CSE 574

Finite State Machines for Information Extraction

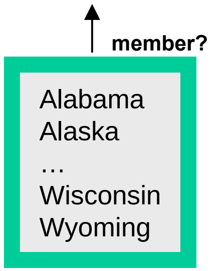
- **Today and Friday**
  - Dan @ IUI on the Canary Islands
  - I am presenting

- **Topics**
  - HMMs, Conditional Random Fields
  - Inference and Learning

Landscape of IE Techniques: Models

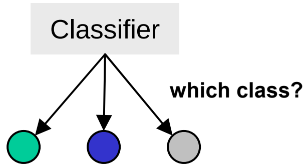
Lexicons

Abraham Lincoln was born in Kentucky.



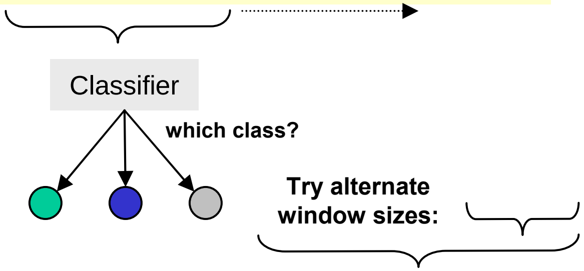
Classify Pre-segmented Candidates

Abraham Lincoln was born in Kentucky.

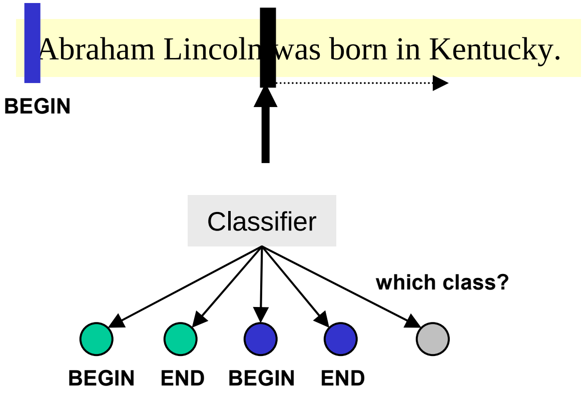


Sliding Window

Abraham Lincoln was born in Kentucky.

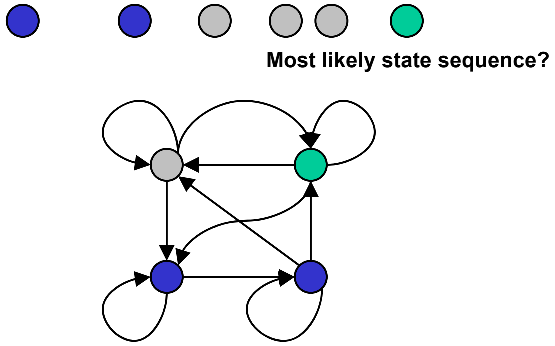


Boundary Models



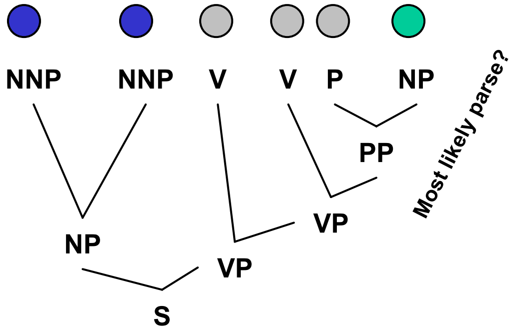
Finite State Machines

Abraham Lincoln was born in Kentucky.



Context Free Grammars

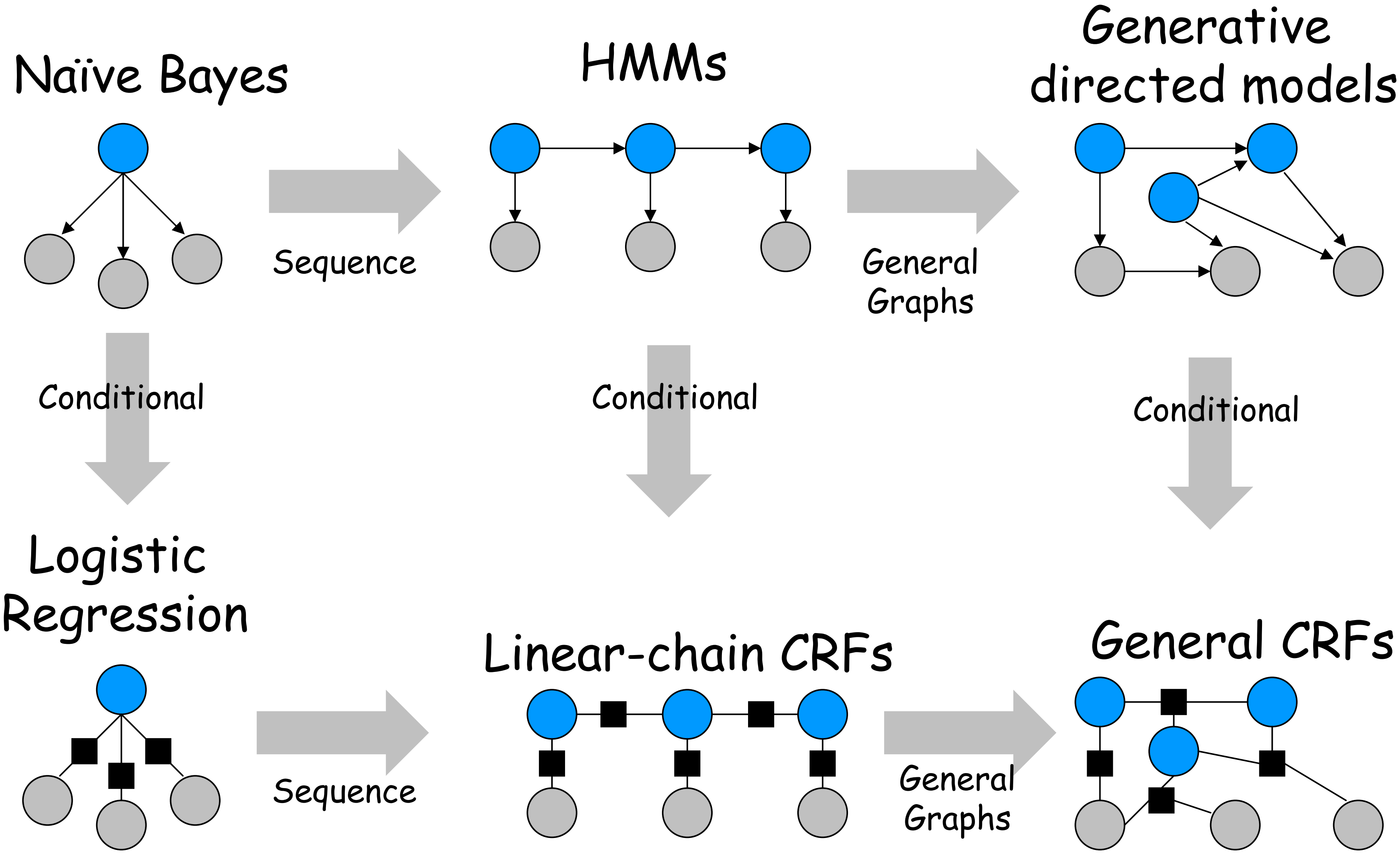
Abraham Lincoln was born in Kentucky.



Each model can capture words, formatting, or both

Slides from Cohen & McCallum

Finite State Models



Naïve Bayes

Sequence

HMMs

General Graphs

Generative directed models

Conditional

Conditional

Conditional

Logistic Regression

Sequence

Linear-chain CRFs

General Graphs

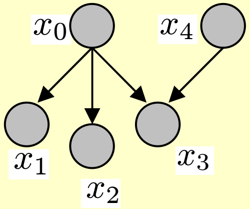
General CRFs

Graphical Models

- Family of probability certain way

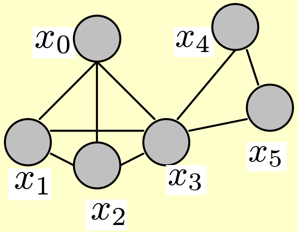
Node is independent of its non-descendants given its parents

- Directed (Bayes Nets)



Node is independent all other nodes given its neighbors

- Undirected (Markov Random Field)

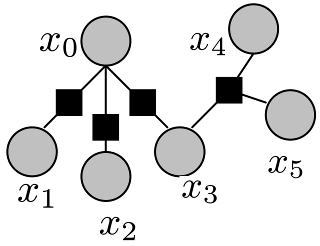


$$p(\mathbf{x}) = \tfrac{1}{Z}\prod_C \Psi_C(\mathbf{x}_C)$$

$C \subset \{x_1, \ldots, x_K\}$ clique

Ψ_C potential function

- Factor Graphs



$$p(\mathbf{x}) = \tfrac{1}{Z}\prod_A \Psi_A(\mathbf{x}_A)$$

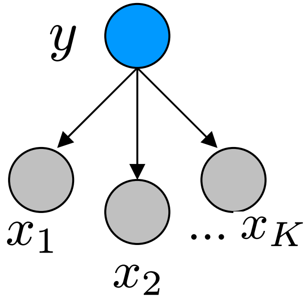
$A \subset \{x_1, \ldots, x_K\}$

Ψ_A factor function

Recap: Naïve Bayes

- **Assumption: features independent given label**
- **Generative Classifier**
  - Model joint distribution p(x,y)

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k|y)$$

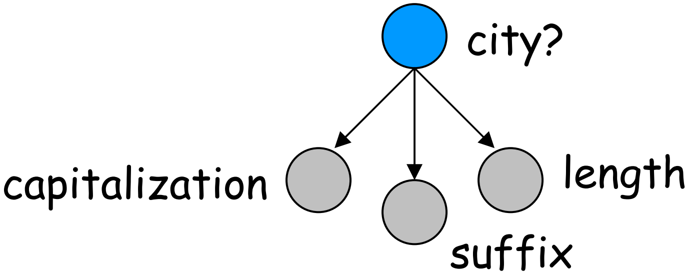


  - Inference

$$p(y|\mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k|y) \frac{1}{p(\mathbf{x})}$$

  - Learning: counting
  - Example

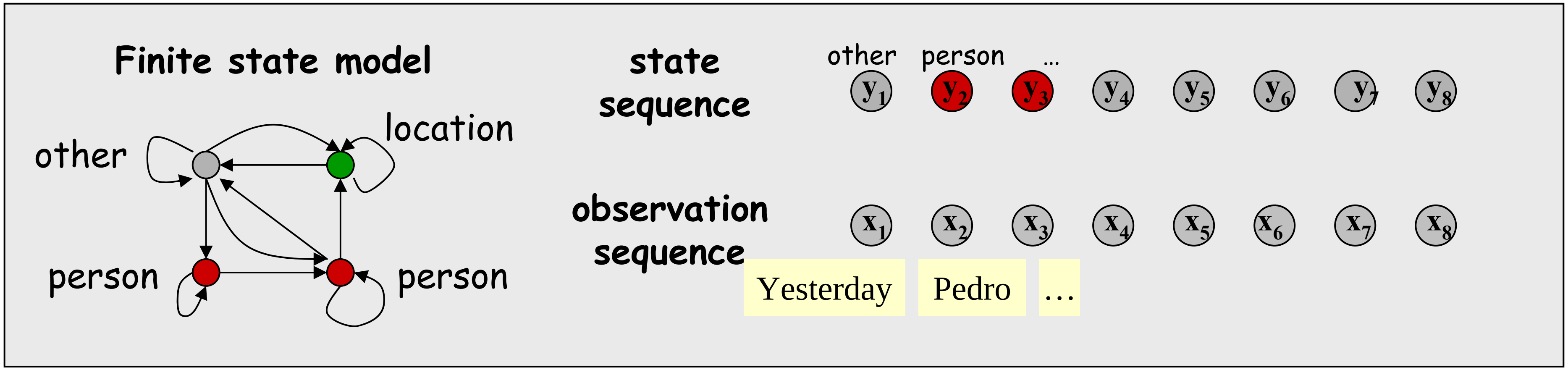
The article appeared in the Seattle Times.



Labels of neighboring words dependent!

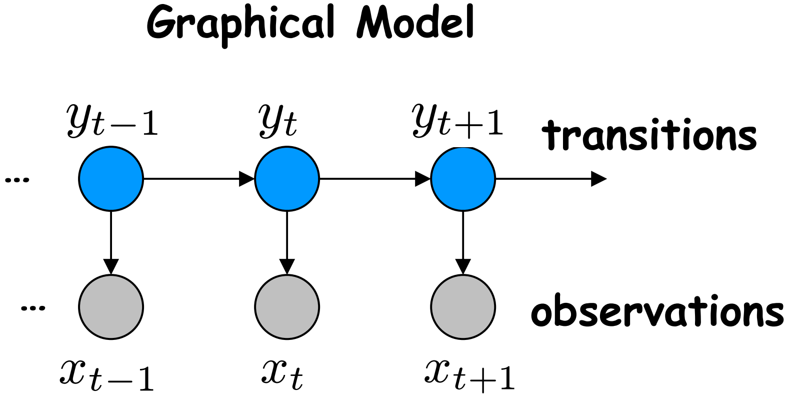
Need to consider sequence!

Hidden Markov Models

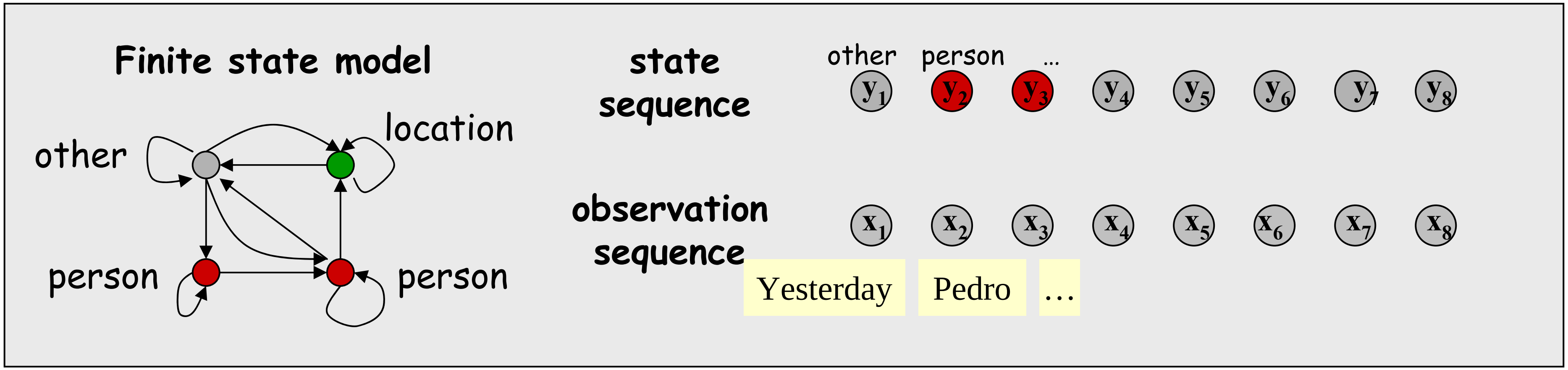


- *Generative Sequence Model*
  - 2 assumptions to make joint distribution tractable

1. Each state depends only on its immediate predecessor.
2. Each observation depends only on current state.



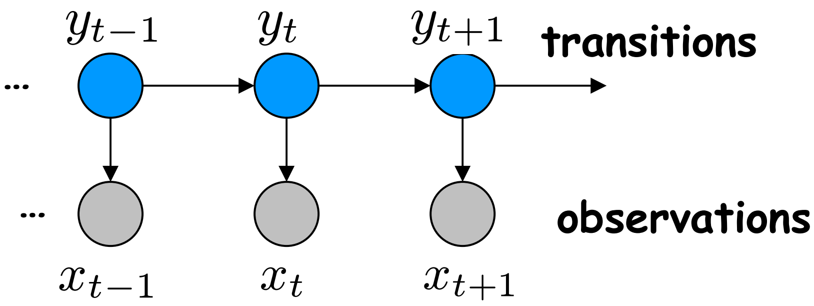
Hidden Markov Models



- **Generative Sequence Model**

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

Graphical Model



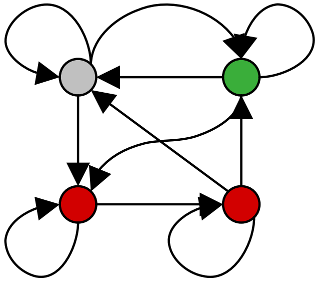
- **Model Parameters**
  - Start state probabilities $p(y_1) := p(y_1|y_0)$
  - Transition probabilities $p(y_t|y_{t-1})$
  - Observation probabilities $p(x_t|y_t)$

IE with Hidden Markov Models

Given a sequence of observations:

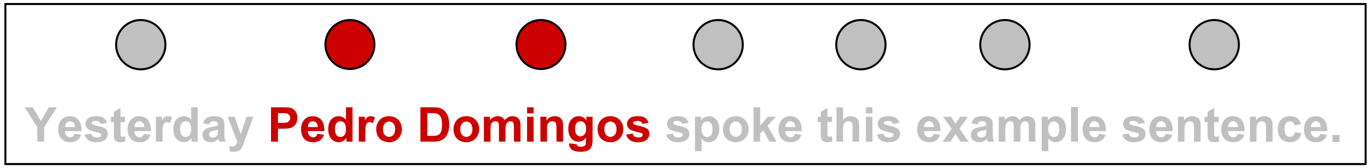
Yesterday Pedro Domingos spoke this example sentence.

and a trained HMM:



🟥 person name
🟩 location name
⬜ background

Find the most likely state sequence: (Viterbi) $\arg\max_{\vec{s}} P(\vec{s},\vec{o})$



Yesterday **Pedro Domingos** spoke this example sentence.

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

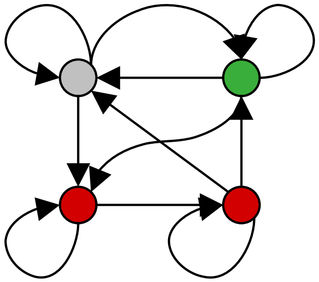
Person name: Pedro Domingos

Slide by Cohen & McCallum

IE with Hidden Markov Models

For sparse extraction tasks :

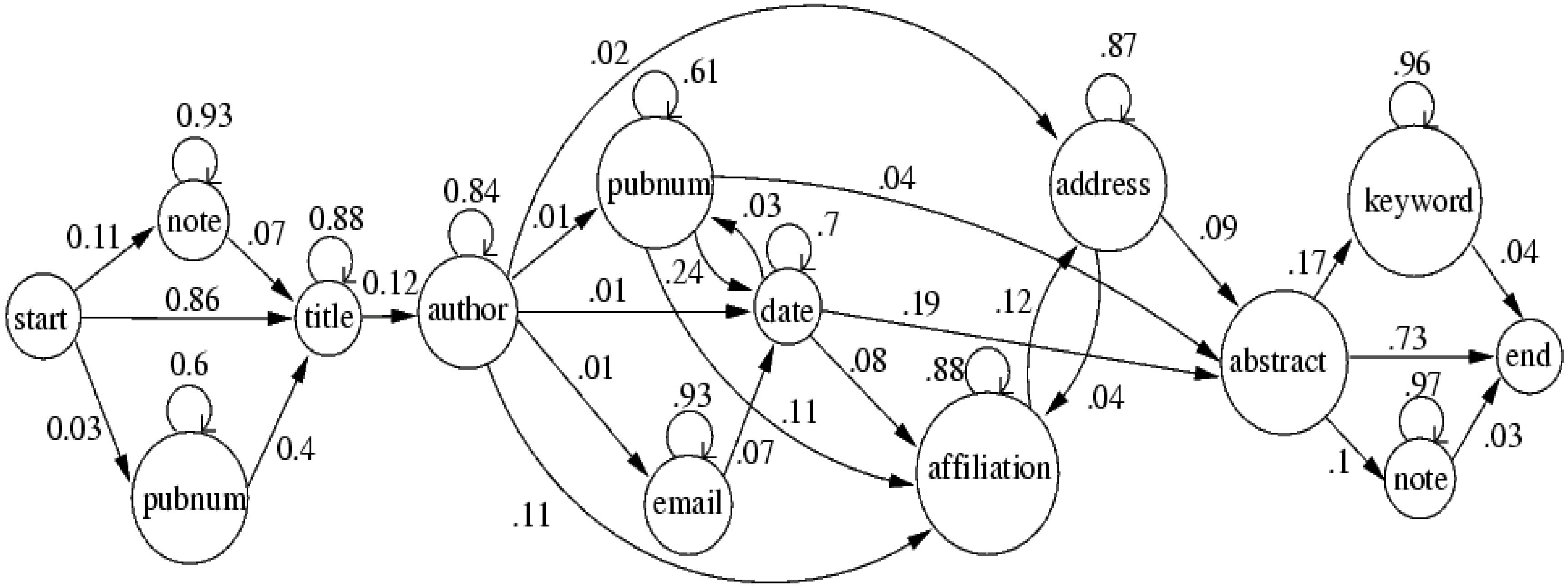
- Separate HMM for each type of target
- Each HMM should
  - Model entire document
  - Consist of ***target*** and ***non-target*** states
  - Not necessarily fully connected



Slide by Okan Basegmez

Information Extraction with HMMs

- **Example – Research Paper Headers**

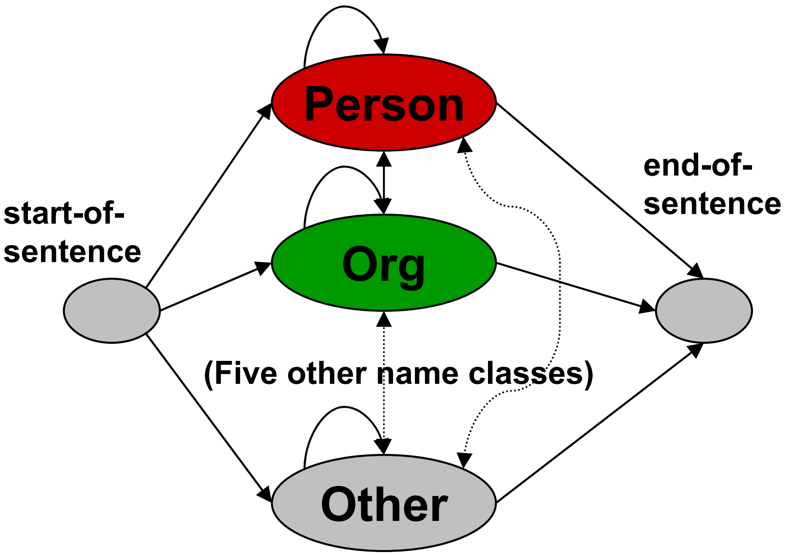


Slide by Okan Basegmez

HMM Example: "Nymble"

[Bikel, et al 1998], [BBN "IdentiFinder"]

Task: Named Entity Extraction



Transition probabilities

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

Back-off to:

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

$p(y_t)$

Observation probabilities

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

or $p(x_t | y_t, x_{t-1})$

Back-off to:

$p(x_t | y_t)$

$p(x_t)$

Train on ~500k words of news wire text.

Results:

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

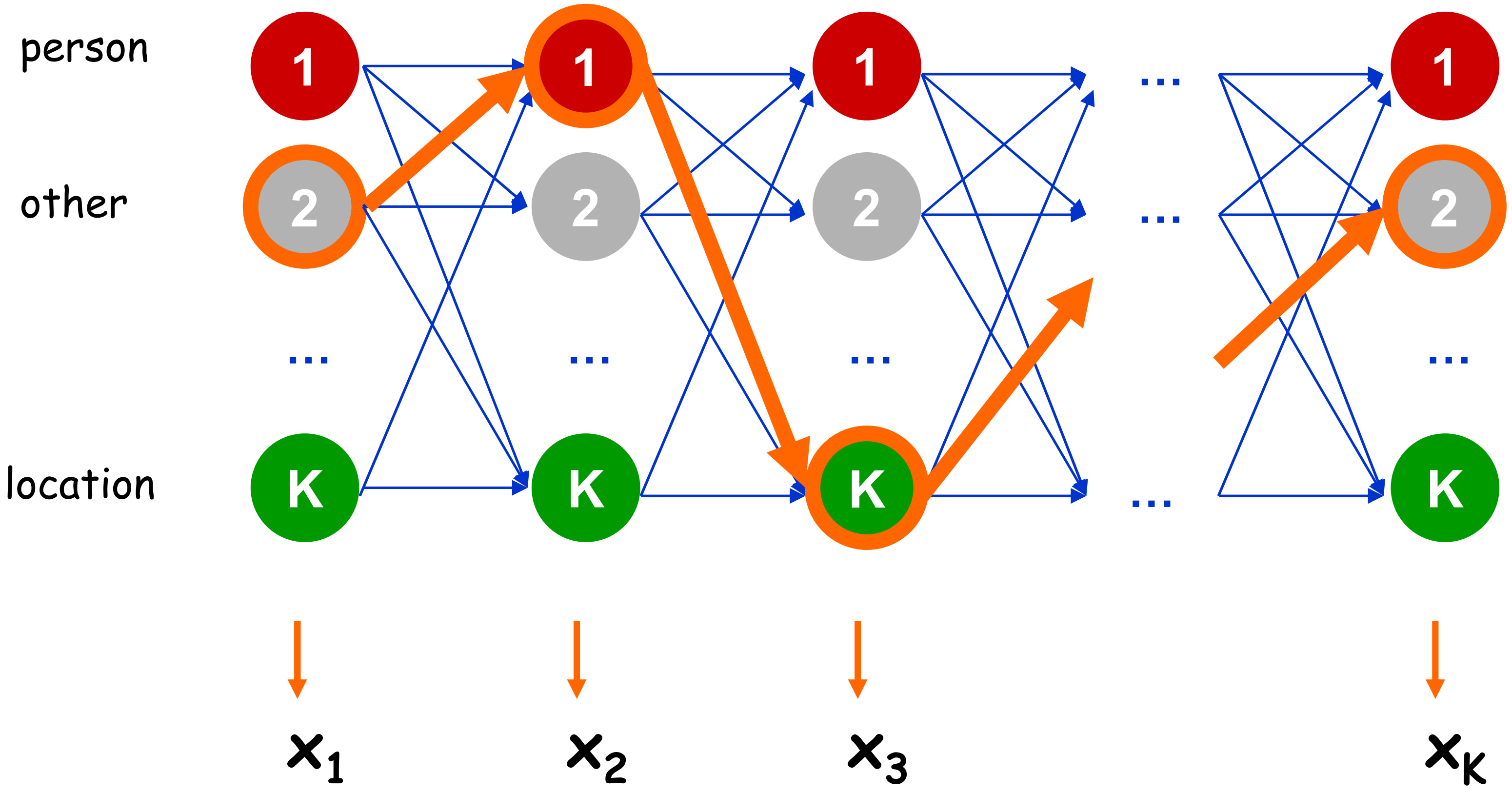
Other examples of shrinkage for HMMs in IE: *[Freitag and McCallum '99]*

Slide by Cohen & McCallum

A parse of a sequence

Given a sequence $x = x_1 \ldots\ldots x_N$,

A parse of o is a sequence of states $y = y_1, \ldots\ldots, y_N$



Slide by Serafim Batzoglou

Question #1 - Evaluation

GIVEN

A sequence of observations $x_1\ x_2\ x_3\ x_4 \ldots\ldots x_N$

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!

Question #2 - Decoding

GIVEN

A sequence of observations $x_1\ x_2\ x_3\ x_4\ \ldots\ldots x_N$

A trained HMM

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

QUESTION

How dow we choose the corresponding parse (state sequence) $y_1\ y_2\ y_3\ y_4\ \ldots\ldots y_N$, which "best" explains $x_1\ x_2\ x_3\ x_4\ \ldots\ldots x_N$?

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

Question #3 - Learning

GIVEN

A sequence of observations $x_1\ x_2\ x_3\ x_4\ \ldots\ldots x_N$

QUESTION

How do we learn the model parameters

$\theta = (\ p(y_t|y_{t-1}) \quad p(x_t|y_t), \quad p(y_1))$ to maximize $P(x, \lambda)$?

Solution to #1: Evaluation

Given observations $x=x_1 \ldots x_N$ and HMM θ, what is $p(x)$?

Naïve: enumerate every possible state sequence $y=y_1 \ldots y_N$

Probability of x and given particular y

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

2T multiplications per sequence

Probability of particular y

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

Summing over all possible state sequences we get

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

N^T state sequences!

For small HMMs
T=10, N=10
there are 10
billion sequences!

Solution to #1: Evaluation

Use Dynamic Programming:

Define forward variable

$$\alpha_t(i) = P(\mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_t, y_t = S_i)$$

probability that at time t

- the state is y_i
- the partial observation sequence $x=x_1 \ldots x_t$ has been omitted

Solution to #1: Evaluation

- **Use Dynamic Programming**

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

- **Cache and reuse inner sums**
- **Define *forward variables***

$$
\alpha_t(i) \quad := \quad P(\mathbf{x}_1 \mathbf{x}_2 \ldots \mathbf{x}_t, y_t = S_i)
$$

probability that at time t

- the state is $y_t = S_i$
- the partial observation sequence $x=x_{1 \ldots}x_t$ has been omitted

The Forward Algorithm

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

INITIALIZATION

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

INDUCTION

$$\begin{aligned} \alpha_t(i) \quad &= \quad p(x_1 x_2 \ldots x_t, y_t = S_i) \\ &= \quad \sum_{j \in S} \alpha_{t-1}(j) p(y_t = S_i | y_{t-1} = S_j) p(x_t | y_t) \end{aligned}$$

TERMINATION

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

Time:

$O(K^2N)$

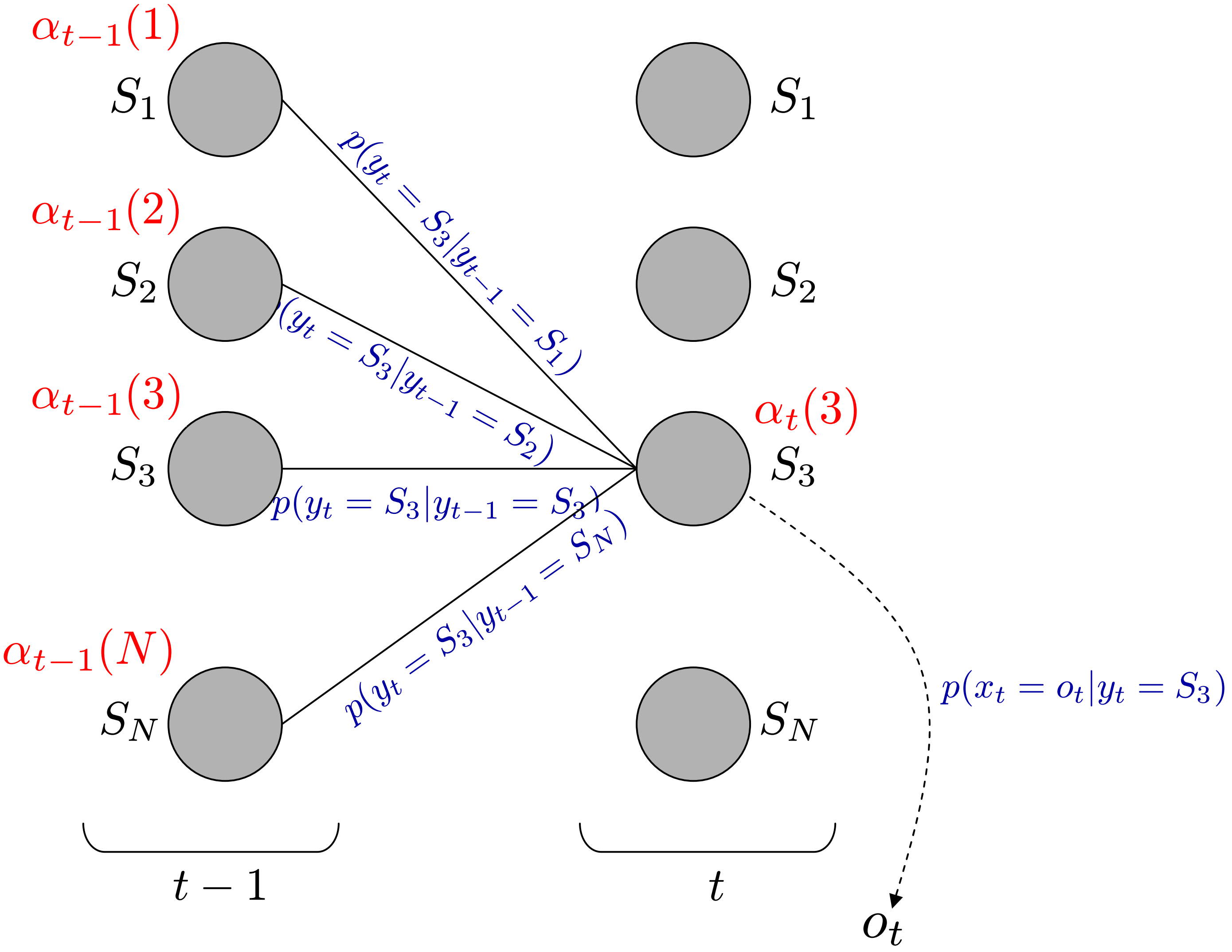
Space:

$O(KN)$

K = |S|	#states
N	length of sequence

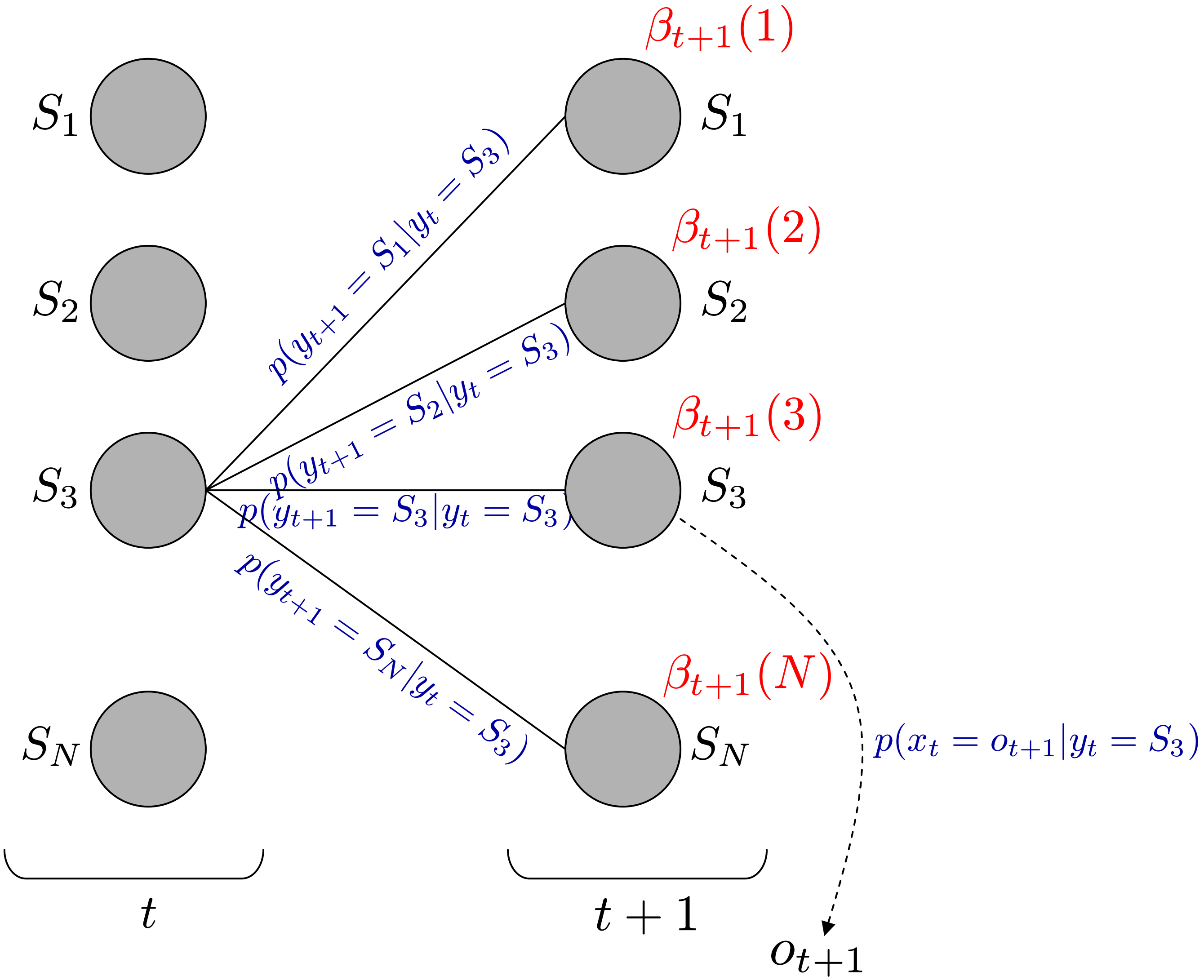
The Forward Algorithm

$$\boxed{\alpha_t(i) \quad := \quad P(\mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_t, y_t = S_i)}$$



The Backward Algorithm

$$\boxed{\beta_t(i) \quad := \quad P(y_t = S_i, x_{t+1}x_{t+2}\ldots x_T)}$$


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

The Backward Algorithm

$$\boxed{\beta_t(i) \quad := \quad P(y_t = S_i, x_{t+1}x_{t+2}...x_T)}$$

INITIALIZATION

$$\beta_T(i) = 1$$

INDUCTION

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

TERMINATION

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

Time:

O(K^{2}N)

Space:

O(KN)

Solution to #2 - Decoding

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

Several optimal solutions

- 1. States which are individually most likely:

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

most likely state y^*_t at time t is then

$$y^*_t = \arg\max_{1\leq i\leq N} P(y_t = S_i|\mathbf{x})$$

But some transitions may have 0 probability!

Solution to #2 - Decoding

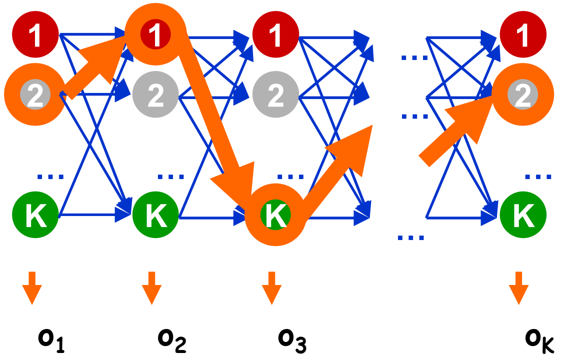
Given $x=x_1 \ldots x_N$ and HMM θ, what is "best" parse $y_1 \ldots y_N$?

Several optimal solutions

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

We want to find sequence $y_1 \ldots y_N$, such that $P(x,y)$ is maximized

$$y^* = \text{argmax}_y \, P(\, x, \, y \,)$$



Again, we can use dynamic programming!

The Viterbi Algorithm

DEFINE

$$\delta_t(i) = \max_{y_1, y_2, \ldots, y_{t-1}} P(y_1, y_2, \ldots, y_{t-1}, y_t = i, o_1, o_2, \ldots, o_t | \lambda)$$

INITIALIZATION

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

INDUCTION

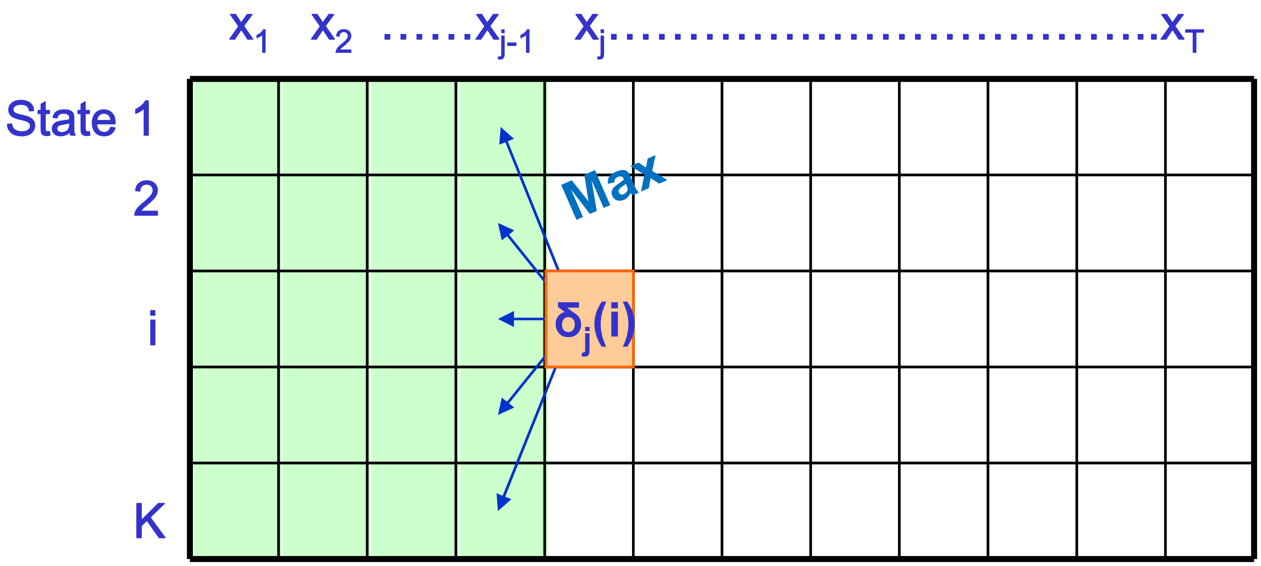
$$\delta_t(j) = \max_{i \in S} \delta_{t-1}(i) p(y_t = S_j | y_{t-1} = S_i) p(x_t | y_t = S_j)$$

TERMINATION

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

Backtracking to get state sequence y*

The Viterbi Algorithm



Time:

O(K²T) ← Linear in length of sequence

Space:

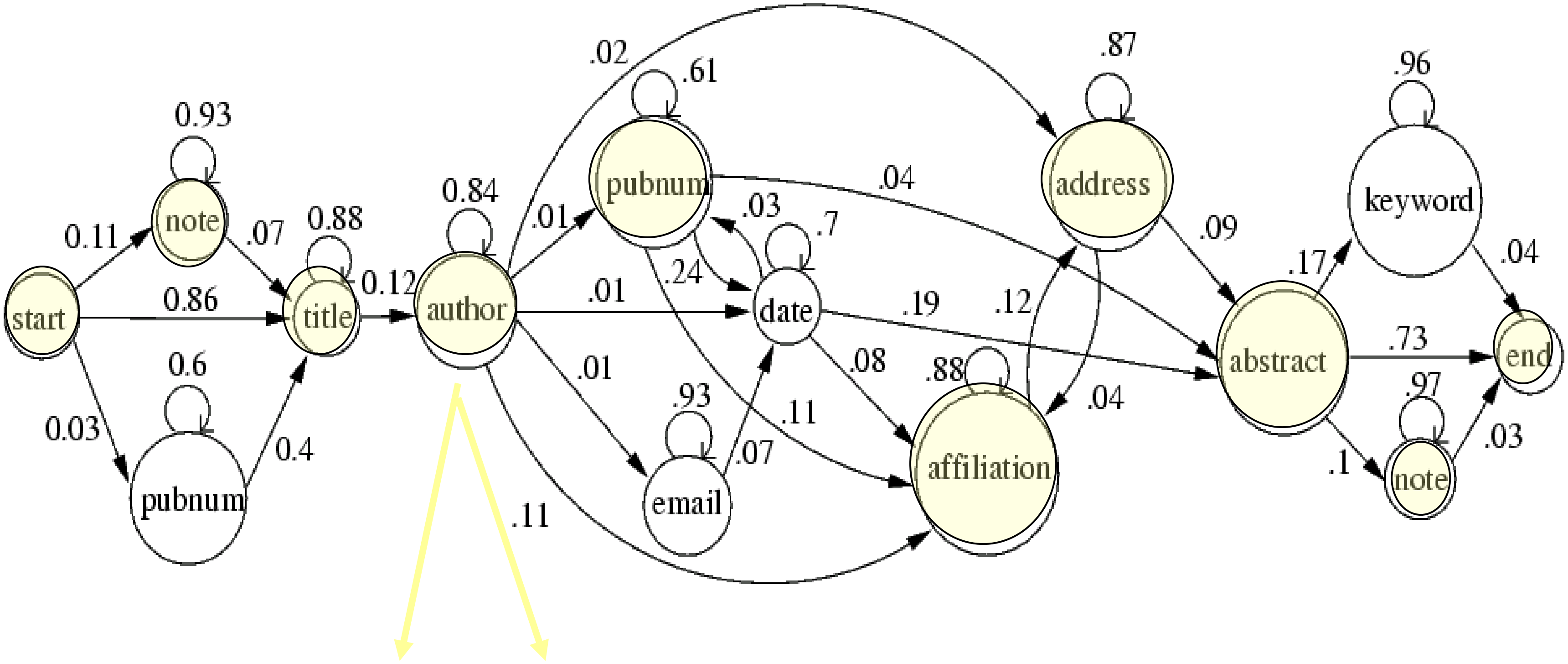
O(KT)

Remember:

$\delta_k(i)$ = probability of most likely state seq ending with state S.

Slides from Serafim Batzoglou

The Viterbi Algorithm



Pedro Domingos

Solution to #3 - Learning

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

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

  θ * = arg max P(o | θ)

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

  P(o | θ') ≥ P(o | θ)

Solution to #3 - Learning

- **Use hill-climbing**
  - Called the forward-backward (or Baum/Welch) 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

Expectation Maximization

- **The forward-backward algorithm is an instance of the more general EM algorithm**
  - The E Step:
    Compute the forward and backward probabilities for given model parameters and our observations
  - The M Step:
    Re-estimate the model parameters

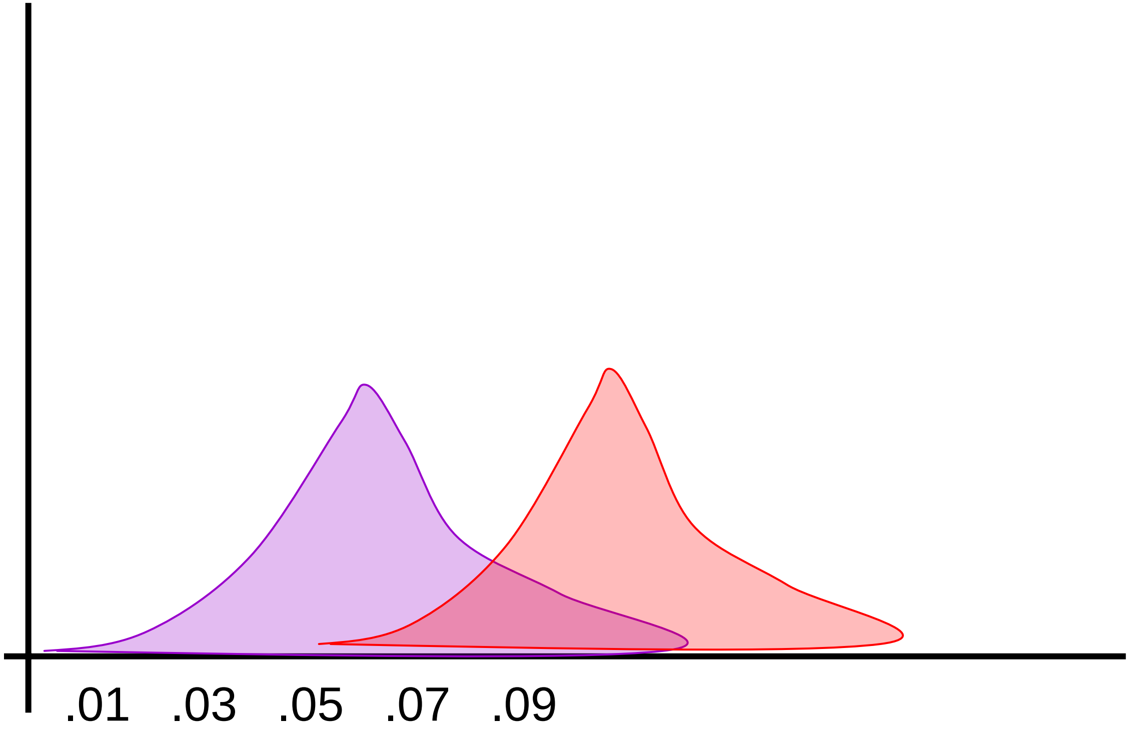
Chicken & Egg Problem

- **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!

Slide by Daniel S. Weld

Simplest Version

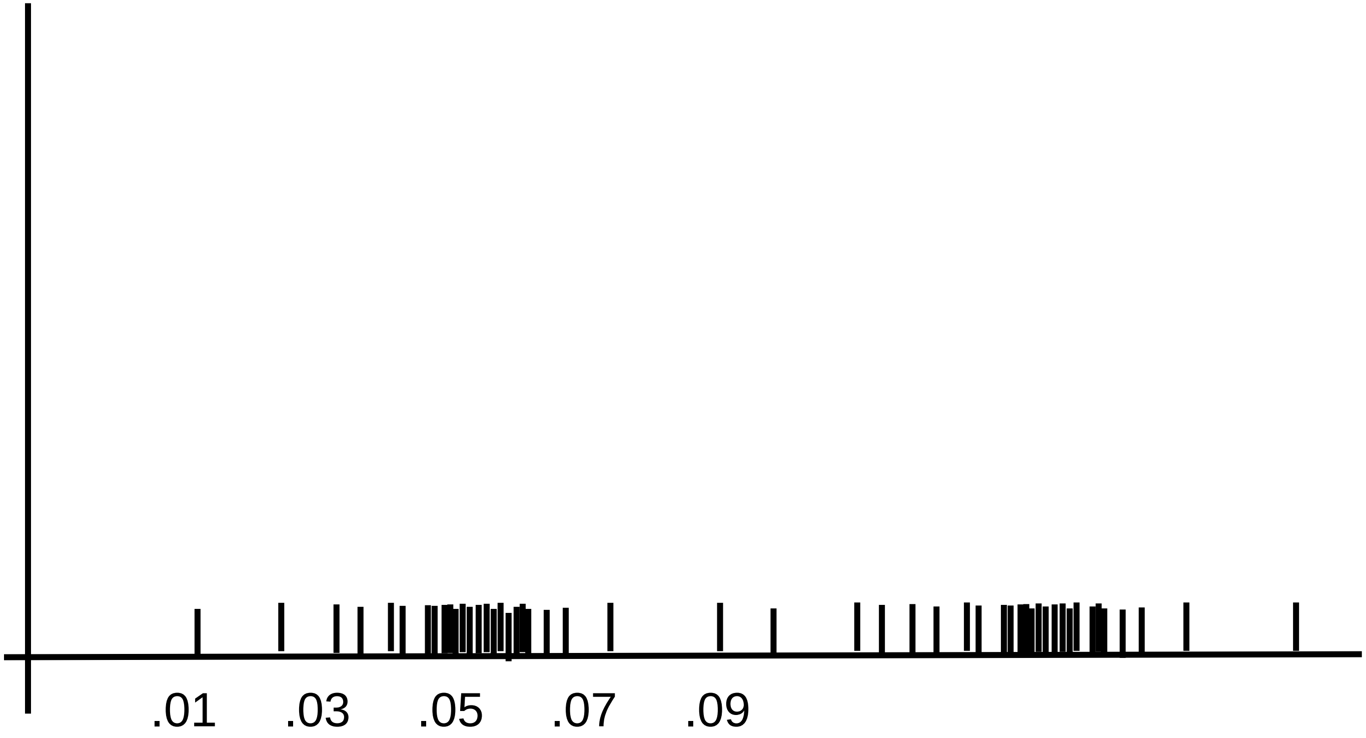
- Mixture of two distributions



- Know: form of distribution & variance,
  % =5
- Just need *mean* of each distribution

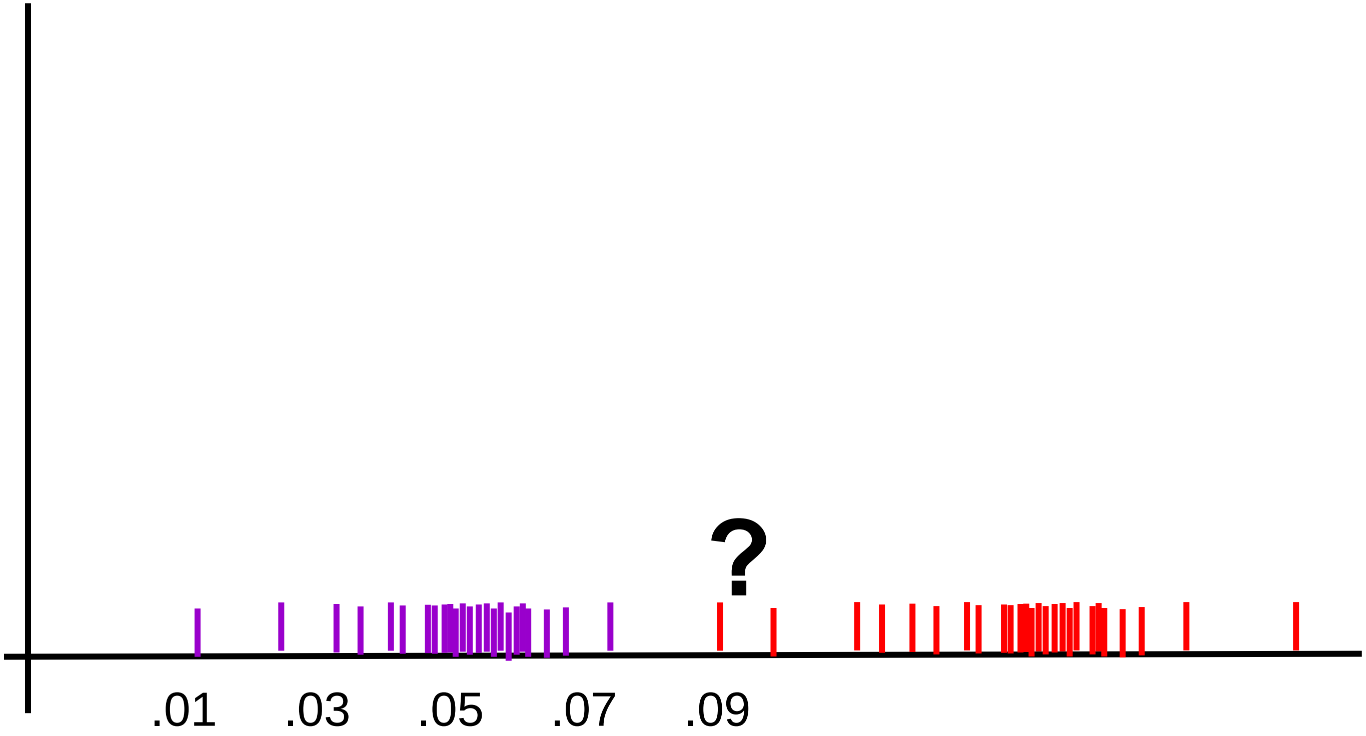
Slide by Daniel S. Weld

Input Looks Like



Slide by Daniel S. Weld

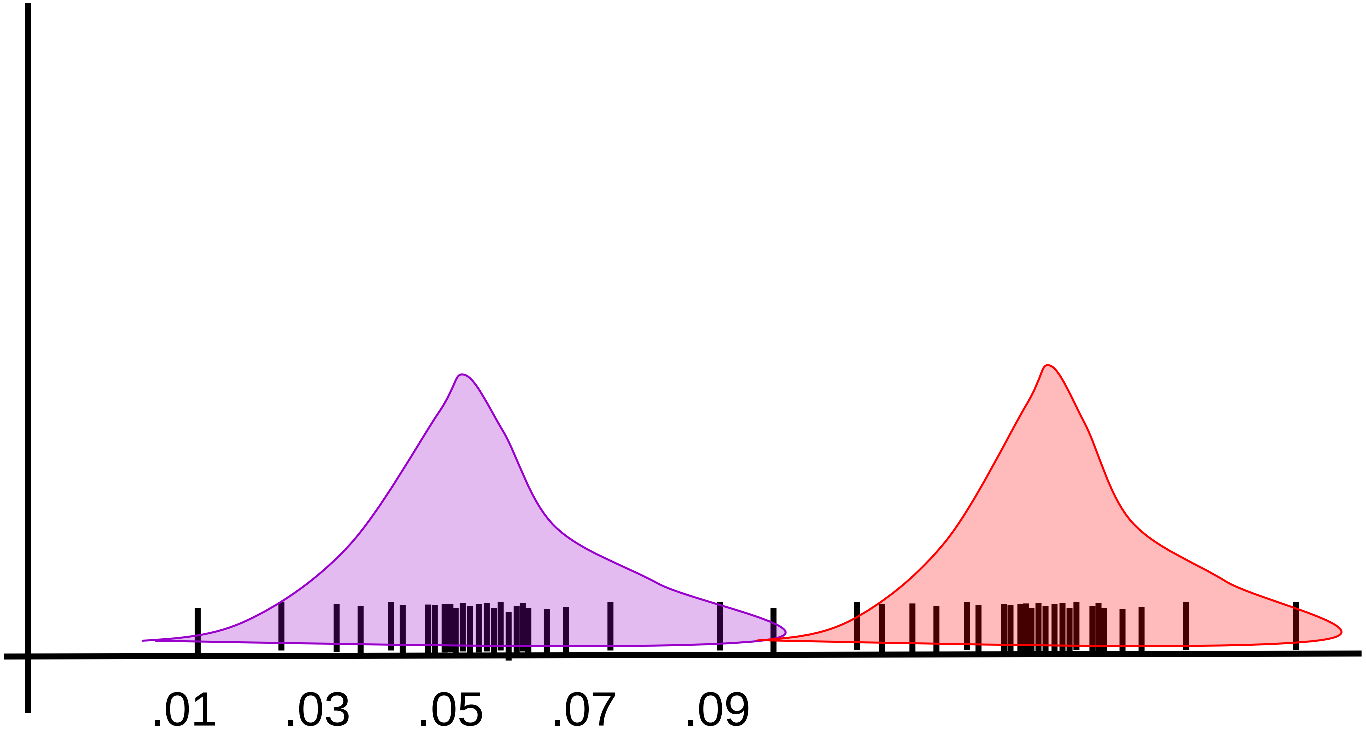
We Want to Predict



Slide by Daniel S. Weld

Chicken & Egg

Note that coloring instances would be easy *if* we knew Gausians….

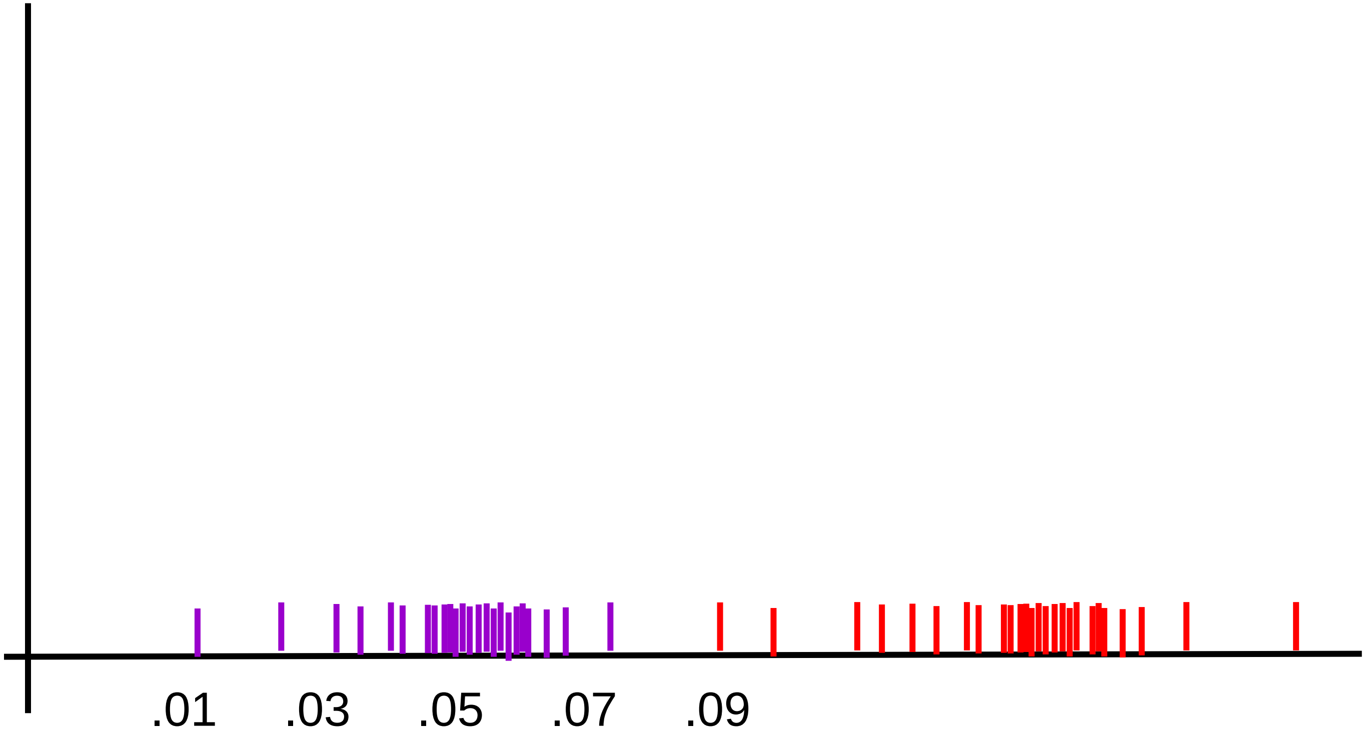


Slide by Daniel S. Weld

Chicken & Egg

And finding the Gausians would be easy

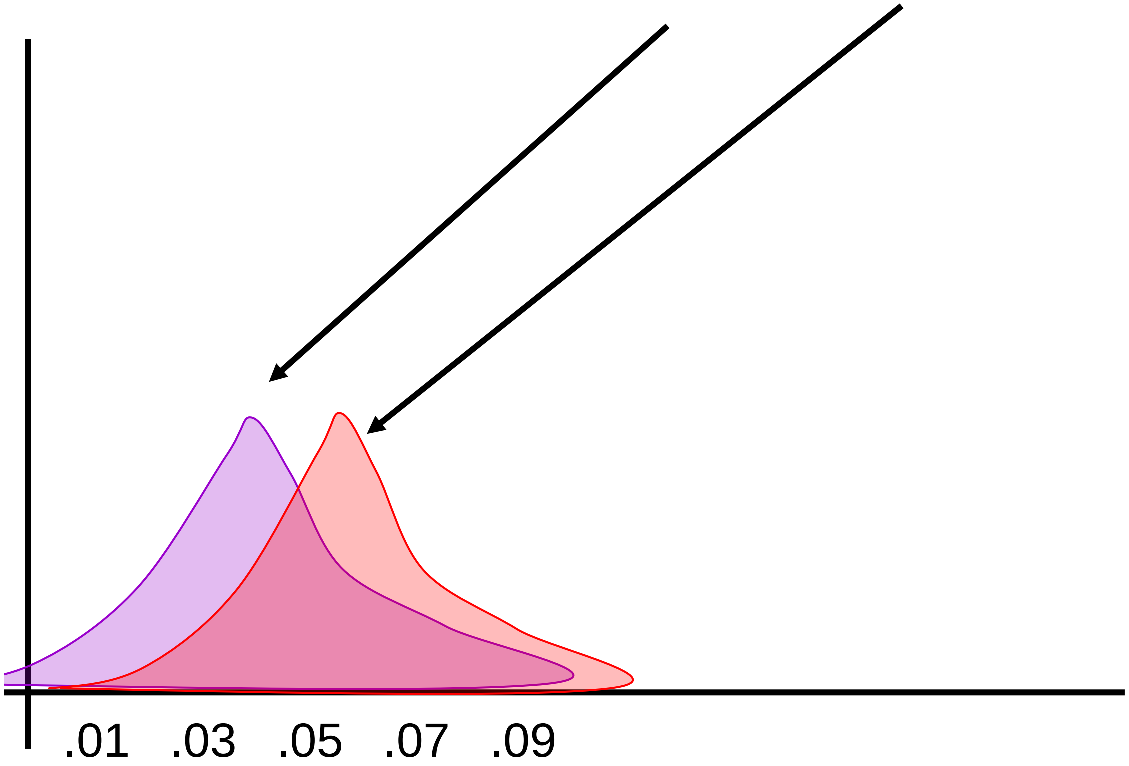
If we knew the coloring



Slide by Daniel S. Weld

Expectation Maximization (EM)

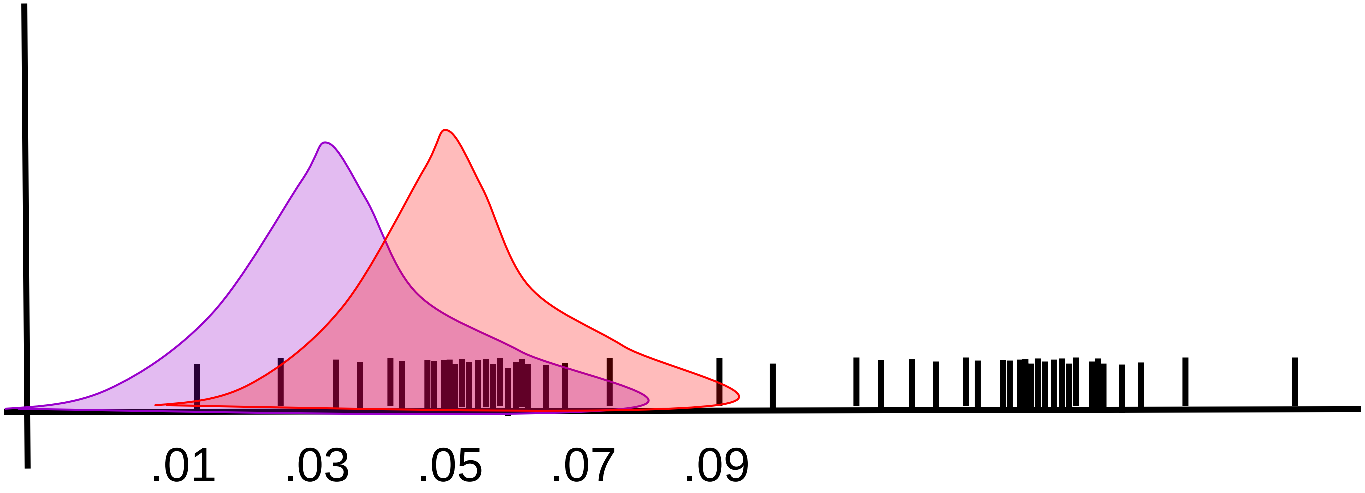
- **Pretend we *do* know the parameters**
  - Initialize randomly: set θ_1=?; θ_2=?



Slide by Daniel S. Weld

Expectation Maximization (EM)

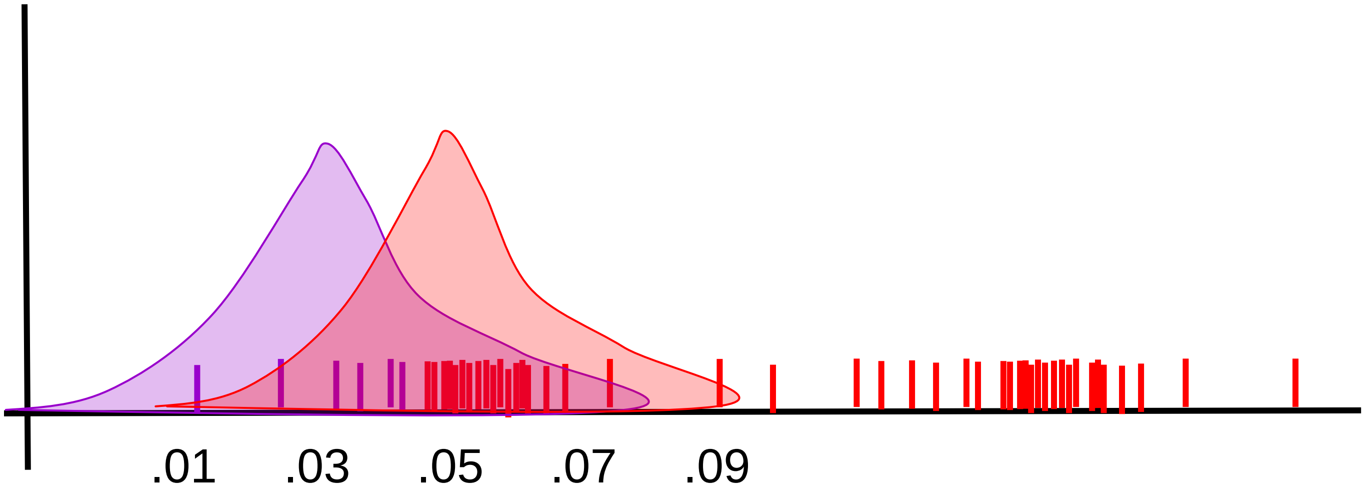
- **Pretend we *do* know the parameters**
  - Initialize randomly
- [E step] Compute probability of instance having each possible value of the hidden variable



Slide by Daniel S. Weld

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

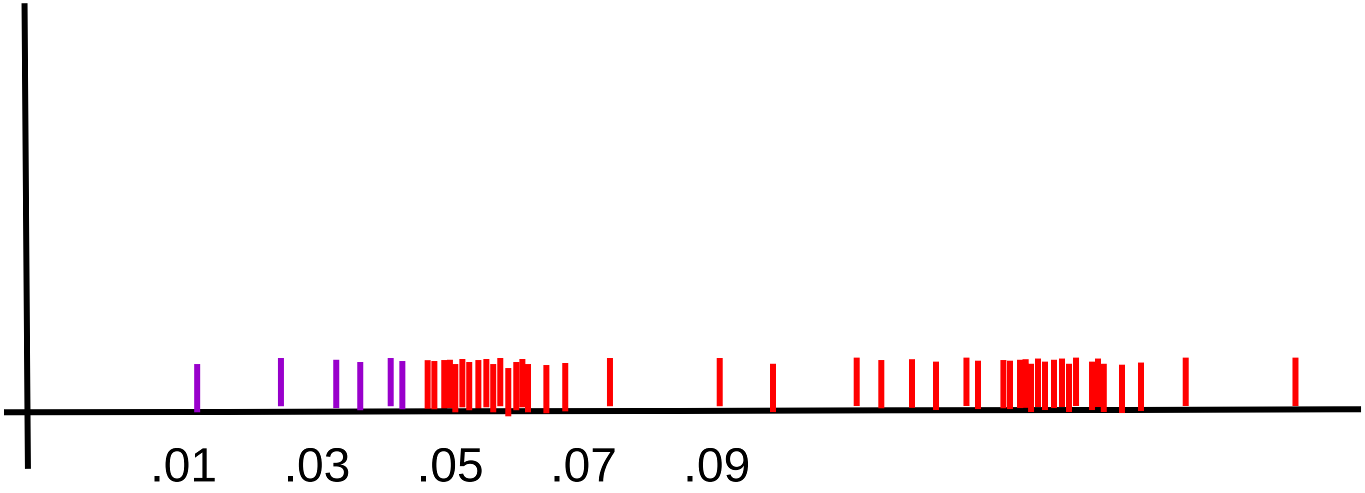


Slide by Daniel S. Weld

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

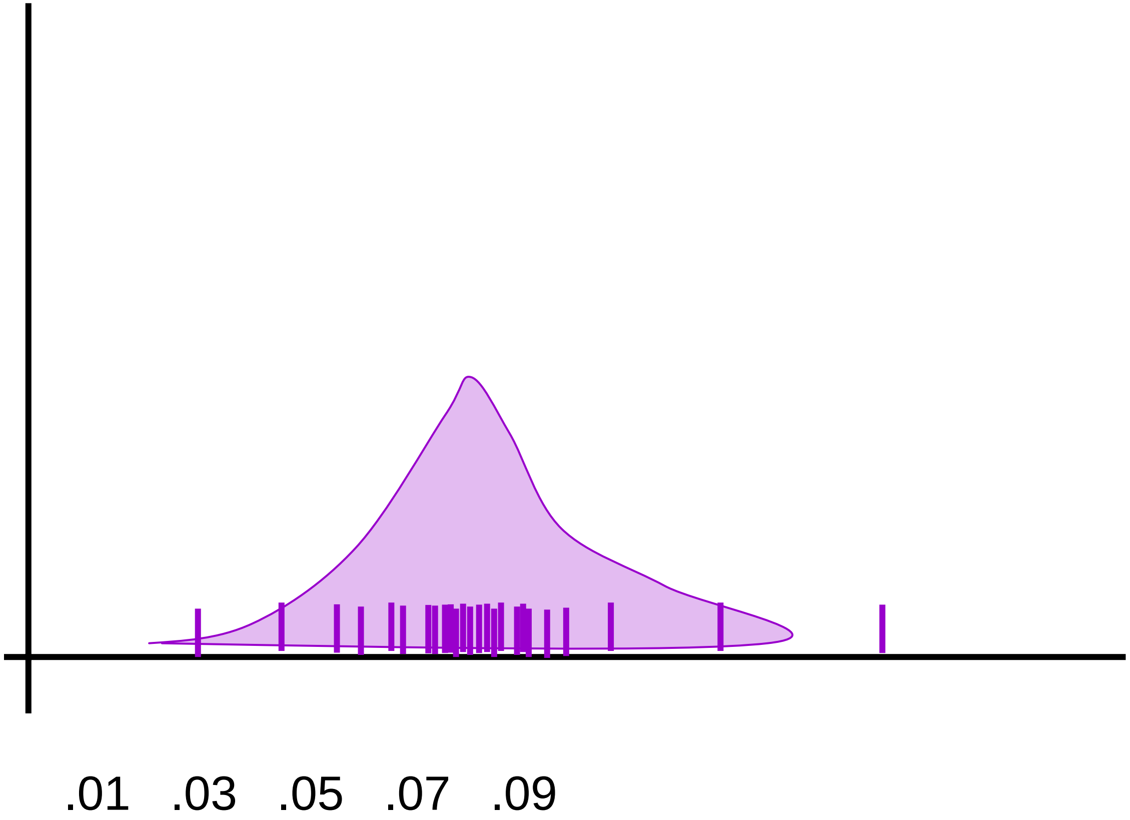
  [M step] Treating each instance as ***fractionally*** having **both** values compute the new parameter values



Slide by Daniel S. Weld

ML Mean of Single Gaussian

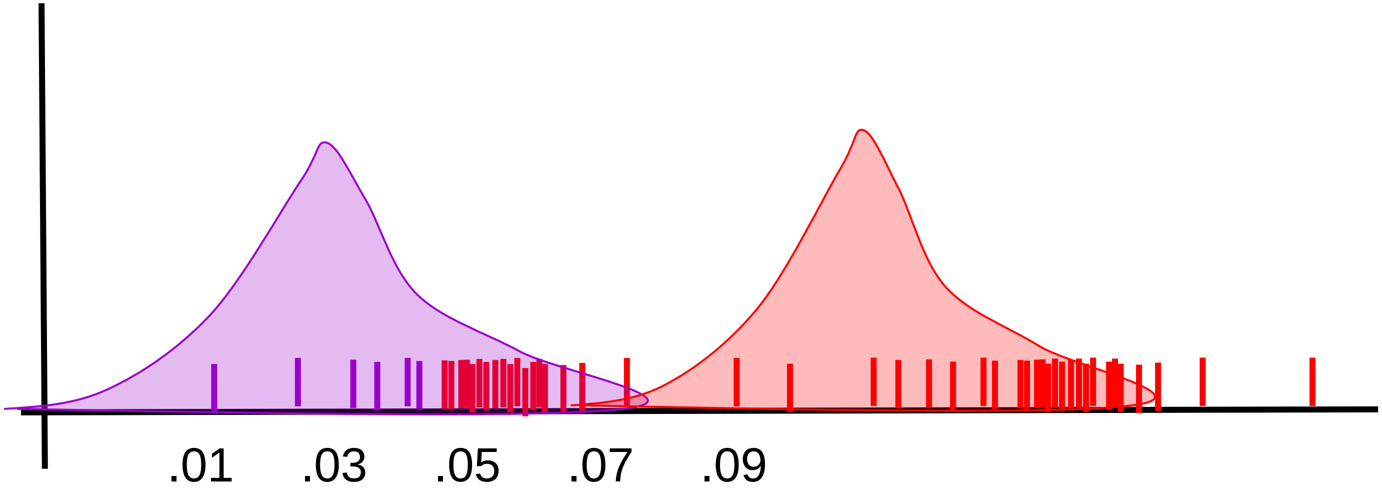
$$U_{ml} = \text{argmin}_u \sum_i (x_i - u)^2$$



Slide by Daniel S. Weld

Expectation Maximization (EM)

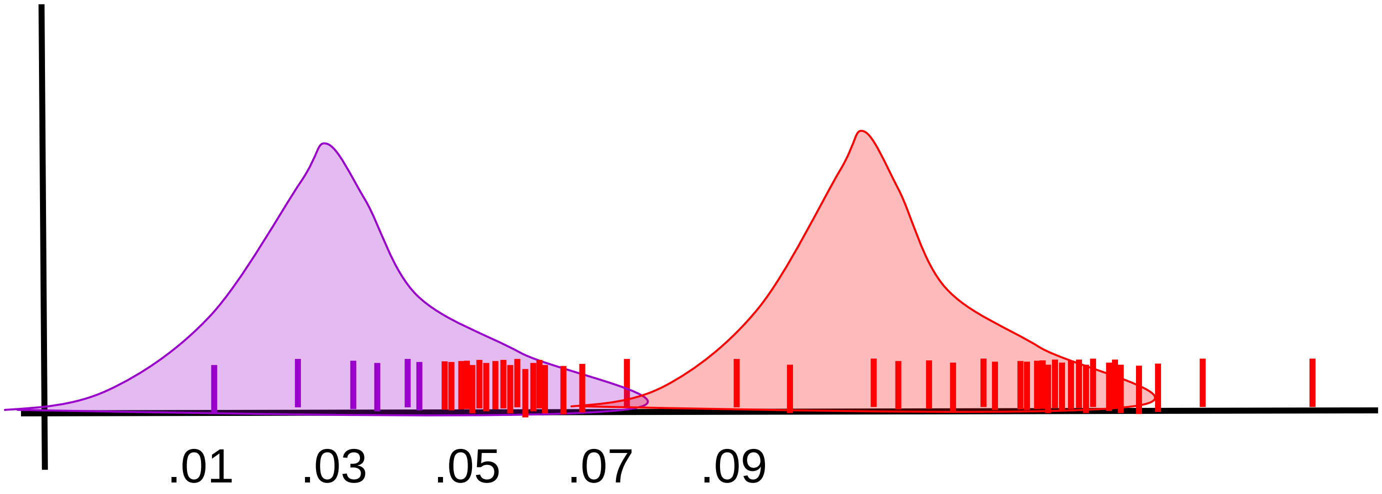
[M step] Treating each instance as fractionally having **both** values compute the new parameter values



Slide by Daniel S. Weld

Expectation Maximization (EM)

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

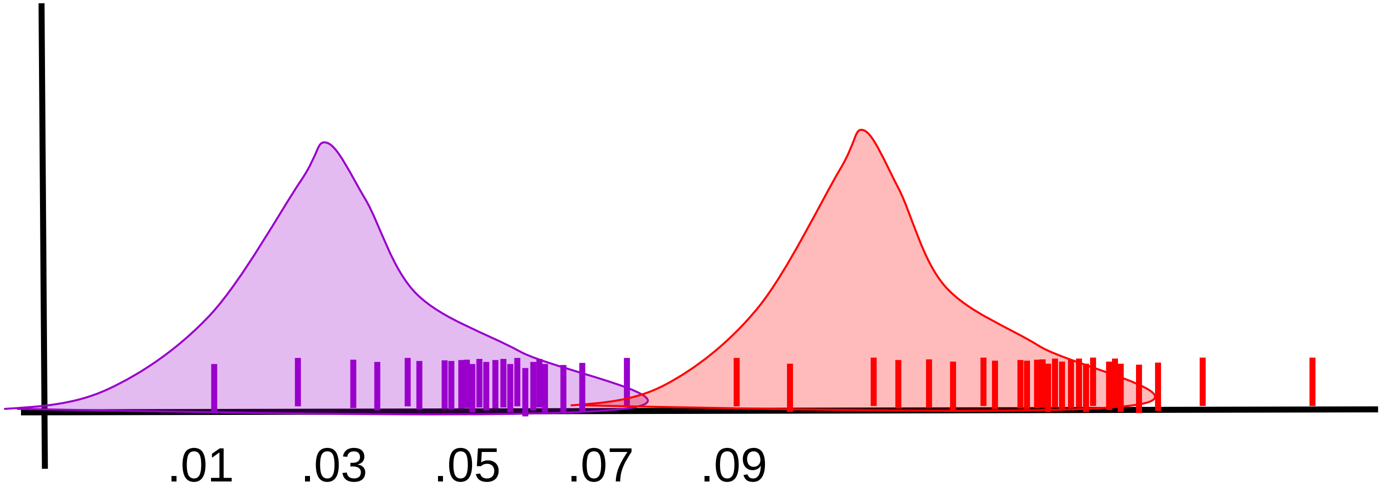


Slide by Daniel S. Weld

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

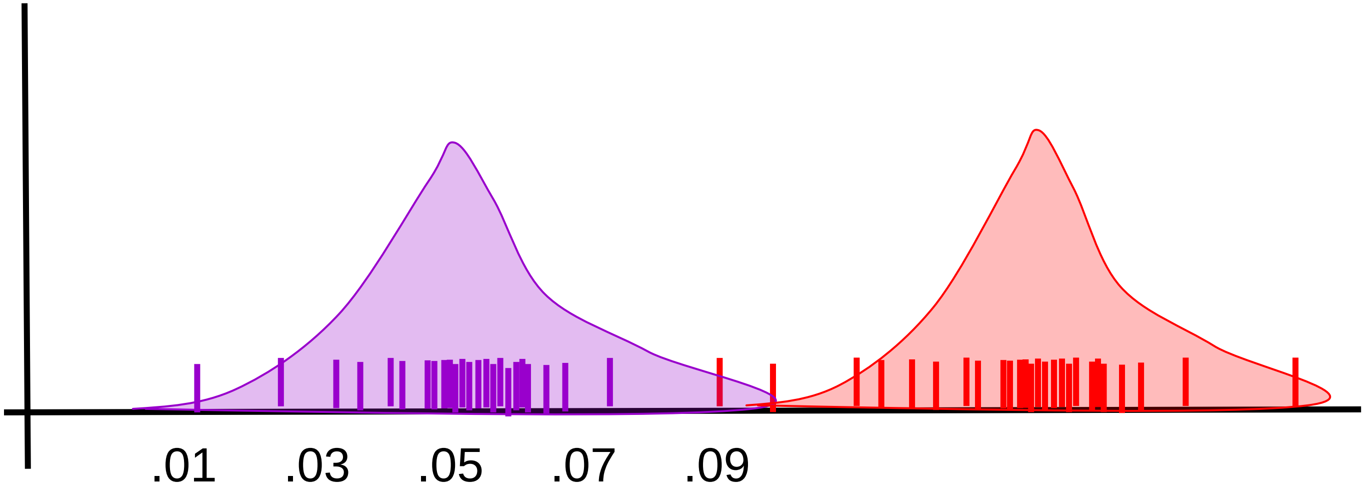


Slide by Daniel S. Weld

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

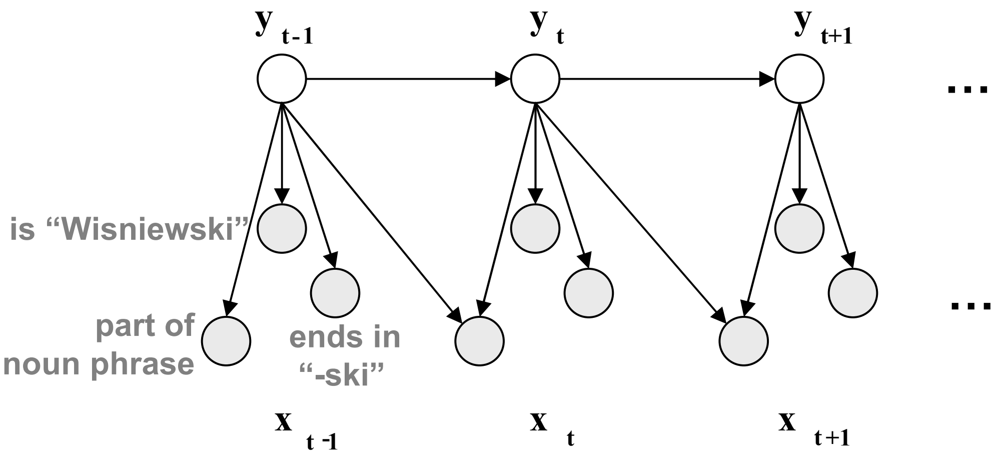


Slide by Daniel S. Weld

The Problem with HMMs

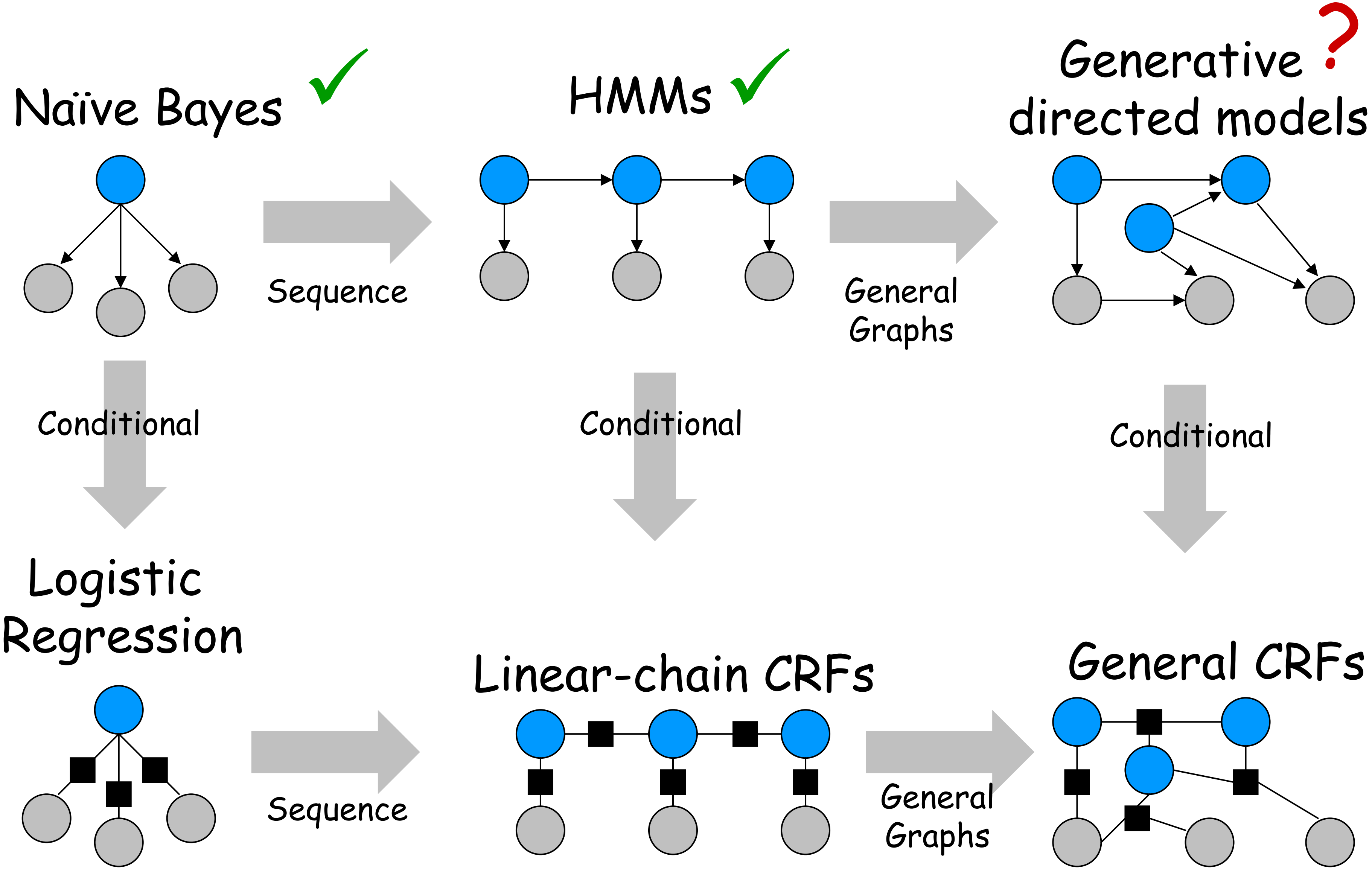
- We want more than an Atomic View of Words
- We want many arbitrary, overlapping features of words

identity of word
ends in "-ski"
is capitalized
is part of a noun phrase
is in a list of city names
is under node X in WordNet
is in bold font
is indented
is in hyperlink anchor
last person name was female
next two words are "and Associates"



Slide by Cohen & McCallum

Finite State Models



Naïve Bayes ✓

Sequence

HMMs ✓

General Graphs

Generative directed models ?

Conditional

Conditional

Conditional

Logistic Regression

Sequence

Linear-chain CRFs

General Graphs

General CRFs

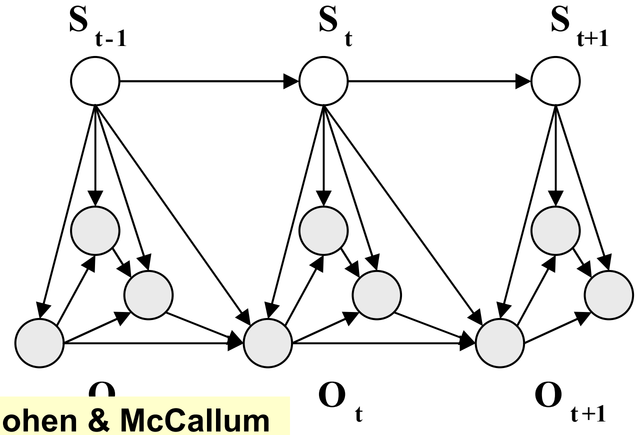
Problems with Richer Representation and a Joint Model

These arbitrary features are not independent.

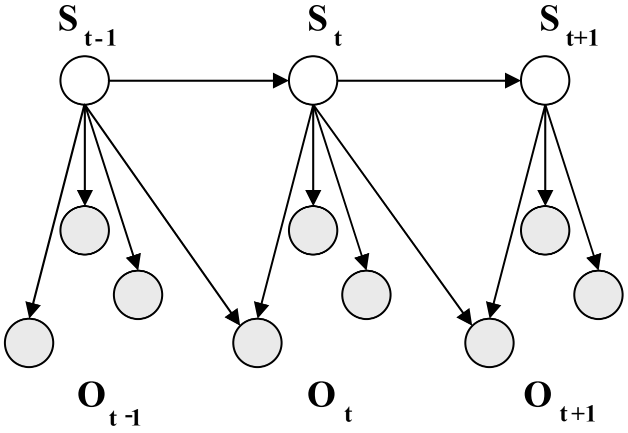
- Multiple levels of granularity (chars, words, phrases)
- Multiple dependent modalities (words, formatting, layout)
- Past & future

Two choices:

Model the dependencies. Each state would have its own Bayes Net. *But we are already starved for training data!*



Ignore the dependencies. This causes “over-counting” of evidence (ala naïve Bayes). *Big problem when combining evidence, as in Viterbi!*



Slide by Cohen & McCallum

Discriminative and Generative Models

- **So far: all models generative**
- **Generative Models …**

  model $P(x,y)$
- **Discriminative Models …**

  model $P(x|y)$

$P(x|y)$ 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 among y, but not among x.**

Conditional Sequence Models

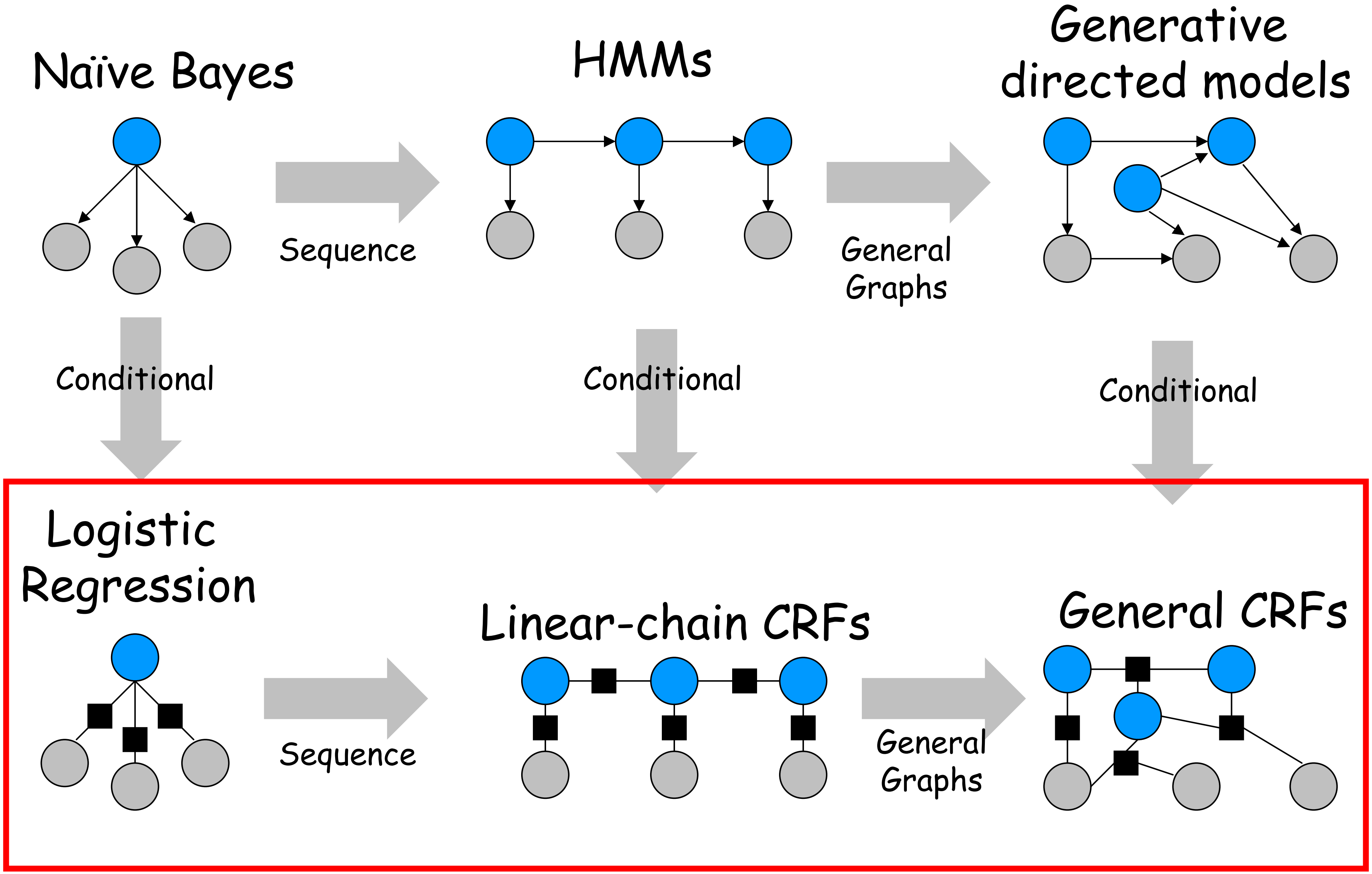
- **We prefer a model that is trained to maximize a *conditional* probability rather than *joint* probability:**

  P(y|x) instead of P(y,x):

  - Can examine features, but not responsible for generating them.
  - Don't have to explicitly model their dependencies.
  - Don't "waste modeling effort" trying to generate what we are given at test time anyway.

Slide by Cohen & McCallum

Finite State Models



Naïve Bayes

Sequence

HMMs

General Graphs

Generative directed models

Conditional

Conditional

Conditional

Logistic Regression

Sequence

Linear-chain CRFs

General Graphs

General CRFs

Linear-Chain Conditional Random Fields

- From HMMs to CRFs

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

can also be written as

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} exp \left(\sum_t \sum_{i,j \in S} \lambda_{ij} \mathbf{1}_{\{y_t = i\}} \mathbf{1}_{\{y_{t-1} = j\}} + \sum_t \sum_{i \in S} \sum_{o \in O} \mu_{oi} \mathbf{1}_{\{y_t = i\}} \mathbf{1}_{\{x_t = o\}} \right)$$

(set $\lambda_{ij} := \log p(y' = i | y = j)$, ...)

We let new parameters vary freely, so we need normalization constant Z.

Linear-Chain Conditional Random Fields

$$p(\mathbf{y},\mathbf{x}) = \frac{1}{Z} exp \left(\sum_t \sum_{i,j \in S} \lambda_{ij} \mathbf{1}_{\{y_t=i\}} \mathbf{1}_{\{y_{t-1}=j\}} + \sum_t \sum_{i \in S} \sum_{o \in O} \mu_{\right.$$

- **Introduce feature functions** $f_k(y_t, y_{t-1}, x_t$

One feature per transition

One feature per state-observati

$$(f_{ij}(y, y', x_t) := \mathbf{1}_{y=i} \mathbf{1}_{y'=j} \;, \quad f_{io}(y, y', x_t) := \mathbf{1}_{y=i} \mathbf{1}_{x=o}$$

$$p(\mathbf{y},\mathbf{x}) = \frac{1}{Z} exp \left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t) \right)$$

- **Then the conditional distribution is**

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y},\mathbf{x})}{\sum_{y'} p(\mathbf{y}',\mathbf{x})} = \frac{exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right)}{\sum_{y'} exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right)}$$

This is a linear-chain CRF, but includes only current word's identity as a feature

Linear-Chain Conditional Random Fields

- Conditional $p(y|x)$ that follows from joint $p(y,x)$ of HMM is a linear CRF with certain feature functions!

Linear-Chain Conditional Random Fields

- Definition:

  A *linear-chain CRF* is a distribution that takes the form

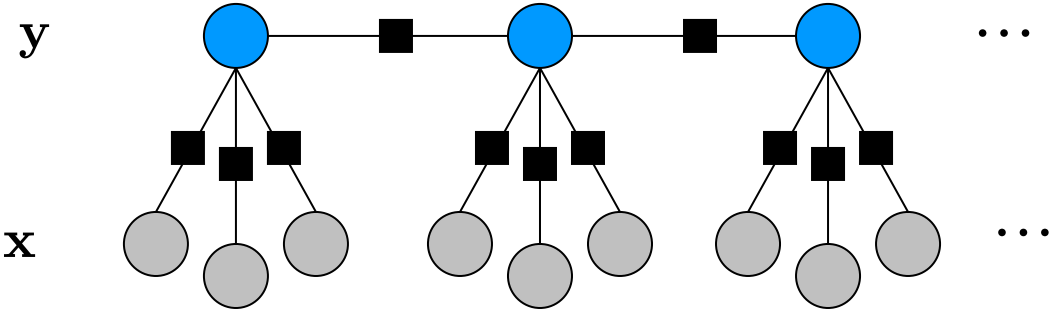
$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} exp\left(\sum_{k=1}^{K} \overset{\substack{\text{parameters}\\ \downarrow}}{\lambda_k} \overset{\substack{\text{feature functions}\\ \downarrow}}{f_k}(y_t, y_{t-1}, \mathbf{x}_t)\right)$$

where Z(x) is a normalization function

$$Z(x) = \sum_{\mathbf{y}} exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right)$$

Linear-Chain
Conditional Random Fields

- HMM-like linear-chain CRF



- Linear-chain CRF, in which transition score depends on the current observation