Hidden Markov Models (HMMs)

이영기

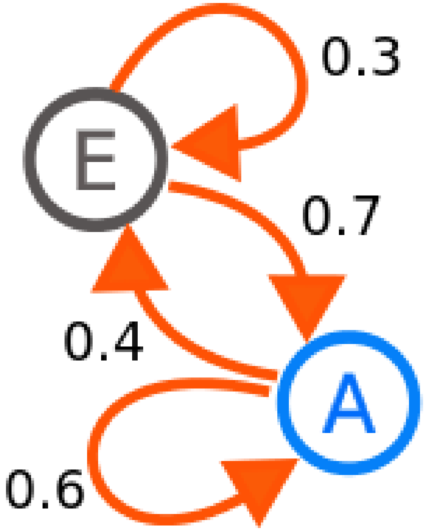
서울대학교 컴퓨터공학부

서울대학교
SEOUL NATIONAL UNIVERSITY

Daniel S. Weld
CSE 454

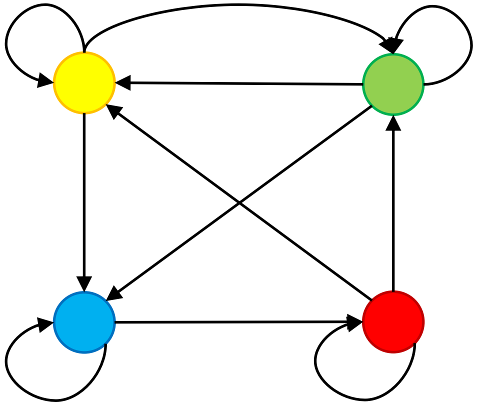
Markov Chain

- *Markov Chain* is a stochastic model describing a sequence of possible events.
- The probability of each event depends only on the state attained in the previous event.
- Markov Process is a Markov Chain for sequential time.



What’s an HMM?

- **Set of states**
  - Initial probabilities
  - Transition probabilities



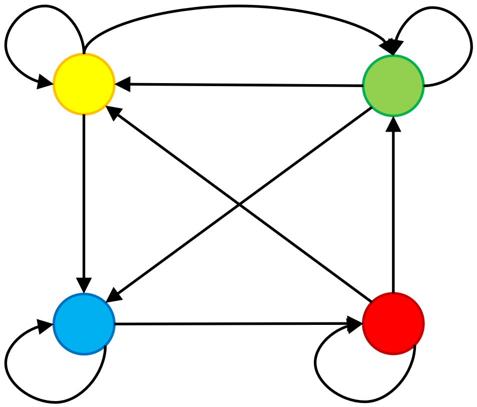
- **Set of potential observations**
  - Emission probabilities

x_1 x_2 x_3 x_4 x_5

HMM generates observation sequence

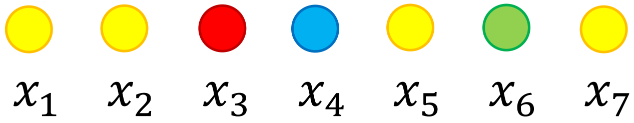
Hidden Markov Models (HMMs)

Finite state machine



Generates

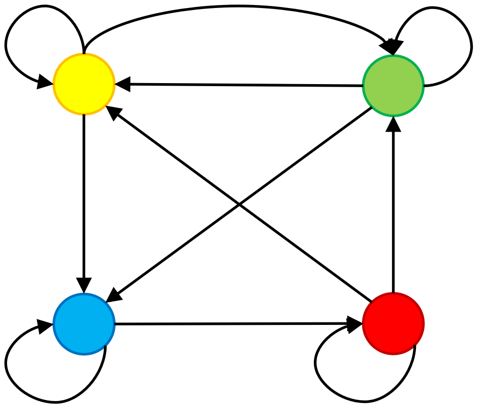
Hidden state sequence

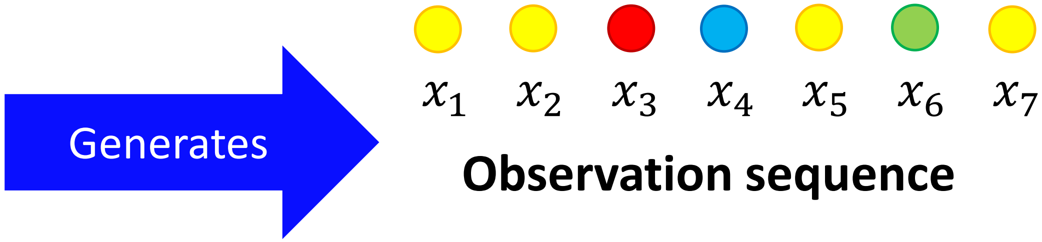


Observation sequence

Hidden Markov Models (HMMs)

Finite state machine





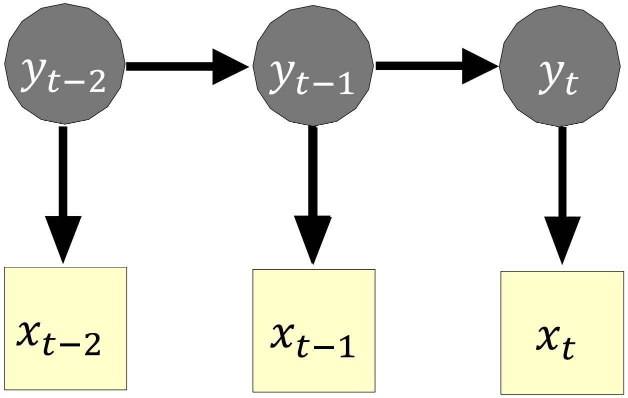
Hidden state sequence

Observation sequence

Graphical Model

Hidden states …

Observations …



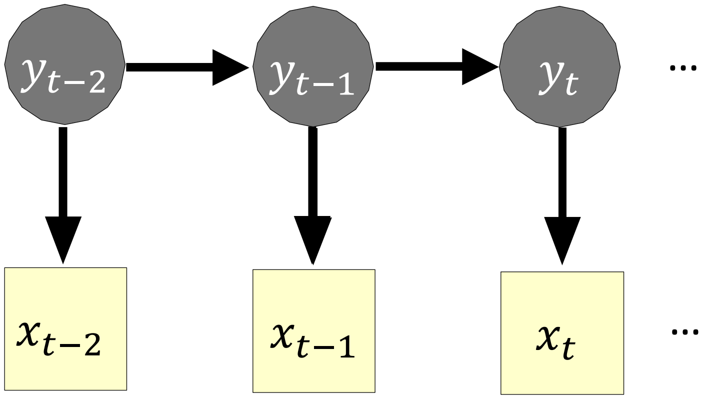
Random variable y_t takes values from $\{S_1, S_2, S_3, S_4\}$

Random variable x_t takes values from $\{x_1, x_2, x_3, x_4, x_5, \cdots\}$

HMM Graphical Model

Hidden states …

Observations …



Random variable y_t takes values from $\{S_1, S_2, S_3, S_4\}$

Random variable x_t takes values from $\{x_1, x_2, x_3, x_4, x_5, \cdots\}$

Need Parameters:

Start state probabilities: $P(y_1 = S_i)$

Transition probabilities: $P(y_t = S_j | y_{t-1} = S_i)$

Observation probabilities: $P\left(x_t = o_j \middle| y_t = S_i\right)$

Training:

Maximize probability of training observations

Example: The Dishonest Casino

A casino has two dice:

- Fair die

  P(1) = P(2) = P(3) = P(4) = P(5) = P(6) = 1/6

- Loaded die

  P(1) = P(2) = P(3) = P(4) = P(5) = 1/10

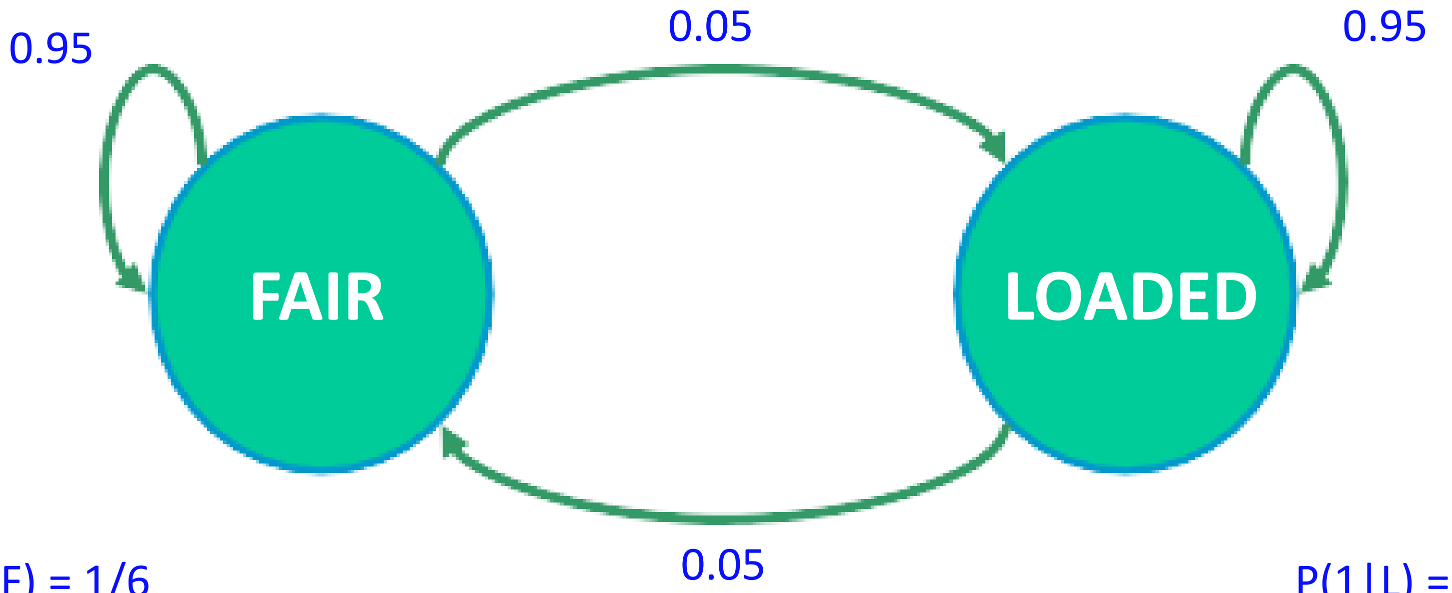
  P(6) = ½

Dealer switches back-&-forth between fair and loaded die about once every 20 turns

Game:

1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2

The Dishonest Casino HMM



P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

Question # 1 - Evaluation

GIVEN

A sequence of rolls by the casino player

1245526462146146136136661664661636616366163616515615115146123562344 …

QUESTION

How likely is this sequence, given our model of how the casino works?

This is the EVALUATION problem in HMMs

Question # 2 - Decoding

GIVEN

A sequence of rolls by the casino player

124552646214614613613666166466163661636616361 …

QUESTION

What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the DECODING problem in HMMs

Question # 3 - Learning

GIVEN

A sequence of rolls by the casino player

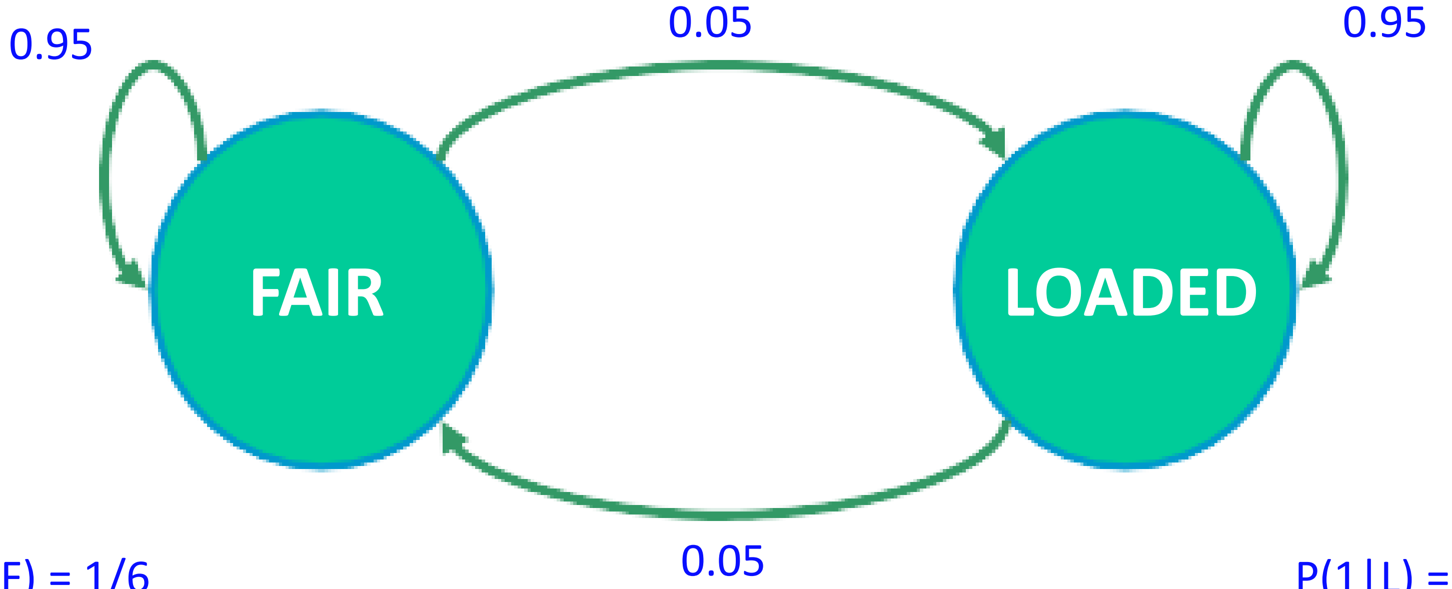
124552646214614613613666166466163661636616361 …

QUESTION

How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?

This is the LEARNING problem in HMMs

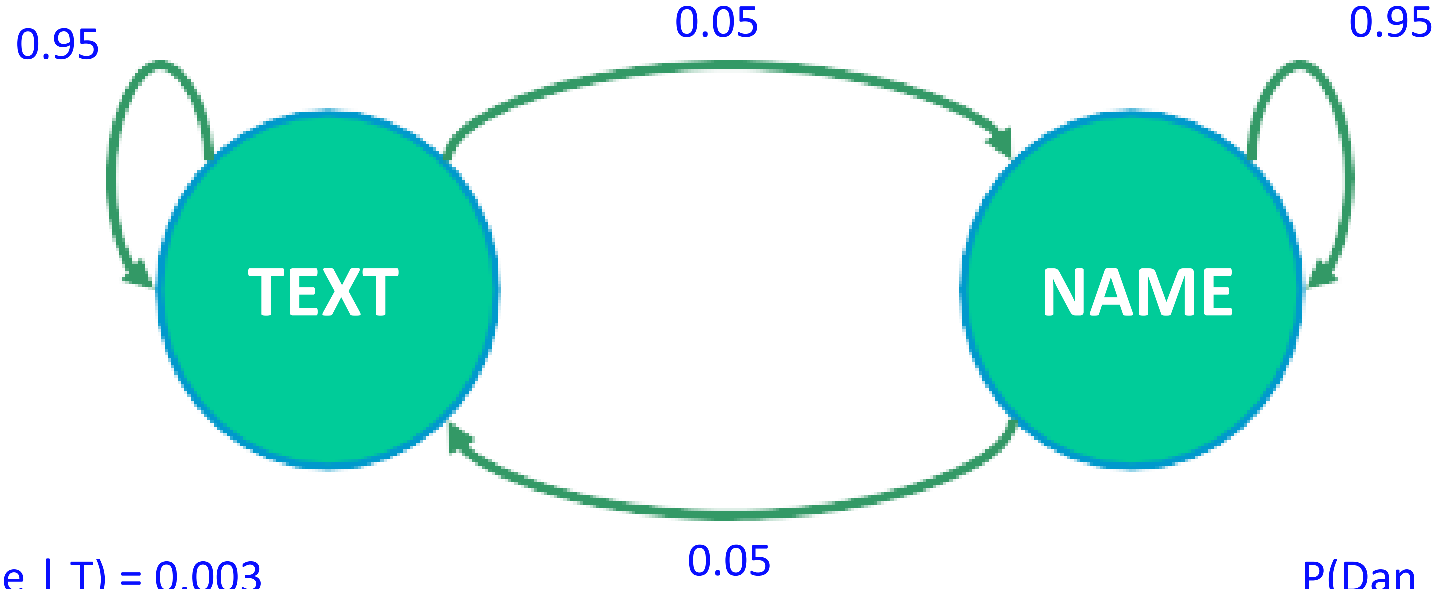
What’s This Have to Do with Info Extraction?



P(1|F) = 1/6
P(2|F) = 1/6
P(3|F) = 1/6
P(4|F) = 1/6
P(5|F) = 1/6
P(6|F) = 1/6

P(1|L) = 1/10
P(2|L) = 1/10
P(3|L) = 1/10
P(4|L) = 1/10
P(5|L) = 1/10
P(6|L) = 1/2

What’s This Have to Do with Info Extraction?



P(the | T) = 0.003
P(from | T) = 0.002
…

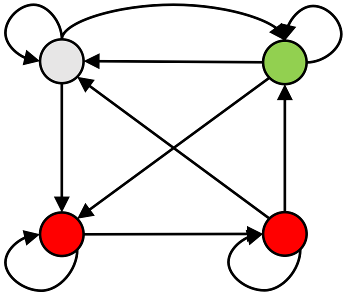
P(Dan | N) = 0.005
P(Sue | N) = 0.003
…

IE with Hidden Markov Models

Given a sequence of observations:

Yesterday Pedro Domingos spoke this example sentence.

And a trained HMM:

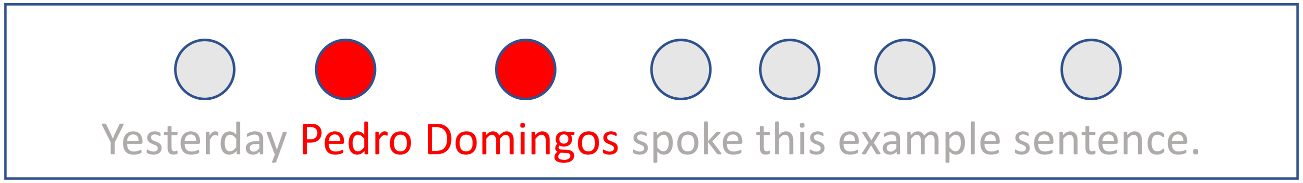


🟥 person name

🟩 location name

⬜ background

Find the most likely state sequence: (Viterbi) $\mathrm{argmax}_{\vec{s}} \mathrm{P}(\vec{s}, \vec{o})$



Yesterday Pedro Domingos spoke this example sentence.

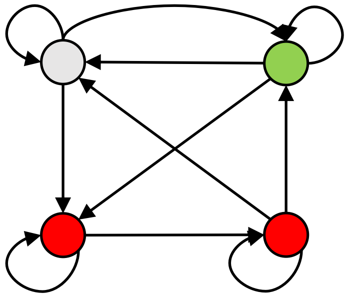
Any words said to be generated by the designated "person name“ state extract as a person name:

Person name: Pedro Domingos

IE with Hidden Markov Models

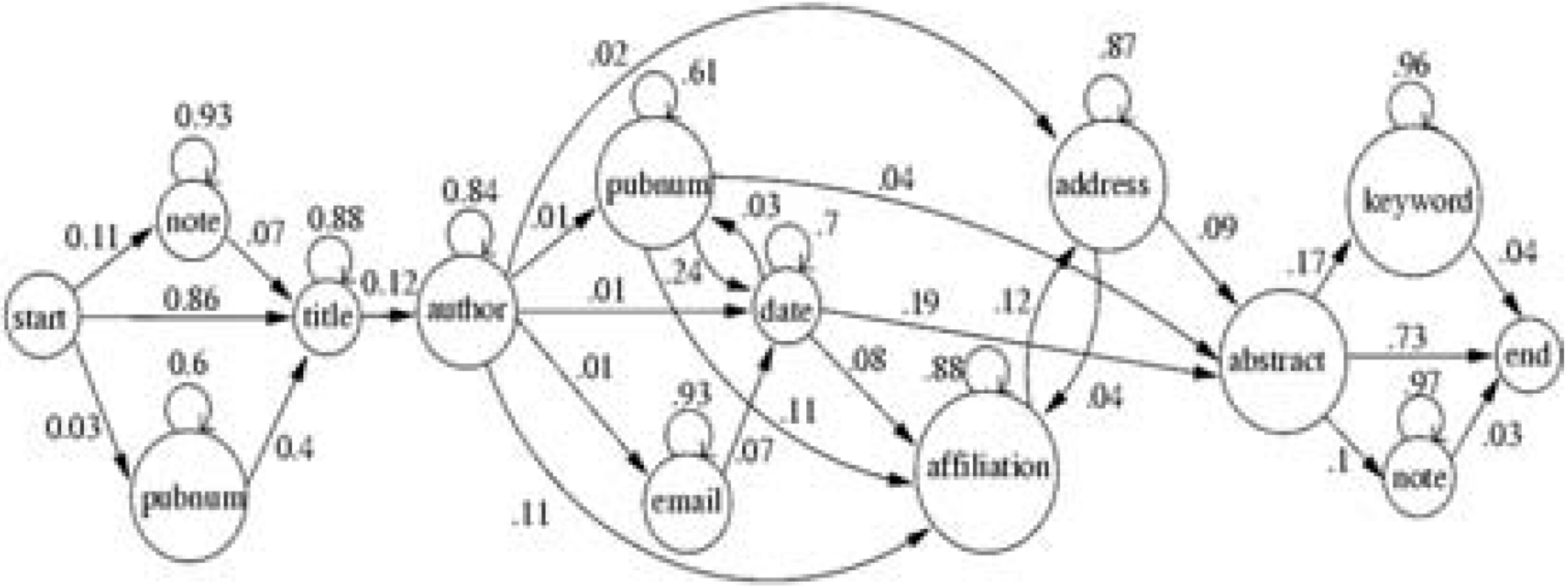
For sparse extraction tasks :

- Separate HMM for each type of target
- Each HMM should
  - Model entire document
  - Consist of ***target*** and ***non-target*** states
  - Not necessarily fully connected



Or … Combined HMM

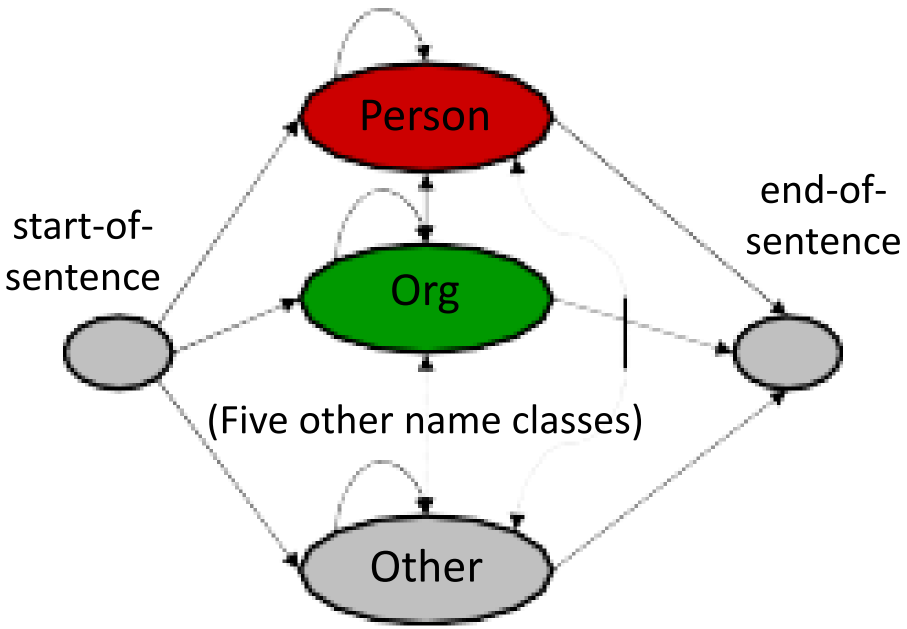
- Example – Research Paper Headers



HMM Example: “Nymble”

Task: Named Entity Extraction

[bikel, et al. 1998].
[BBN “IdentiFinder”]



Train on ~500k words of news wire text.

Results:

Case	Language	F1
Mixed	English	93%
Upper	English	91%
Mixed	Spanish	90%

Question # 1 - Evaluation

GIVEN

A sequence of observations $x = x_1 x_2 x_3 \ldots x_T$

A trained HMM

$$\theta = (p(y_t|y_{t-1}), p(x_t|y_t), p(y_1))$$

QUESTION

How likely is this sequence, given our HMM?

$$P(x|\theta)$$

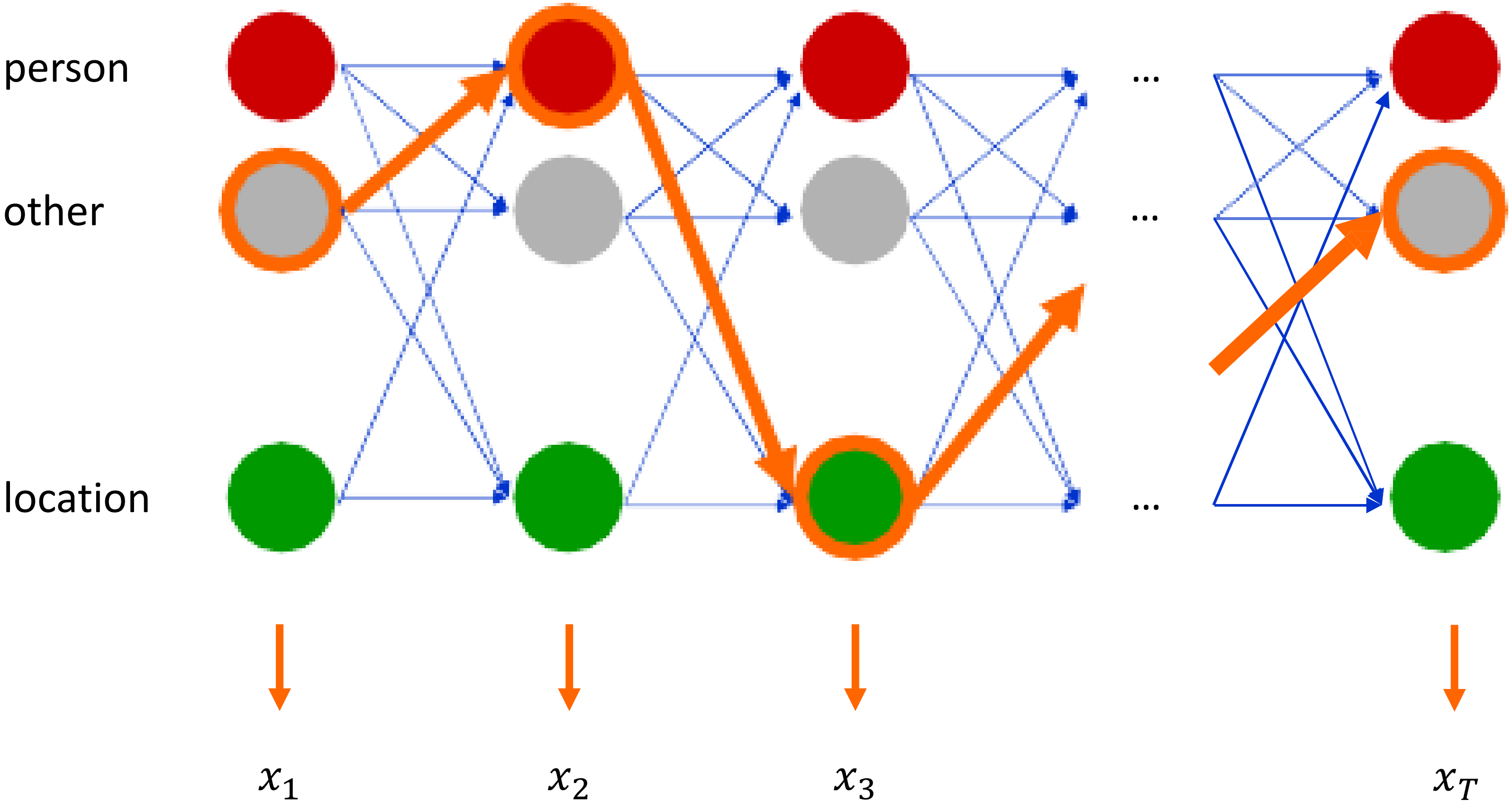
Why do we care?

Need it for learning to choose among competing models!

A Parse of a Sequence

Given a sequence $x = x_1 x_2 x_3 \dots x_T$

A parse of x is a sequence of states $y = y_1 \dots y_T$



Question # 2 - Decoding

GIVEN

A sequence of observations $x = x_1 x_2 x_3 \ldots x_T$

A trained HMM

$$\theta = (p(y_t | y_{t-1}), p(x_t | y_t), p(y_1))$$

QUESTION

How do we choose the corresponding parse (state sequence) $y_1 y_2 y_3 \ldots y_T$, which “best” explains $x_1 x_2 x_3 \ldots x_T$?

There are several reasonable optimality criteria: single optimal sequence, average statistics for individual states, …

Question # 3 - Learning

GIVEN

A sequence of observations $x = x_1 x_2 x_3 \dots x_T$

QUESTION

How do we learn the model parameters

$$\theta = (P(y_t|y_{t-1}), P(x_t|y_t), p(y_1))$$

Which maximize $P(x|\theta)$?

Three Questions

- Evaluation

  - Forward algorithm

- Decoding

  - Viterbi algorithm

- Learning

  - Baum-Welch Algorithm (aka “forward-backward”)

  - A kind of EM (expectation maximization)

Naive Solution to #1: Evaluation

Given observations $x = x_1 \dots x_T$ and HMM θ, what is $p(x|\theta)$?

Enumerate every possible state sequence $y = y_1 \dots y_T$

Probability of x and given particular y

$$p(x|y) = \prod_{t=1}^{T} p(x_t|y_t)$$

Probability of particular y

$$p(y) = \prod_{t=1}^{T} p(y_t|y_{t-1})$$

2T multiplications per sequence

Summing over all possible state sequences we get

$$p(x) = \sum_{all\ y} p(x|y)p(y)$$

N^T state sequences!

For small HMMs
T=10, N=10
There are 10
Billion sequences!

Many Calculations Repeated

- Use Dynamic Programming

$$
\begin{aligned}
p(x) &= \sum_{y} p(y)p(x|y) \\
&= \sum_{y} \prod_{t=1\ldots T} p(y_t|y_{t-1})p(x_t|y_t) \\
&\boxed{= \sum_{y_T} \sum_{y_{T-1}} p(y_T, y_{T-1}, x_T) \boxed{\sum_{y_{T-2}} p(y_{T-1}|y_{T-2})p(x_{T-1}|y_{T-1}) \boxed{\sum_{y_{T-3}} \ldots}}}
\end{aligned}
$$

- Cache and reuse inner sums

“forward variables”

Solution to #1: Evaluation

Use Dynamic Programming

Define forward variable

$$\alpha_t(i) = P(x_1 x_2 \ldots x_t, y_t = S_i)$$

Probability that at time t

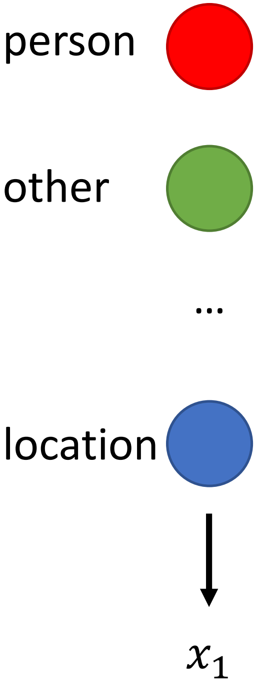
- the state is S_i
- the partial observation sequence $x = x_1 \ldots x_T$ has been emitted

Base case: Forward Variable $\alpha_t(i)$

$$\alpha_t(i) = P(x_1 x_2 \ldots x_t, y_t = S_i)$$

Prob - that the state at t has value S_i and

- the partial observation sequence $x = x_1 \ldots x_t$ has been seen



Base Case: t=1

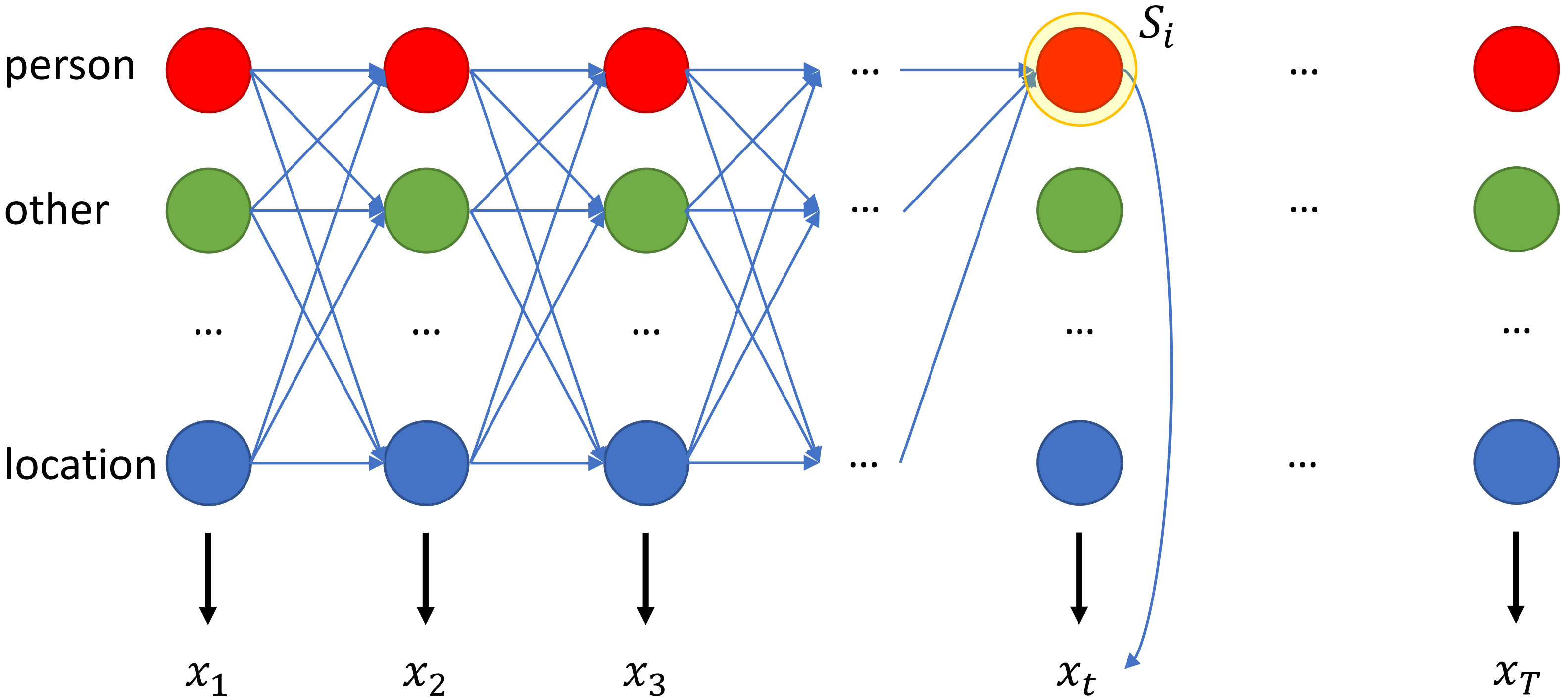
$$\boldsymbol{\alpha_1(i) = p(y_1 = S_i)p(x_1 = o_1)}$$

Inductive Case: Forward Variable $\alpha_t(i)$

$$\alpha_t(i) = P(x_1 x_2 \dots x_t, y_t = S_i)$$

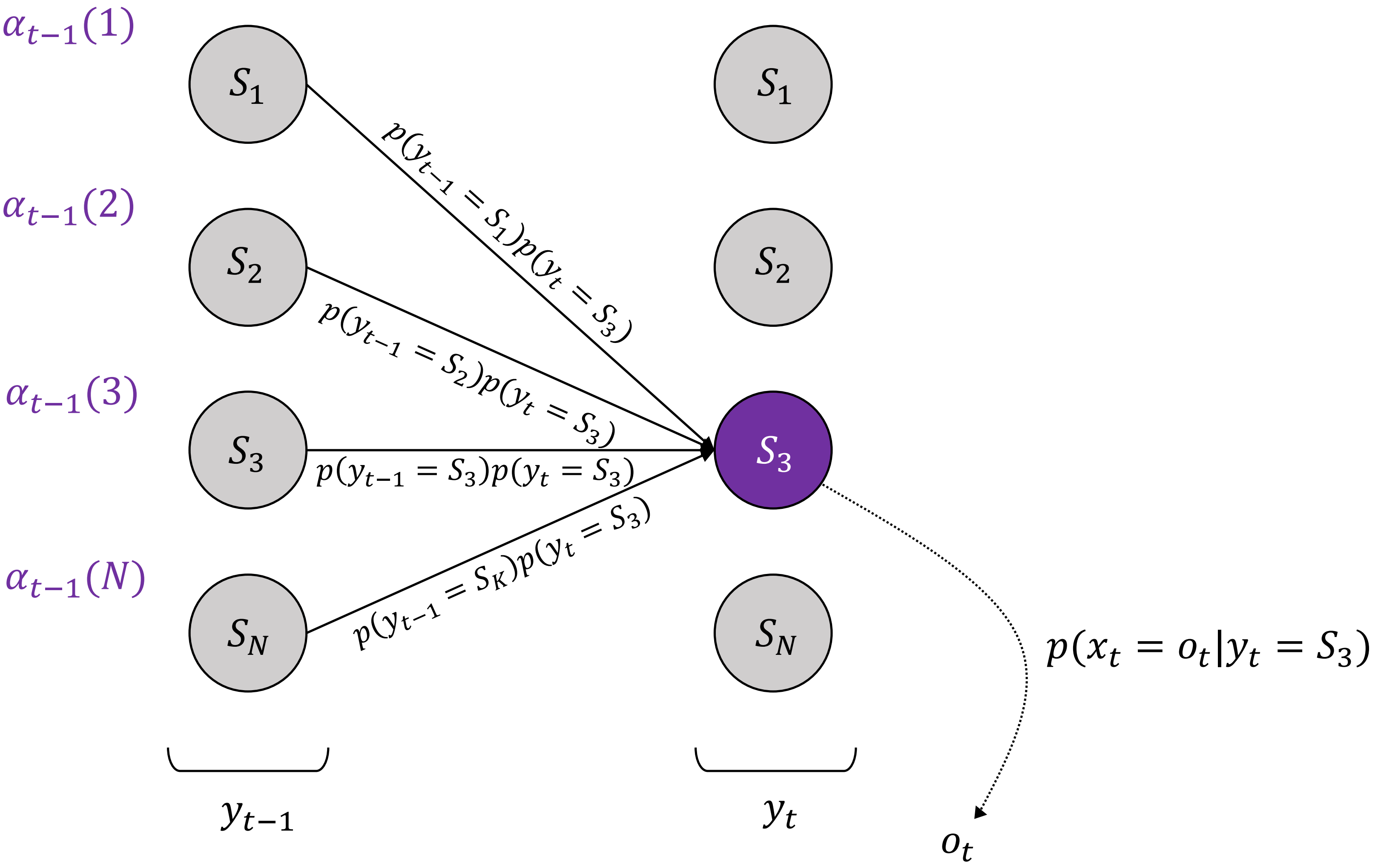
Prob - that the state at t has value S_i and

- the partial observation sequence $x = x_1 \dots x_t$ has been seen



The Forward Algorithm

$$\alpha_t(i) = P(x_1 x_2 \ldots x_t, y_t = S_i)$$


$\alpha_{t-1}(1)$
$\alpha_{t-1}(2)$
$\alpha_{t-1}(3)$
$\alpha_{t-1}(N)$
S_1
S_2
S_3
S_N
S_1
S_2
S_3
S_N
$p(y_{t-1} = S_1)p(y_t = S_3)$
$p(y_{t-1} = S_2)p(y_t = S_3)$
$p(y_{t-1} = S_3)p(y_t = S_3)$
$p(y_{t-1} = S_K)p(y_t = S_3)$
$p(x_t = o_t | y_t = S_3)$
y_{t-1}
y_t
o_t

The Forward Algorithm

$$\boxed{\alpha_t(i) = P(x_1 x_2 \dots x_t, y_t = S_i)}$$

INITIALIZATION

$$\alpha_1(i) = p(y_1 = S_i)p(x_1|y_1)$$

INDUCTION

$$\alpha_t(i) = p(x_1 x_2 \dots x_t, y_t = S_i)$$
$$= \sum_{j \in S} \alpha_{t-1}(j)p\left(y_t = S_i \middle| y_{t-1} = S_j\right)p(x_t|y_t)$$

TERMINATION

$$p(x) = \sum_{j \in S} \alpha_T(j)$$

Time:

$O(K^2 N)$

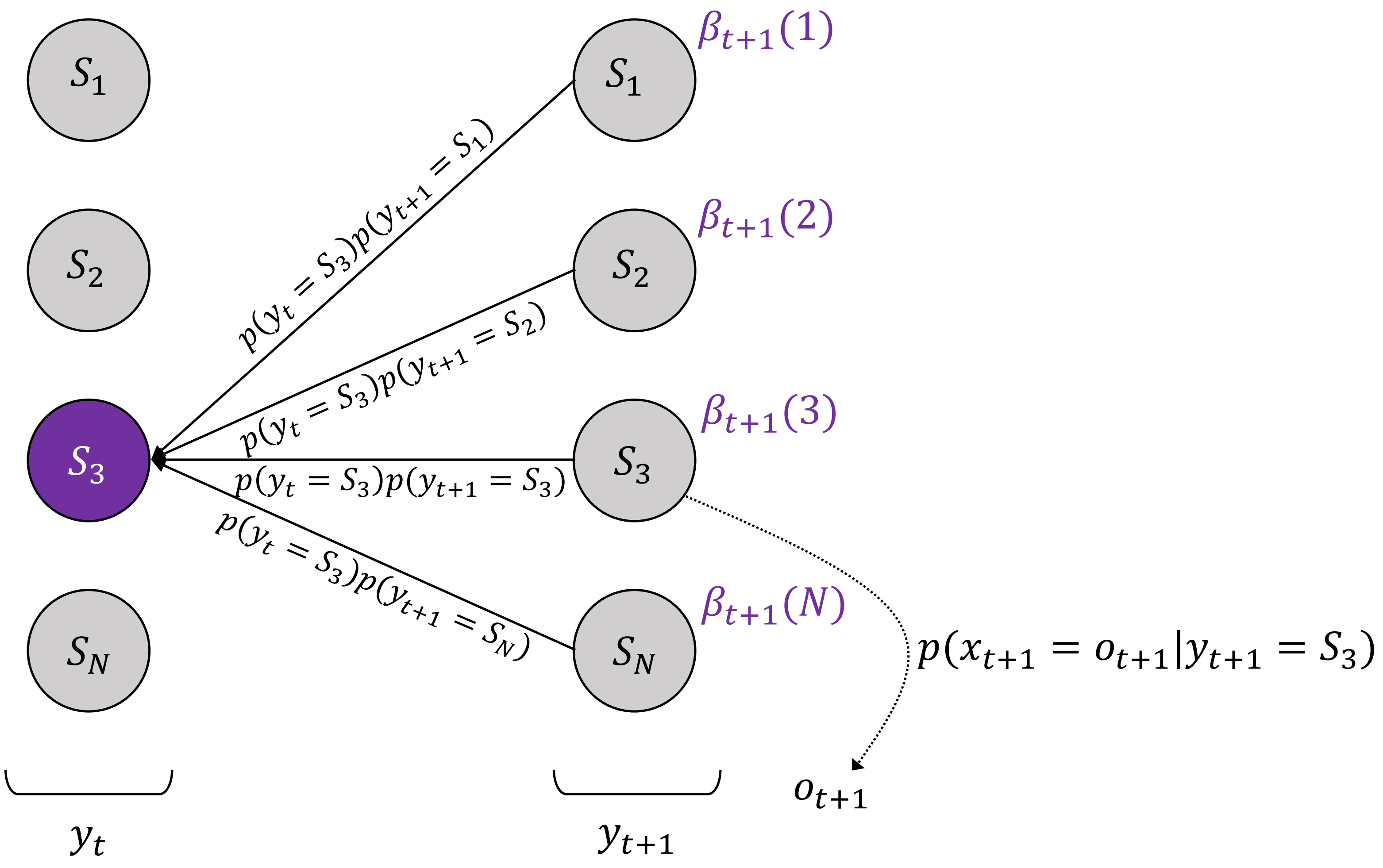
Space:

$O(KN)$

$K = \lvert S\rvert$	#states
N	length of sequence

The Backward Algorithm

$$\beta_t(i) = P(y_t = \mathrm{S}_i, x_{t+1}x_{t+2} \ldots x_T)$$



$\beta_{t+1}(1)$
$\beta_{t+1}(2)$
$\beta_{t+1}(3)$
$\beta_{t+1}(N)$
S_1
S_2
S_3
S_N
S_1
S_2
S_3
S_N
$p(y_t = S_3)p(y_{t+1} = S_1)$
$p(y_t = S_3)p(y_{t+1} = S_2)$
$p(y_t = S_3)p(y_{t+1} = S_3)$
$p(y_t = S_3)p(y_{t+1} = S_N)$
$p(x_{t+1} = o_{t+1} | y_{t+1} = S_3)$
o_{t+1}
y_t
y_{t+1}

The Backward Algorithm

$$\boxed{\beta_t(i) = P(y_t = \mathrm{S}_i, x_{t+1}x_{t+2} \dots x_T)}$$

INITIALIZATION

$$\beta_T(i) = 1$$

INDUCTION

$$\beta_t(i) = p(y_t = S_i, x_{t+1}x_{t+2} \dots x_T)$$
$$= \sum_{j \in S} p(y_{t+1} = S_j | y_t = S_i) p(x_{t+1} | y_{t+1}) \beta_{t+1}(j)$$

TERMINATION

$$p(x) = \sum_{j \in S} p(y_1 = S_j) p(x_1 | y_1) \beta_1(j)$$

Time:

$O(K^2 N)$

Space:

$O(KN)$

Three Questions

- Evaluation

  - Forward algorithm

- Decoding

  - Viterbi algorithm

- Learning

  - Baum-Welch Algorithm (aka “forward-backward”)

  - A kind of EM (expectation maximization)

#2 – Decoding Problem

Given $x = x_1 \ldots x_T$ and HMM θ, what is "best" parse $y_1 \ldots y_T$?

Several possible meanings

1. States which are individually most likely:

$$P(y_t = S_i|x) = \frac{\alpha_t(i)\beta_t(i)}{P(x)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{K} \alpha_t(i)\beta_t(i)}$$

most likely state y_t^* at time t is then

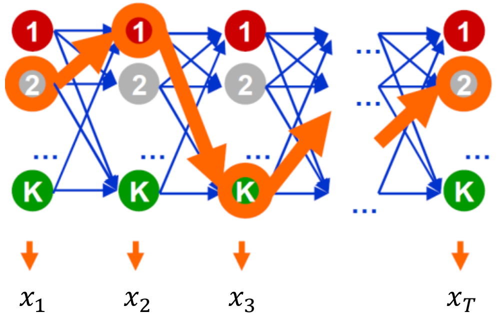
$$y_t^* = argmax_{1 \le i \le K} P(y_t = S_i|x)$$

#2 – Decoding Problem

Given $x = x_1 \dots x_T$ and HMM θ, what is "best" parse $y_1 \dots y_T$?

Several possible meanings of 'solution'

1. States which are individually most likely:
2. Single best state sequence



We want ***sequence*** $y_1 \dots y_T$,
such that $P(y|x)$ is maximized

$$y^* = argmax_y P(y|x)$$

$\boldsymbol{\delta_t(i)}$

- Like

  $\alpha_t(i)$ = prob that the state, y, at time t has value S_i

  and the partial observation sequence $x = x_1 \dots x_t$ has been seen

- Define

  $\delta_t(i)$ = probability of ***most likely*** state sequence

  ending with state S_i, given observations $x_1 \dots x_t$

$$\delta_t(i) = \max_{y_1, y_2, \dots y_{t-1}} P(y_1, \dots, y_{t-1}, y_t = S_i | x_1, \dots, x_t, \theta)$$

$\delta_t(i)$ = **probability of** most likely

state sequence ending with state S_i,
given observations $x_1, \ldots, x_t$

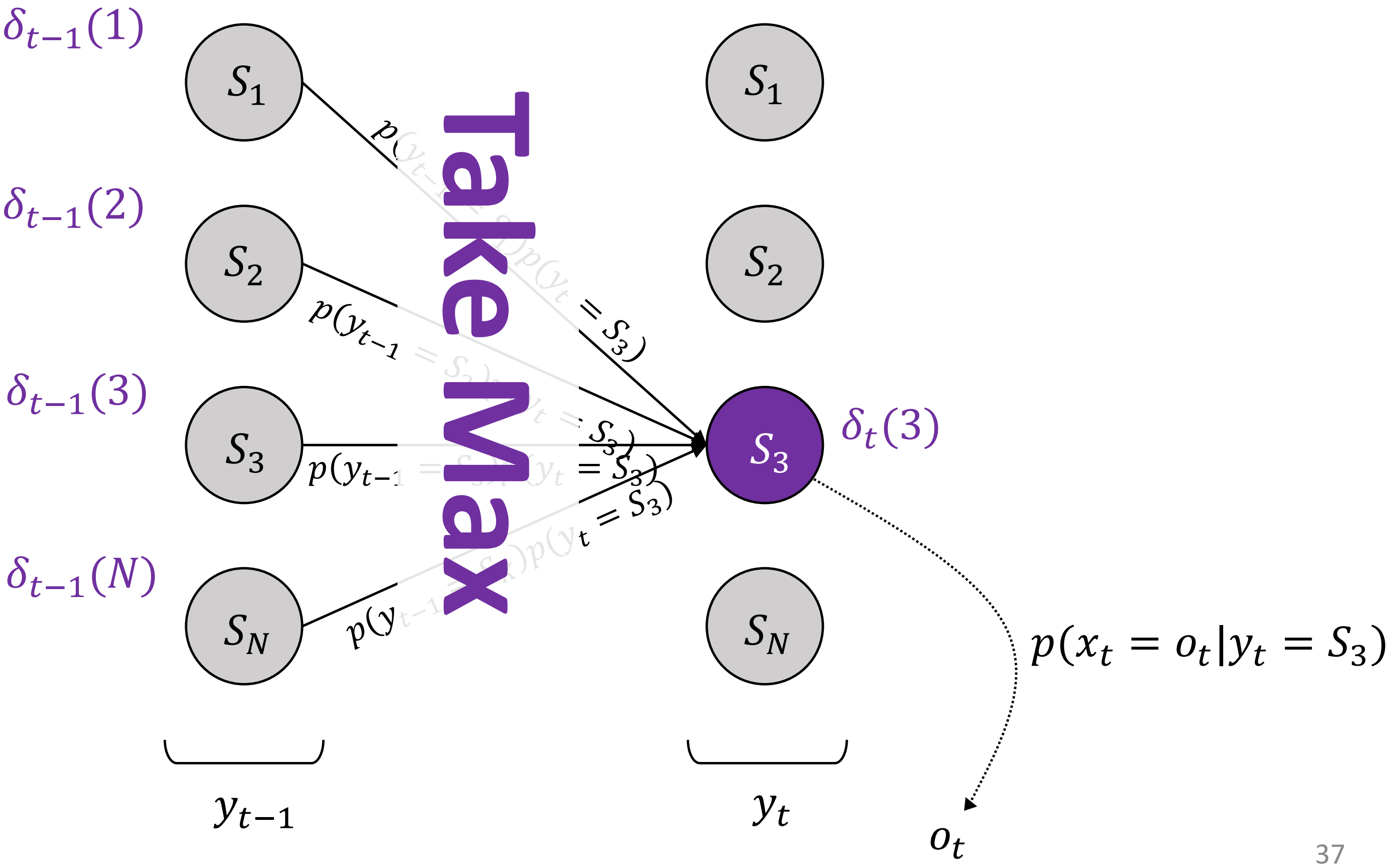
$$\delta_t(i) = \max_{y_1, y_2, \ldots y_{t-1}} P(y_1, \ldots, y_{t-1}, y_t = S_i | x_1, \ldots, x_t, \theta)$$

Base Case: t=1

$$Max_i \; P(y_1 = S_i) P(x_1 = o_1 | y_1 = S_i)$$

Inductive Step

$$\delta_t(i) = \max_{y_1, y_2, \ldots y_{t-1}} P(y_1, \ldots, y_{t-1}, y_t = S_i | x_1, \ldots, x_t, \theta)$$



The Viterbi Algorithm

DEFINE

$$\delta_t(i) = \max_{y_1, y_2, \ldots, y_{t-1}} P(y_1, y_2, \ldots, y_{t-1}, y_t = i | x_1, \ldots, x_t, \theta)$$

INITIALIZATION

$$\delta_1(i) = p(y_1 = S_i) p(x_1 | y_1 = S_i)$$

INDUCTION

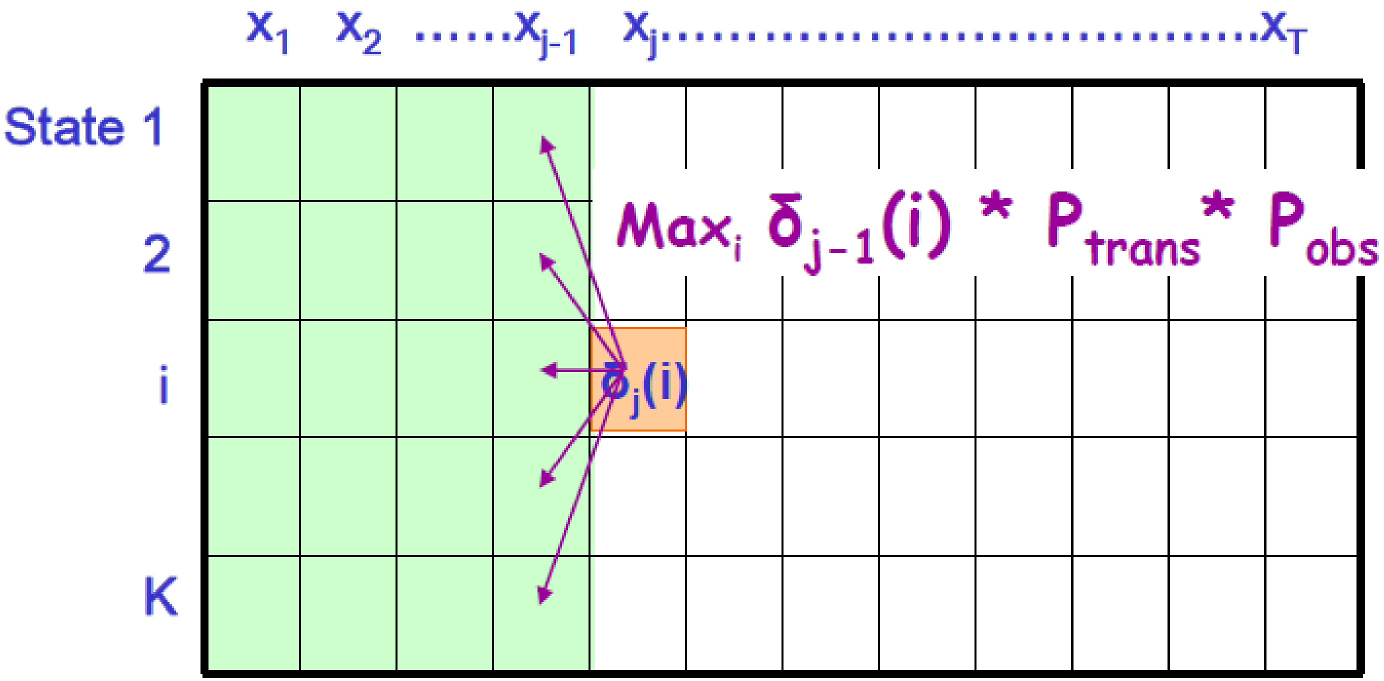
$$\delta_t(j) = \max_{i \in S} \delta_{t-1}(i) p\big(y_t = S_j \big| y_{t-1} = S_i\big) p(x_i | y_t = S_j)$$

TERMINATION

$$p^* = \max_{i \in S} \delta_T(i)$$

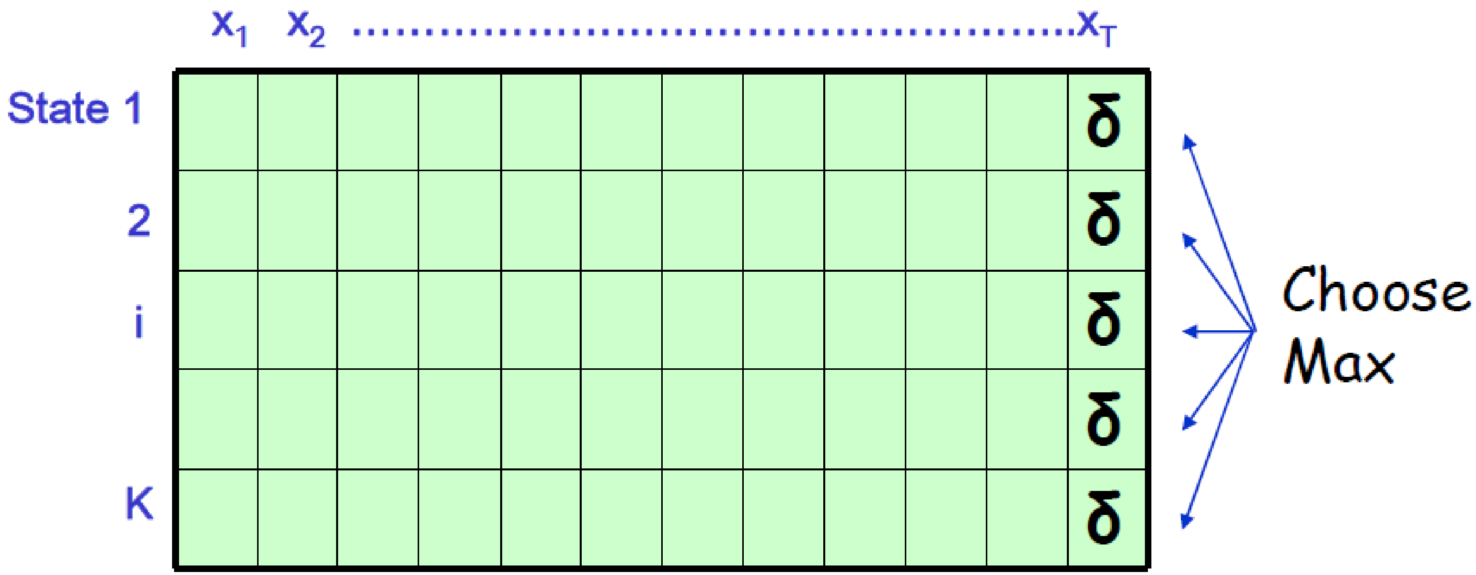
Backtracking to get state sequence y^*

The Viterbi Algorithm


x_1 x_2 ……x_{j-1} x_j……………………………x_T
State 1
2
i
K
Max$_i$ $\delta_{j-1}(i)$ * P_{trans} * P_{obs}
$\delta_j(i)$

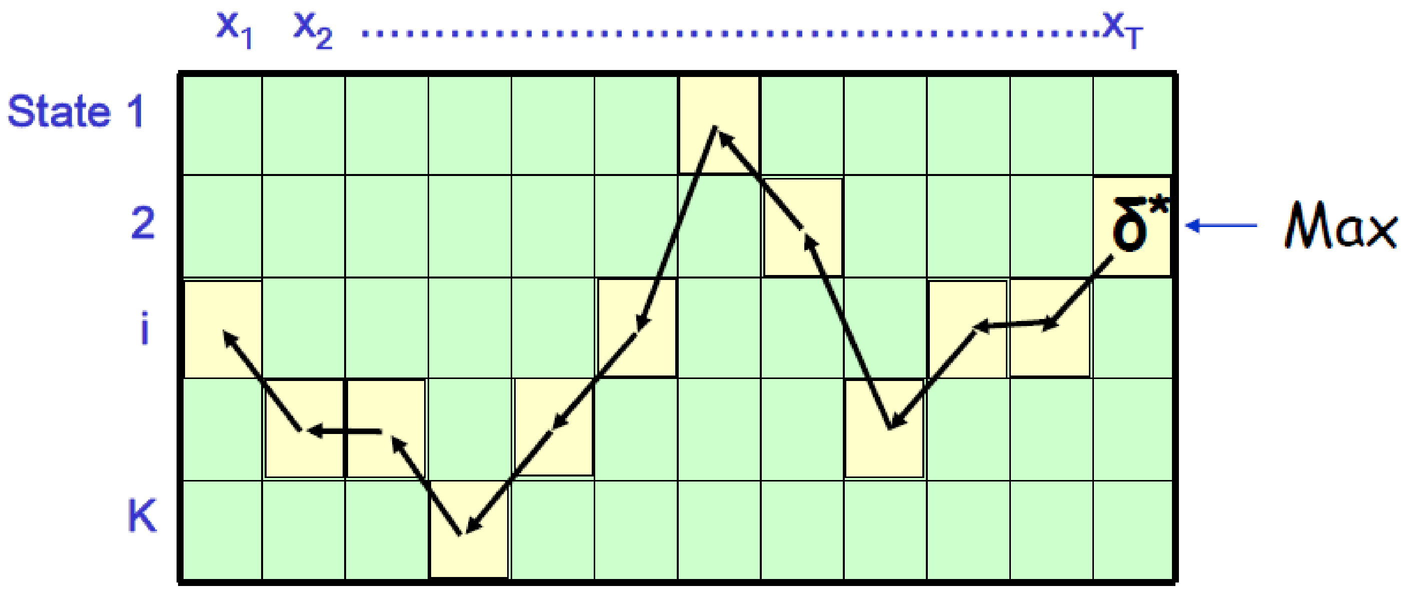
Remember: $\delta_t(i)$ = probability of most likely state seq ending with $y_t = state\ S_t$

Terminating Viterbi



Choose Max

Terminating Viterbi



How did we compute δ^*? $\max_i \delta_{T-1}(i) * P_{trans} * P_{obs}$

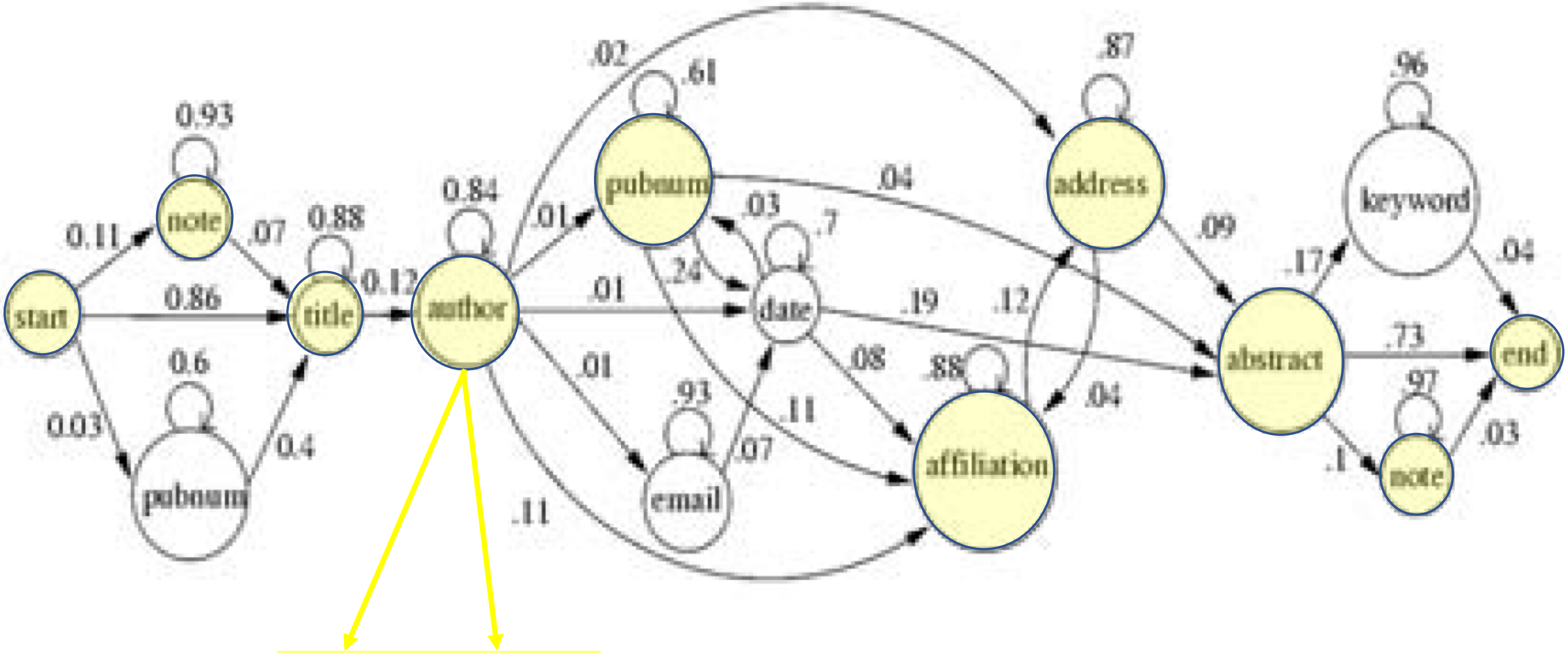
Now Backchain to Find Final Sequence

Time: $O(K^2T)$

Space: $O(KT)$

Linear in length of sequence

The Viterbi Algorithm



Pedro Domingos

Three Questions

- Evaluation

  - Forward algorithm

- Decoding

  - Viterbi algorithm

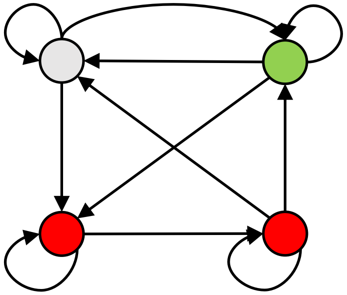
- Learning

  - Baum-Welch Algorithm (aka “forward-backward”)

  - A kind of EM (expectation maximization)

Solution to #3 - Learning

- If we have labeled training data!
- Input:

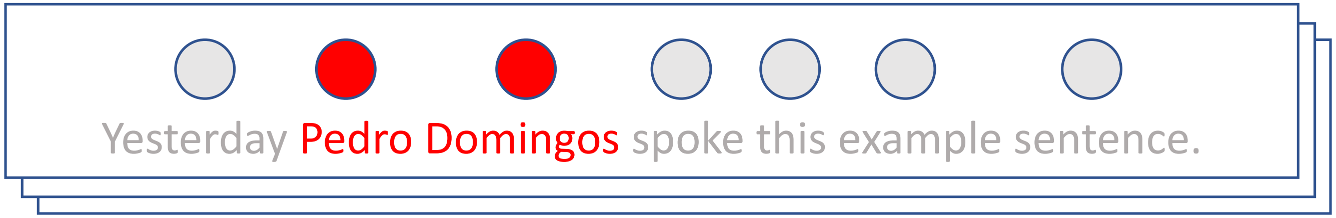


🟥 person name

🟩 location name

⬜ background

states & edges, but no probabilities



Yesterday Pedro Domingos spoke this example sentence.

Many labeled sentences

- Output:

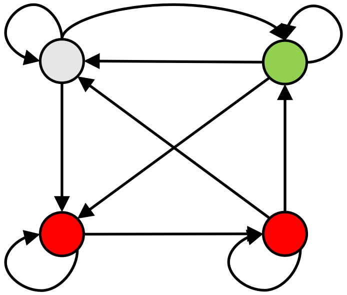
  - Initial state & transition probabilities:

$$p(y_1), p(y_t|y_{t-1})$$

  - Emission probabilities: $p(x_t|y_t)$

Supervised Learning

- Input:



🟥 person name

🟩 location name

⬜ background

states & edges, but no probabilities

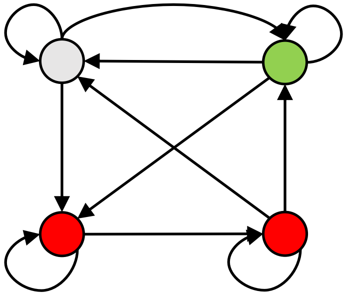
Yesterday Pedro Domingos spoke this example sentence.
Daniel Weld gave his talk in Mueller 153.
Sieg 128 is a nasty lecture hall, don’t you think?
The next distinguished lecture is by Oren Etzioni on Thursday.

- Output:

  - Initial state probabilities: $p(y_1)$
    - $P(y_1 = name) = 1/4$
    - $P(y_1 = location) = 1/4$
    - $P(y_1 = background) = 2/4$

Supervised Learning

- Input:



🟥 person name
🟩 location name
⬜ background

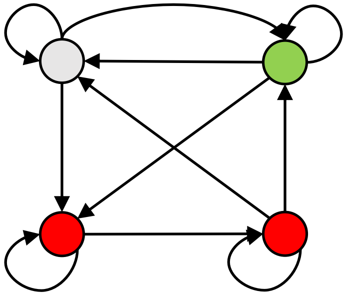
states & edges, but no probabilities

Yesterday Pedro Domingos spoke this example sentence.
Daniel Weld gave his talk in Mueller 153.
Sieg 128 is a nasty lecture hall, don't you think?
The next distinguished lecture is by Oren Etzioni on Thursday.

- Output:
  - State transition probabilities: $p(y_t|y_{t-1})$
    - $P(y_t = name|y_{t-1} = name) = 3/6$
    - $P(y_t = name|y_{t-1} = background) = 2/22$
    - Etc…

Supervised Learning

- Input:

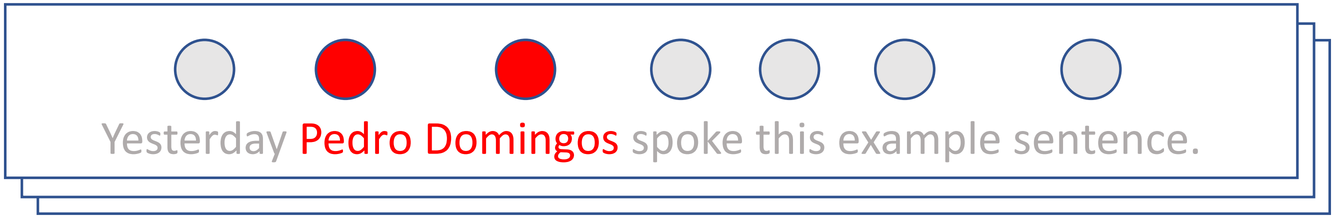


🟥 person name

🟩 location name

⬜ background

states & edges, but no probabilities



Yesterday Pedro Domingos spoke this example sentence.

Many labeled sentences

- Output:

  - State transition probabilities: $\mathrm{p}(\mathrm{y}_1), p(y_t|y_{t-1})$

  - Emission probabilities: $p(x_t|y_t)$

Solution to #3 - Learning

Given $x_1 \ldots x_T$, how do we learn $\theta = (p(y_t|y_{t-1}), p(x_t|y_t), p(y_1))$ to maximize $P(x|\theta)$?

- Unfortunately, there is no known way to analytically find a global maximum θ^* such that

$$\theta^* = argmax\, P(x|\theta)$$

- But it is possible to find a local maximum; given an initial model θ, we can always find a model θ' such that

$$P(x|\theta') \geq P(x|\theta)$$

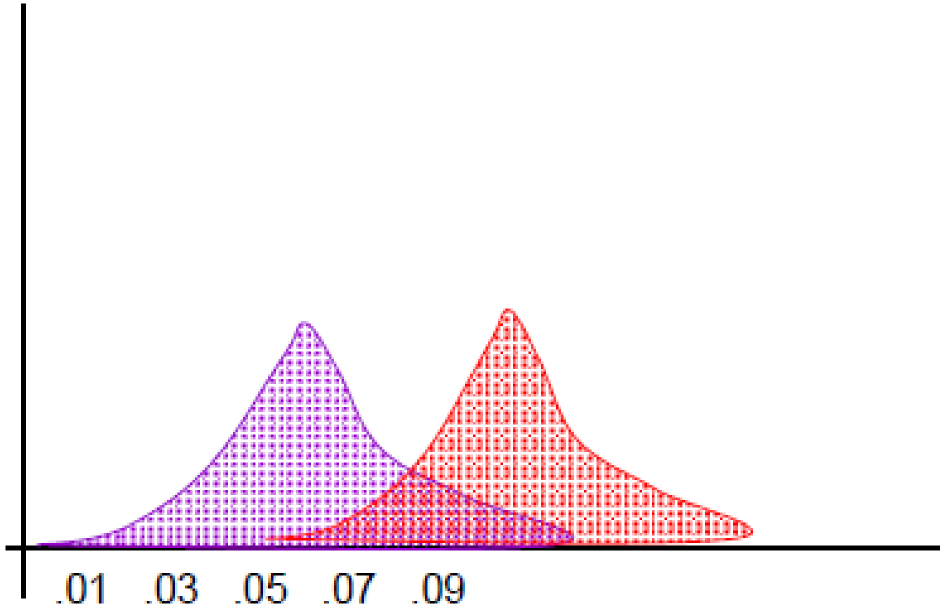
Chicken & Egg Problem

- If we knew the actual sequence of states
  - It would be easy to learn transition and emission probabilities
  - But we can’t observe states, so we don’t!

- If we knew transition & emission probabilities
  - Then it’d be easy to estimate the sequence of states (Viterbi)
  - But we don’t know them!

Simplest Version

- Mixture of two distributions

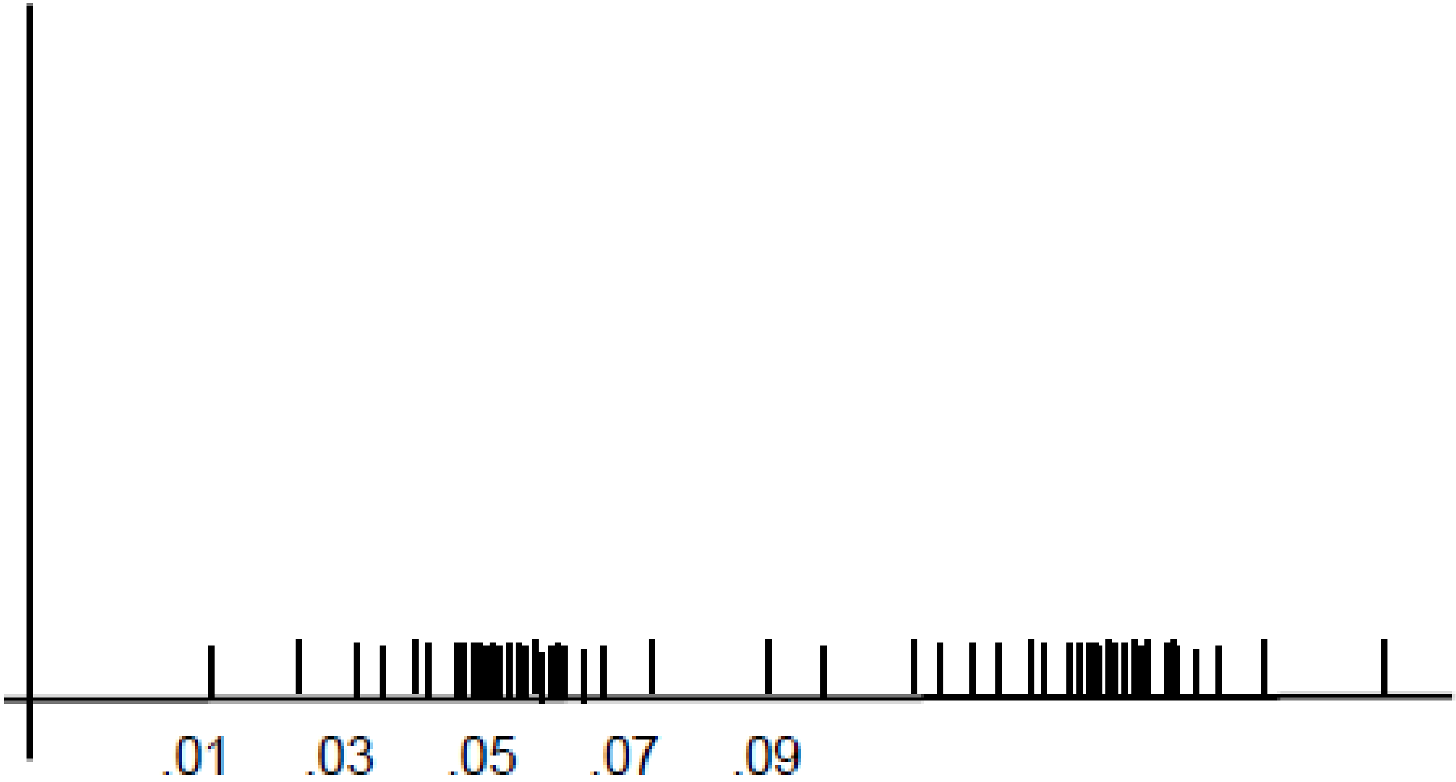


- Know: form of distribution & variance,

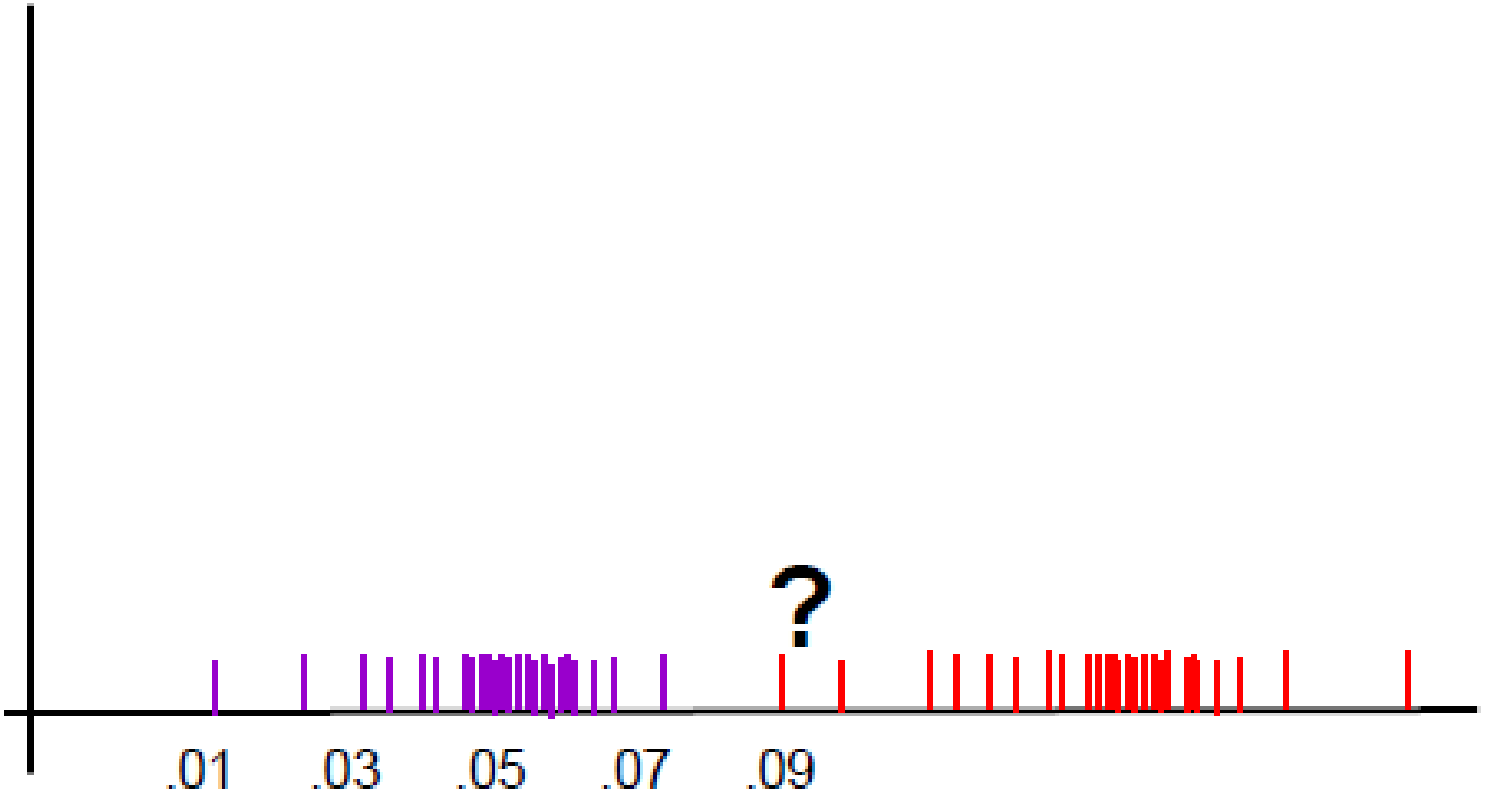
  % =5

- Just need *mean* of each distribution

Input Looks Like

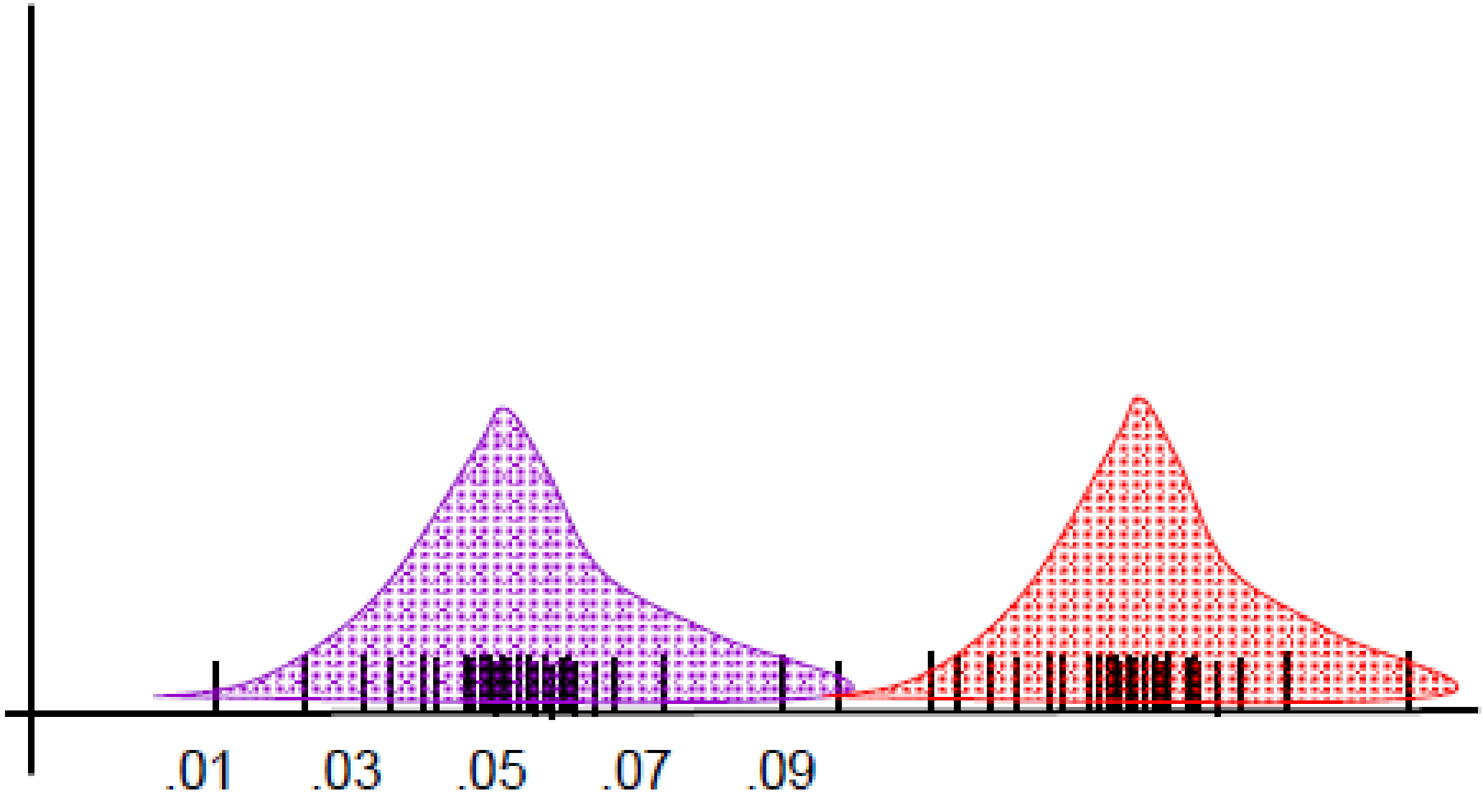


We Want to Predict



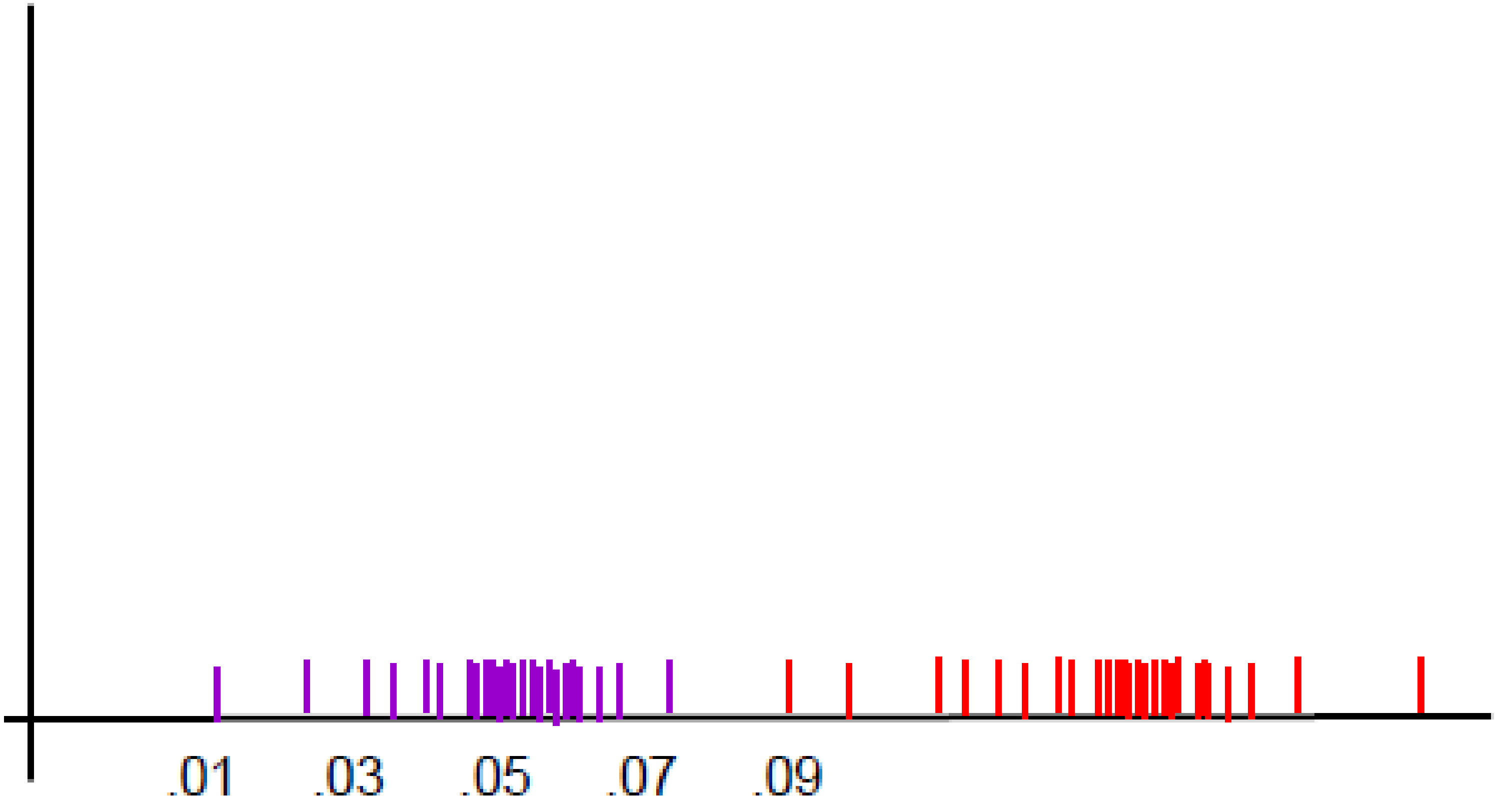
Chicken & Egg

Note that coloring instances would be easy *If* we knew Gaussians…



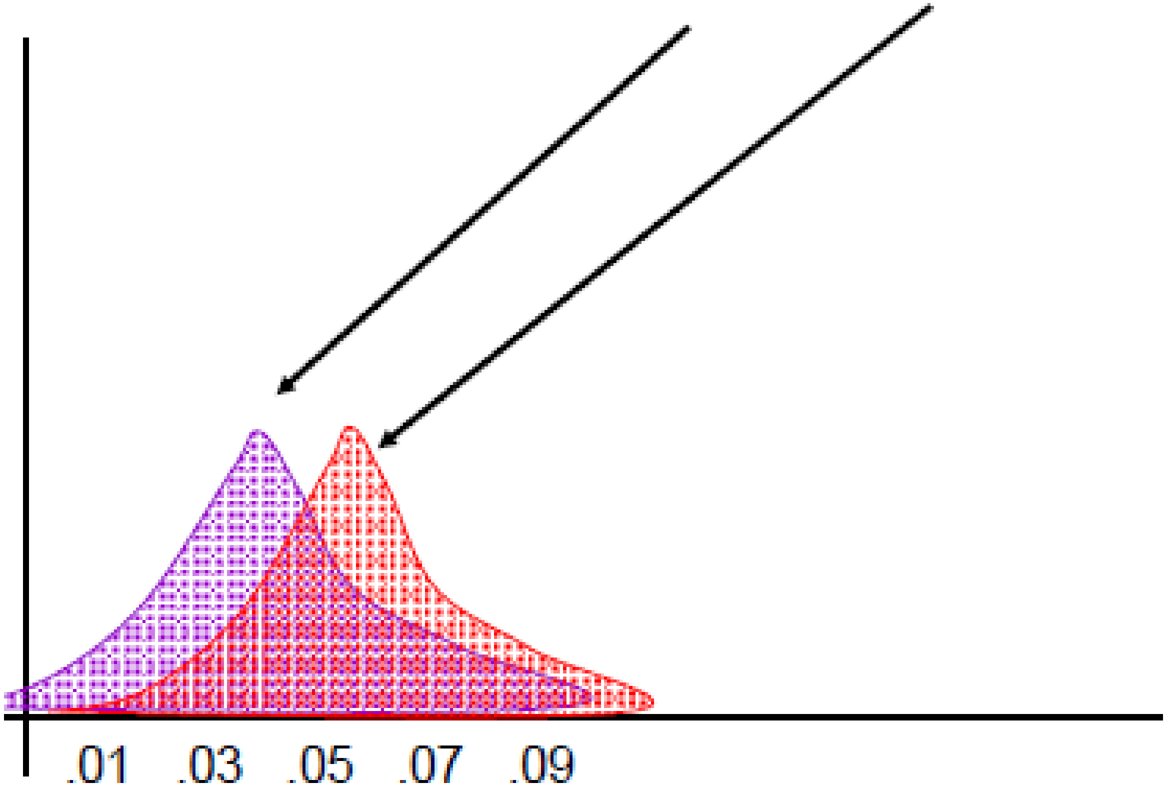
Chicken & Egg

And finding the Gaussians would be easy
If we knew the coloring



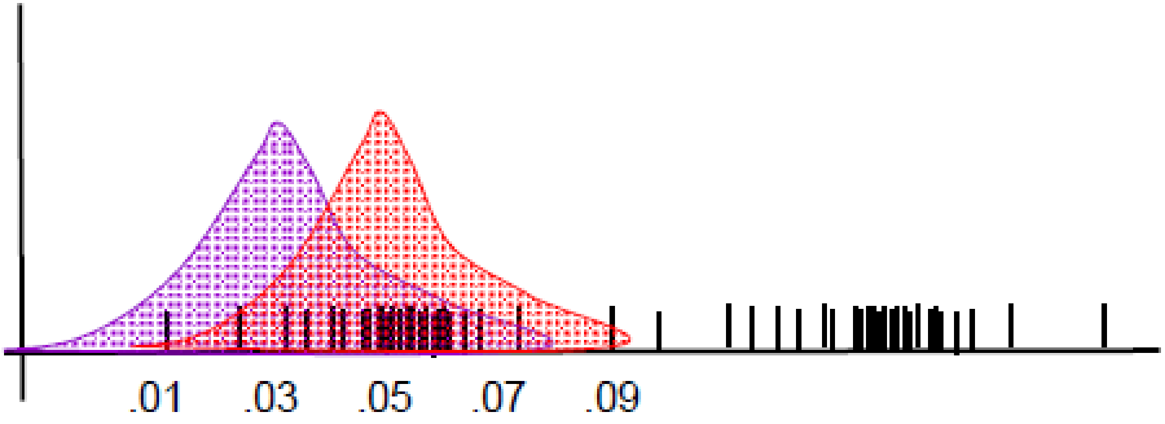
Expectation Maximization (EM)

- Pretend we *do* know the parameters
  - Initialize randomly: set $\theta_1 = ?$; $\theta_2 = ?$



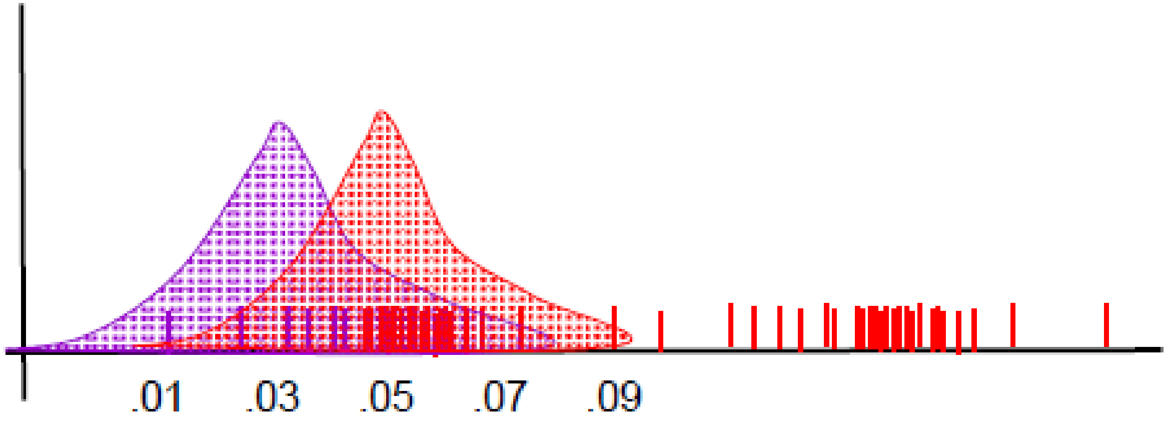
Expectation Maximization (EM)

- Pretend we *do* know the parameters
  - Initialize randomly
- [E step] Compute probability of instance having each possible value of the hidden variable



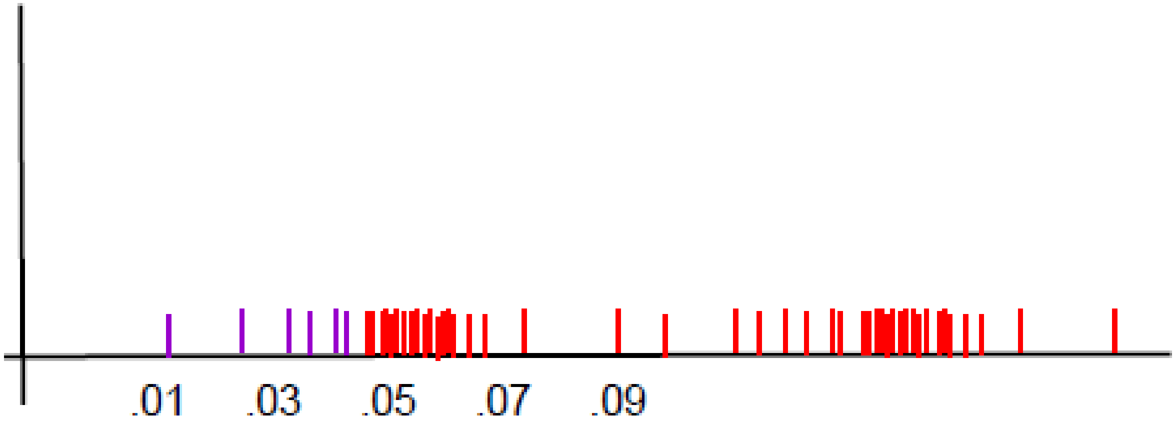
Expectation Maximization (EM)

- Pretend we *do* know the parameters
  - Initialize randomly
- [E step] Compute probability of instance having each possible value of the hidden variable



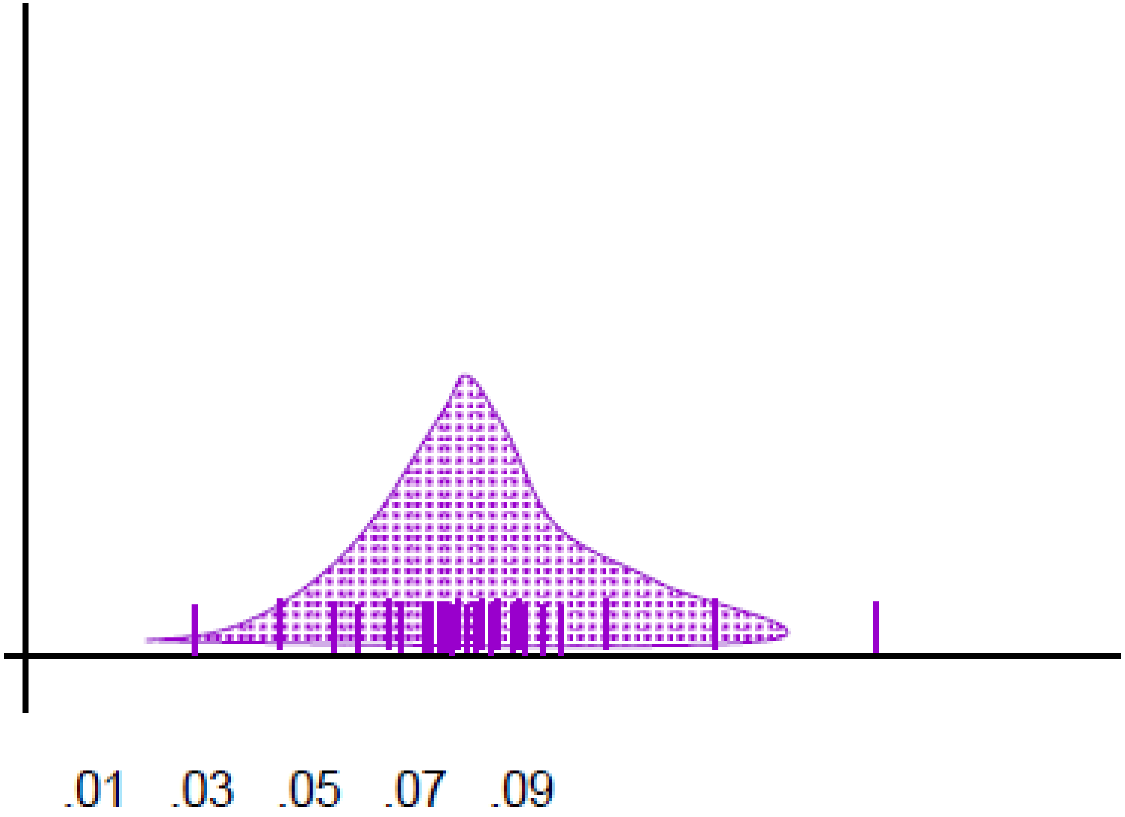
Expectation Maximization (EM)

- Pretend we *do* know the parameters
  - Initialize randomly
- [E step] Compute probability of instance having each possible value of the hidden variable
- **[M step]** Treating each instance as ***fractionally*** having **both** values compute the new parameter values



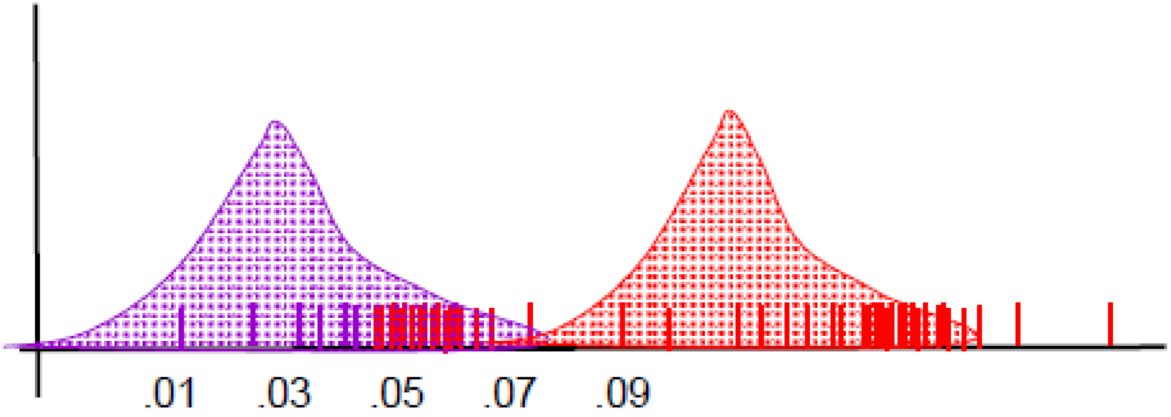
ML Mean of Single Gaussian

$$U_{ml} = argmin_u \sum_i (x_i - u)^2$$



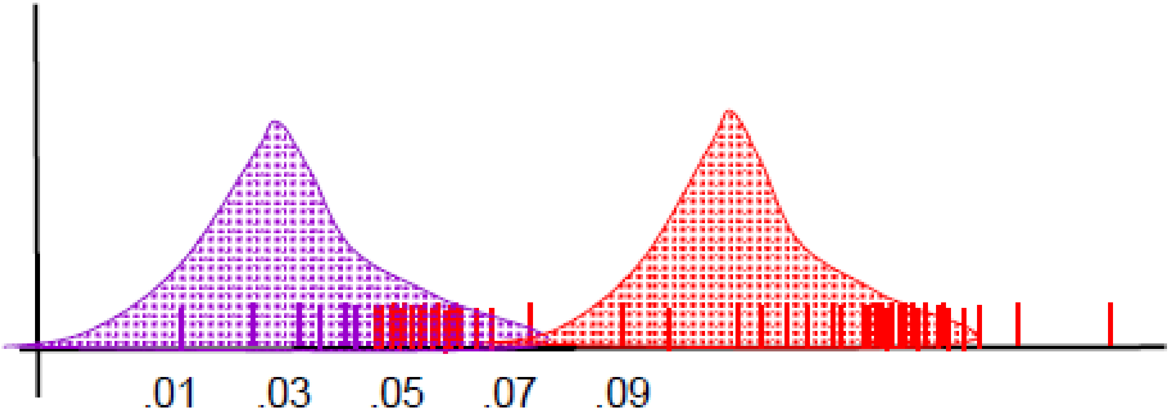
Expectation Maximization (EM)

- [M step] Treating each instance as ***fractionally*** having **both** values compute the new parameter values



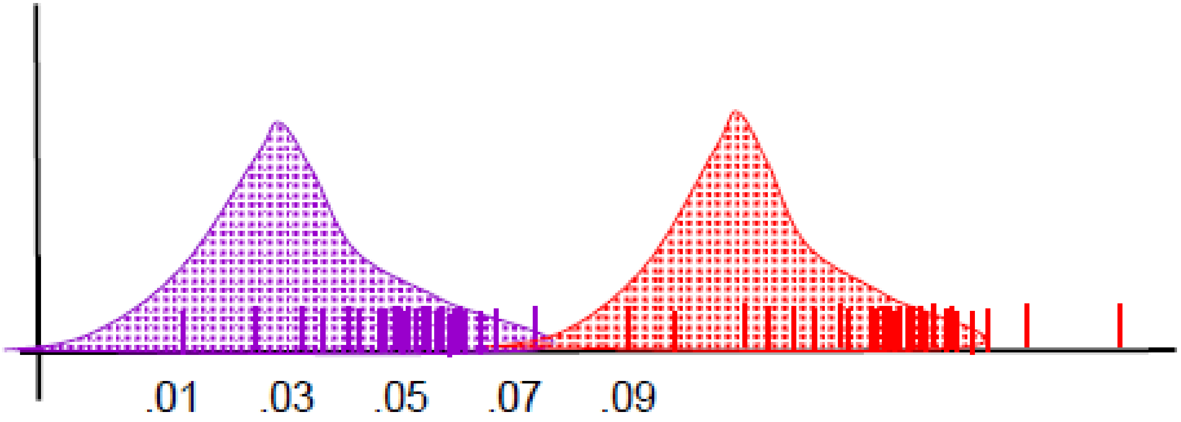
Expectation Maximization (EM)

- [E step] Compute probability of instance having each possible value of the hidden variable



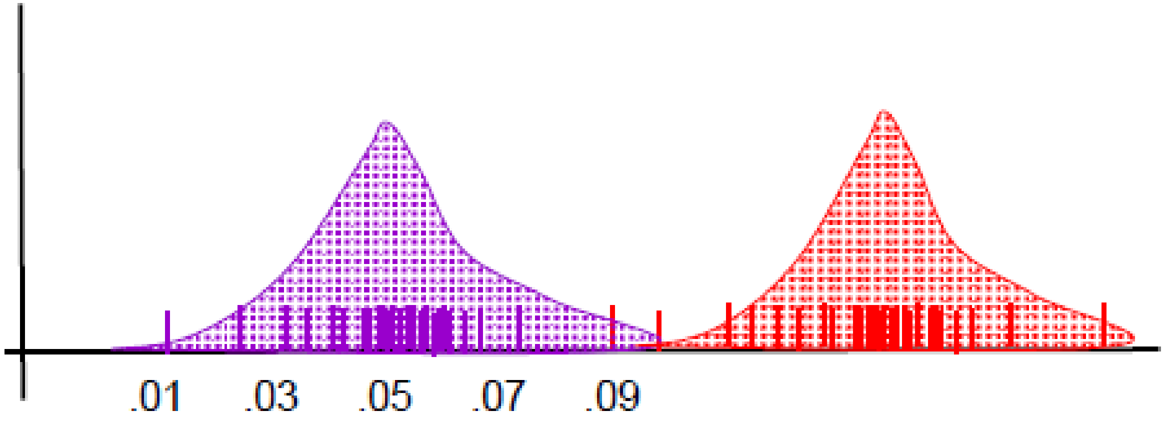
Expectation Maximization (EM)

- [E step] Compute probability of instance having each possible value of the hidden variable
- [M step] Treating each instance as fractionally having both values compute the new parameter values



Expectation Maximization (EM)

- [E step] Compute probability of instance having each possible value of the hidden variable
- [M step] Treating each instance as fractionally having both values compute the new parameter values



EM for HMMs

[E step] Compute probability of instance having each possible value of the hidden variable

- Compute the forward and backward probabilities for given model parameters and our observations

[M step] Treating each instance as fractionally having every value compute the new parameter values

- Re-estimate the model parameters
- Simple counting

Summary - Learning

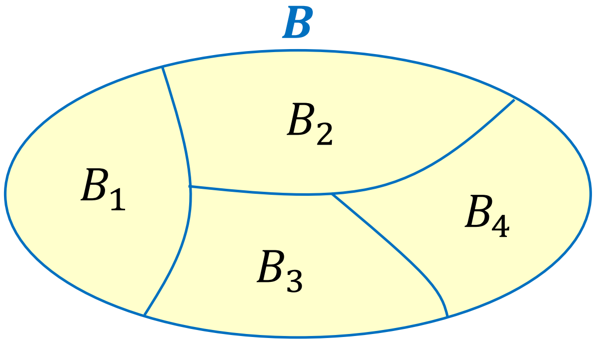
- Use hill-climbing
  - Called the Baum/Welch algorithm
  - Also “forward-backward algorithm”

- Idea
  - Use an initial parameter instantiation
  - Loop
    - Compute the forward and backward probabilities for given model parameters and our observations
    - Re-estimate the parameters
  - Until estimates don’t change much

Bayes' Theorem

When $B = B_1 \cup B_2 \cup \cdots \cup B_K$ and $B_1, B_2, \dots, B_K$ are coprime

$$\Rightarrow A = (A \cap B_1) \cup (A \cap B_2) \cup \cdots \cup (A \cap B_K)$$



Then,

$$\therefore P(B_j|A) = \frac{P(A|B_j)P(B_j)}{P(A)}$$

$$= \frac{P(A|B_j)P(B_j)}{P(A|B_1)P(B_1) + \cdots + P(A|B_K)P(B_K)}$$

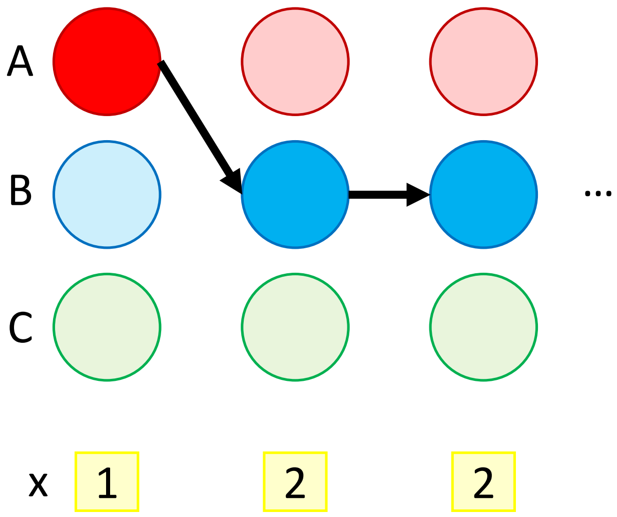
Baum-Welch Algorithm

- Estimating Emission probabilities $\boxed{\gamma_t(j) = P(y_t = S_j|\theta)}$

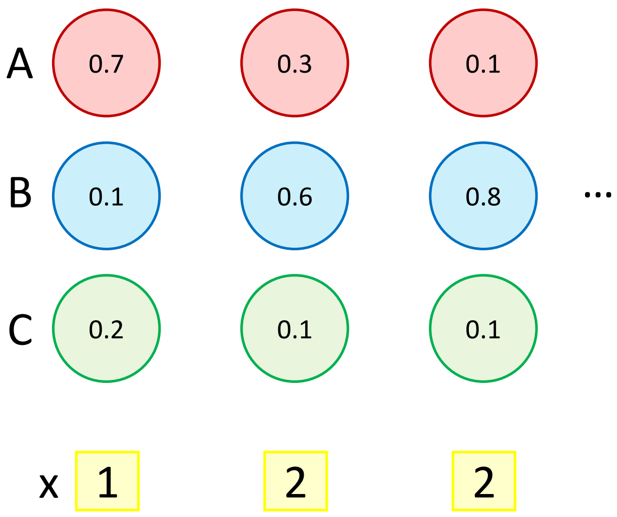
$$\gamma_t(j) = P\left(y_t = S_j \middle| x_1x_2 \dots x_T, \theta\right) = \frac{P\left(y_t = S_j, x_1x_2 \dots x_T \middle| \theta\right)}{P(x_1x_2 \dots x_T|\theta)}$$

$$= \frac{\alpha_t(j) \times \beta_t(j)}{\sum_{n=1}^{N} \alpha_t(n) \times \beta_t(n)}$$

Decoding



γ

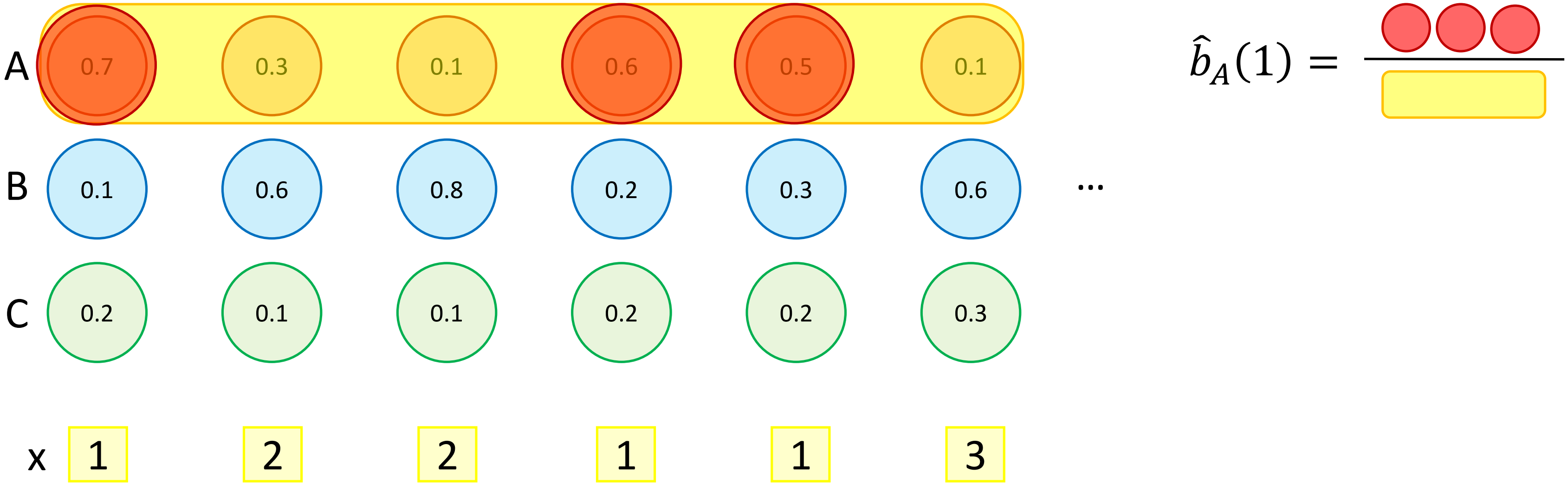


A	0.7	0.3	0.1	
B	0.1	0.6	0.8	…
C	0.2	0.1	0.1	
x	1	2	2	

Baum-Welch Algorithm

- Estimated Emission probability

$$\hat{b}_j(o_k) = \frac{\sum_{t=1,s.t.x_t=o_t}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$



A	0.7	0.3	0.1	0.6	0.5	0.1	
B	0.1	0.6	0.8	0.2	0.3	0.6	…
C	0.2	0.1	0.1	0.2	0.2	0.3	
x	1	2	2	1	1	3	

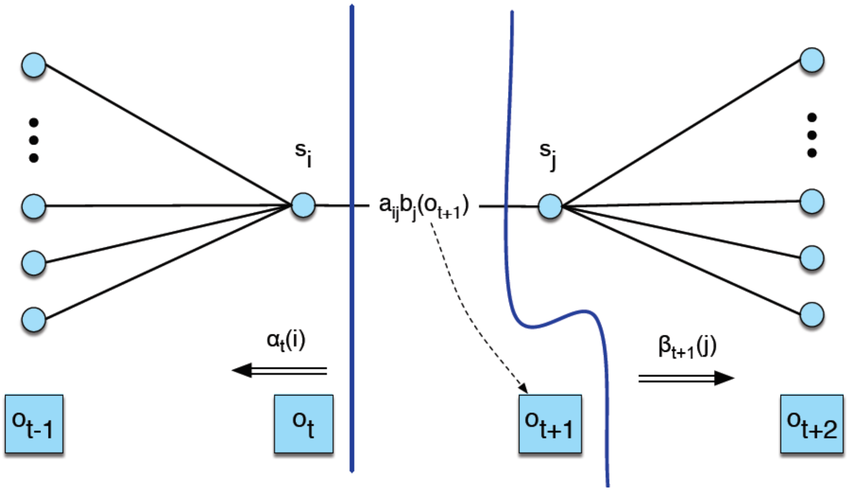
$\hat{b}_A(1) =$

Baum-Welch Algorithm

- Estimating Transition probabilities

$$\boxed{\xi_t(i,j) = P(y_t = S_i, y_{t+1} = S_j | x_1 x_2 \dots x_T, \theta)}$$

$$\begin{aligned}
\xi_t(i,j) &= P\left(y_t = S_i, y_{t+1} = S_j \middle| x_1 x_2 \dots x_T, \theta\right) \\
&= \frac{P\left(y_t = S_i, y_{t+1} = S_j, x_1 x_2 \dots x_T \middle| \theta\right)}{P(x_1 x_2 \dots x_T | \theta)} \\
&= \frac{\alpha_t \times P\left(y_t = S_j \middle| y_{t-1} = S_i\right) \times P\left(x_{t+1} = o_{t+1} \middle| y_t = S_j\right) \times \beta_{t+1}(j)}{\sum_{n=1}^{N} \alpha_t(n) \times \beta_t(n)}
\end{aligned}$$

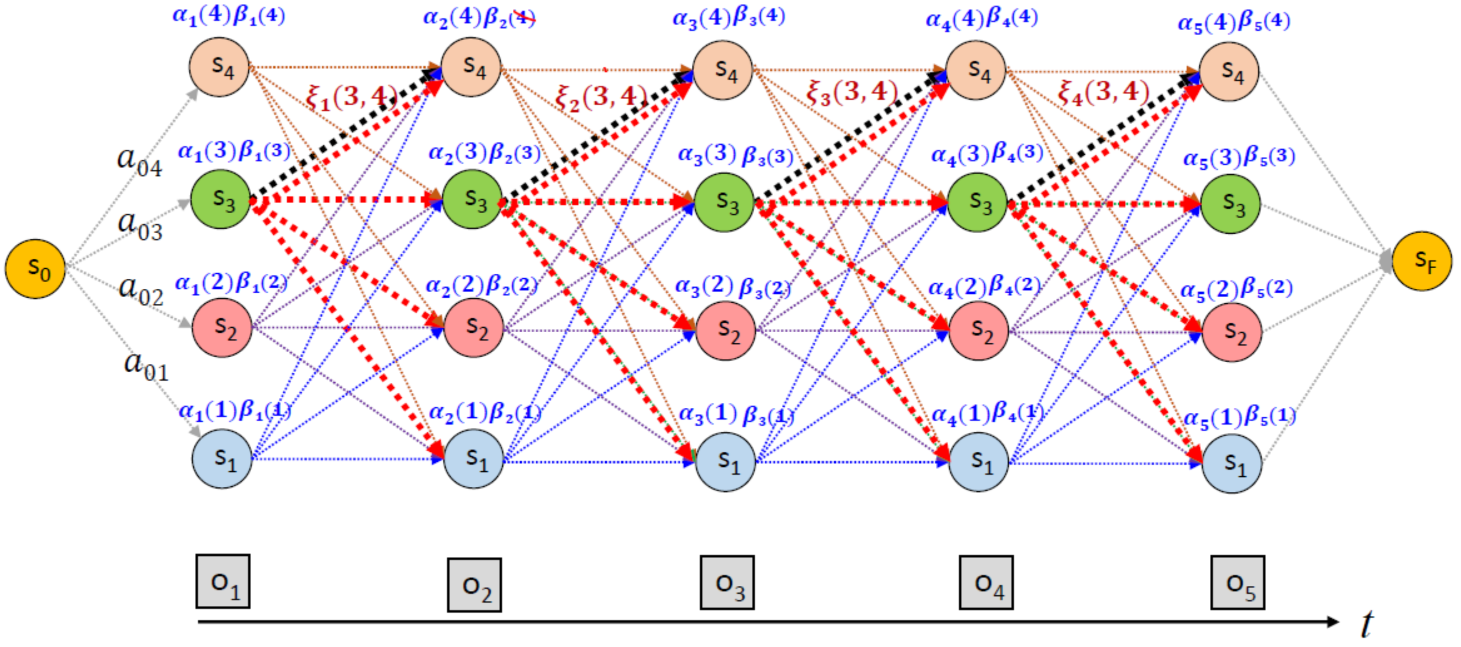


Baum-Welch Algorithm

- Estimating Transition probabilities

$$\boxed{\xi_t(i,j) = P(y_t = S_i, y_{t+1} = S_j | x_1 x_2 \dots x_T, \theta)}$$

$$P\left(y_t = S_j \middle| y_{t-1} = S_i\right) = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$



Discriminative vs Generative Models

- So far: all models generative
- Generative Models …

$$model\ P(y, x)$$

- Discriminative Models …

$$model\ P(y \mid x)$$

$P(y|x)$ does not include a model of $P(x)$. So it does not need to model the dependencies between features!

Discriminative Models Often Better

- Eventually, what we care about is $p(y|x)$!
  - Bayes Net describes a family of joint distributions of.

    whose conditionals take certain form
  - But there are many other joint models, whose conditionals

    also have that form.
- We want to make independence assumptions amon g y, but not among x.

Thank you!