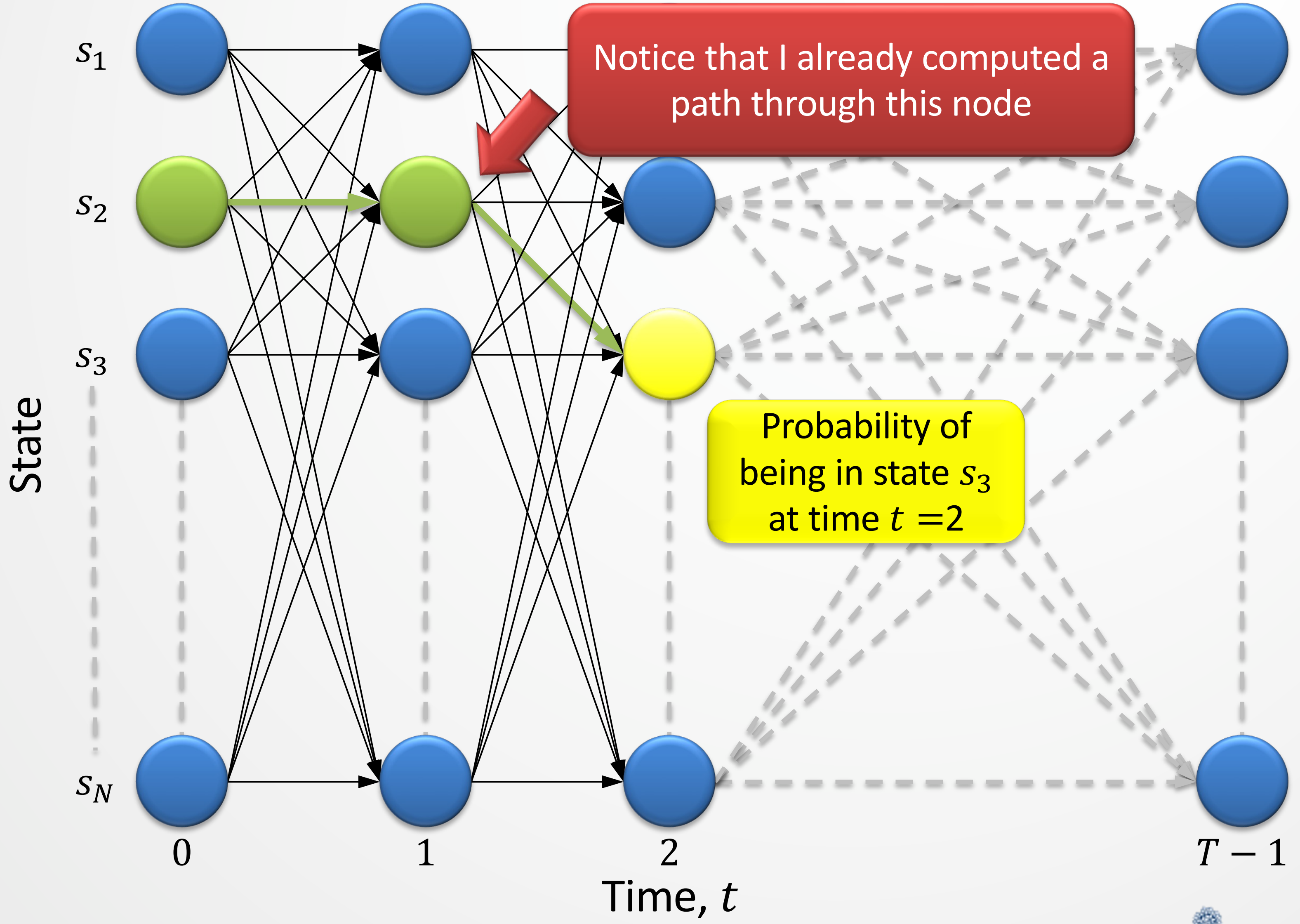
Probability of being in state s_3 at time $t = 2$

Alternative paths through the trellis



Probability of being in state s_3 at time $t = 2$

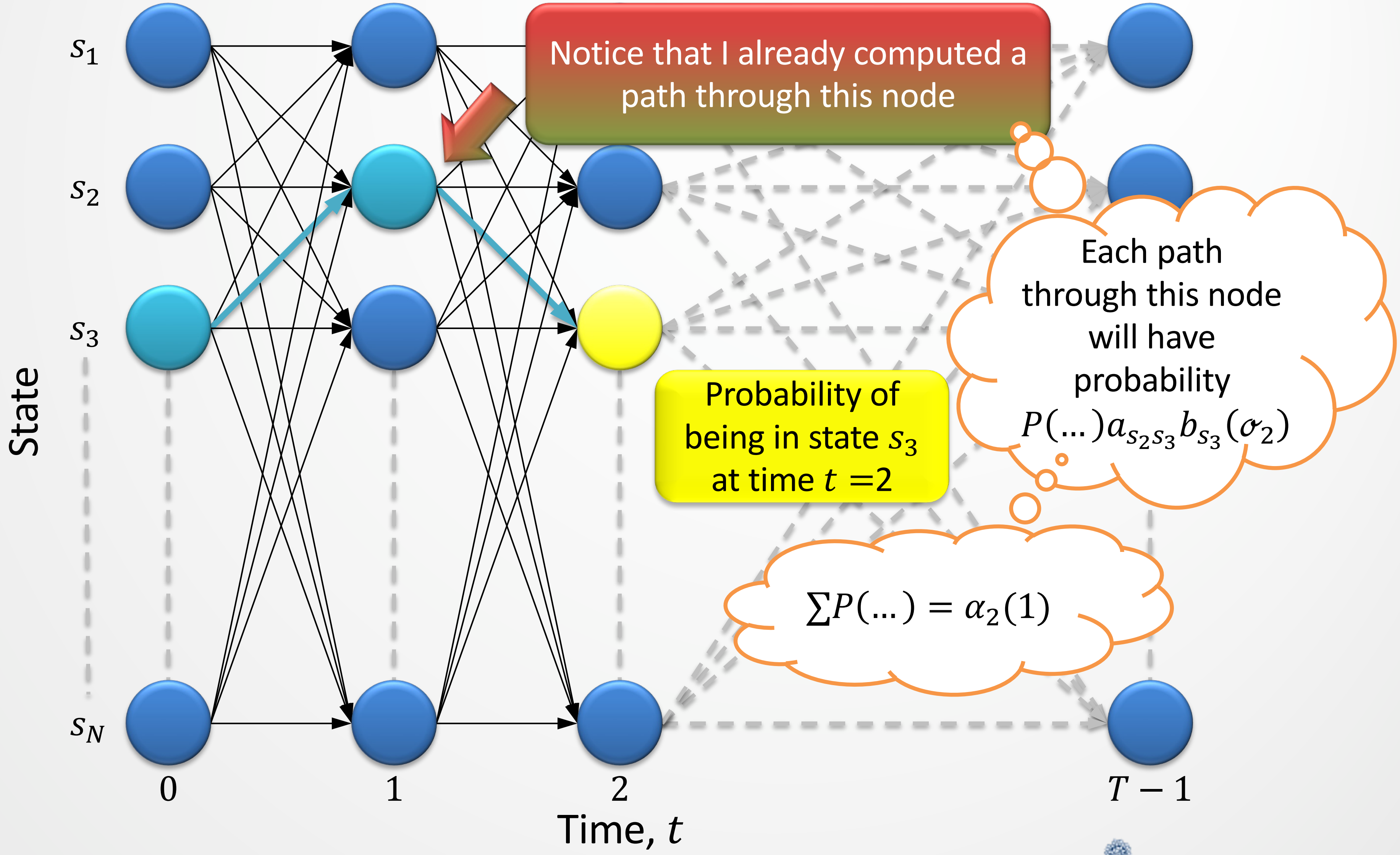
Alternative paths through the trellis



Notice that I already computed a path through this node

Probability of being in state s_3 at time t =2

Alternative paths through the trellis



Notice that I already computed a path through this node

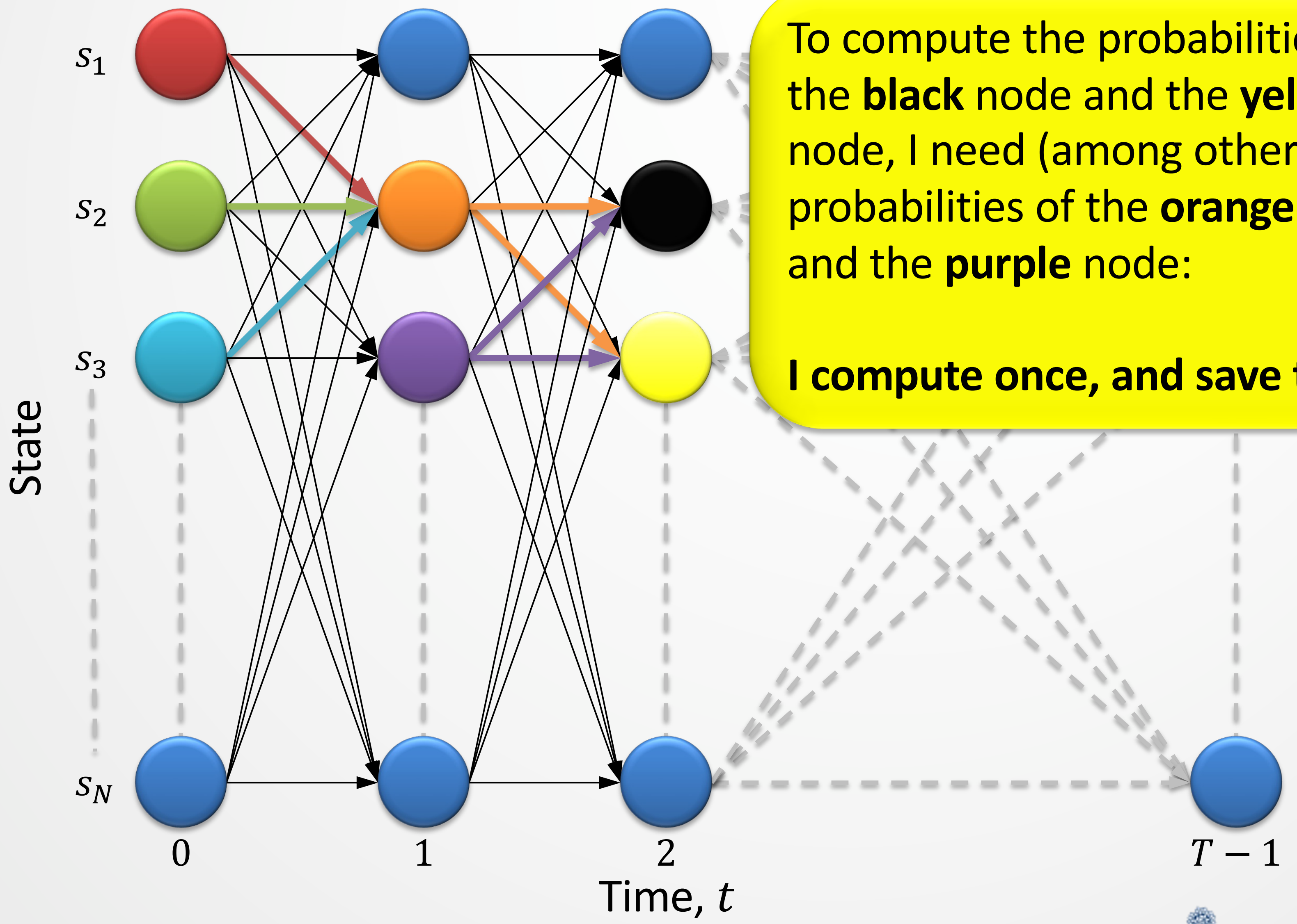
Each path through this node will have probability $P(\dots)a_{s_2 s_3}b_{s_3}(\sigma_2)$

Probability of being in state s_3 at time $t = 2$

$\sum P(\dots) = \alpha_2(1)$

AND SO ON…

Trellis



To compute the probabilities of the **black** node and the **yellow** node, I need (among others) the probabilities of the **orange** node and the **purple** node:

I compute once, and save them.

The Forward procedure

- To compute

$$\alpha_i(t) = P(\sigma_{0:t}, q_t = s_i; \theta)$$

  we can compute $\alpha_j(t-1)$ for possible *previous* states s_j, then use our knowledge of a_{ji} and $b_i(\sigma_t)$

- We compute the trellis **left-to-right** (because of the convention of time) and **top-to-bottom** (‘just because’).

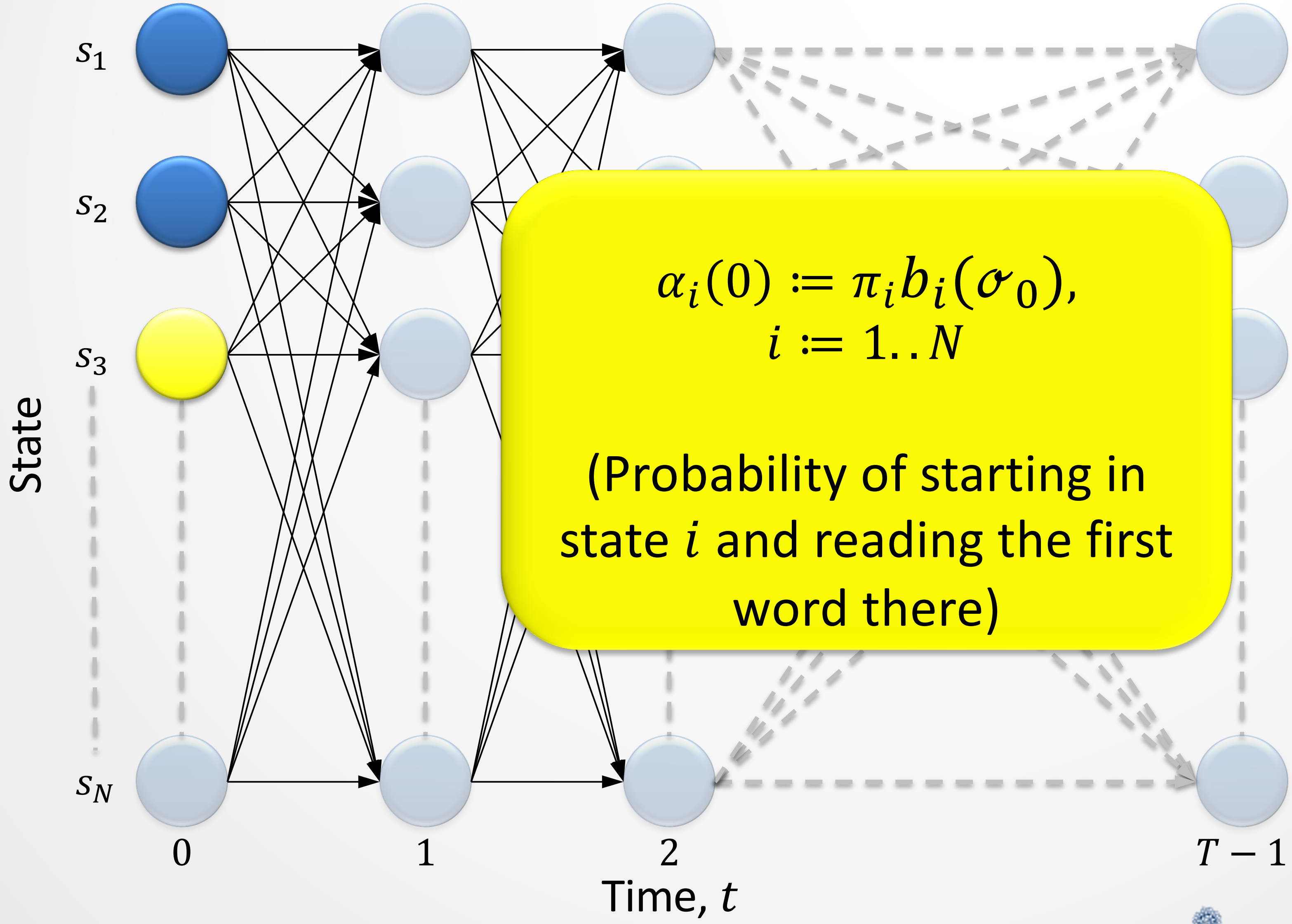
- **Remember**: σ_t is fixed and known.
  $\alpha_i(t)$ is agnostic of the future.

The Forward procedure

- The trellis is computed **left-to-right** and **top-to-bottom**.

- There are <u>three steps</u> in this procedure:
  - **Initialization**: Compute the nodes in the ***first** column* of the trellis ($t = 0$).
  - **Induction**: Iteratively compute the nodes in the *rest* of the trellis ($1 \leq t < T$).
  - **Conclusion**: Sum over the nodes in the ***last** column* of the trellis ($t = T - 1$).

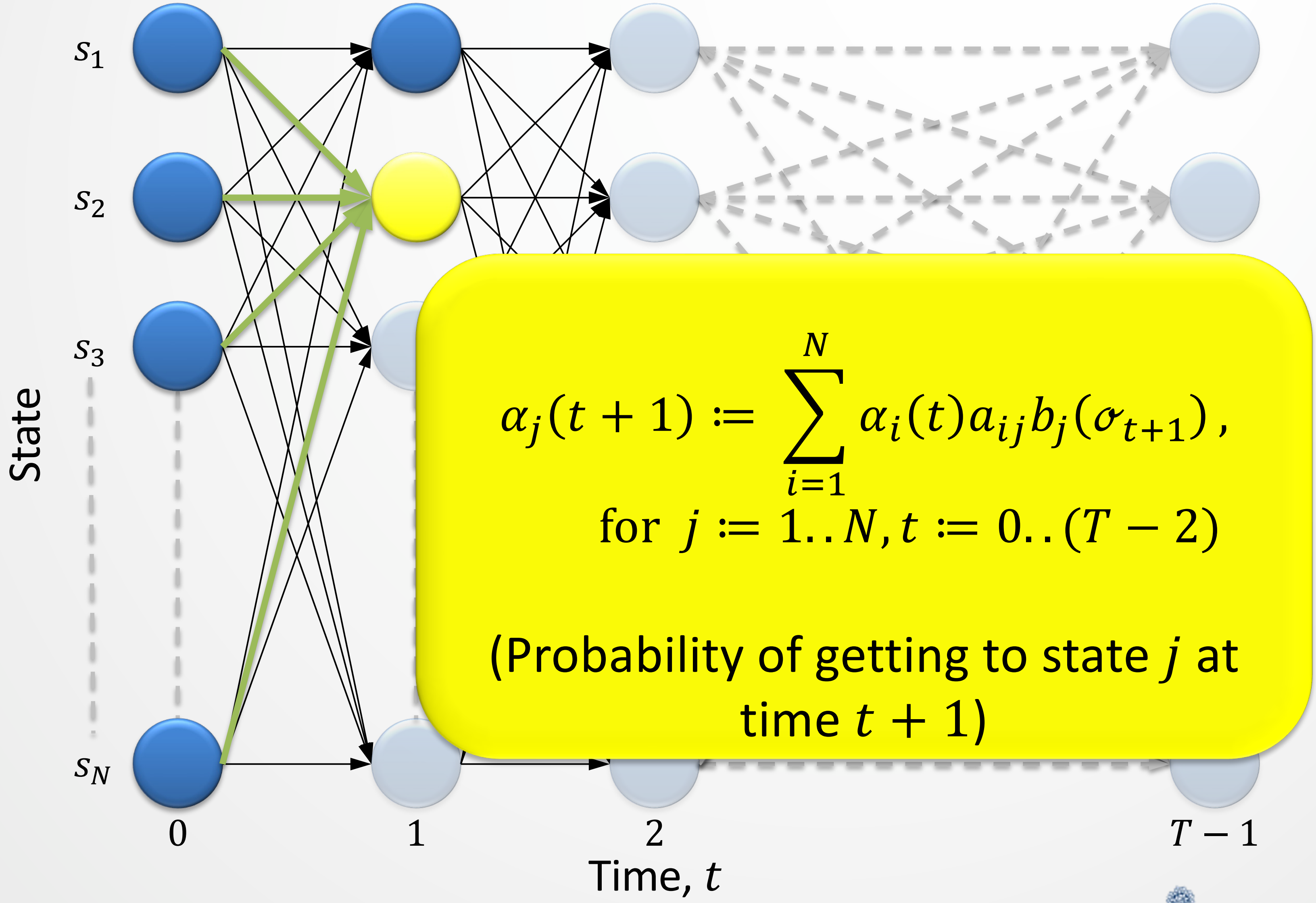
Initialization of Forward procedure



$$\alpha_i(0) \coloneqq \pi_i b_i(\sigma_0),$$
$$i \coloneqq 1..N$$

(Probability of starting in state i and reading the first word there)

Induction of Forward procedure

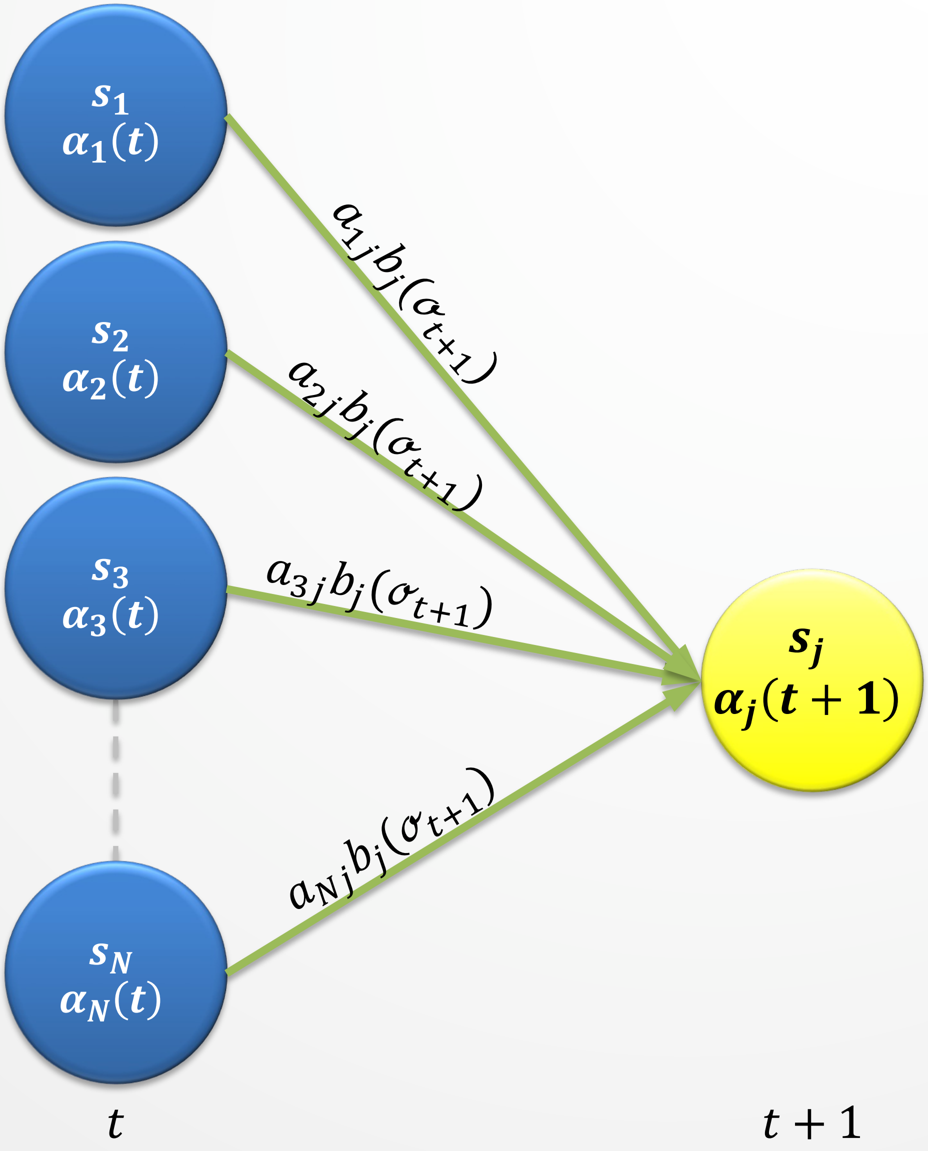


$$\alpha_j(t+1) \coloneqq \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_j(\sigma_{t+1}),$$

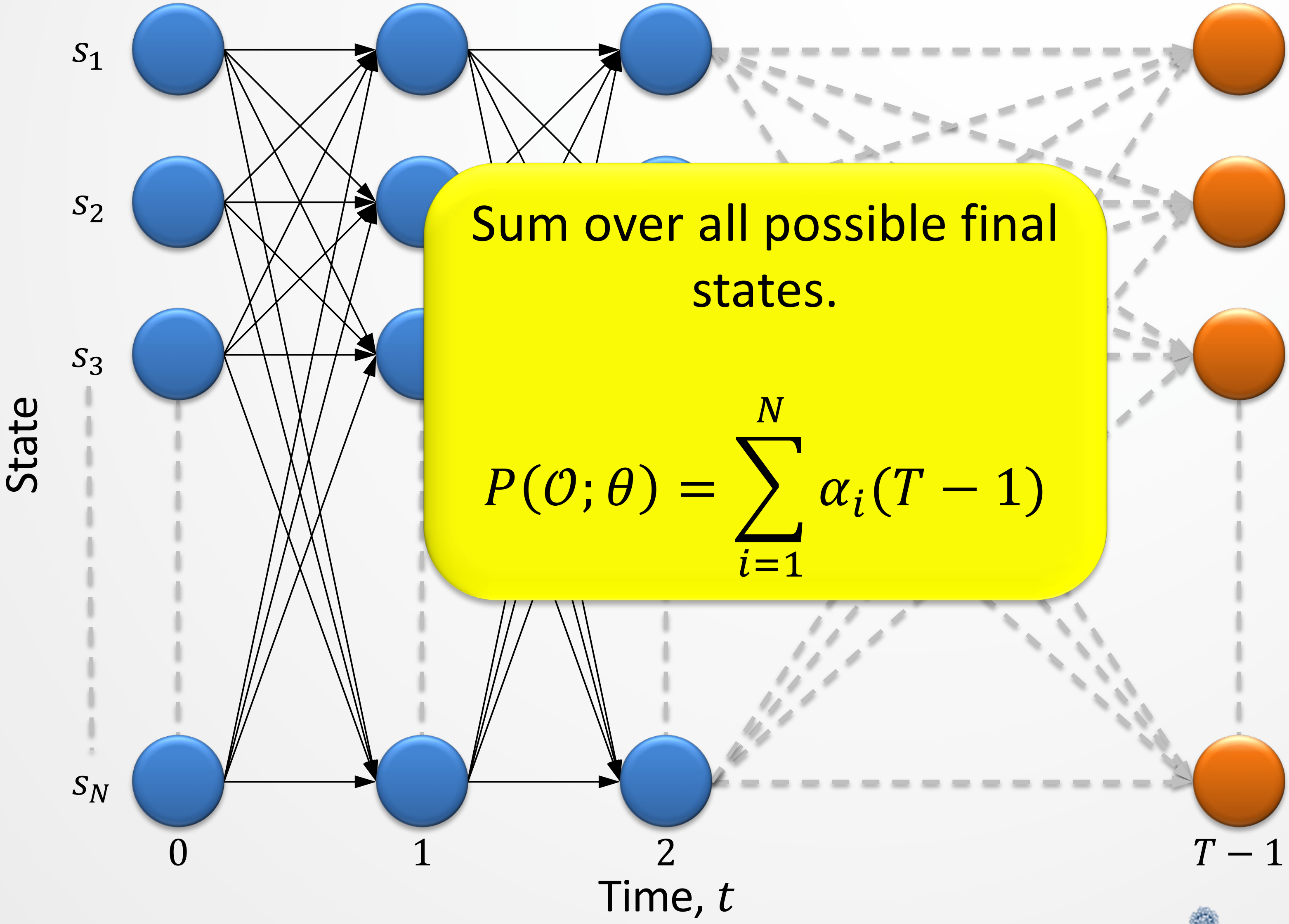
$$\text{for } j \coloneqq 1..N, t \coloneqq 0..(T-2)$$

(Probability of getting to state j at time $t+1$)

Induction of Forward procedure



Conclusion of Forward procedure

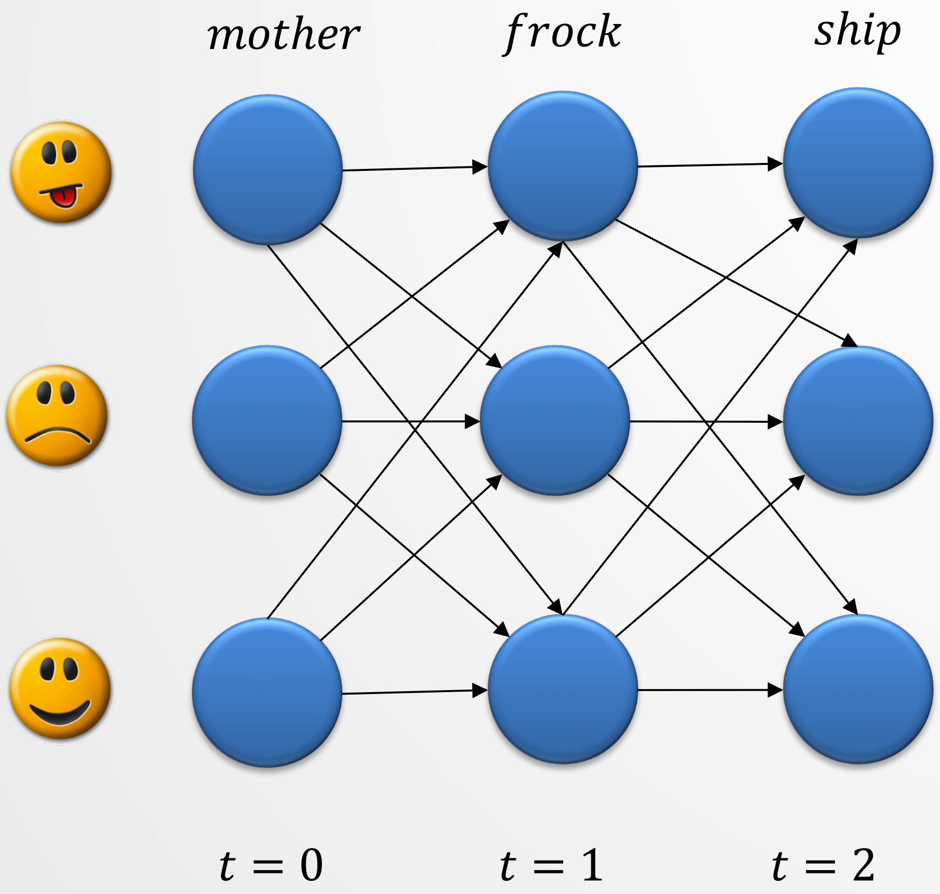


Sum over all possible final states.

$$P(\mathcal{O};\theta) = \sum_{i=1}^{N} \alpha_i(T-1)$$

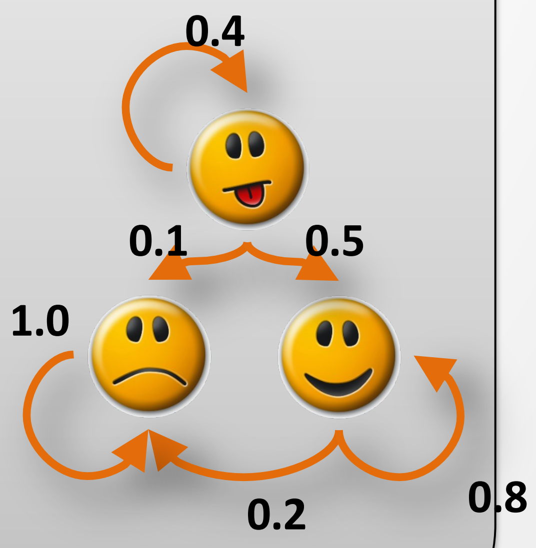
The Forward procedure - Example

- Let’s compute $\mathrm{P}(\langle \mathrm{mother}, \mathrm{frock}, \mathrm{ship} \rangle)$



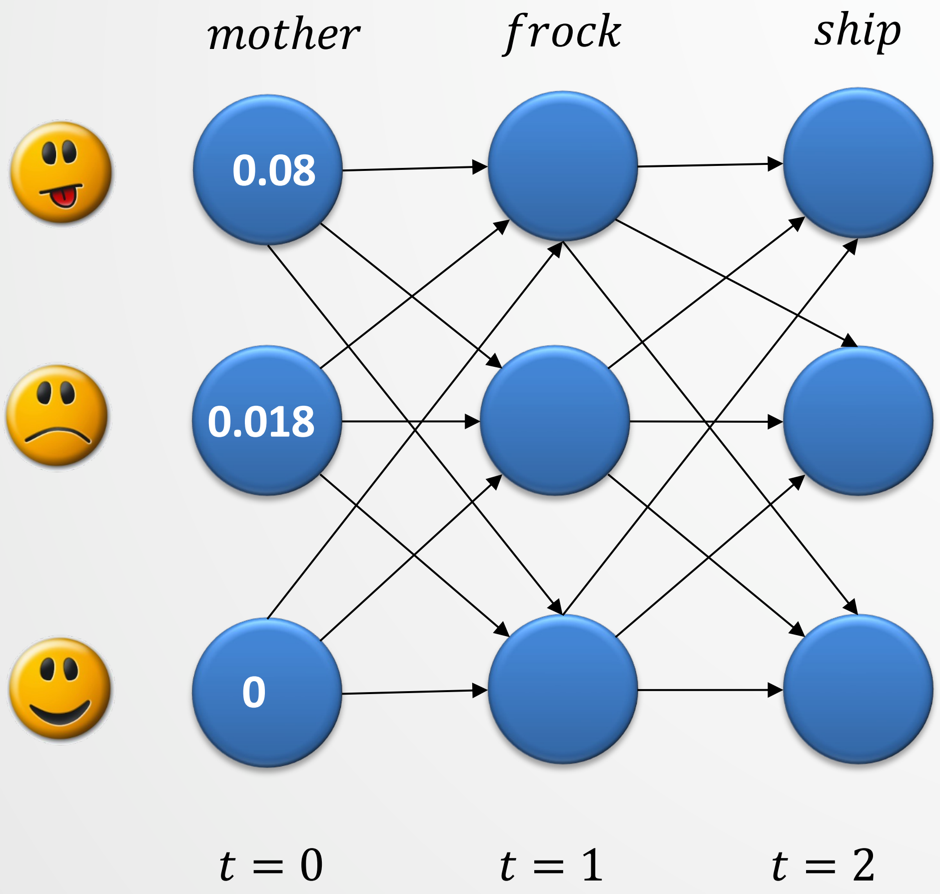
We need initial state probabilities $\mathbf{\Pi}$ and transition probabilities $\boldsymbol{\alpha_{ij}}$

😛	$\mathbf{\Pi}$ = 0.80
🙁	$\mathbf{\Pi}$ = 0.20
🙂	$\mathbf{\Pi}$ = 0



The Forward procedure - Example

- Let’s compute $\mathrm{P}(\langle \mathrm{mother}, \mathrm{frock}, \mathrm{ship} \rangle)$



Initialization

Compute the probability of starting in state i and reading the first word there

$$\boldsymbol{\alpha_i(0) \coloneqq \pi_i b_i(\sigma_0)}$$

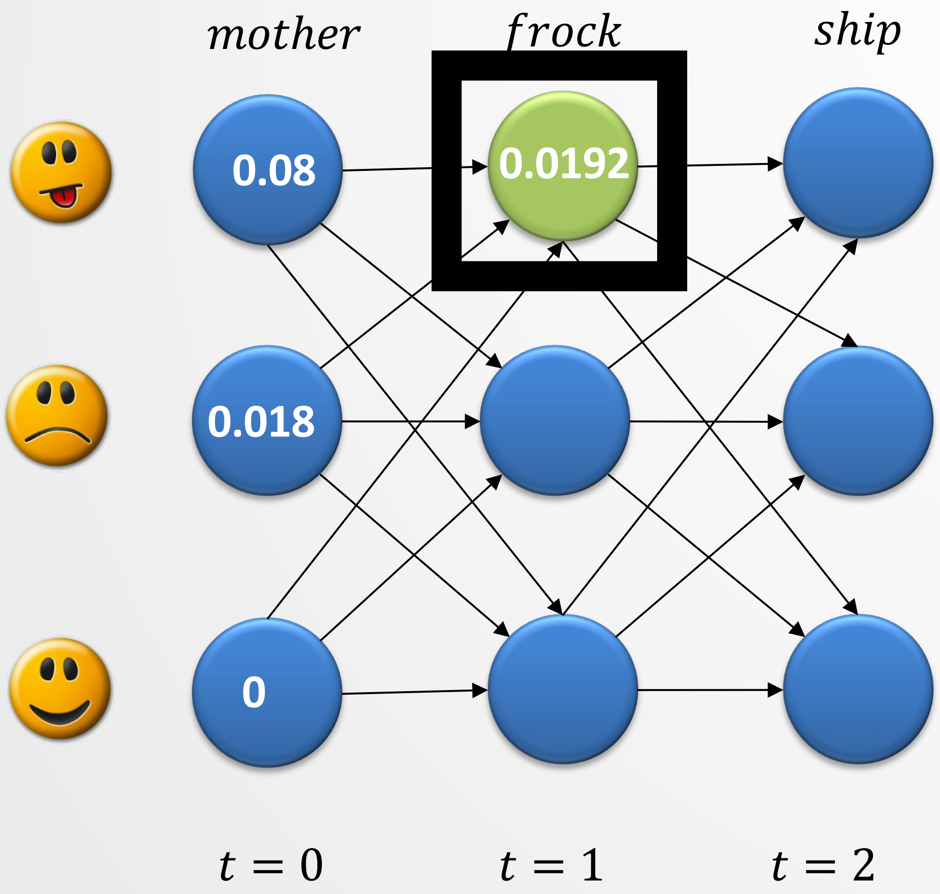
😛 $\boldsymbol{\alpha(0) = 0.80 \times 0.10 = 0.08}$

☹ $\boldsymbol{\alpha(0) = 0.20 \times 0.09 = 0.018}$

🙂 $\boldsymbol{\alpha(0) = 0 \times 0.05 = 0}$

The Forward procedure - Example

- Let's compute $\mathrm{P}(\langle \mathrm{mother}, \mathrm{frock}, \mathrm{ship} \rangle)$



Induction

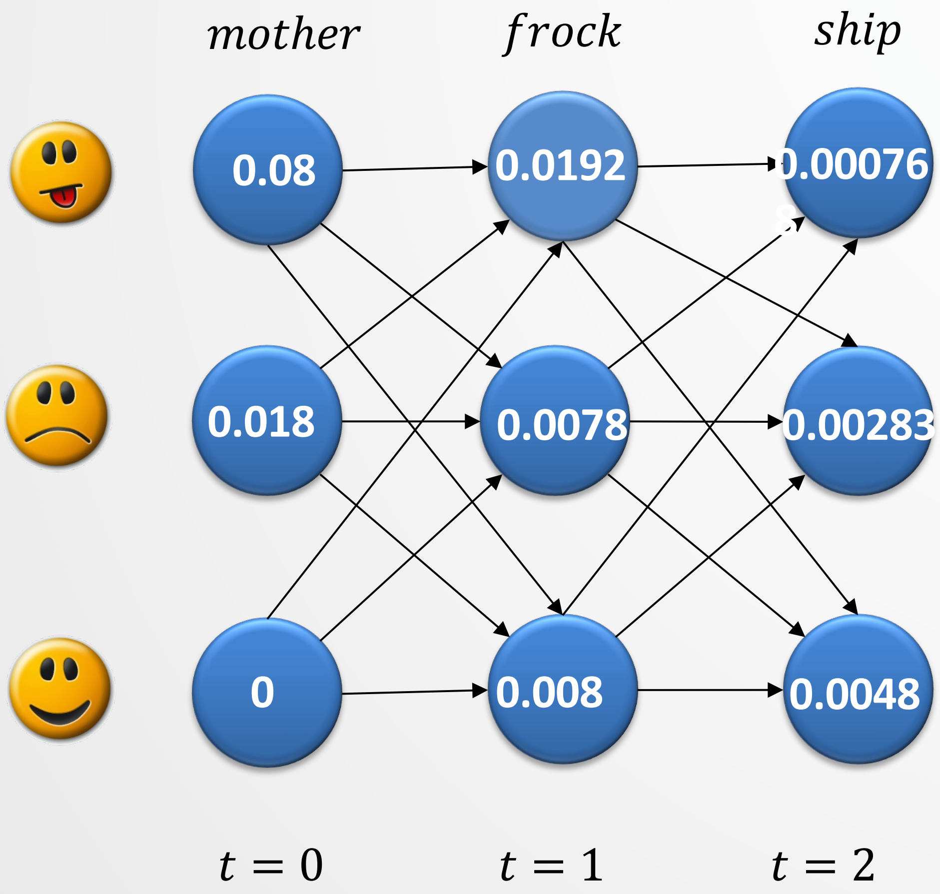
Iteratively compute the rest of the nodes in the trellis; i.e., the probability of getting to state *j* at time *t+1*

$$\alpha_j(t+1) \coloneqq \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_j(\sigma_{t+1})$$

$$\begin{aligned} \boldsymbol{\alpha(t+1)} &= \mathbf{0.08(0.4)(0.6)} \\ &\quad \mathbf{+ 0.018(0)(0.6)} \\ &\quad \mathbf{+ 0(0)(0.6)} \\ &= \mathbf{0.0192} \end{aligned}$$

The Forward procedure - Example

- Let’s compute $P(\langle \text{mother}, \text{frock}, \text{ship} \rangle)$



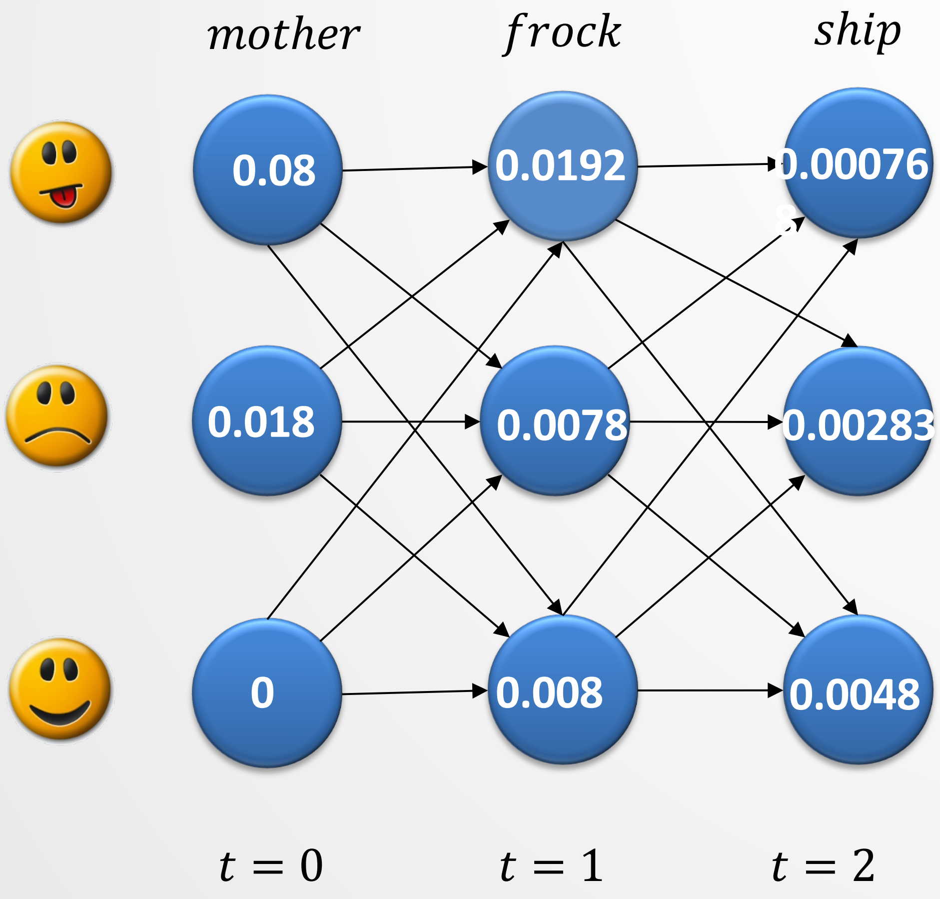
Induction

Iteratively compute the rest of the nodes in the trellis; i.e., the probability of getting to state *j* at time *t+1*

$$\alpha_j(t+1) \coloneqq \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_j(\sigma_{t+1})$$

The Forward procedure - Example

- Let’s compute $\mathrm{P}(\langle \mathrm{mother}, \mathrm{frock}, \mathrm{ship} \rangle)$



Conclusion

Sum over all possible final states

$$\boldsymbol{P(\mathcal{O};\theta) = \sum_{i=1}^{N} \alpha_i(T-1)}$$

$$P(\mathcal{O};\theta)$$
$$= 0.00076 + 0.00283 + 0.0048$$
$$= \mathbf{0.00839}$$

The Forward procedure

- The naïve approach needed $(2T) \cdot N^T$ multiplications.
- The Forward procedure (using **dynamic programming**) needs only $2N^2T$ multiplications. 🙂
- The Forward procedure gives us $P(\mathcal{O}; \theta)$.
- Clearly, but less intuitively, we can also compute the trellis from back-to-front, i.e., *backwards in time*…

Remember the point

- The point was to compute the equivalent of

$$P(\mathcal{O};\theta) = \sum_{Q} P(\mathcal{O},Q;\theta)$$

where

$$P(\mathcal{O},Q;\theta) = P(\mathcal{O}|Q;\theta)P(Q;\theta)$$

$$= \pi_{q_0} b_{q_0}(\sigma_0) a_{q_0 q_1} b_{q_1}(\sigma_1) a_{q_1 q_2} b_{q_2}(\sigma_2) \ldots$$

The Forward algorithm stores all possible 1-state sequences (from the start), to store all possible 2-state sequences (from the start), to store all possible 3-state sequences (from the start)…

Remember the point

- But, we can compute these factors in reverse

$$P(\mathcal{O}, Q; \theta) = P(\mathcal{O}|Q; \theta)P(Q; \theta)$$

$$= \pi_{q_0} \ldots b_{q_{T-3}}(\sigma_{T-3}) a_{q_{T-3}q_{T-2}} b_{q_{T-2}}(\sigma_{T-2}) a_{q_{T-2}q_{T-1}} b_{q_{T-1}}(\sigma_{T-1})$$

$\beta_i(T-2)$

$\beta_i(T-3)$

$\beta_i(T-4)$

We can still deal with sequences that evolve *forward* in time, but simply **store temporary results in reverse**…

The Backward procedure

- In the $(i,t)^{th}$ node of the **trellis**, we store

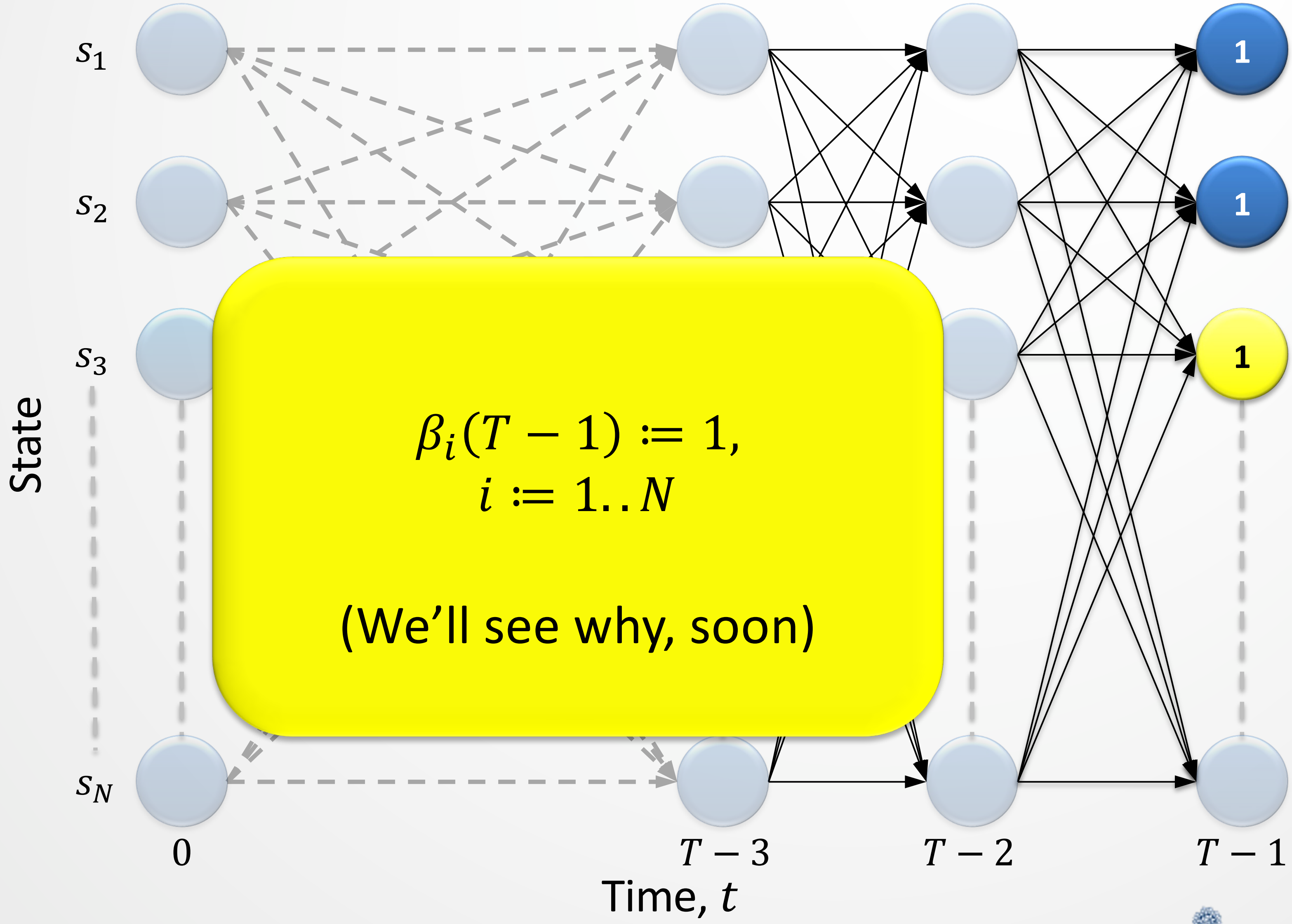
$$\begin{aligned}\beta_i(t) &= P(\sigma_{t+1:T-1} | \sigma_{0:t}, q_t = s_i; \theta) \\ &= P(\sigma_{t+1:T-1} | q_t = s_i; \theta)\end{aligned}$$

  which is computed by summing probabilities on **outgoing** arcs **from** that node.

$\beta_i(t)$ *is the probability of starting in state* i *at time* t *then observing everything that comes thereafter.*

- The trellis is computed **right-to-left** and **top-to-bottom**.

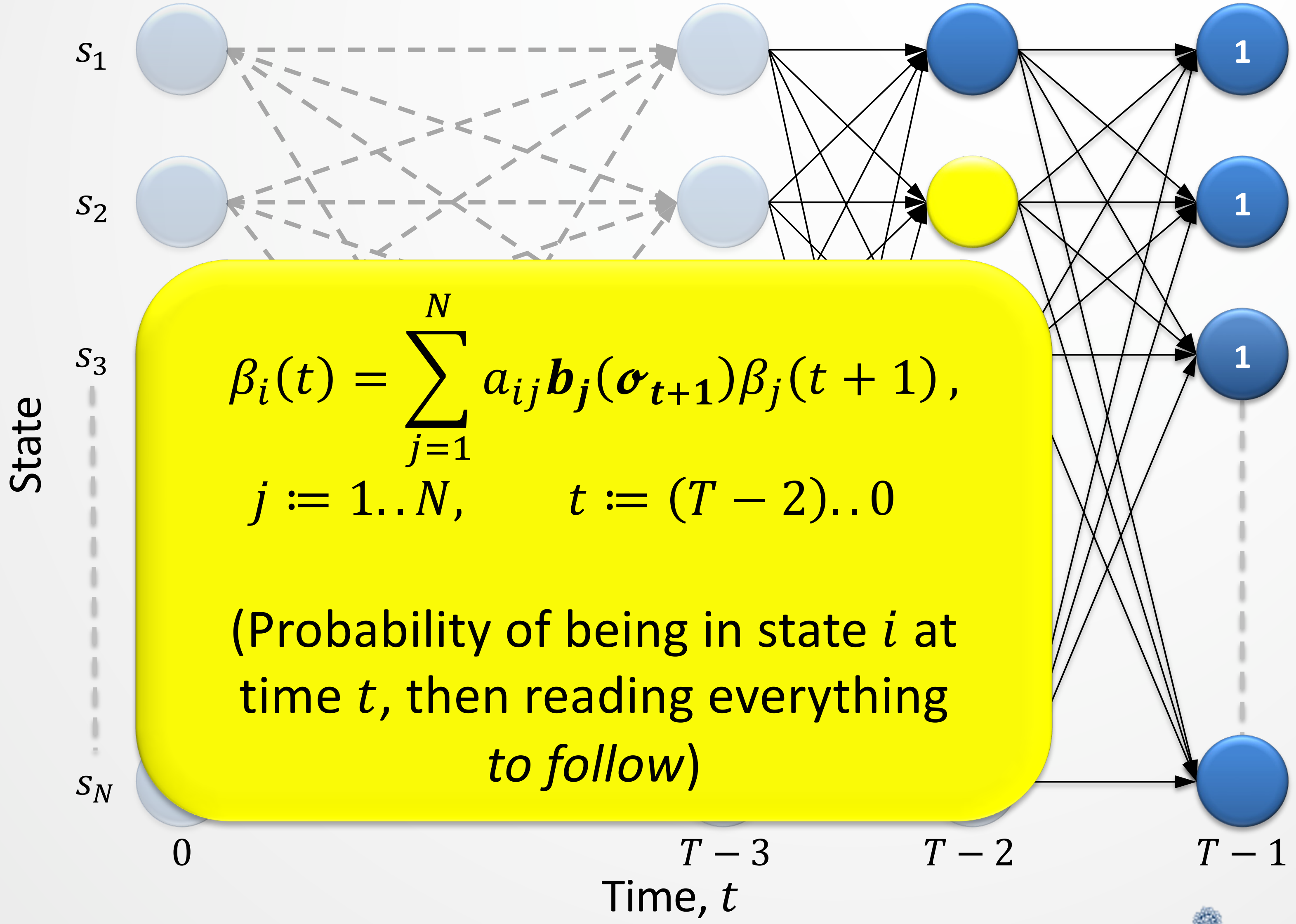
Step 1: Backward initialization



$$\beta_i(T-1) \coloneqq 1,$$
$$i \coloneqq 1..N$$

(We'll see why, soon)

Step 2: Backward induction

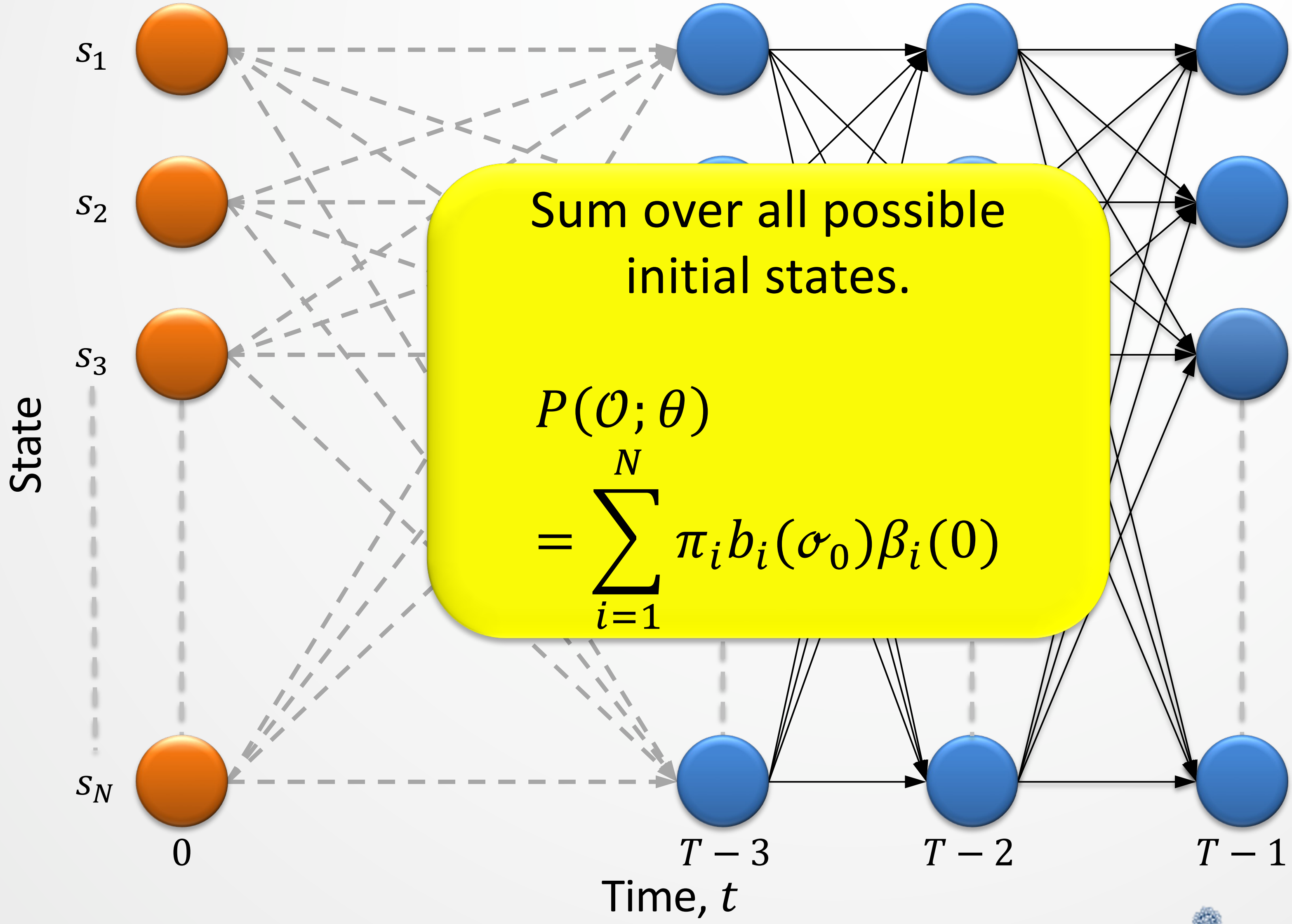


$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} \boldsymbol{b_j}(\boldsymbol{o_{t+1}}) \beta_j(t+1)\,,$$

$$j \coloneqq 1..N, \qquad t \coloneqq (T-2)..0$$

(Probability of being in state i at time t, then reading everything *to follow*)

Step 3: Backward conclusion



Sum over all possible initial states.

$$P(\mathcal{O};\theta) = \sum_{i=1}^{N} \pi_i b_i(\sigma_0)\beta_i(0)$$

The Backward procedure

- Initialization

$$\beta_i(T-1) = 1, \qquad i \coloneqq 1..N$$

- Induction

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(\sigma_{t+1}) \beta_j(t+1), \qquad i \coloneqq 1..N$$

$$t \coloneqq T-2..0$$

- Conclusion

$$P(\mathcal{O};\theta) = \sum_{i=1}^{N} \pi_i b_i(\sigma_0) \beta_i(0)$$

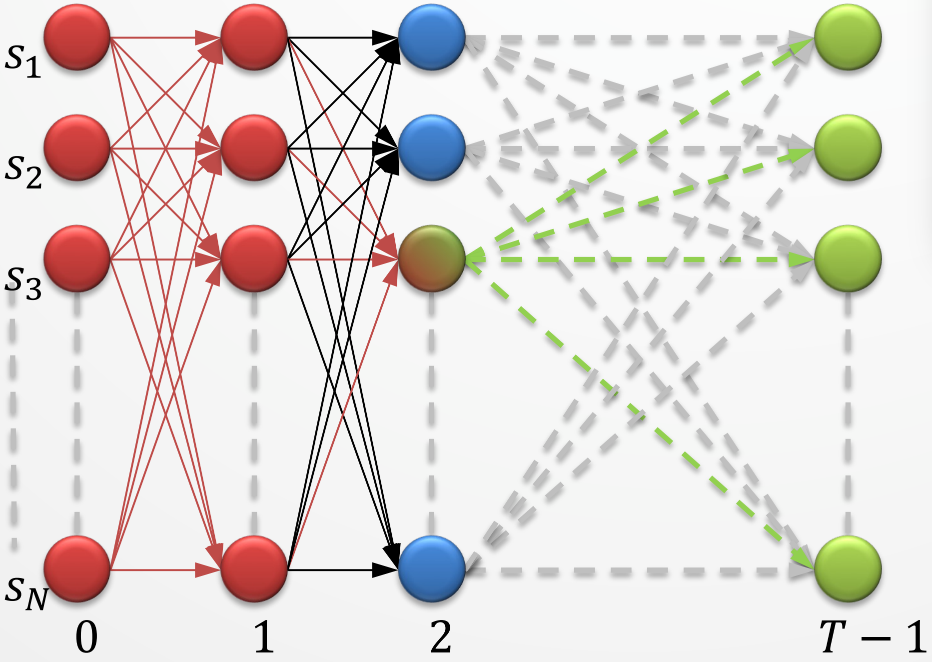
The Backward procedure – so what?

- The **combination** of Forward and Backward procedures will be vital for solving parameter re-estimation, i.e., **training**.
- Generally, we can **combine** α and β at any point in time to represent the probability of an **entire** observation sequence…

Combining α and β

$$P(\mathcal{O}, q_t = i; \theta) = \alpha_i(t)\beta_i(t)$$

$$\therefore P(\mathcal{O}; \theta) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$



This requires the current word to be incorporated by $\alpha_i(t)$, but **not** $\beta_i(t)$.

This isn't merely for fun – it will soon become useful…

Fundamental tasks for HMMs

2. Given an **observation sequence** $\mathcal{O}$ and a **model** θ, how do we choose a **state sequence** $Q^* = \{q_0, \dots, q_{T-1}\}$ that *best explains* the observations?

This is the task of **inference** – i.e., guessing at the best explanation of unknown (‘latent’) variables given our model.

This is often an important part of **classification**.

Task 2: Choosing $\boldsymbol{Q}^* = \{\boldsymbol{q}_0 \ldots \boldsymbol{q}_{T-1}\}$

- The purpose of finding **the best state sequence $\boldsymbol{Q}^*$** out of all possible state sequences Q is that it tells us what is **most likely** to be going on 'under the hood'.

- With the **Forward** algorithm, we didn't care about specific state sequences – we were summing over **all possible** state sequences.

Task 2: Choosing $Q^* = \{q_0 \ldots q_{T-1}\}$

- In other words,

$$Q^* = \underset{Q}{\operatorname{argmax}} P(\mathcal{O}, Q; \theta)$$

where

$$P(\mathcal{O}, Q; \theta) = \pi_{q_0} b_{q_0}(\sigma_0) \prod_{t=1}^{T-1} a_{q_{t-1}q_t} b_{q_t}(\sigma_t)$$

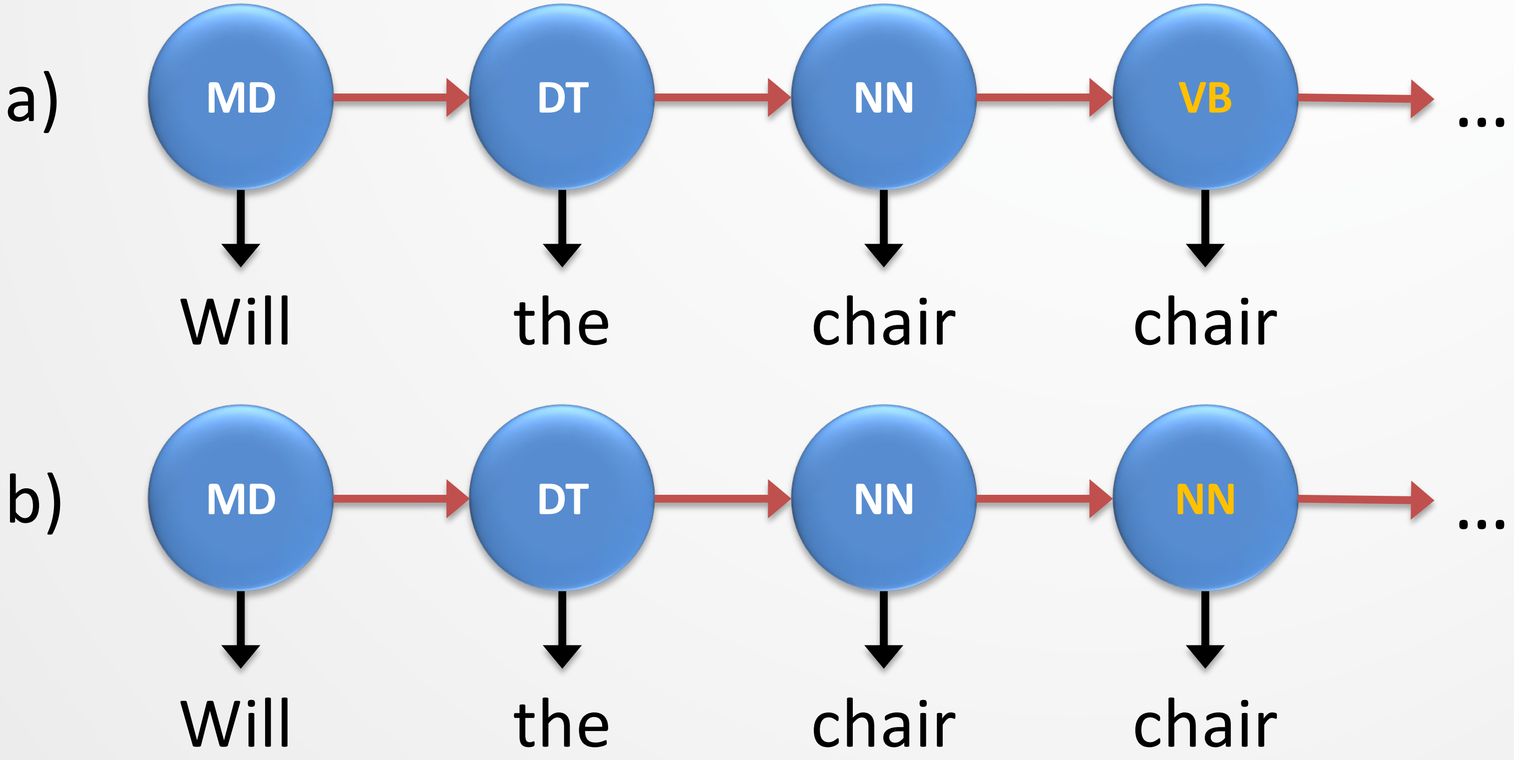
Why choose $Q^* = \{q_0 \dots q_{T-1}\}$?

- Recall the purpose of HMMs:
  - To represent multivariate systems where some variable is unknown/hidden/latent.

- Finding the best hidden-state sequence Q^* allows us to:
  - Identify **unseen parts-of-speech** given **words**,
  - Identify **equivalent English** words given **French words**,
  - Identify **unknown phonemes** given **speech sounds**,
  - Decipher **hidden messages** from **encrypted symbols**,
  - Identify **hidden relationships** from **gene sequences**,
  - Identify **hidden market conditions** given **stock prices**,
  - …

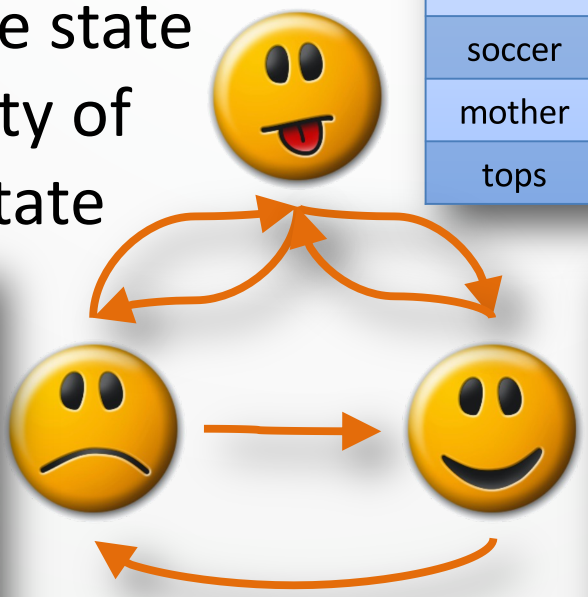
Example – PoS state sequences

- *Will/MD the/DT* ***chair/NN chair/??*** *the/DT meeting/NN from/IN that/DT* ***chair/NN****?*



Recall

- Observation likelihoods depend on the state, which changes over time
- We **cannot** simply choose the state that maximizes the probability of o_t without considering the state *sequence*.



word	P(word)
ship	0.1
pass	0.05
camp	0.05
frock	0.6
soccer	0.05
mother	0.1
tops	0.05

word	P(word)
ship	0.25
pass	0.25
camp	0.05
frock	0.3
soccer	0.05
mother	0.09
tops	0.01

word	P(word)
ship	0.3
pass	0
camp	0
frock	0.2
soccer	0.05
mother	0.05
tops	0.4

The Viterbi algorithm

- **The Viterbi algorithm** is an inductive dynamic-programming algorithm that uses a ***new*** *kind* of trellis.

- We define **the probability of the most probable path** leading to the trellis node at $(\text{state } i, \text{time } t)$ as

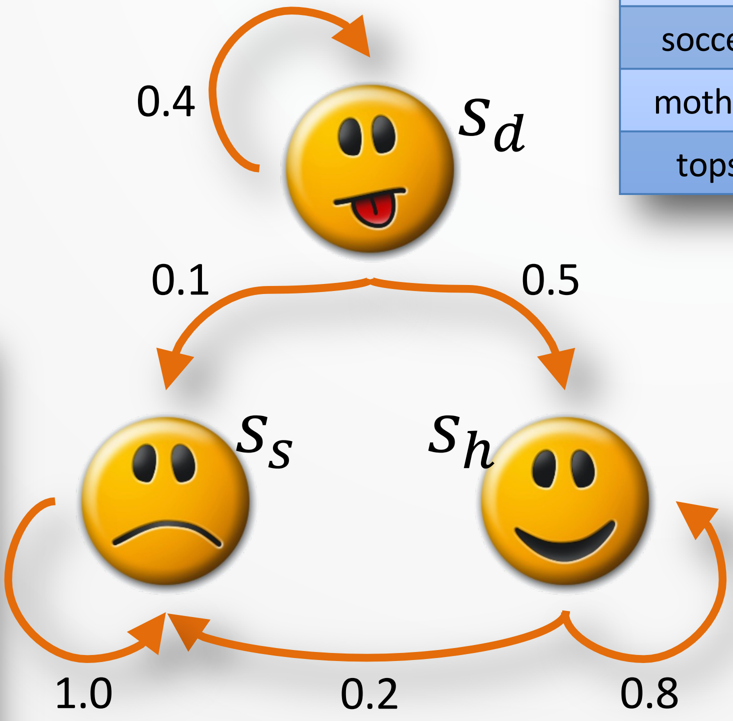
$$\boldsymbol{\delta_i(t)} = \max_{q_0 \dots q_{t-1}} P(q_0 \dots q_{t-1}, \sigma_0 \dots \sigma_t, \boldsymbol{q_t} = \boldsymbol{s_i}; \theta)$$

- $\boldsymbol{\psi_i(t)}$: The best possible previous state,
  if If I'm in state i at time t.

Viterbi example

- For illustration, we assume a simpler state-transition topology:

word	P(word)
ship	0.1
pass	0.05
camp	0.05
frock	0.6
soccer	0.05
mother	0.1
tops	0.05



word	P(word)
ship	0.25
pass	0.25
camp	0.05
frock	0.3
soccer	0.05
mother	0.09
tops	0.01

word	P(word)
ship	0.3
pass	0
camp	0
frock	0.2
soccer	0.05
mother	0.05
tops	0.4

Step 1: Initialization of Viterbi

- Initialize with $\delta_i(0) = \pi_i b_i(\sigma_0)$ and $\psi_i(0) = 0$ for all states.

$\boldsymbol{\pi_s b_s(\sigma_0)}$

0

$\boldsymbol{\pi_h b_h(\sigma_0)}$

0

$\boldsymbol{\pi_d b_d(\sigma_0)}$

0

δ: max probability

ψ: backtrace

0 1 2

Time, t

Step 1: Initialization of Viterbi

- For example, let's assume

$\pi_d = 0.8, \pi_h = 0.2$ and $\mathcal{O} = ship, frock, tops$

$\mathbf{0 \cdot 0.25}$

$\mathbf{0}$

$\mathbf{0.2 \cdot 0.3}$

$\mathbf{0}$

$\mathbf{0.8 \cdot 0.1}$

$\mathbf{0}$

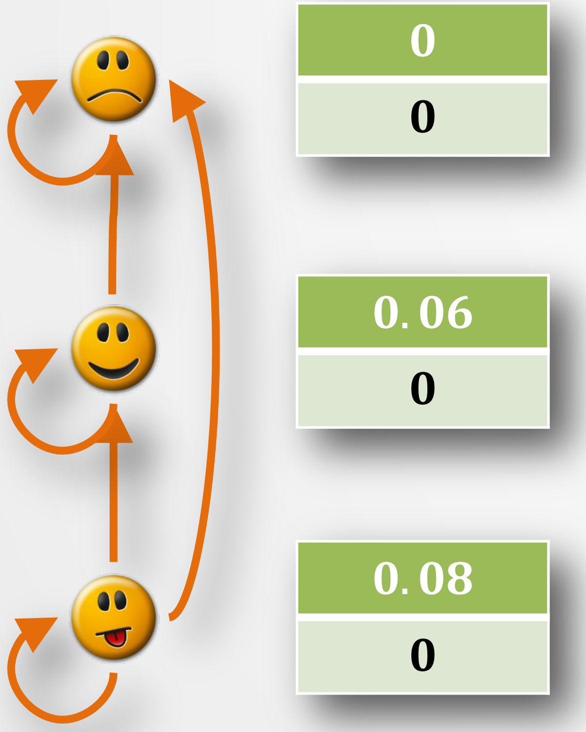
δ: max probability

ψ: backtrace

$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

Observations, σ_t

Step 2: Induction of Viterbi



0

0

0.06

0

0.08

0

The best path to state s_j at time t, $\delta_j(t)$, depends on the best path to each possible previous state, $\delta_i(t-1)$, and their transitions to j, a_{ij}

$$\boldsymbol{\delta_j(t) = \max_i \left[\delta_i(t-1)a_{ij}\right] b_j(\sigma_t)}$$

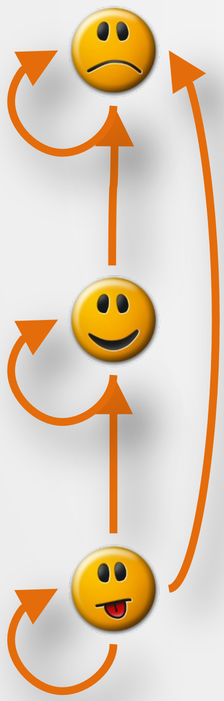
$$\boldsymbol{\psi_j(t) = \operatorname*{argmax}_i \left[\delta_i(t-1)a_{ij}\right]}$$

$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

Observations, σ_t

Step 2: Induction of Viterbi

Specifically…



$\mathbf{0}$

$\mathbf{0}$

$$\boldsymbol{\delta_s(1)} = \max_{\boldsymbol{i}} [\boldsymbol{\delta_i(0) a_{is}}]\, \boldsymbol{b_s(\sigma_1)}$$

$$\boldsymbol{\psi_s(1)} = \operatorname*{argmax}_{\boldsymbol{i}} [\boldsymbol{\delta_i(0) a_{is}}]$$

$\mathbf{0.06}$

$\mathbf{0}$

$$\boldsymbol{\delta_h(1)} = \max_{\boldsymbol{i}} [\boldsymbol{\delta_i(0) a_{ih}}]\, \boldsymbol{b_h(\sigma_1)}$$

$$\boldsymbol{\psi_h(1)} = \operatorname*{argmax}_{\boldsymbol{i}} [\boldsymbol{\delta_i(0) a_{ih}}]$$

$\mathbf{0.08}$

$\mathbf{0}$

$$\boldsymbol{\delta_d(1)} = \max_{\boldsymbol{i}} [\boldsymbol{\delta_i(0) a_{id}}]\, \boldsymbol{b_d(\sigma_1)}$$

$$\boldsymbol{\psi_d(1)} = \operatorname*{argmax}_{\boldsymbol{i}} [\boldsymbol{\delta_i(0) a_{id}}]$$

$\sigma_0 = ship$

$\sigma_1 = frock$

$\sigma_2 = tops$

Observations, σ_t

Step 2: Induction of Viterbi

0

0

0.06

0

0.08

0

$\psi_0(1) = \underset{i}{\text{argmax}}\,[\delta_i(0)a_{ih}]$

$\underset{i}{\max}\,[\boldsymbol{\delta_i(0)a_{id}}]\,\boldsymbol{b_d(\sigma_1)}$

$\underset{i}{\text{argmax}}\,[\boldsymbol{\delta_i(0)a_{id}}]$

$$\delta_s(0) = 0, a_{sd} = 0, \qquad \therefore \delta_s(0)a_{sd} = 0$$

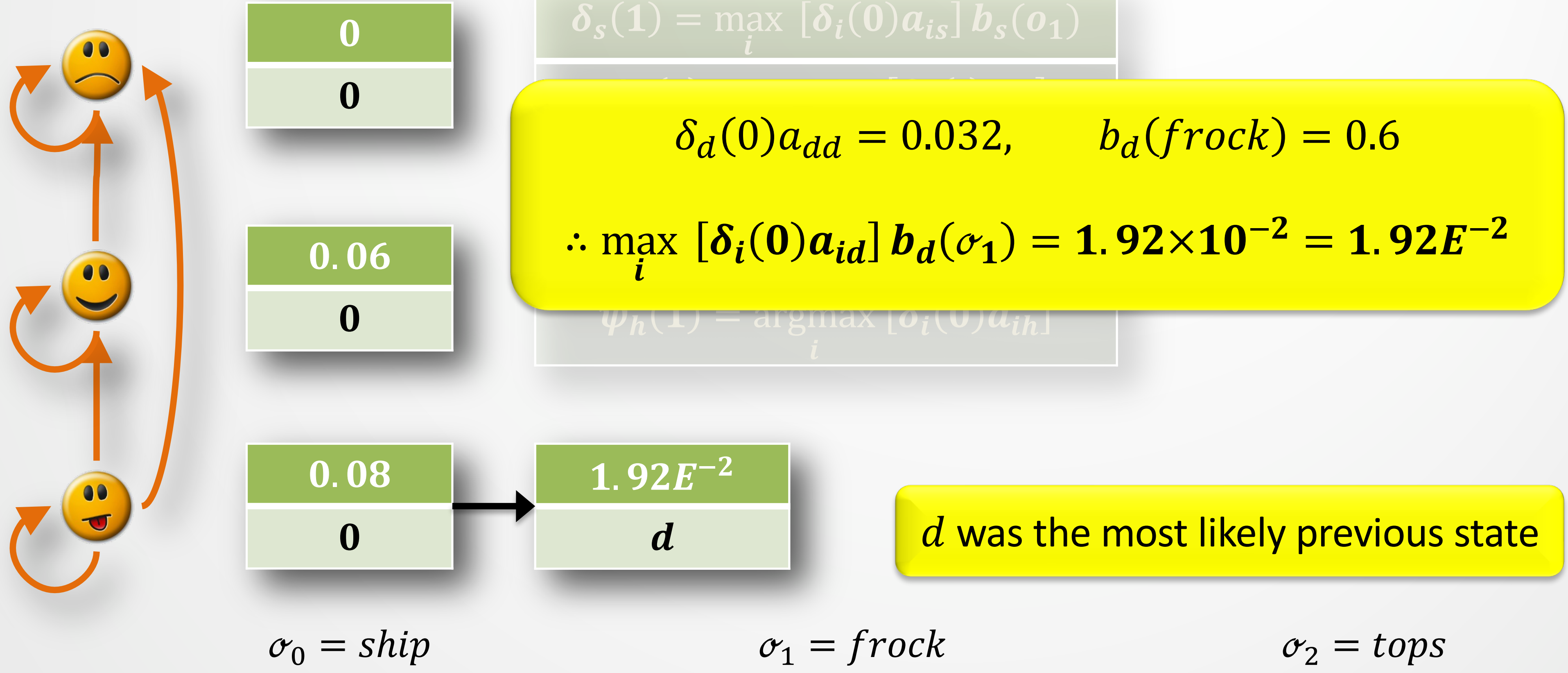
$$\delta_h(0) = 0.06, a_{hd} = 0, \qquad \therefore \delta_h(0)a_{hd} = 0$$

$$\delta_d(0) = 0.08, a_{dd} = 0.4, \qquad \therefore \boldsymbol{\delta_d(0)a_{dd} = 0.032}$$

$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

Observations, σ_t

Step 2: Induction of Viterbi



$$\delta_s(1) = \max_i [\delta_i(0)a_{is}]\, b_s(o_1)$$

$$\delta_d(0)a_{dd} = 0.032, \qquad b_d(frock) = 0.6$$

$$\therefore \max_i [\boldsymbol{\delta_i(0)a_{id}}]\, \boldsymbol{b_d(o_1)} = \mathbf{1.92{\times}10^{-2}} = \mathbf{1.92E^{-2}}$$

$$\psi_h(1) = \operatorname{argmax}_i [\delta_i(0)a_{ih}]$$

0	
0	
0.06	
0	
0.08	$\mathbf{1.92E^{-2}}$
0	$\boldsymbol{d}$

d was the most likely previous state

$o_0 = ship$ $o_1 = frock$ $o_2 = tops$

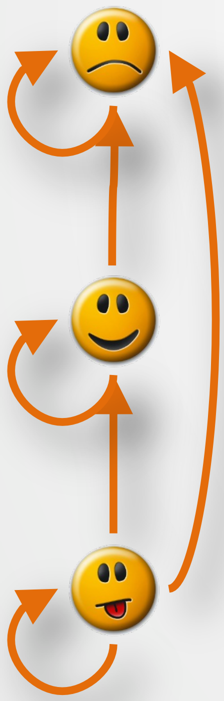
Observations, o_t

Step 2: Induction of Viterbi

$$\delta_s(0) = 0, a_{sh} = 0, \qquad \therefore \delta_s(0)a_{sh} = 0$$

$$\delta_h(0) = 0.06, a_{hh} = 0.8, \qquad \therefore \boldsymbol{\delta_h(0)a_{hh} = 0.048}$$

$$\delta_d(0) = 0.08, a_{dh} = 0.5, \qquad \therefore \delta_d(0)a_{dh} = 0.04$$



$\mathbf{0}$

$\mathbf{0}$

$\mathbf{0.06}$

$\mathbf{0}$

$$\max_i \left[\boldsymbol{\delta_i(0)a_{ih}}\right] \boldsymbol{b_h(\sigma_1)}$$

$$\operatorname{argmax}_i \left[\boldsymbol{\delta_i(0)a_{ih}}\right]$$

$\mathbf{0.08}$

$\mathbf{0}$

$$\mathbf{0.08} \rightarrow \mathbf{1.92E^{-2}}$$

$\boldsymbol{d}$

$\sigma_0 = ship$ $\qquad$ $\sigma_1 = frock$ $\qquad$ $\sigma_2 = tops$

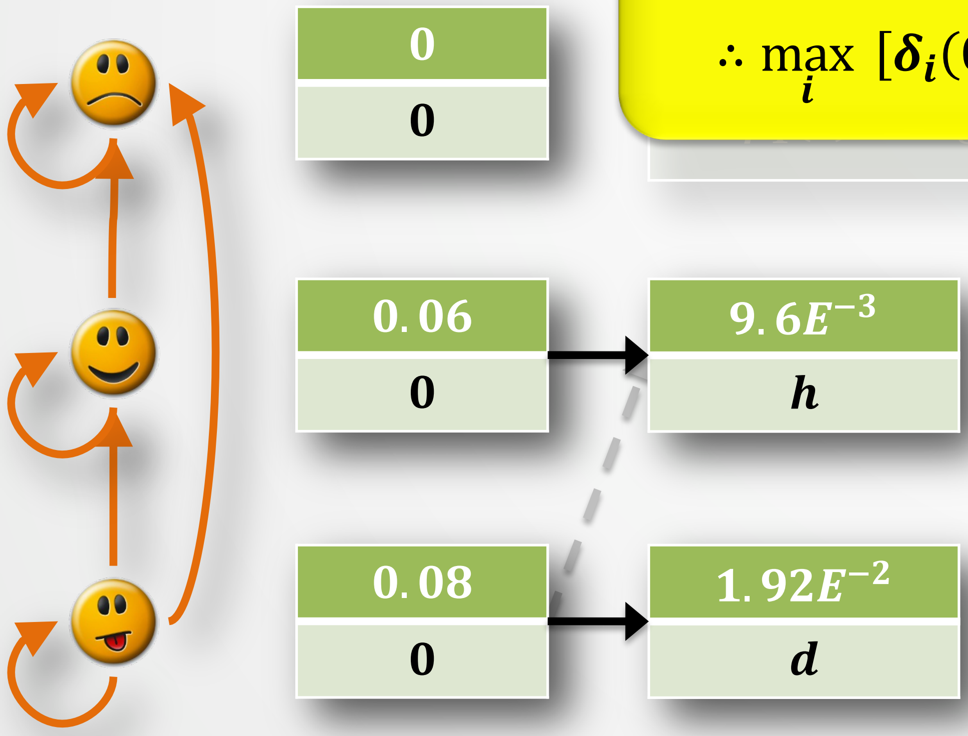
Observations, σ_t

UNIVERSITY OF TORONTO

Step 2: Induction of Viterbi

$$\delta_h(0)a_{hh} = 0.048, \qquad b_h(frock) = 0.2$$

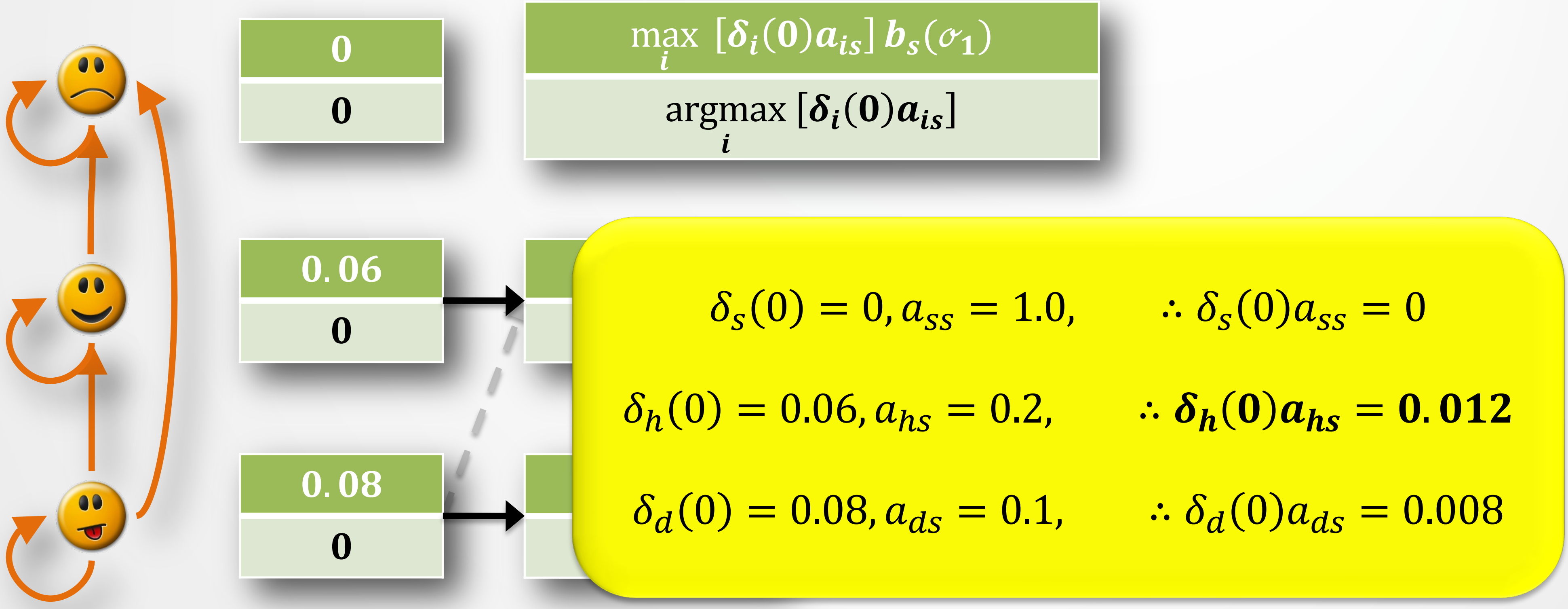
$$\therefore \max_i \left[\boldsymbol{\delta_i(0)a_{ih}}\right] \boldsymbol{b_h(\sigma_1)} = \mathbf{9.6{\times}10^{-3}} = \boldsymbol{9.6E^{-3}}$$



$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

Observations, σ_t

Step 2: Induction of Viterbi



$\mathbf{0}$

$\mathbf{0}$

$\mathbf{0.06}$

$\mathbf{0}$

$\mathbf{0.08}$

$\mathbf{0}$

$\max_{i}\,[\boldsymbol{\delta_i(0)a_{is}}]\,\boldsymbol{b_s(\sigma_1)}$

$\operatorname{argmax}_{\boldsymbol{i}}\,[\boldsymbol{\delta_i(0)a_{is}}]$

$\delta_s(0) = 0, a_{ss} = 1.0, \quad \therefore \delta_s(0)a_{ss} = 0$

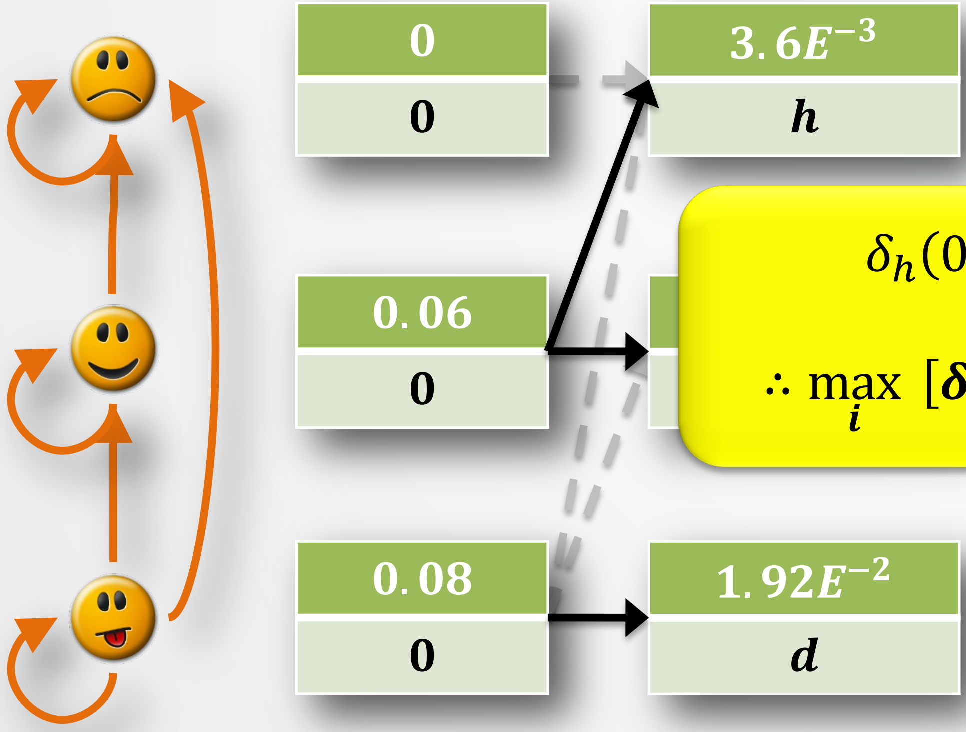
$\delta_h(0) = 0.06, a_{hs} = 0.2, \quad \therefore \boldsymbol{\delta_h(0)a_{hs} = 0.012}$

$\delta_d(0) = 0.08, a_{ds} = 0.1, \quad \therefore \delta_d(0)a_{ds} = 0.008$

$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

Observations, σ_t

Step 2: Induction of Viterbi



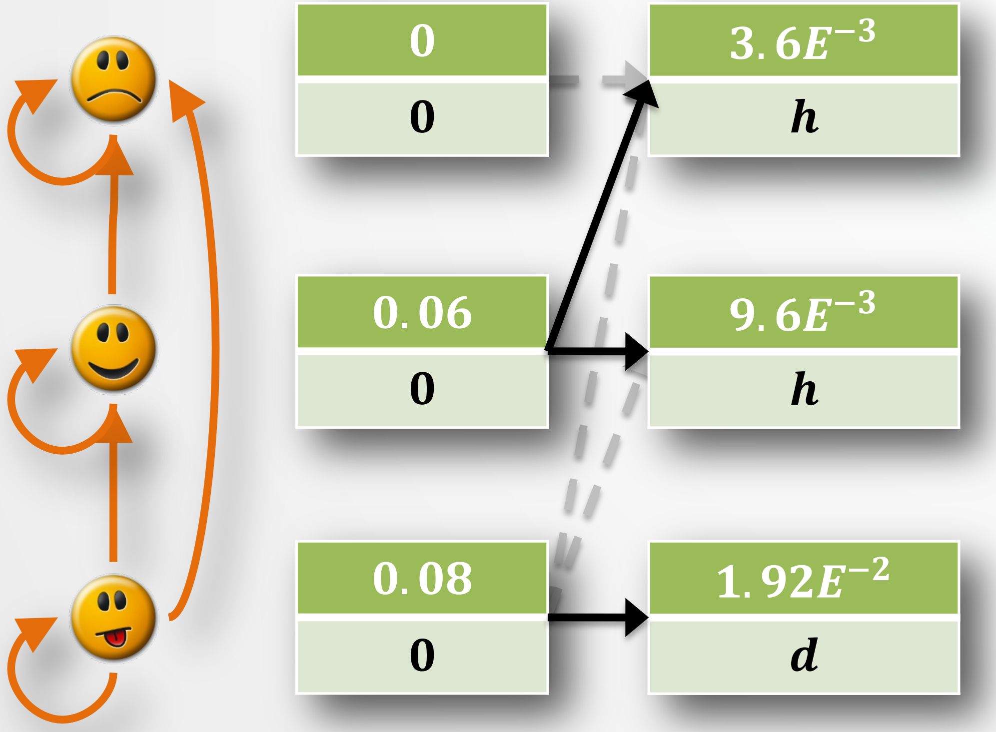
$$\delta_h(0)a_{hh} = 0.012, \qquad b_s(frock) = 0.3$$

$$\therefore \max_{\boldsymbol{i}} \left[\boldsymbol{\delta_i(0)a_{is}}\right] \boldsymbol{b_s(\sigma_1) = 3.6\times10^{-3} = 3.6E^{-3}}$$

$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

Observations, σ_t

Step 2: Induction of Viterbi



$\delta_s(2) = \max_i \, [\delta_i(1) a_{is}] \, b_s(\sigma_2)$

$\psi_s(2) = \operatorname{argmax}_i \, [\delta_i(1) a_{is}]$

$\delta_h(2) = \max_i \, [\delta_i(1) a_{ih}] \, b_h(\sigma_2)$

$\psi_h(2) = \operatorname{argmax}_i \, [\delta_i(1) a_{ih}]$

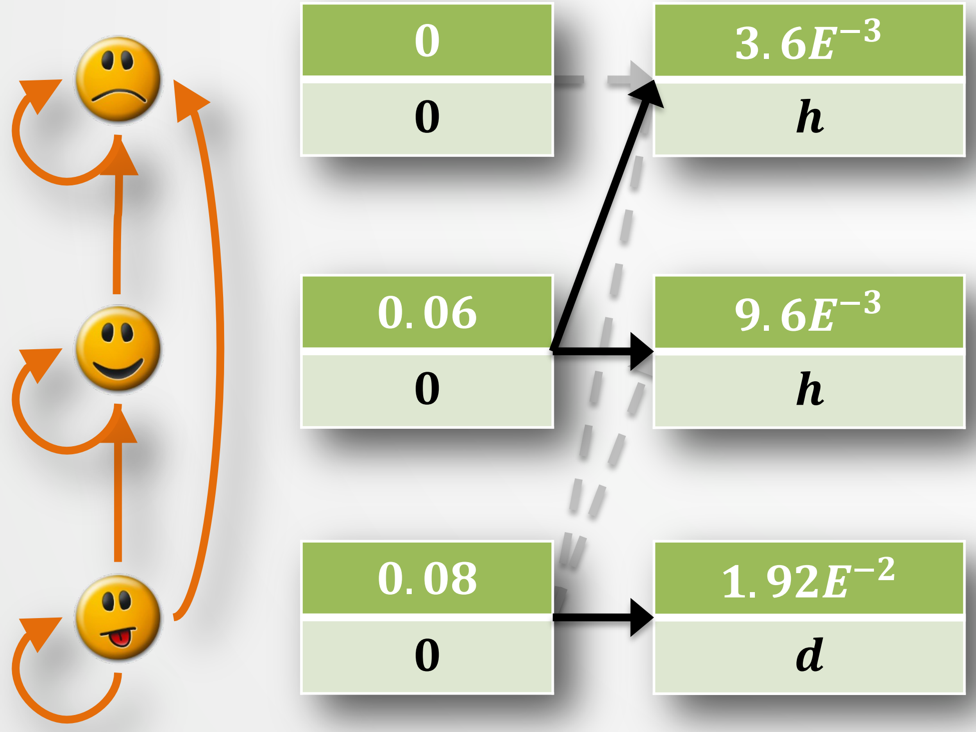
$\delta_d(2) = \max_i \, [\delta_i(1) a_{is}] \, b_s(\sigma_2)$

$\psi_d(2) = \operatorname{argmax}_i \, [\delta_i(1) a_{id}]$

$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

Observations, σ_t

Step 2: Induction of Viterbi



$$\delta_s(1) = 3.6E^{-3}, a_{sd} = 0,$$
$$\therefore \delta_s(1)a_{sd} = 0$$

$$\delta_h(1) = 9.6E^{-3}, a_{hd} = 0,$$
$$\therefore \delta_h(1)a_{hd} = 0$$

$$\delta_d(1) = 1.92E^{-2}, a_{dd} = 0.4,$$
$$\therefore \boldsymbol{\delta_d(1)a_{dd} = 0.00768}$$

$$\boldsymbol{\psi_h(2) = \underset{i}{\operatorname{argmax}} [\delta_i(1)a_{ih}]}$$

$$\delta_d(2) = \max_i \, [\delta_i(1)a_{is}] \, b_s(\sigma_2)$$

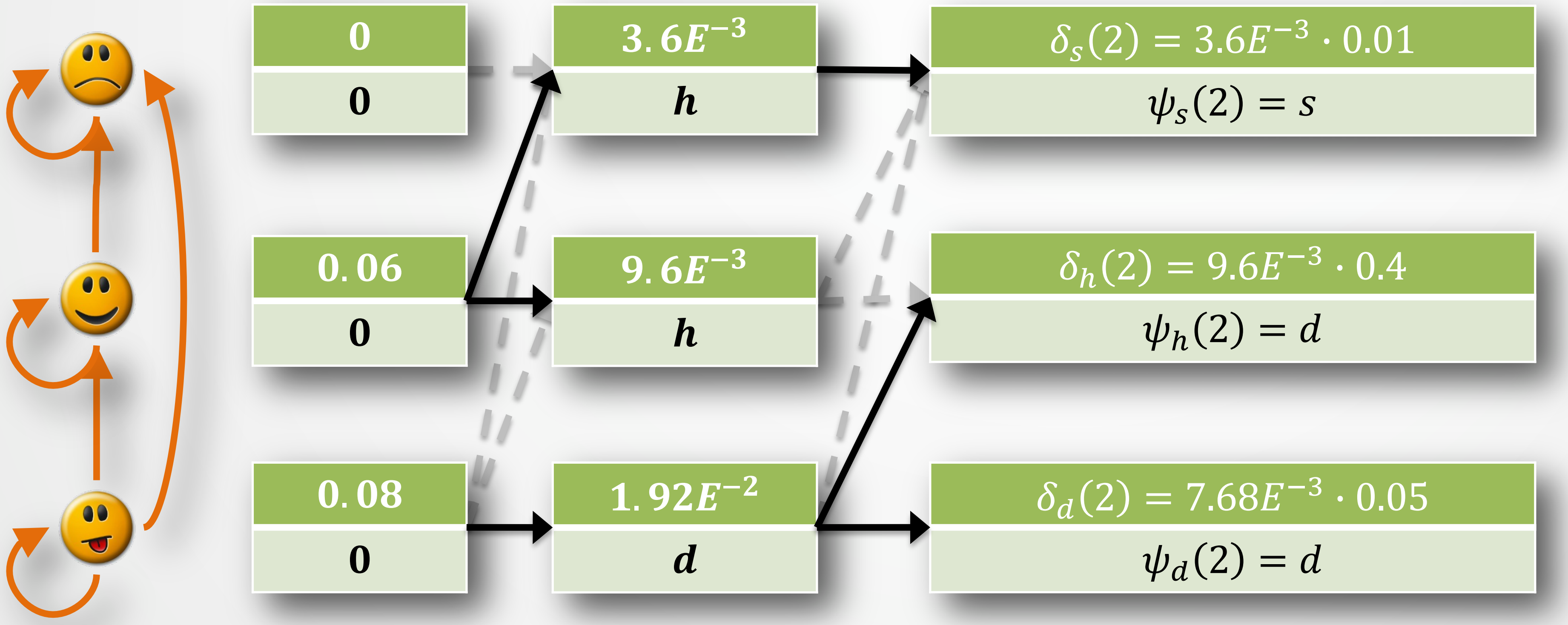
$$\psi_d(2) = \underset{i}{\operatorname{argmax}} [\delta_i(1)a_{id}]$$

$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

Observations, σ_t

Step 2: Induction of Viterbi

Continuing…



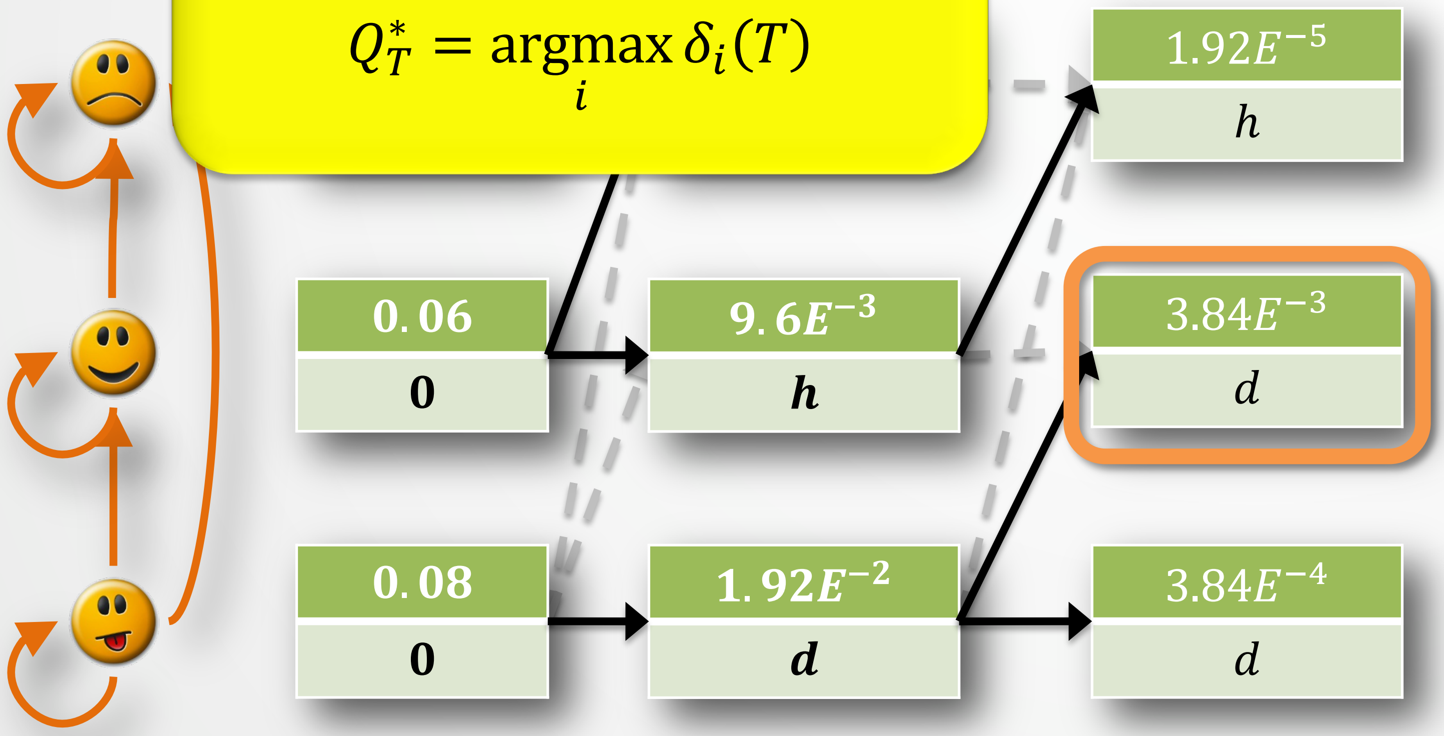
$\mathbf{0}$ $\mathbf{0}$	$\mathbf{3.6E^{-3}}$ $\boldsymbol{h}$	$\delta_s(2) = 3.6E^{-3} \cdot 0.01$ $\psi_s(2) = s$
$\mathbf{0.06}$ $\mathbf{0}$	$\mathbf{9.6E^{-3}}$ $\boldsymbol{h}$	$\delta_h(2) = 9.6E^{-3} \cdot 0.4$ $\psi_h(2) = d$
$\mathbf{0.08}$ $\mathbf{0}$	$\mathbf{1.92E^{-2}}$ $\boldsymbol{d}$	$\delta_d(2) = 7.68E^{-3} \cdot 0.05$ $\psi_d(2) = d$
$\sigma_0 = ship$	$\sigma_1 = frock$	$\sigma_2 = tops$

Observations, σ_t

Step 3: Conclusion of Viterbi

Choose the best final state:

$$Q_T^* = \underset{i}{\operatorname{argmax}}\, \delta_i(T)$$



$1.92E^{-5}$

h

0.06

0

$9.6E^{-3}$

h

$3.84E^{-3}$

d

0.08

0

$1.92E^{-2}$

d

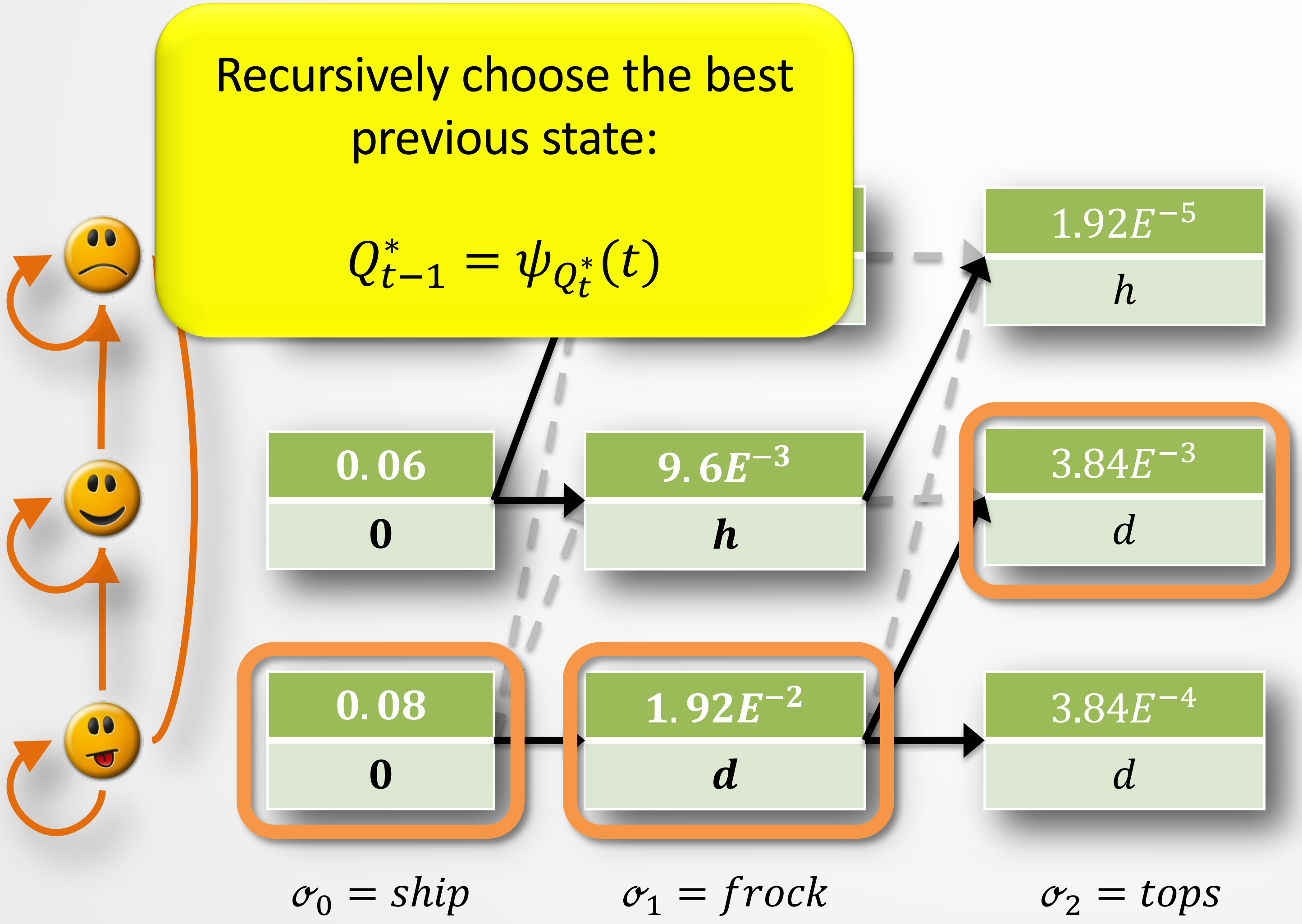
$3.84E^{-4}$

d

$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

Observations, σ_t

Step 3: Conclusion of Viterbi



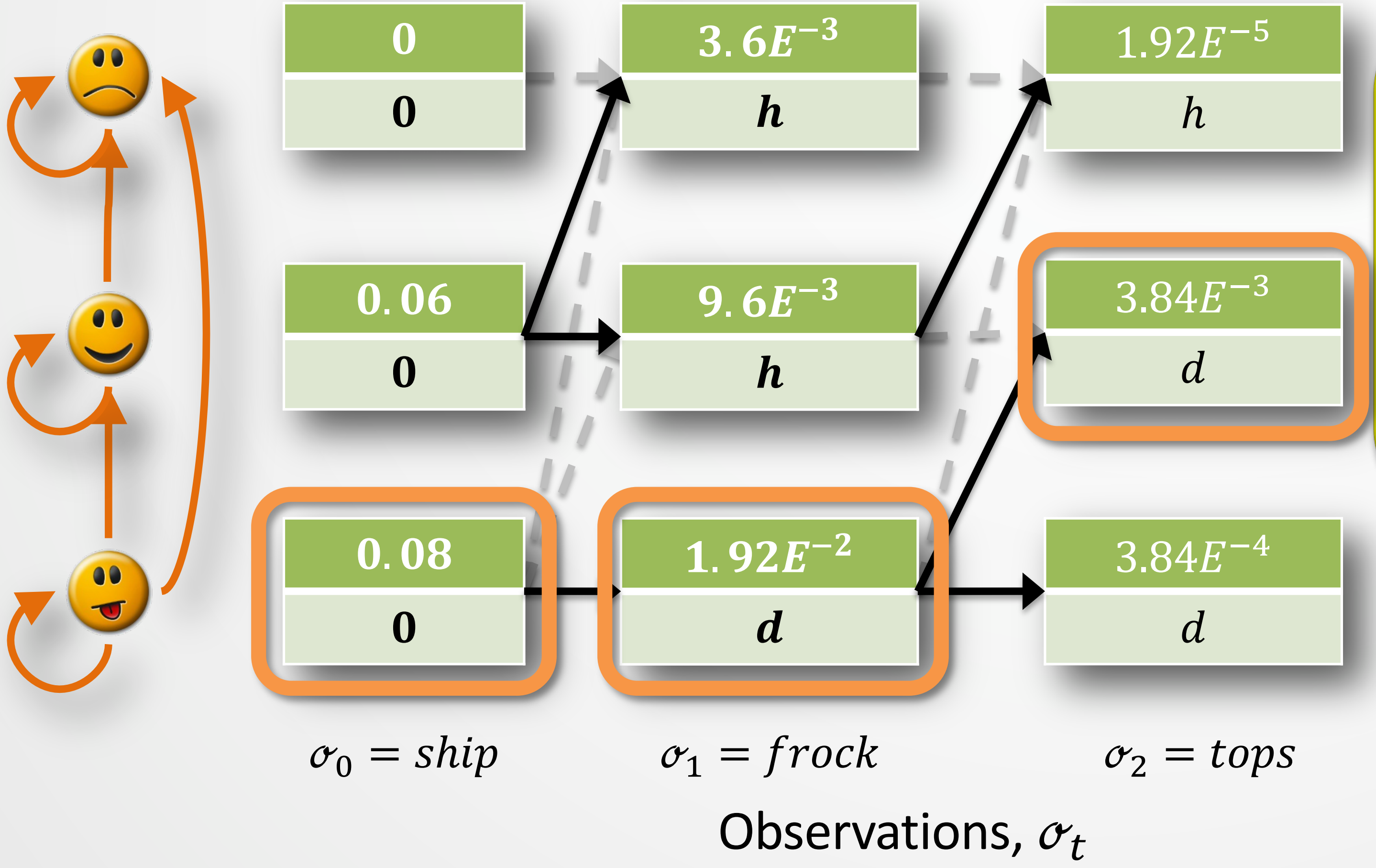
Recursively choose the best previous state:

$$Q^*_{t-1} = \psi_{Q^*_t}(t)$$

Observations, o_t

Step 3: Conclusion of Viterbi

Breaking ties: any tie-breaking heuristic algorithm like **random** path **choice** can be applied



Sequence probability:

$$P(\mathcal{O}, Q^*; \theta) = \max_i \delta_i(T)$$

Aside - Working in the log domain

- Our formulation was

$$Q^* = \text{argmax}_Q\, P(\mathcal{O}, Q; \theta)$$

this is equivalent to

$$Q^* = \underset{Q}{\text{argmin}}\, -\log_2 P(\mathcal{O}, Q; \theta)$$

where

$$-\log_2 P(\mathcal{O}, Q; \theta)$$

$$= -\log{}_2 \left(\pi_{q_0} b_{q_0}(\sigma_0)\right) - \sum_{t=1}^{T-1} \log_2 \left(a_{q_{t-1}q_t} b_{q_t}(\sigma_t)\right)$$

Fundamental tasks for HMMs

3. Given a large **observation sequence** $\mathcal{O}$ for **training**, but ***not*** the state sequence, how do we choose the 'best' parameters $\theta = \langle \Pi, A, B \rangle$ that explain the data $\mathcal{O}$?

This is the task of **training**.

As with observable Markov models and **MLE**, we want our parameters to be set so that
the available training data is maximally likely,
But doing so will involve **guessing unseen information…**

Task 3: Choosing $\boldsymbol{\theta} = \langle \boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B} \rangle$

- We want to **modify** the parameters of our model $\theta = \langle \Pi, A, B \rangle$ so that $P(\mathcal{O}; \theta)$ is maximized for some **training** data $\mathcal{O}$:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \, P(\mathcal{O}; \theta)$$

- Why? E.g., if we later want to choose the **best state sequence** Q^* for previously unseen **test data**, the parameters of the HMM should be tuned to similar **training data.**

Task 3: Choosing $\boldsymbol{\theta} = \langle \boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B} \rangle$

- $\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, P(\mathcal{O};\theta) \;\; = \underset{\theta}{\operatorname{argmax}} \sum_Q P(\mathcal{O},Q;\theta)$

Can we do this?

- $P(\mathcal{O},Q;\theta) = P(q_{0:T-1})P(w_{0:t}|q_{0:t}) \approx \prod_{i=0}^{t} \underline{P(q_i|q_{i-1})}\underline{P(w_i|q_i)}$

Recall that we could use MLE when Q was known

Task 3: Choosing $\boldsymbol{\theta} = \langle \boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B} \rangle$

- $P(\mathcal{O}, Q; \theta) = P(q_{0:t})P(w_{0:t}|q_{0:t}) \approx \prod_{i=0}^{t} \underline{P(q_i|q_{i-1})}\,\underline{P(w_i|q_i)}$
- If the training data <u>contained</u> state sequences, we could simply do **maximum likelihood estimation**, as before:
  - $\cancel{P(q_i|q_{i-1}) = \frac{Count(q_{i-1}\, q_i)}{Count(q_{i-1})}}$ $\cancel{P(w_i|q_i) = \frac{Count(w_i \wedge q_i)}{Count(q_i)}}$
- But we <u>***don't***</u> know the states; we <u>***can't***</u> count them.
- However, we *can* use an **iterative hill-climbing** approach if we can ***guess*** the counts using a "good" pre-existing model

Expecting and maximizing

- If we knew θ, we could make **expectations** such as
  - Expected number of times in state s_i,
  - Expected number of transitions $s_i \rightarrow s_j$

- If we knew:
  - Expected number of times in state s_i,
  - Expected number of transitions $s_i \rightarrow s_j$

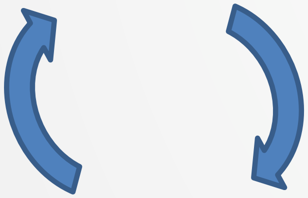
then we could compute the **maximum likelihood estimate** of

$$\theta = \langle \pi_i, \{a_{ij}\}, \{b_i(w)\} \rangle$$

Expectation-maximization

- **Expectation-maximization** (EM) is an **iterative** training algorithm that alternates between two steps:

  - **Expectation (E)**: guesses the **expected** counts for the hidden sequence using the current model θ_k.



  - **Maximization (M)**: computes a new θ that **maximizes** the likelihood of the data, given the guesses of the E-step. This θ_{k+1} is then used in the next E-step.

  - Continue until $\overbrace{\text{convergence}}^{\boxed{|\theta_{k+1} - \theta_k| < \epsilon}}$ or stopping condition…

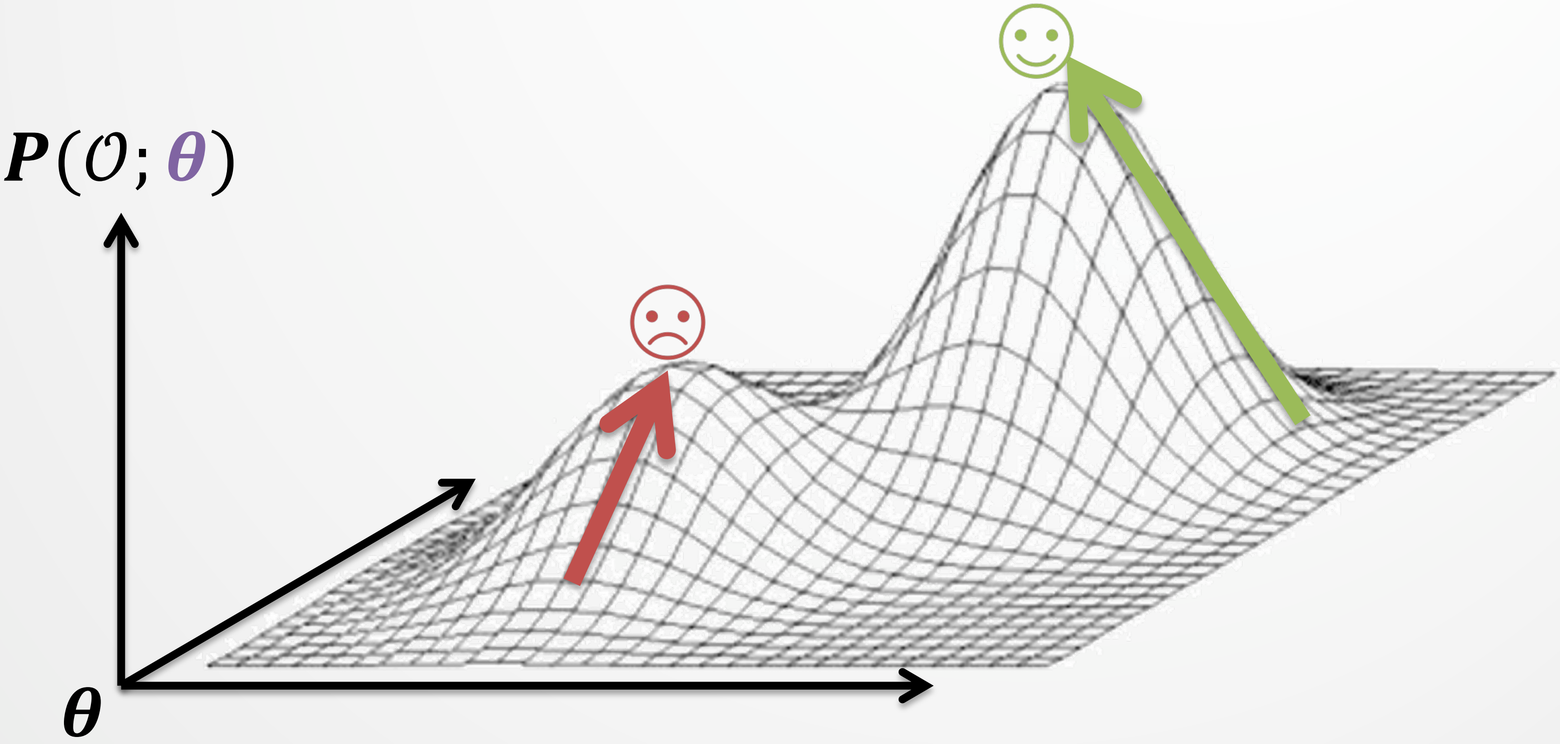
Baum-Welch re-estimation

- **Baum-Welch** (BW): *n.* a specific version of EM for HMMs.
  a.k.a. '**forward-backward**' algorithm.

1. Initialize the model.
2. E-step: Compute **expectations** for $Count(q_{t-1}q_t)$ and $Count(q_t \wedge w_t)$ given model, training data $\mathcal{O}$.
3. M-step: Adjust our start, transition, and observation probabilities to **maximize** the likelihood of $\mathcal{O}$.
4. **Go to 2**. and repeat until convergence or stopping condition…

Local maxima

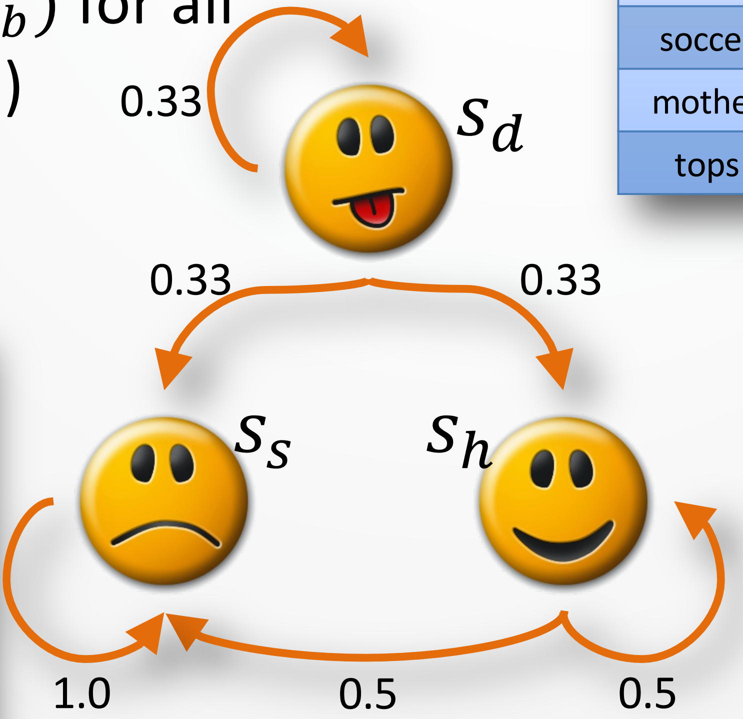
- Baum-Welch changes θ to climb a `hill' in $P(\mathcal{O};\theta)$.
  - How we initialize θ can have a big effect.



Step 1: BW initialization

- Our **initial guess** for the parameters, θ_0, can be:
  a) All probabilities are **uniform** (e.g., $b_i(w_a) = b_i(w_b)$ for all states i and words w)

word	P(word)
ship	0.143
pass	0.143
camp	0.143
frock	0.143
soccer	0.143
mother	0.143
tops	0.143



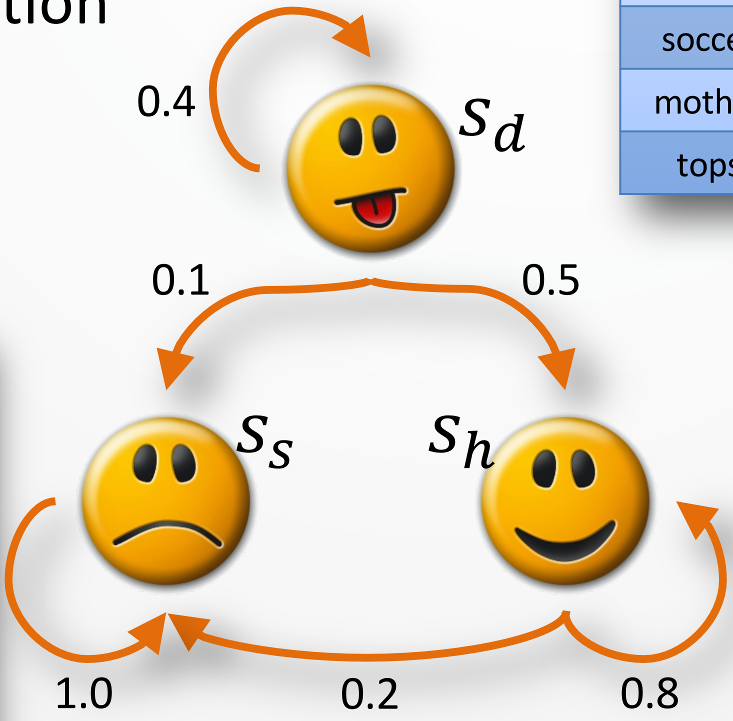
word	P(word)
ship	0.143
pass	0.143
camp	0.143
frock	0.143
soccer	0.143
mother	0.143
tops	0.143

word	P(word)
ship	0.143
pass	0.143
camp	0.143
frock	0.143
soccer	0.143
mother	0.143
tops	0.143

Step 1: BW initialization

- Our **initial guess** for the parameters, θ_0, can be:
  b) All probabilities are drawn **randomly** (subject to the condition that $\sum_i P(i) = 1$)

word	P(word)
ship	0.1
pass	0.05
camp	0.05
frock	0.6
soccer	0.05
mother	0.1
tops	0.05



word	P(word)
ship	0.25
pass	0.25
camp	0.05
frock	0.3
soccer	0.05
mother	0.09
tops	0.01

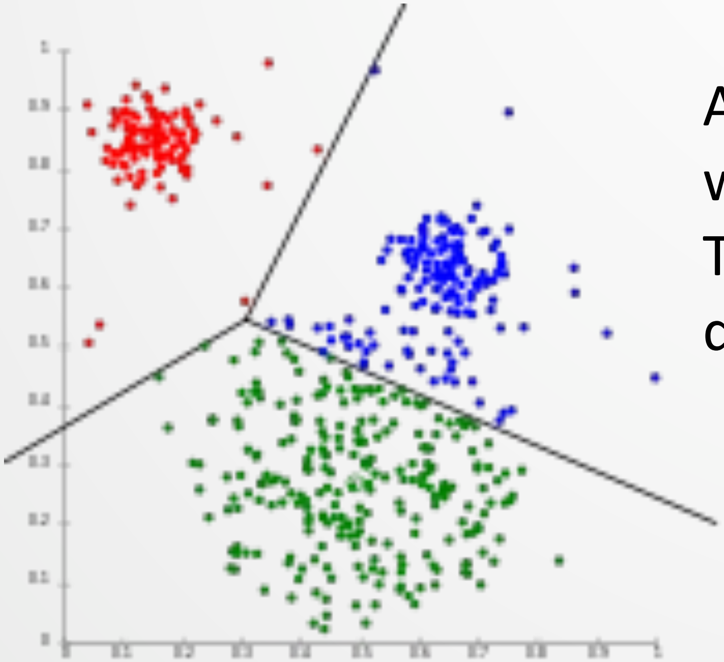
word	P(word)
ship	0.3
pass	0
camp	0
frock	0.2
soccer	0.05
mother	0.05
tops	0.4

Step 1: BW initialization

- Our **initial guess** for the parameters, θ_0, can be:
  - c) Observation distributions are drawn from **prior** distributions:

    e.g., $b_i(w_a) = P(w_a)$ for all states i.

    sometimes this involves pre-clustering, e.g. k-means



All blue dots are words in state BLUE. Their probability distribution is →

word	P(word)
ship	0.2
pass	0.1
camp	0.03
frock	0.5
soccer	0.07
mother	0.02
tops	0.08

What to expect when you're expecting

- If we knew θ, we could estimate **expectations** such as
  - Expected number of times in state s_i,
  - Expected number of transitions $s_i \rightarrow s_j$

- If we knew:
  - Expected number of times in state s_i,
  - Expected number of transitions $s_i \rightarrow s_j$

then we could compute the **maximum likelihood estimate** of

$$\theta = \langle \{a_{ij}\}, \{b_i(w)\}, \pi_i \rangle$$

BW E-step (occupation)

- We define

$$\boldsymbol{\gamma_i(t) = P(q_t = i | \mathcal{O}; \theta_k)}$$

as the probability of **being** in state i at time t, based on our current model, θ_k, ***given*** the ***entire*** observation, $\mathcal{O}$.

and rewrite as:

$$\gamma_i(t) = \frac{P(q_t = i, \mathcal{O}; \theta_k)}{P(\mathcal{O}; \theta_k)}$$

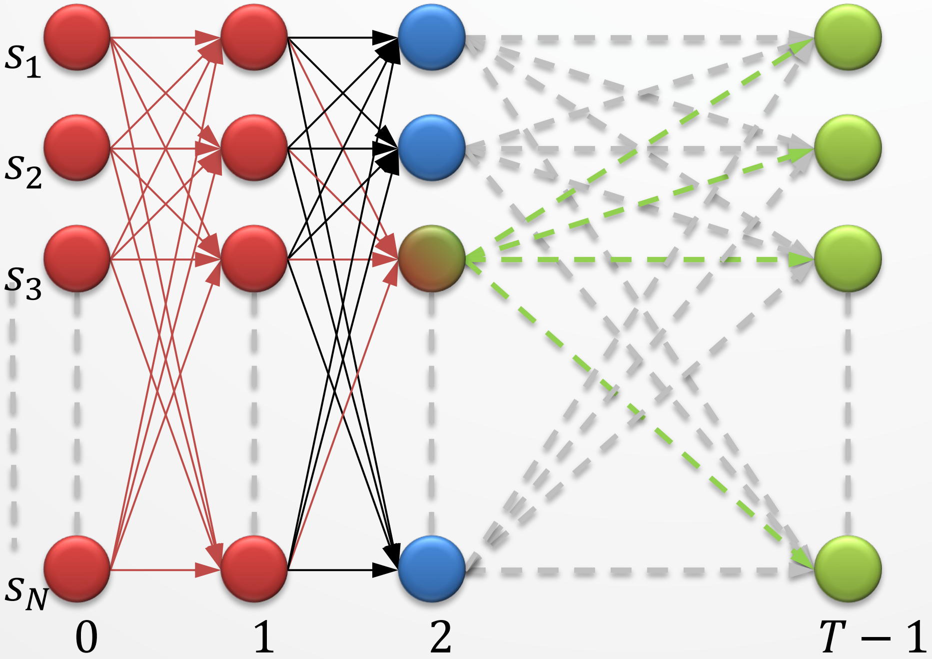
$$= \frac{\alpha_i(t)\beta_i(t)}{P(\mathcal{O}; \theta_k)}$$

Remember, $\alpha_i(t)$ and $\beta_i(t)$ depend on values from $\theta = \langle \pi_i, a_{ij}, b_i(w) \rangle$

Combining α and β

$$P(\mathcal{O}, q_t = i; \theta) = \alpha_i(t)\beta_i(t)$$

$$\therefore P(\mathcal{O}; \theta) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$



BW E-step (transition)

- We define

$$\boldsymbol{\xi_{ij}(t) = P(q_t = i, q_{t+1} = j | \mathcal{O}; \theta_k)}$$

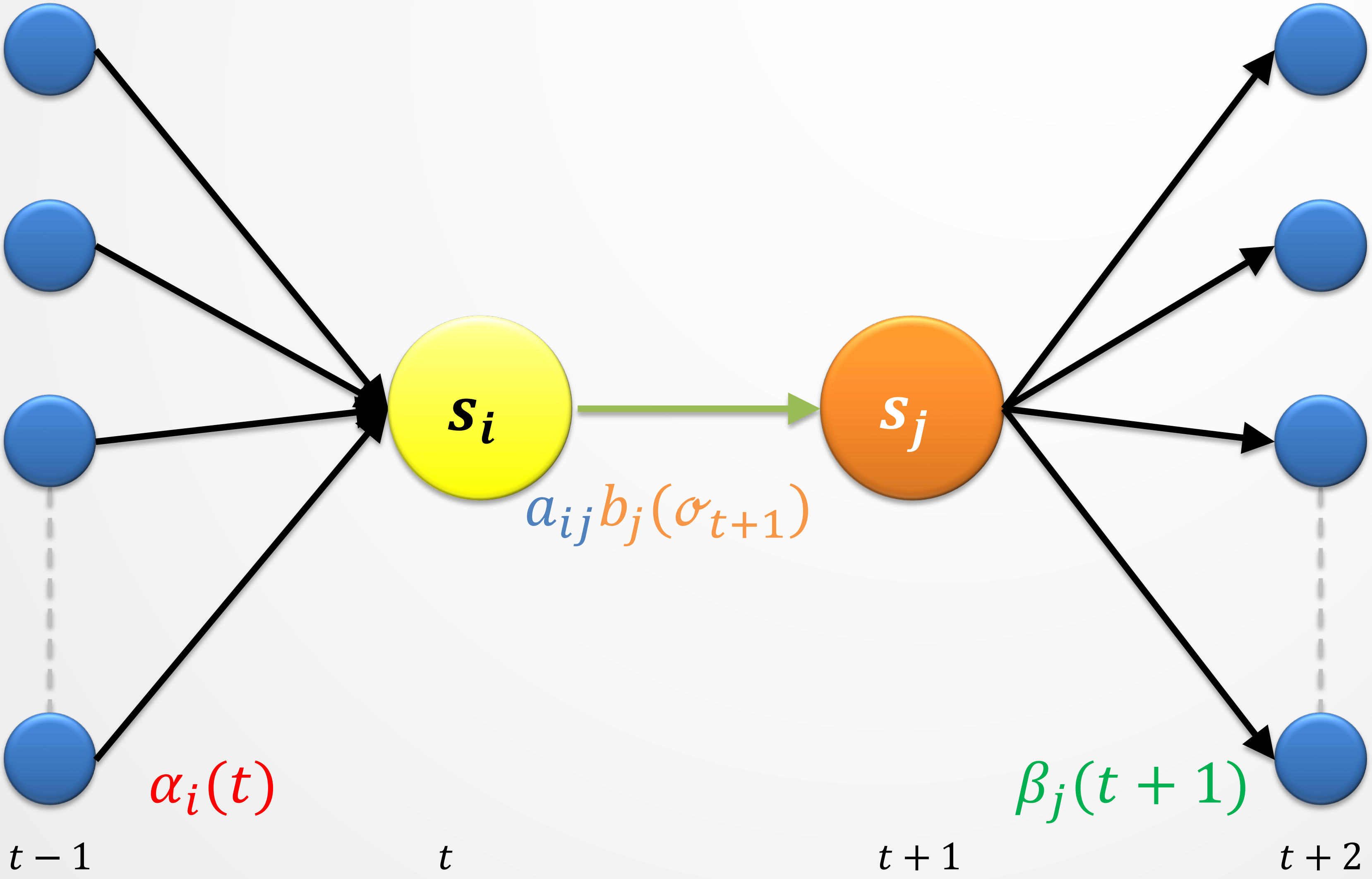
as the probability of **transitioning** from state i at time t to state j at time $t+1$ ***based on*** our current model, θ_k, and **given** the ***entire*** observation, $\mathcal{O}$. This is:

$$\xi_{ij}(t) = \frac{P(q_t = i, q_{t+1} = j, \mathcal{O}; \theta_k)}{P(\mathcal{O}; \theta_k)}$$

$$= \frac{\alpha_i(t) a_{ij} b_j(\sigma_{t+1}) \beta_j(t+1)}{P(\mathcal{O}; \theta_k)}$$

Again, these estimates come from our model at iteration k, θ_k.

BW E-step (transition)



Expecting and maximizing

- If we knew θ, we could estimate **expectations** such as
  - Expected number of times in state s_i,
  - Expected number of transitions $s_i \rightarrow s_j$

$\boldsymbol{\gamma_i(t)}$ - from the E-step (occupation)
$\boldsymbol{\xi_{ij}(t)}$ - from the E-step (transition)

- If we knew:
  - Expected number of times in state s_i,
  - Expected number of transitions $s_i \rightarrow s_j$

then we could compute the **maximum likelihood estimate** of

$$\theta = \langle \{a_{ij}\}, \{b_i(w)\}, \pi_i \rangle$$

BW M-step

We **update our parameters** as if we were doing **MLE**:

I. Initial-state probabilities:

$$\hat{\pi}_i = \gamma_i(0) \qquad \text{for } i \coloneqq 1..N$$

II. State-transition probabilities:

$$P(q_j|q_i) = \frac{Count(q_i\, q_j)}{Count(q_i)}$$

$$\hat{a}_{ij} = \frac{\sum_{t=0}^{T-2} \xi_{ij}(t)}{\sum_{t=0}^{T-2} \gamma_i(t)} \qquad \text{for } i,j \coloneqq 1..N$$

III. Discrete observation probabilities:

$$P(w_i|q_i) = \frac{Count(w_i \wedge q_i)}{Count(q_i)}$$

$$\hat{b}_j(w) = \frac{\sum_{t=0}^{T-1} \gamma_j(t)|_{\sigma_t=w}}{\sum_{t=0}^{T-1} \gamma_j(t)} \qquad \text{for } j \coloneqq 1..N \text{ and } w \in \mathcal{V}$$

Baum-Welch iteration

- We update our parameters after **each iteration**

$$\theta_{k+1} = \langle \hat{\pi}_i, \hat{a}_{ij}, \hat{b}_j(w) \rangle$$

rinse, and repeat until $\theta_k \approx \theta_{k+1}$ (until change almost stops).

- Baum proved that

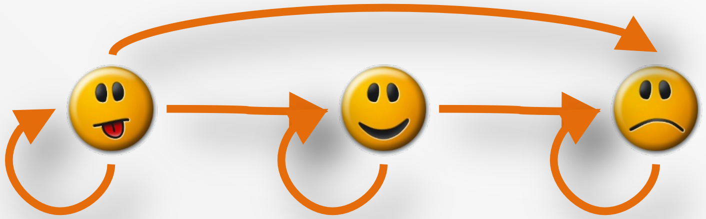
$$P(\mathcal{O}; \theta_{k+1}) \geq P(\mathcal{O}; \theta_k)$$

although this method does *not* guarantee a ***global maximum***.

Features of Baum-Welch

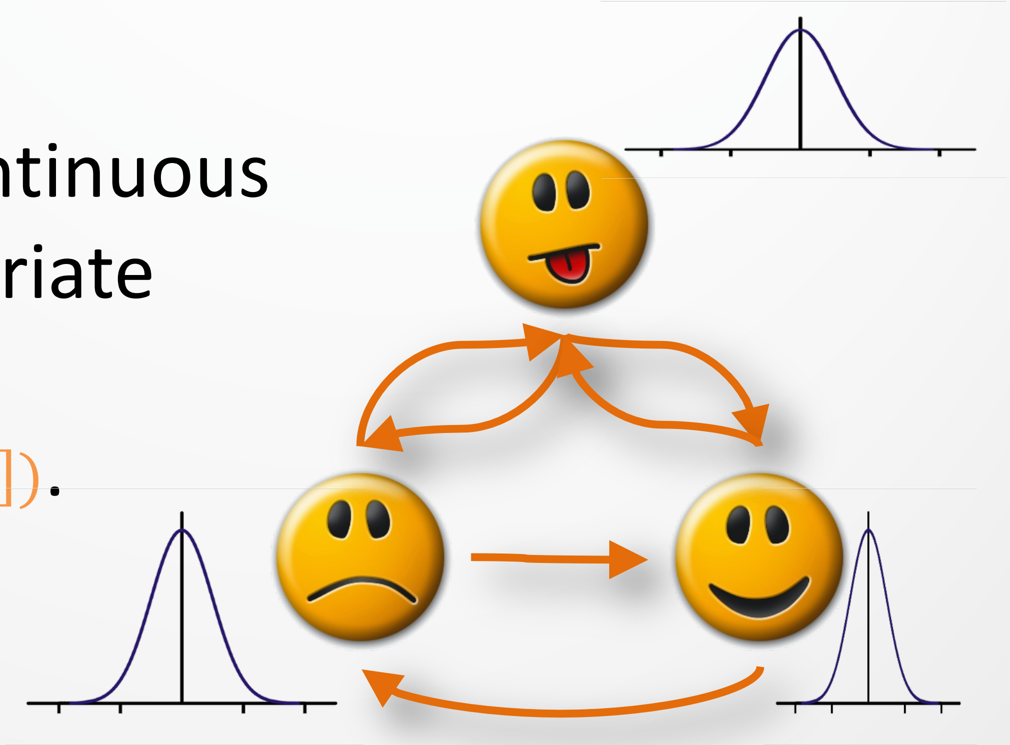
- Although we’re **not guaranteed** to achieve a **global optimum**, the **local optima** are often ‘**good enough**’.

- BW does **not** estimate the number of states, which must be ‘known’ beforehand.
  - Moreover, some constraints on topology are often imposed beforehand to assist training.



Discrete vs. continuous

- If our observations are drawn from a **continuous** space (e.g., speech acoustics), the probabilities $b_i(X)$ must also be continuous.
- HMMs **generalize** to continuous distributions, or multivariate observations, e.g., $b_i([-14.28,\ \ 0.85,\ \ 0.21])$.



Adaptation

- It can take a ***LOT*** of data to train HMMs.
- Imagine that we're given a **trained** HMM but not the data.
    - Also imagine that this HMM has been trained with data from **many** sources (e.g., many speakers).

- We want to use this HMM with a **particular new source** for whom we have **some** data (but not enough to fully train the HMM properly from scratch).
    - To be **more accurate for that source**, we want to **change** the original HMM parameters *slightly* given the new data.

HMM interpolation

- For added robustness, we can combine estimates of a **generic** HMM, G, trained with **lots of data** from many sources with a **specific** HMM, S, trained with **a little data** from a single source.

$$P_{Interp}(\sigma) = \lambda P(\sigma; \theta_G) + (1 - \lambda) P(\sigma; \theta_S)$$

- This gives us a model tuned to our target source (S), but with some general ‘knowledge’ (G) built in.
  - How do we pick $\lambda \in [\mathbf{0..1}]$?

EM for interpolated models

- Strategy can be used for any $P(\mathcal{O};\lambda) = \sum_i \lambda_i P_i(\mathcal{O})$
- Introduce latent states s such that $P(s = i;\lambda) = \lambda_i$
- Once in state i, $P(\mathcal{O}|s = i;\lambda) = P_i(\mathcal{O})$
- Like with HMMs, we estimate $Count(s = i)$ using EM:

$$\lambda_i^{new} = \frac{P(s = i, \mathcal{O};\lambda^{old})}{P(\mathcal{O};\lambda^{old})}$$

- This is a (simplified) version of what is done for **Jelinek-Mercer interpolation**, as well as **Gaussian Mixture Models** (covered in ASR lecture)

Held-out data

- Let $T_\lambda = \{\mathcal{O}\}$ be the data used to **learn** λ, T_i for $P_i(\cdot)$
- If for most $\mathcal{O} \in T_\lambda,\ j. P_i(\mathcal{O}) \geq P_j(\mathcal{O})$, then $\lambda_i \to 1$
- This can easily occur when $T_i = T_\lambda$, e.g.:
    - If $P_i(\cdot)$ is an MLE *i*-gram model trained on T_λ, it will outperform $P_{<i}(\cdot)$ (even if also trained on T_λ)
    - If $P(\sigma; \theta_S)$ was trained on T_λ but not $P(\sigma; \theta_G)$
- Less likely to happen when $T_i \cap T_\lambda = \emptyset$
- A disjoint T_λ is often called **held-out** or **development data**

Aside – Maximum a Posteriori (MAP)

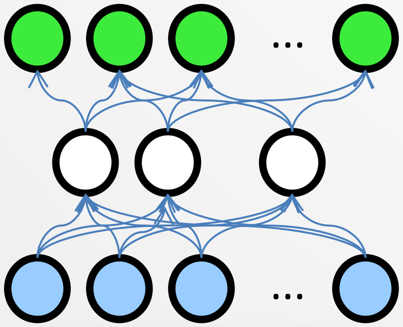
- Given adaptation data $\mathcal{O}_a$, the MAP estimate is
$$\hat{\theta} = \text{argmax}_{\theta} P(\mathcal{O}_a|\theta)P(\theta)$$

- If we can guess some structure for $P(\theta)$, we can use EM to estimate new parameters (or **Monte Carlo**).

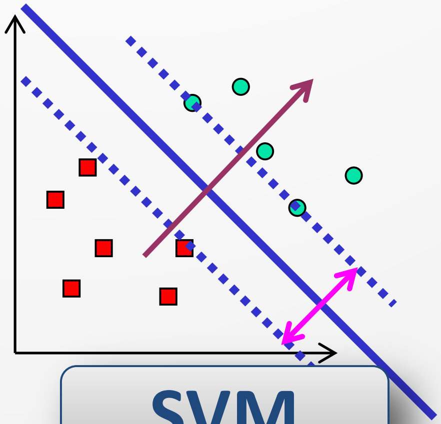
- For continuous $b_i(\sigma)$, we use **Dirichlet distribution** that defines the hyper-parameters of the model and the **Lagrange method** to describe the change in parameters $\theta \Longrightarrow \hat{\theta}$.

Generative vs. discriminative

- HMMs are **generative** classifiers. You can **generate** synthetic samples from because they model the phenomenon itself. (e.g. $P(\mathcal{O}, Q; \theta)$ or $P(\mathcal{O}; \theta)$)
- Other classifiers (e.g., artificial neural networks and support vector machines) are **discriminative** in that their probabilities are trained specifically to reduce the *error in classification*. (e.g. $P(Q|\mathcal{O}; \theta)$)



ANN



SVM

Summary

- Important ideas to know:
  - The definition of an HMM (e.g., its parameters).
  - The purpose of the **Forward algorithm**.
    - How to compute $\alpha_i(t)$ and $\beta_i(t)$
  - The purpose of the **Viterbi algorithm**.
    - How to compute $\delta_i(t)$ and $\psi_i(t)$.
  - The purpose of the **Baum-Welch algorithm**.
    - Some understanding of EM.
    - Some understanding of the equations.

Reading

- (optional) Manning & Schütze: Section 9.2—9.4.1
  - Note that they use another formulation…

- Rabiner, L. (1990) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: *Readings in speech recognition*. Morgan Kaufmann.
(posted on course website)

- Optional software:
  - Hidden Markov Model Toolkit (http://htk.eng.cam.ac.uk/)
  - Sci-kit's HMM (https://github.com/hmmlearn/hmmlearn)