Hidden Markov Models

Chapter 15

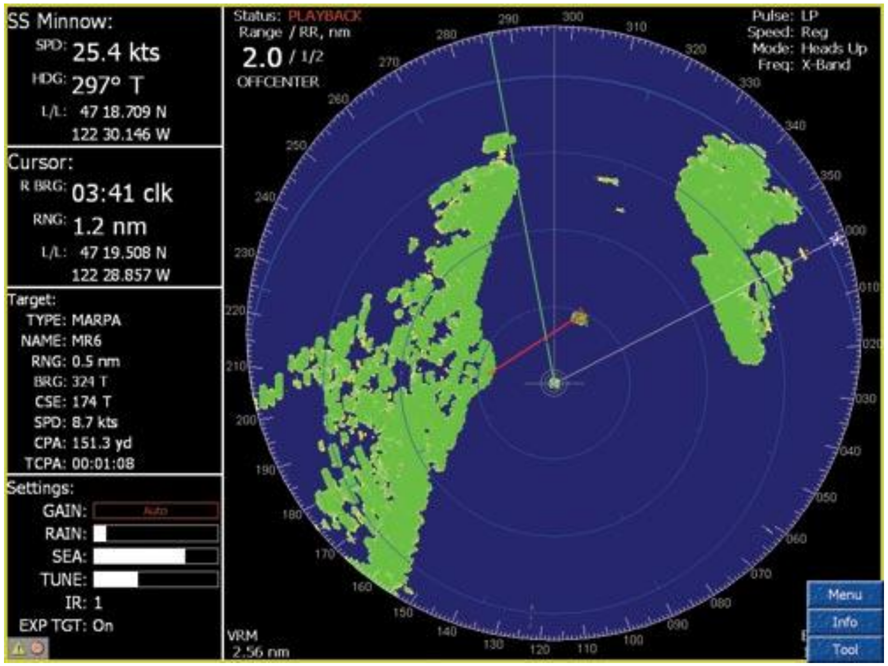
Mausam

(Slides based on Dan Klein, Luke Zettlemoyer, Alex Simma, Erik Sudderth, David Fernandez-Baca, Drena Dobbs, Serafim Batzoglou, William Cohen, Andrew McCallum, Dan Weld)

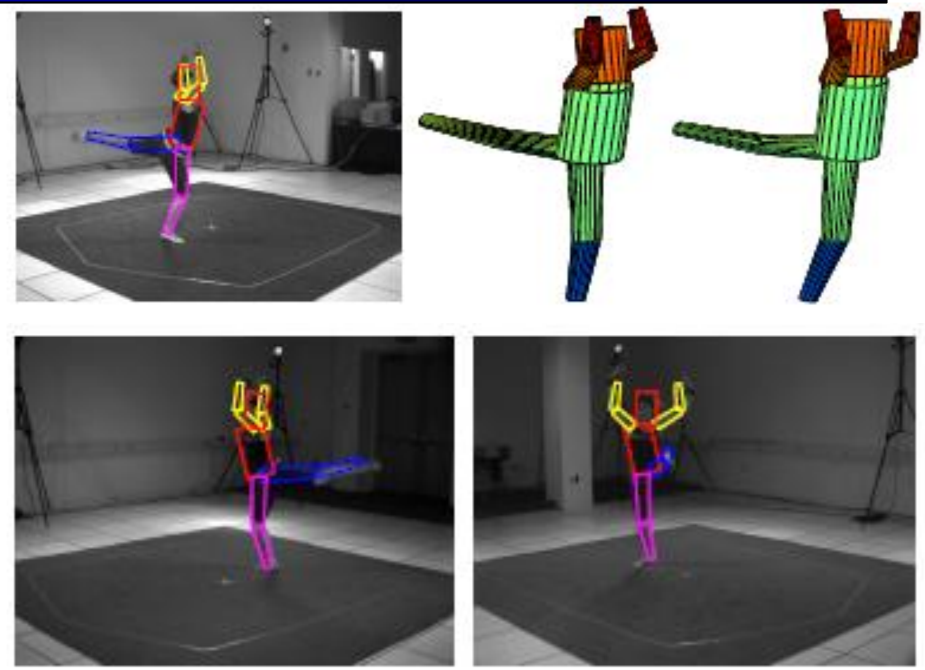
Temporal Models

- Graphical models with a temporal component
- S_t/X_t = set of unobservable variables at time t
- W_t/Y_t = set of evidence variables at time t
- Notation $X_{a:b} = X_a, X_{a+1}, \ldots, X_b$

Target Tracking



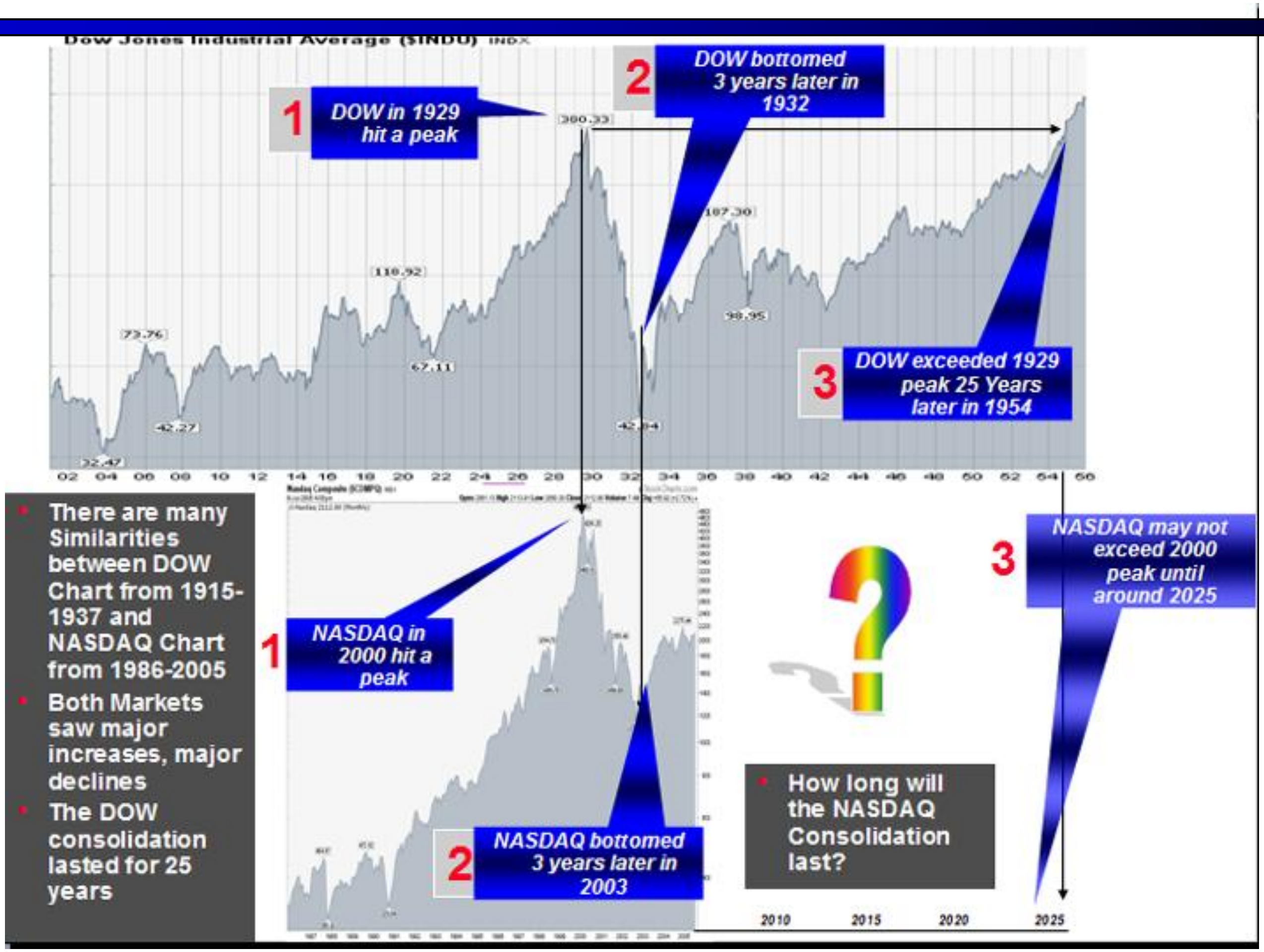
Radar-based tracking of multiple targets



Visual tracking of articulated objects
(L. Sigal et. al., 2006)

- Estimate motion of targets in 3D world from indirect, potentially noisy measurements

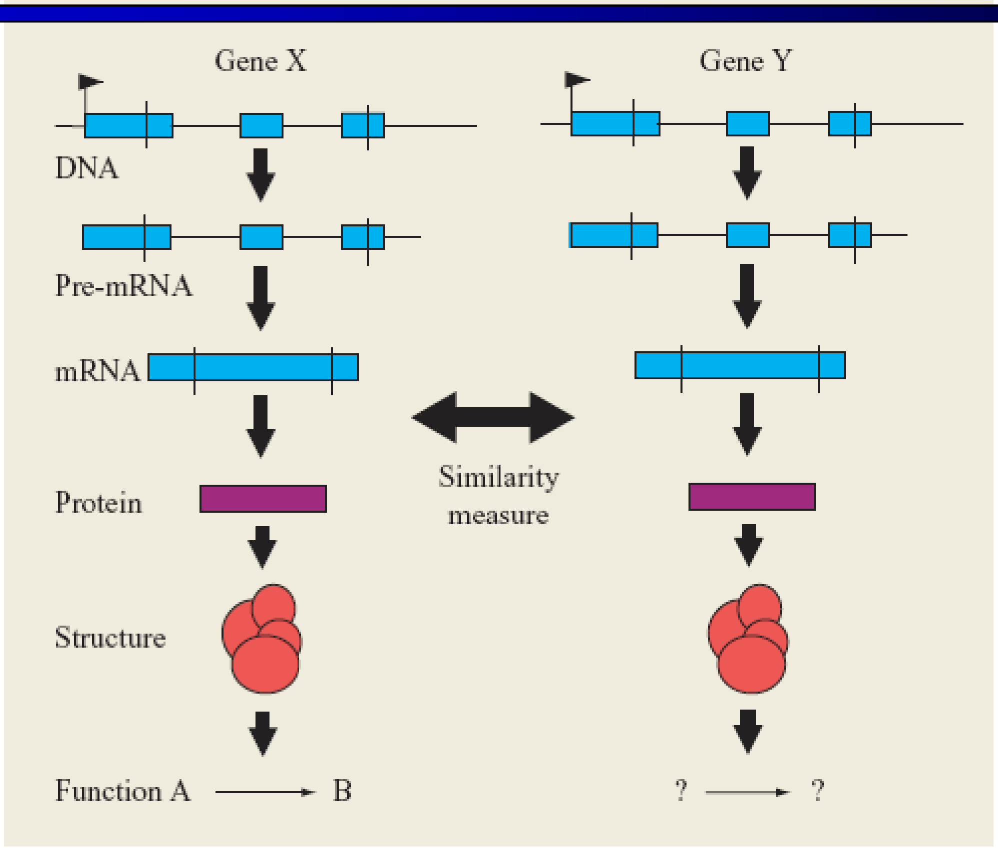
Financial Forecasting



http://www.steadfastinvestor.com/

- Predict future market behavior from historical data, news reports, expert opinions, …

Biological Sequence Analysis

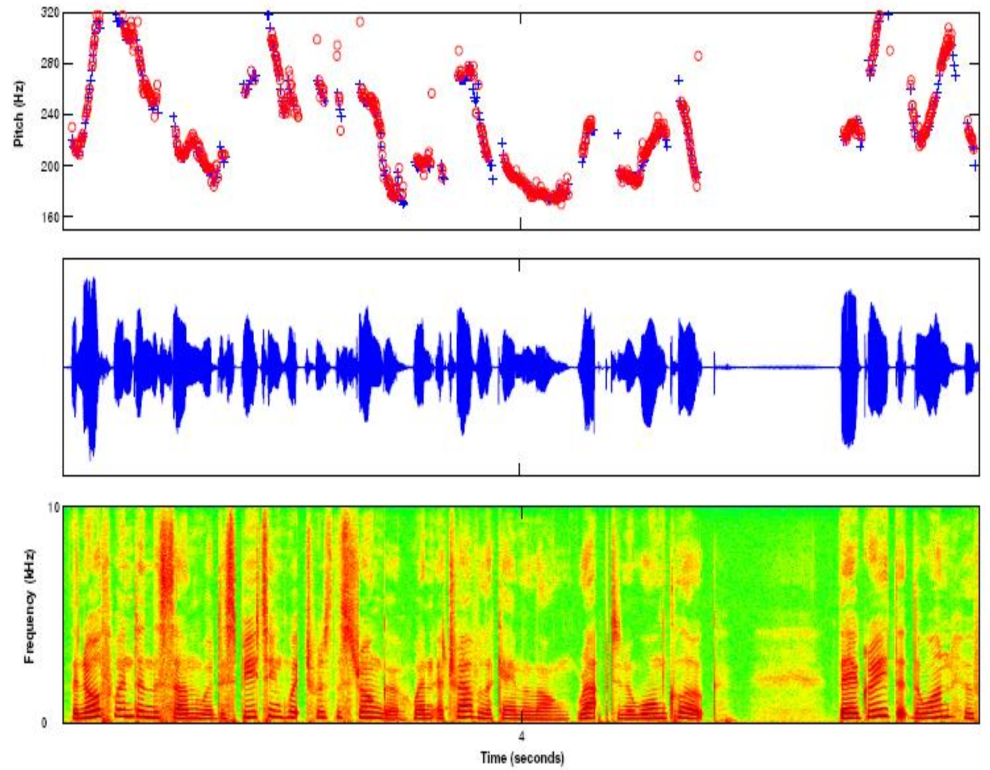


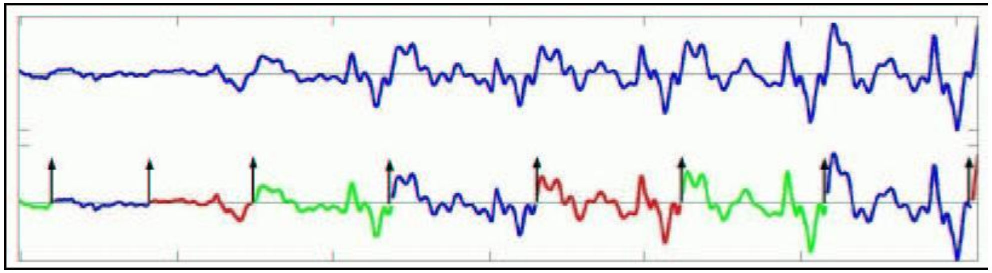
(E. Birney, 2001)

- Temporal models can be adapted to exploit more general forms of *sequential* structure, like those arising in DNA sequences

Speech Recognition

- Given an audio waveform, would like to robustly extract & recognize any spoken words
- Statistical models can be used to
  - Provide greater robustness to noise
  - Adapt to accent of different speakers
  - Learn from training

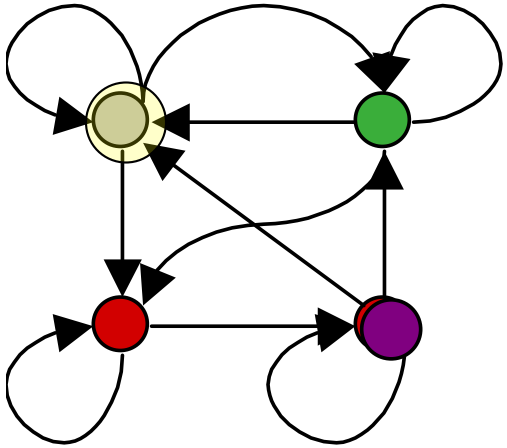




S. Roweis, 2004

Markov Chain

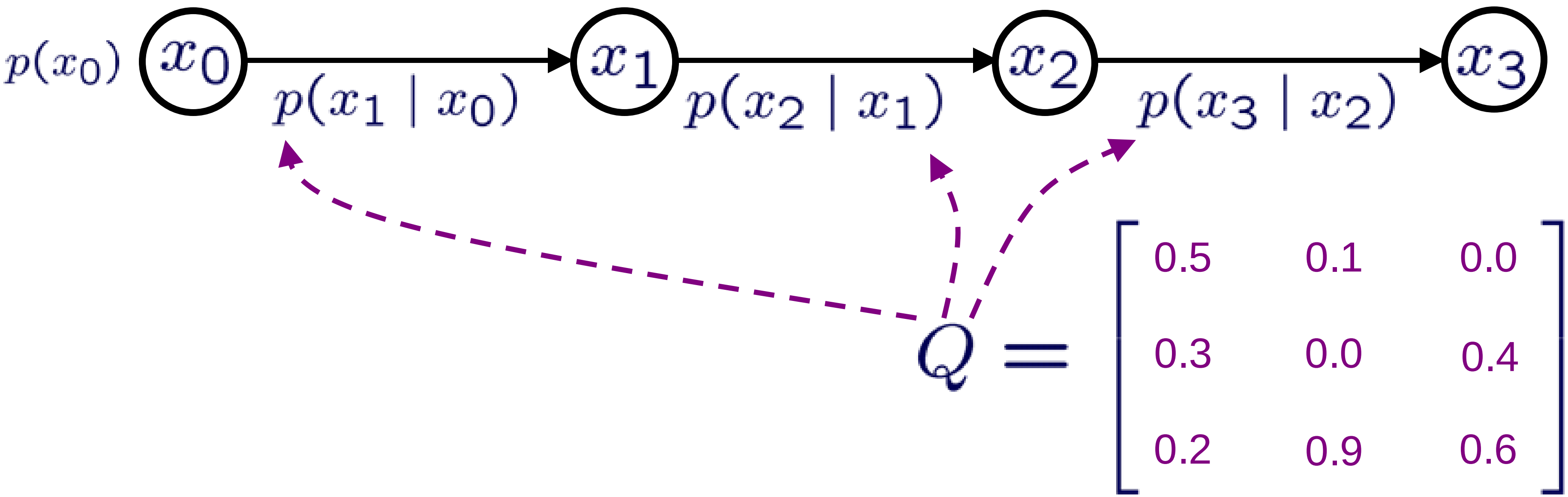
- Set of states
  - Initial probabilities
  - Transition probabilities



Markov Chain models system dynamics

Markov Chains: Graphical Models

$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1})$$

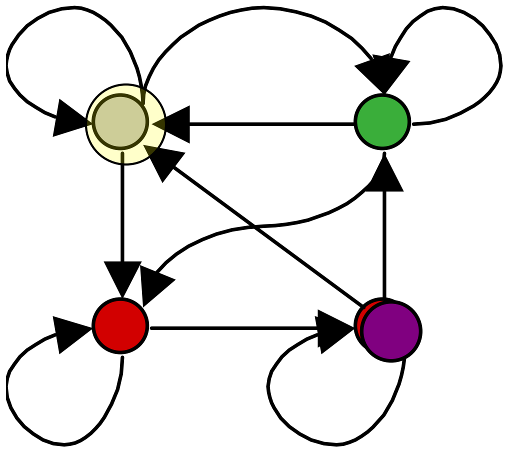


Difference from a Markov Decision Process?

it is a system that transitions by itself

Hidden Markov Model

- Set of states
  - Initial probabilities
  - Transition probabilities
- Set of potential observations
  - Emission/Observation probabilities



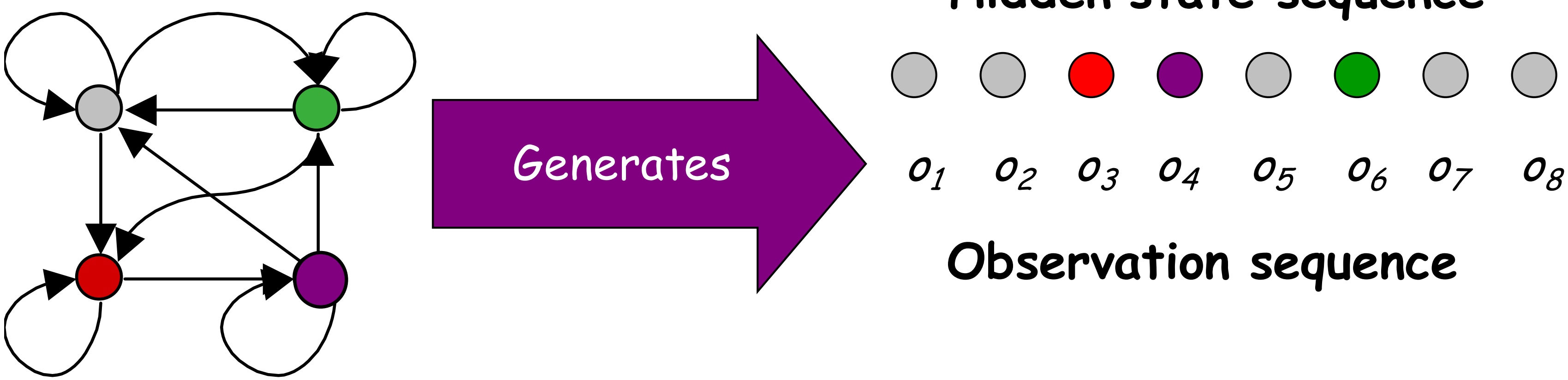
o_1 o_2 o_3 o_4 o_5

HMM generates observation sequence

Hidden Markov Models (HMMs)

Finite state machine

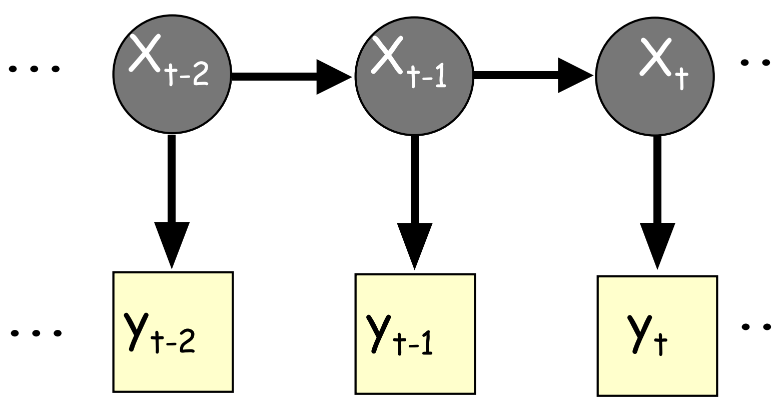
Hidden state sequence



Graphical Model

Hidden states

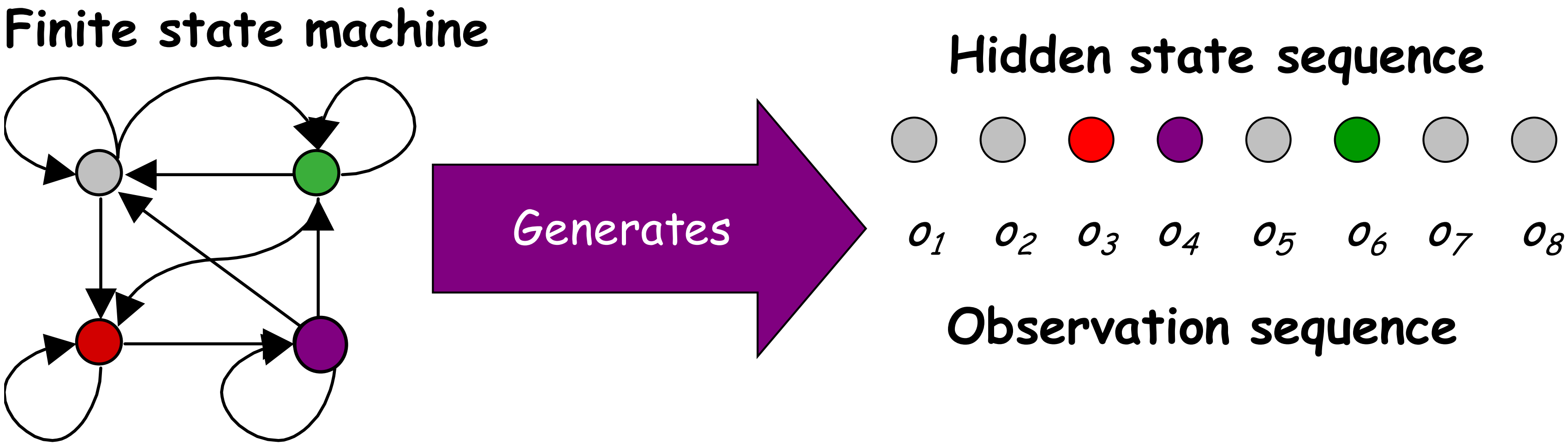
Observations



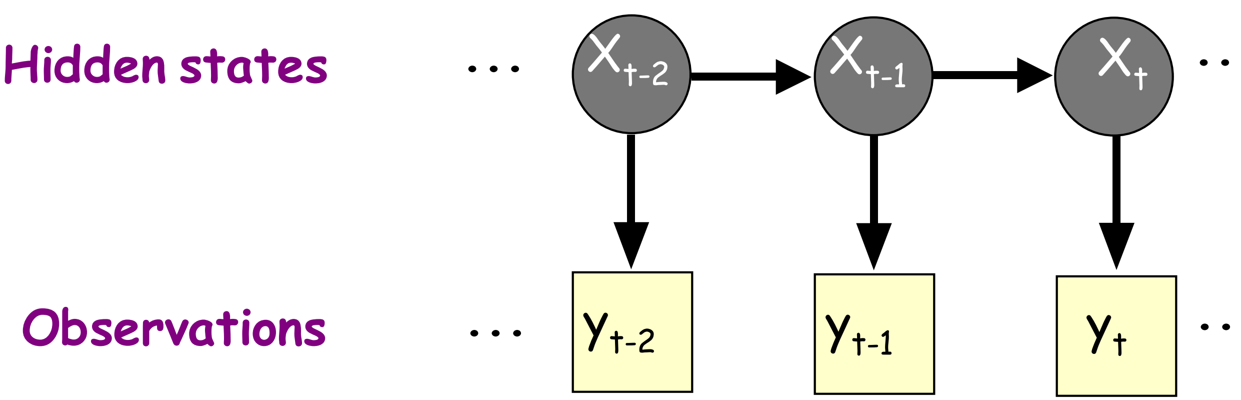
Random variable X_t takes values from $\{s_1, s_2, s_3, s_4\}$

Random variable y_t takes values from $\{o_1, o_2, o_3, o_4, o_5, \ldots\}$

HMM



Graphical Model



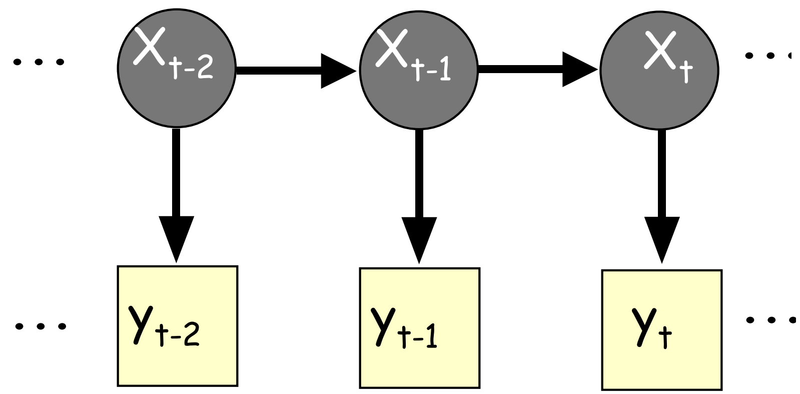
Random variable X_t takes values from $\{s_1, s_2, s_3, s_4\}$

Random variable y_t takes values from $\{o_1, o_2, o_3, o_4, o_5, \ldots\}$

HMM

Graphical Model

Hidden states



Random variable y_t takes values from $\{s_1, s_2, s_3, s_4\}$

Observations

Random variable x_t takes values from $\{o_1, o_2, o_3, o_4, o_5, \ldots\}$

Need Parameters:

Start state probabilities: $P(x_1=s_k)$

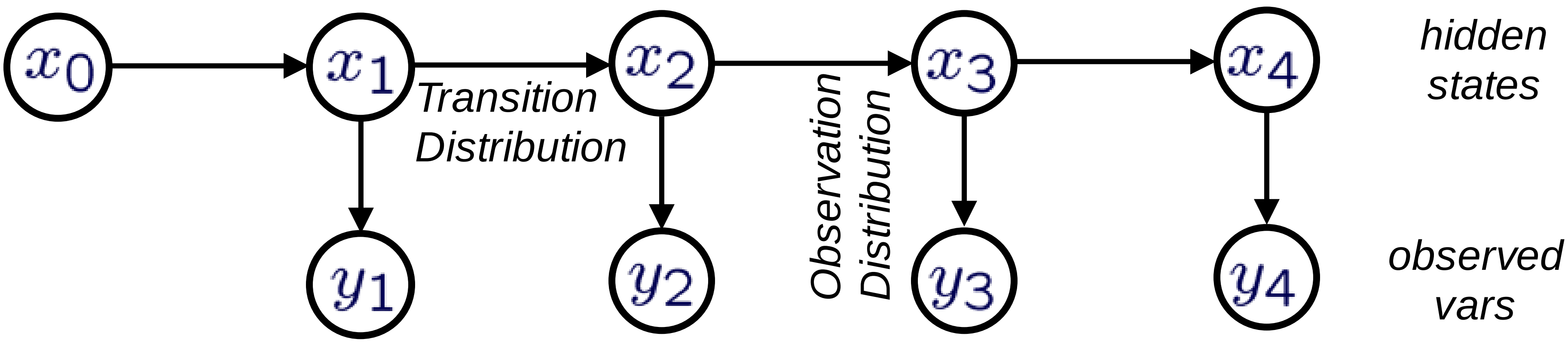
Transition probabilities: $P(x_t=s_i \mid x_{t-1}=s_k)$

Observation probabilities: $P(y_t=o_j \mid x_t=s_k)$

Hidden Markov Models

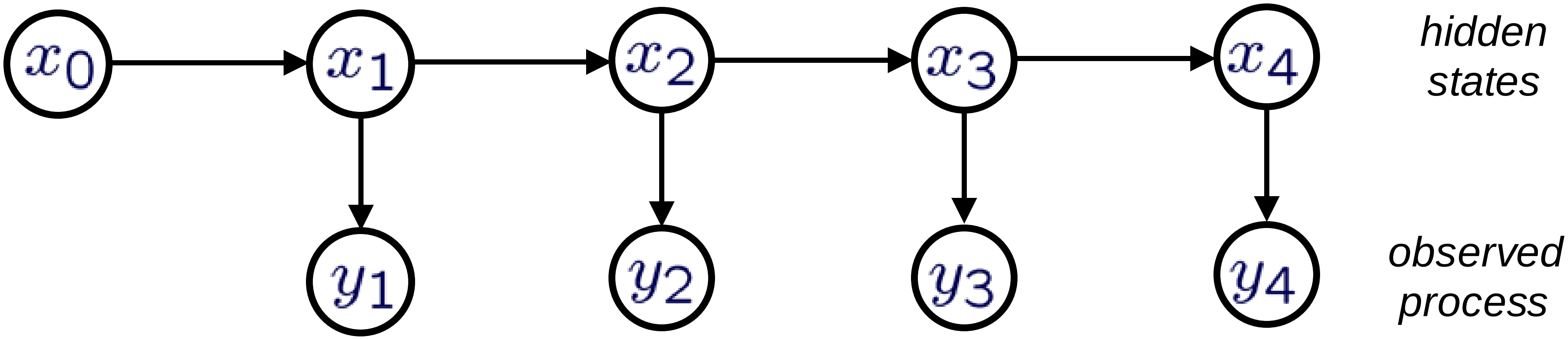
- Just another graphical model…

"Conditioned on the present,
the past & future are independent"



$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1}) p(y_t \mid x_t)$$

Hidden states



- Given x_t, earlier observations provide no *additional information* about the future:

$$p(y_t, y_{t+1}, \ldots \mid x_t, y_{t-1}, y_{t-2}, \ldots) = p(y_t, y_{t+1}, \ldots \mid x_t)$$

HMM Generative Process

- We can easily sample sequences pairs:

  $$\mathbf{X_{0:n}, Y_{0:n} = S_{0:n}, W_{0:n}}$$

- Sample initial state: $P(x_0)$
- For i = 1 ... n
  - Sample $\mathbf{s_i}$ from the distribution $P(s_i|s_{i-1})$
  - Sample $\mathbf{w_i}$ from the distribution $P(w_i|s_i)$

Example: POS Tagging

- Useful as a pre-processing step

DT	NNP	NN	VBD	VBN	RP	NN	NNS
The	Georgia	branch	had	taken	on	loan	commitments …

DT	NN	IN	NN	VBD	NNS	VBD
The	average	of	interbank	offered	rates	plummeted …

- Setup:
  - states S = {DT, NNP, NN, ... } are the POS tags
  - Observations W = V are words
  - Transition dist'n $P(s_i|s_{i-1})$ models the tag sequences
  - Observation dist'n $P(w_i|s_i)$ models words given their POS

Example: Chunking

- Find spans of text with certain properties
- For example: named entities with types
  - (PER, ORG, or LOC)

- Germany 's representative to the European Union's veterinary committee Werner Zwingman said on Wednesday consumers should ...

- [Germany]$_{LOC}$'s representative to the [European Union]$_{ORG}$'s veterinary committee [Werner Zwingman]$_{PER}$ said on Wednesday consumers should ...

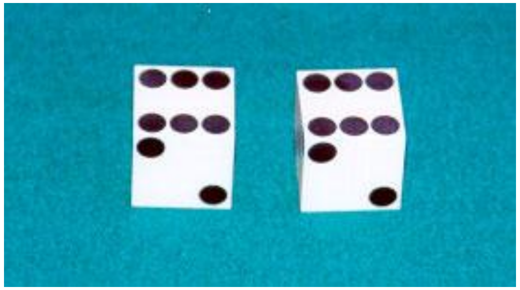
Example: Chunking

- [Germany]LOC ’s representative to the [European Union]ORG ‘s veterinary committee [Werner Zwingman]PER said on Wednesday consumers should ...

- Germany/BL ’s/NA representative/NA to/NA the/NA European/BO Union/CO ‘s/NA veterinary/NA committee/NA Werner/BP Zwingman/CP said/NA on/NA Wednesday/NA consumers/NA should/NA ...

- HMM Model:
  - States $S = \{NA, BL, CL, BO, CO, BL, CL\}$ *represent* beginnings (BL, BO, BP} and continuations (CL, CO, CP) of chunks, and other (NA)
  - Observations $W = V$ *are words*
  - Transition dist’n $P(s_i \mid s_{i-1})$ *models the tag sequences*
  - Observation dist’n $P(w_i \mid s_i)$ *models words given their type*

Example: The Occasionally Dishonest Casino



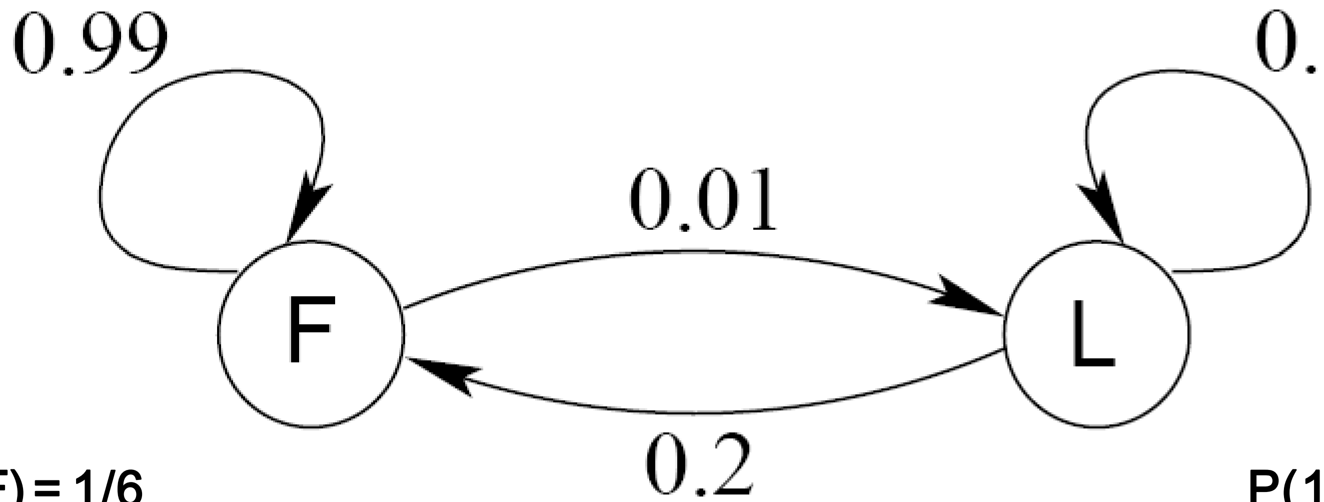
A casino has two dice:

- 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) = 1/2
- Dealer switches between dice as:
  - Prob(Fair $\rightarrow$ Loaded) = 0.01
  - Prob(Loaded $\rightarrow$ Fair) = 0.2
  - Transitions between dice obey a Markov process

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

An HMM for the occasionally dishonest casino



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

124552646214614613613666166466163661636616361…

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

124552646214614613613666166466163661636616361651561511514612356234…

QUESTION

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

This is the DECODING question in HMMs

Question # 3 – Learning

GIVEN

A sequence of rolls by the casino player

124552646214614613613666166466163661636616361651…

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 question in HMMs

HMM Inference

- Evaluation: prob. of observing an obs. sequence
  - Forward Algorithm (very similar to Viterbi)
- Decoding: most likely sequence of hidden states
  - Viterbi algorithm
- Marginal distribution: prob. of a particular state
  - Forward-Backward

Decoding Problem

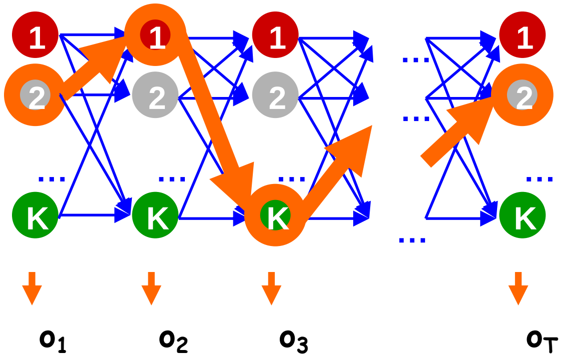
Given $w = w_1 \ldots w_n$ and HMM θ, what is "best" parse $s_1 \ldots s_n$?

Several possible meanings of 'solution'

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

We want ***sequence*** $s_1 \ldots s_n$,
such that $P(s|w)$ is maximized

$$s^* = \text{argmax}_s \, P(\, s \,|\, w \,)$$



Most Likely Sequence

- Problem: find the most likely (Viterbi) sequence under the model

$$s^*_{0:n} = \arg\max_{s_{0:n}} P(s_{0:n}|w_{0:n})$$

- Given model parameters, we can score any sequence pair

NNP	VBZ	NN	NNS	CD	NN	.
Fed	raises	interest	rates	0.5	percent	.

$P(\mathbf{S_{0:n}}, \mathbf{W_{0:n}}) = P(NNP|\phi)\, P(Fed|NNP)\, P(VBZ|NNP)\, P(raises|VBZ)\, P(NN|NNP)$…..

- In principle, we're done – list all possible tag sequences, score each one, pick the best one (the Viterbi state sequence)

NNP VBZ NN NNS CD NN ⇨ logP = -23

NNP NNS NN NNS CD NN ⇨ logP = -29

NNP VBZ VB NNS CD NN ⇨ logP = -27

2n multiplications per sequence

$|S|^n$ state sequences!

The occasionally dishonest casino

- Known:
  - The structure of the model
  - The transition probabilities
- Hidden: What the casino did
  - FFFFFLLLLLLLFFFF...
- Observable: The series of die tosses
  - 3415256664666153...
- What we must infer:
  - When was a fair die used?
  - When was a loaded one used?
    - The answer is a sequence FFFFFFFLLLLLLFFF...

The occasionally dishonest casino

$$w = \langle w_1, w_2, w_3 \rangle = \langle 6,2,6 \rangle$$

$$s^{(2)} = FFF$$

$$\Pr(w, s^{(1)}) = p(F \mid 0)\, p(6 \mid F)\, p(F \mid F)\, p(2 \mid F)\, p(F \mid F)\, p(6 \mid F)$$

$$= 0.5 \times \frac{1}{6} \times 0.99 \times \frac{1}{6} \times 0.99 \times \frac{1}{6}$$

$$\approx 0.00227$$

$$s^{(2)} = LLL$$

$$\Pr(w, s^{(2)}) = p(L \mid 0)\, p(6 \mid L)\, p(L \mid L)\, p(2 \mid L)\, p(L \mid L)\, p(6 \mid L)$$

$$= 0.5 \times 0.5 \times 0.8 \times 0.1 \times 0.8 \times 0.5$$

$$= 0.008$$

$$s^{(3)} = LFL$$

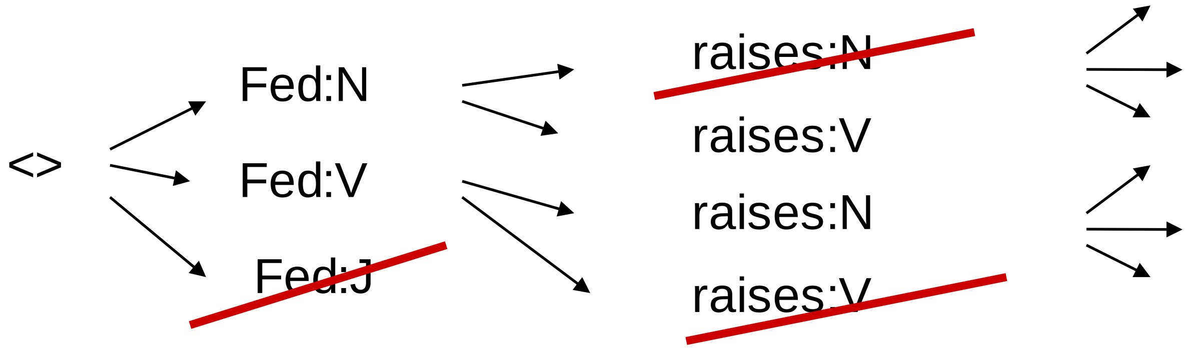
$$\Pr(w, s^{(3)}) = p(L \mid 0)\, p(6 \mid L)\, p(F \mid L)\, p(2 \mid F)\, p(L \mid F)\, p(6 \mid L)$$

$$= 0.5 \times 0.5 \times 0.2 \times \frac{1}{6} \times 0.01 \times 0.5$$

$$\approx 0.0000417$$

Finding the Best Trajectory

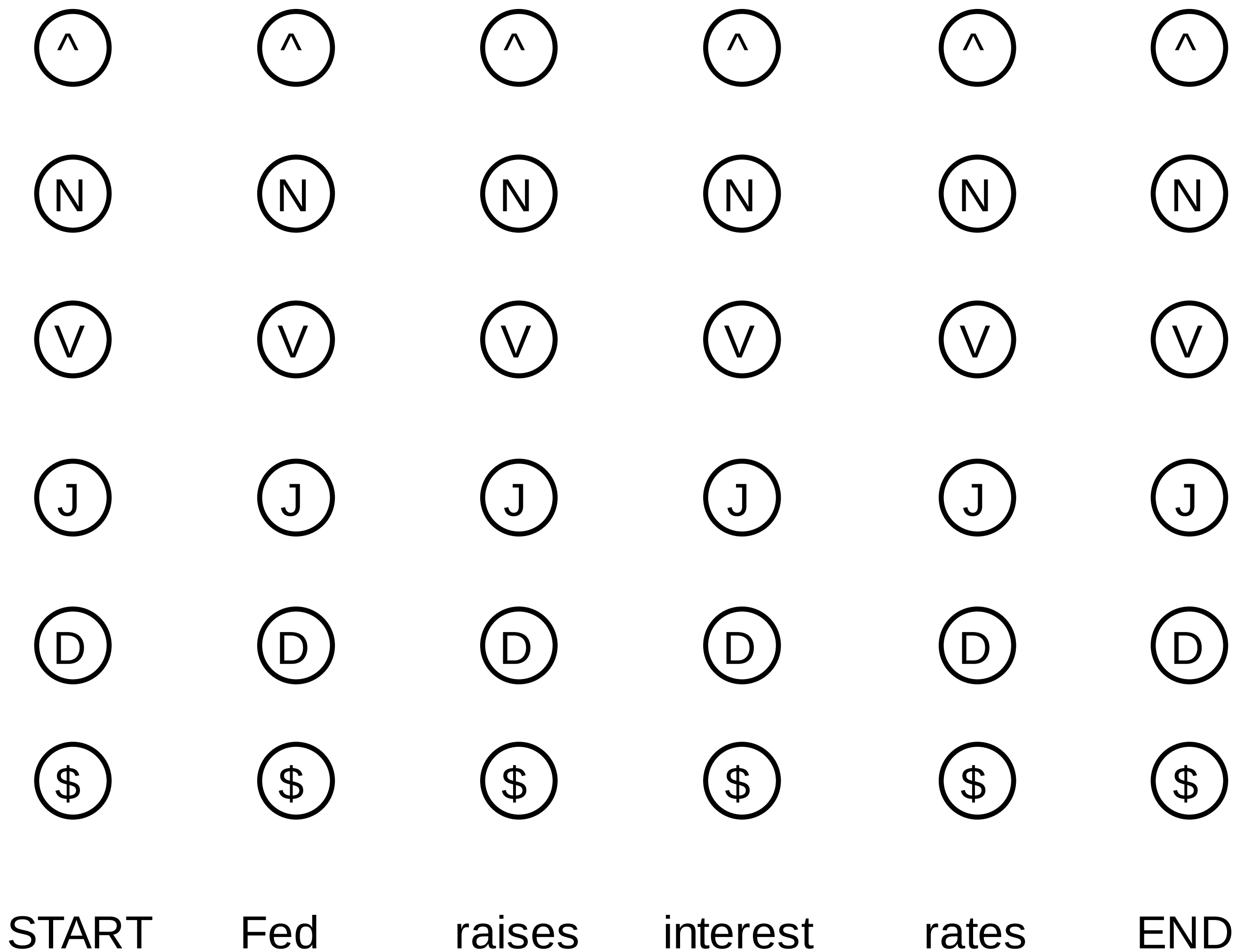
- Too many trajectories (state sequences) to list
- Option 1: Beam Search



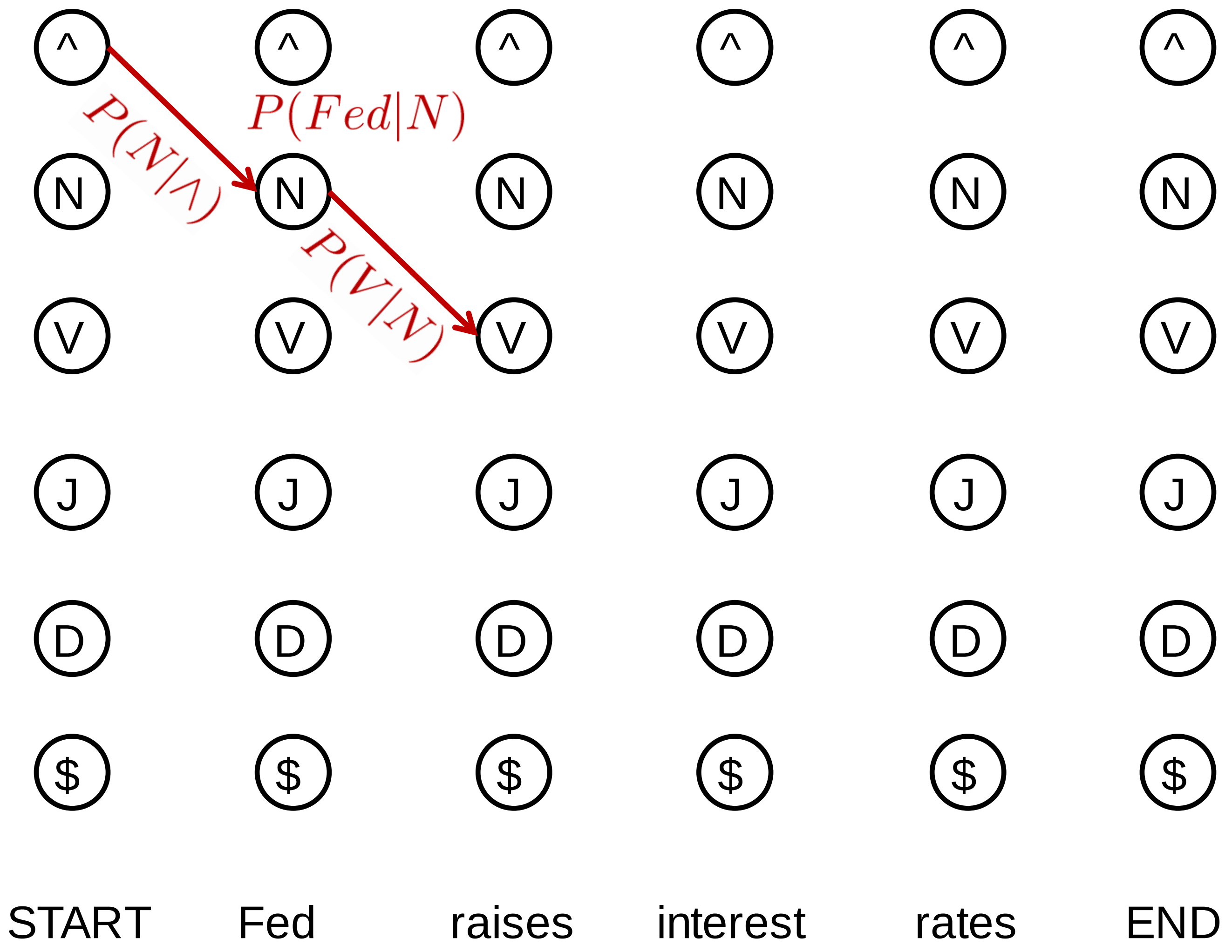
  - A beam is a set of partial hypotheses
  - Start with just the single empty trajectory
  - At each derivation step:
    - Consider all continuations of previous hypotheses
    - Discard most, keep top k

- Beam search works ok in practice
  - … but sometimes you want the optimal answer
  - … and there's usually a better option than naïve beams

The State Lattice / Trellis



The State Lattice / Trellis



Dynamic Programming

$$s^*_{0:n} = \arg\max_{s_{0:n}} P(s_{0:n}|w_{0:n}) = \arg\max_{s_{0:n}} P(s_{0:n}, w_{0:n})$$

First, consider how to compute the max:

Define: $$\delta_i(s) = \max_{s_{0:i-1}} P(s_{0:i-1}, s, w_{0:i})$$

Then:

$$\delta_i(s_i) =$$

$$=$$

$$=$$

$\delta_i(s)$: probability of ***most likely*** state sequence ending with state s, given observations $w_1, \ldots, w_i$

Dynamic Programming

$$s^*_{0:n} = \arg\max_{s_{0:n}} P(s_{0:n}|w_{0:n}) = \arg\max_{s_{0:n}} P(s_{0:n}, w_{0:n})$$

First, consider how to compute the max:

Define: $$\delta_i(s) = \max_{s_{0:i-1}} P(s_{0:i-1}, s, w_{0:i})$$

Then:

$$\delta_i(s_i) = \max_{s_{0:i-1}} P(w_i|s_i)P(s_i|s_{i-1})P(s_{0:i-1}, w_{0:i--1})$$

$$= P(w_i|s_i) \max_{s_{i-1}} P(s_i|s_{i-1}) \max_{s_{0:i-2}} P(s_{0:i-1}, w_{0:i-1})$$

$$= P(w_i|s_i) \max_{s_{i-1}} P(s_i|s_{i-1})\delta_{i-1}(s_{i-1})$$

$\delta_i(s)$: probability of ***most likely*** state sequence ending with state s, given observations $w_1, \ldots, w_i$

Vitterbi Algorithm

- Dynamic program for computing (for all i)

$$\delta_i(s) = \max_{s_{0:i-1}} P(s_{0:i-1}, s, w_{0:i})$$

  - The score of a best path up to position i ending in state s

$$\delta_0(s_0) = \begin{cases} 1 & \text{if } s_0 = START \\ 0 & \text{otherwise} \end{cases}$$

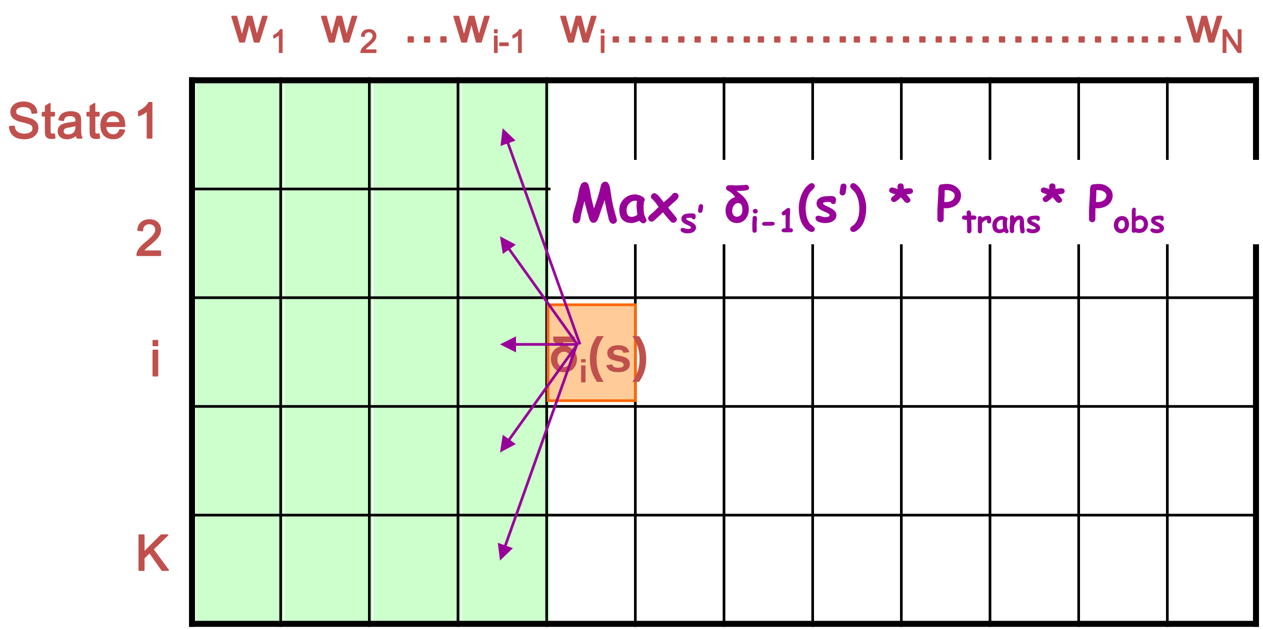
For $i = 1 \ldots n$

$$\delta_i(s) = \max_{s'} P(s \mid s')P(w \mid s)\delta_{i-1}(s')$$

  - Also store a backtrace

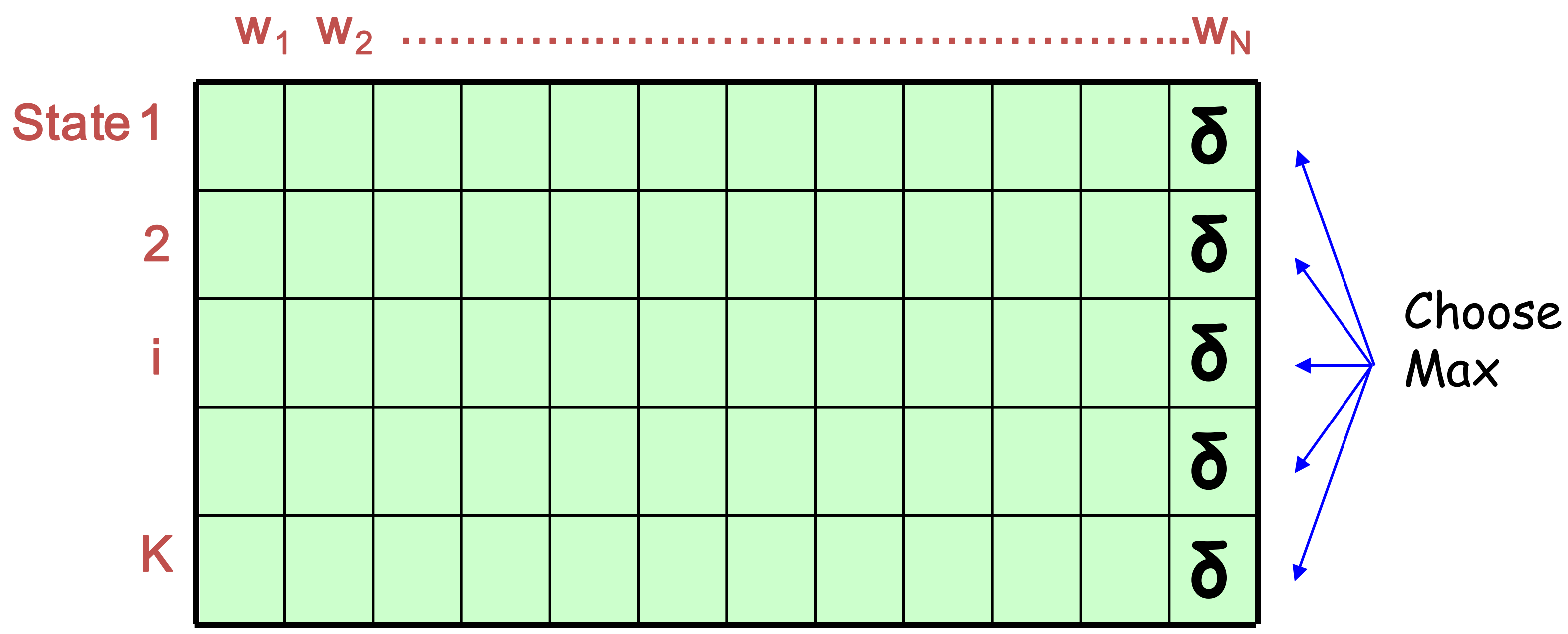
$$\psi_i(s) = \arg\max_{s'} P(s \mid s')P(w \mid s)\delta_{i-1}(s')$$

The Viterbi Algorithm

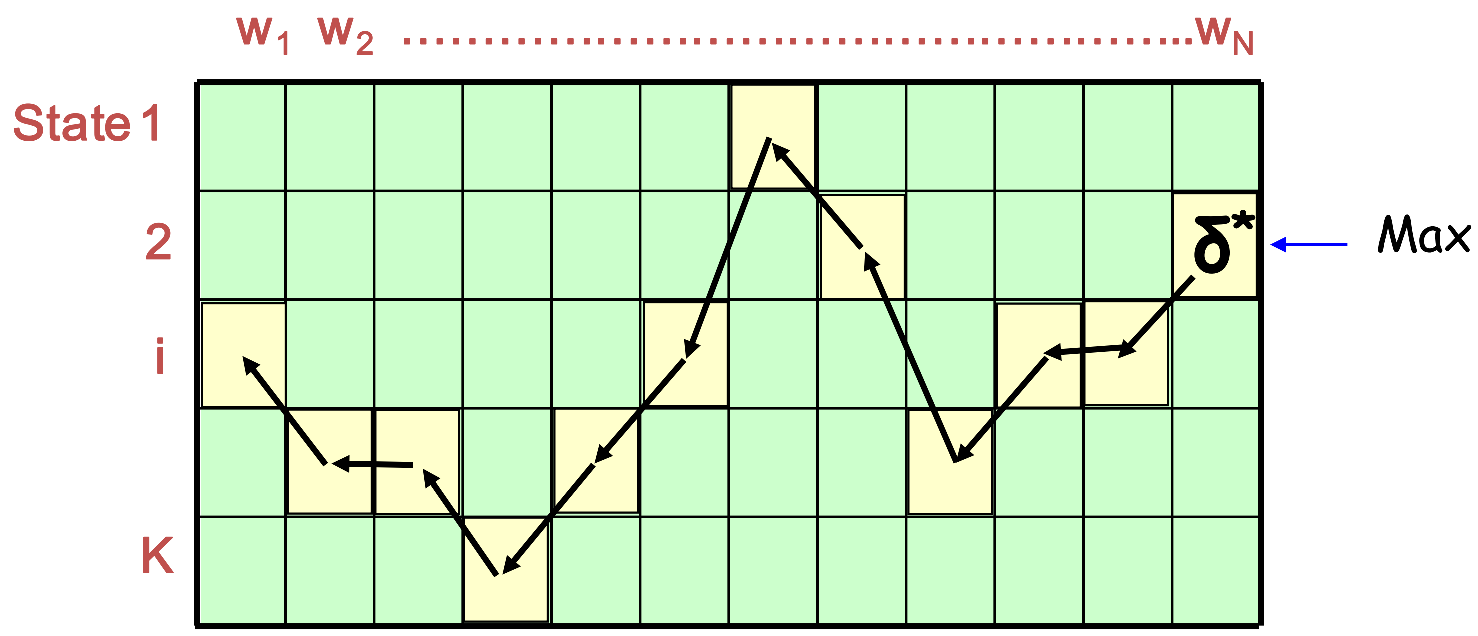


Remember: $\delta_i(s)$ = probability of most likely state seq ending with s at time i

Terminating Viterbi



Terminating Viterbi



How did we compute δ^*? $Max_{s'}\ \delta_{N-1}(s') * P_{trans} * P_{obs}$

Now Backchain to Find Final Sequence

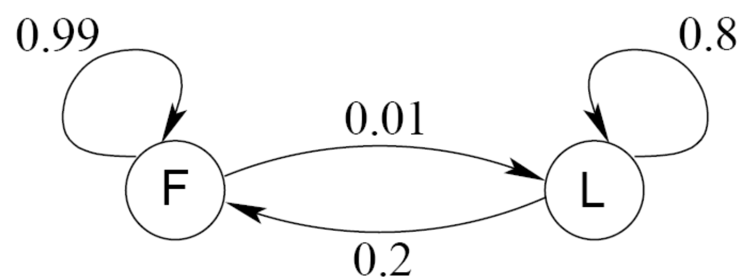
Time: $O(|S|^2N)$
Space: $O(|S|N)$

Linear in length of sequence

Viterbi: Example

s \ w	ε	6	2	6
B	1	0	0	0
F	0	(1/6)×(1/2) = 1/12	(1/6)×max{(1/12)×0.99, (1/4)×0.2} = 0.01375	(1/6)×max{0.01375×0.99, 0.02×0.2} = 0.00226875
L	0	(1/2)×(1/2) = 1/4	(1/10)×max{(1/12)×0.01, (1/4)×0.8} = 0.02	(1/2)×max{0.01375×0.01, 0.02×0.8} = 0.08

$$\delta_i(s) = p(w_i \mid s) \max_{s'} \left(p(s \mid s') \delta_{i-1}(s') \right)$$



Viterbi gets it right more often than not

Rolls	315116246446644245321131631164152133625144543631656626566666
Die	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL
Rolls	651166453132651245636664631636663162326455236266666625151631
Die	LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLLFFFFFFFFF
Viterbi	LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF
Rolls	222555441666566563564324364131513465146353411126414626253356
Die	FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Viterbi	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Rolls	366163666466232534413661661163252562462255265252266435353336
Die	LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi	LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Rolls	233121625364414432335163243633665562466662632666612355245242
Die	FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF

Computing Marginals

- Problem: find the marginal distribution

$$P(s_i|w_{0:n}) \propto P(s_i, w_{0:n}) = \sum_{s_{0:i-1}} \sum_{s_{i+1:n}} P(s_{0:n}, w_{0:n})$$

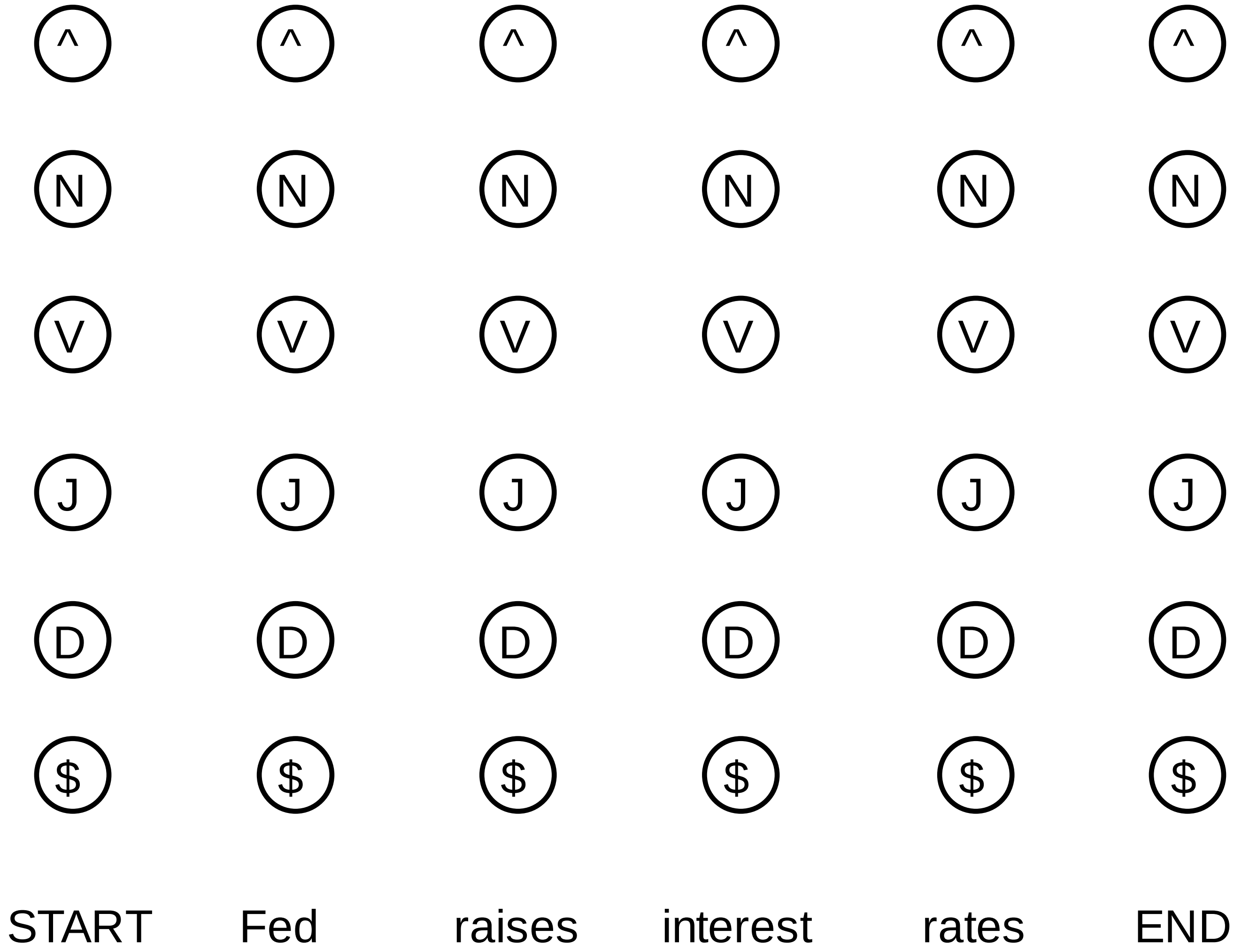
- Given model parameters, we can score any tag sequence

NNP	VBZ	NN	NNS	CD	NN	.
Fed	raises	interest	rates	0.5	percent	.

P(NNP|ϕ) P(Fed|NNP) P(VBZ|NNP) P(raises|VBZ) P(NN|NNP)…..

- In principle, we're done – list all possible tag sequences, score each one, sum up the values

The State Lattice / Trellis



The Forward Backward Algorithm

$$P(s_i, w_{0:n}) = P(w_{0:i}, s_i)P(w_{i+1:n}|s_i)$$

$$\begin{aligned}
P(s_i, w_{0:n}) &= P(s_i, w_{0:i}, w_{i+1:n}) \\
&= P(s_i, w_{0:i})P(w_{i+1:n} \mid s_i, w_{0:i}) \\
&= P(s_i, w_{0:i})P(w_{i+1:n} \mid s_i)
\end{aligned}$$

The Forward Backward Algorithm

$$P(s_i, w_{0:n}) = P(w_{0:i}, s_i)P(w_{i+1:n}|s_i)$$

Sum over all paths, on both sides:

$$\begin{aligned}\alpha_i(s_i) &= P(w_{0:i}, s_i) = \sum_{s_{0:i-1}} P(w_{0:i}, s_{0:i}) \\ &= \sum_{s_{i-1}} p(w_i|s_i)P(s_i|s_{i-1})\alpha_{i-1}(s_{i-1})\end{aligned}$$

$$\begin{aligned}\beta_i(s_i) &= P(w_{i+1:n}|s_i) = \sum_{s_{i+1:n}} P(w_{i+1:n}, s_{i+1:n}|s_i) \\ &= \sum_{s_{i+1}} P(w_{i+1}|s_{i+1})P(s_{i+1}|s_i)\beta_{i+1}(s_{i+1})\end{aligned}$$

The Forward Backward Algorithm

Two passes over entire observation sequence

- Forward:

$$\alpha_0(s_0) = \begin{cases} 1 & \text{if } s_0 = START \\ 0 & \text{otherwise} \end{cases}$$

For $i = 1 \ldots n$

$$\alpha_i(s_i) = \sum_{s_{i-1}} P(w_i|s_i)P(s_i|s_{i-1})\alpha_{i-1}(s_{i-1})$$

- Backward:

$$\beta_n(s_n) = \begin{cases} 1 & \text{if } s_n = STOP \\ 0 & \text{otherwise} \end{cases}$$

For $i = n\text{-}1 \ldots 0$

$$\beta_i(s_i) = \sum_{s_{i+1}} P(w_{i+1}|s_{i+1})P(s_{i+1}|s_i)\beta_{i+1}(s_{i+1})$$

HMM Learning

- Learning from data D
  - Supervised
    - $D = \{(\mathbf{s}_{0:n},\mathbf{w}_{0:n})_i \mid i = 1 \ldots m\}$
  - Unsupervised
    - $D = \{(\mathbf{w}_{0:n})_i \mid i = 1 \ldots m\}$
    - We won't do this case!
    - (~hidden vars) EM
      - Also called Baum Welch algorithm

Supervised Learning

– Given data D = {X_i | i = 1 … m } where X_i=($\mathbf{s_{0:n}},\mathbf{w_{0:n}}$) is a state, observation sequence pair

– Define the parameters Θ to include:

- For every pair of states: $\theta_{s,s'} = P(s'|s)$
- For every state, obs. pair: $\theta_{s,w} = P(w|s)$

– Then the data likelihood is:

$$L(D;\Theta) = P(X_1, X_2, \ldots, X_m|\Theta) = \prod_j P(X_j|\Theta)$$

- And the maximum likelihood solutions is

$$\Theta^* = \arg\max_{\Theta} L(D;\Theta)$$

Final ML Estimates (as in BNs)

– c(s,s') and c(s,w) are the empirical counts of transitions and observations in the data D

– The final, intuitive, estimates:

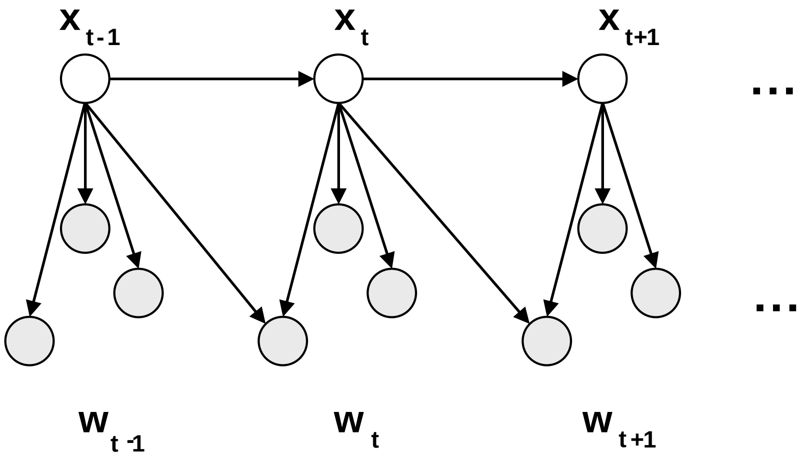
$$\theta_{s,s'} = \frac{c(s,s')}{\sum_{s''} c(s,s'')} \qquad \theta_{s,w} = \frac{c(s,w)}{\sum_{w'} c(s,w')}$$

- Just as with BNs, the counts can be zero
  - use smoothing techniques!

The Problem with HMMs

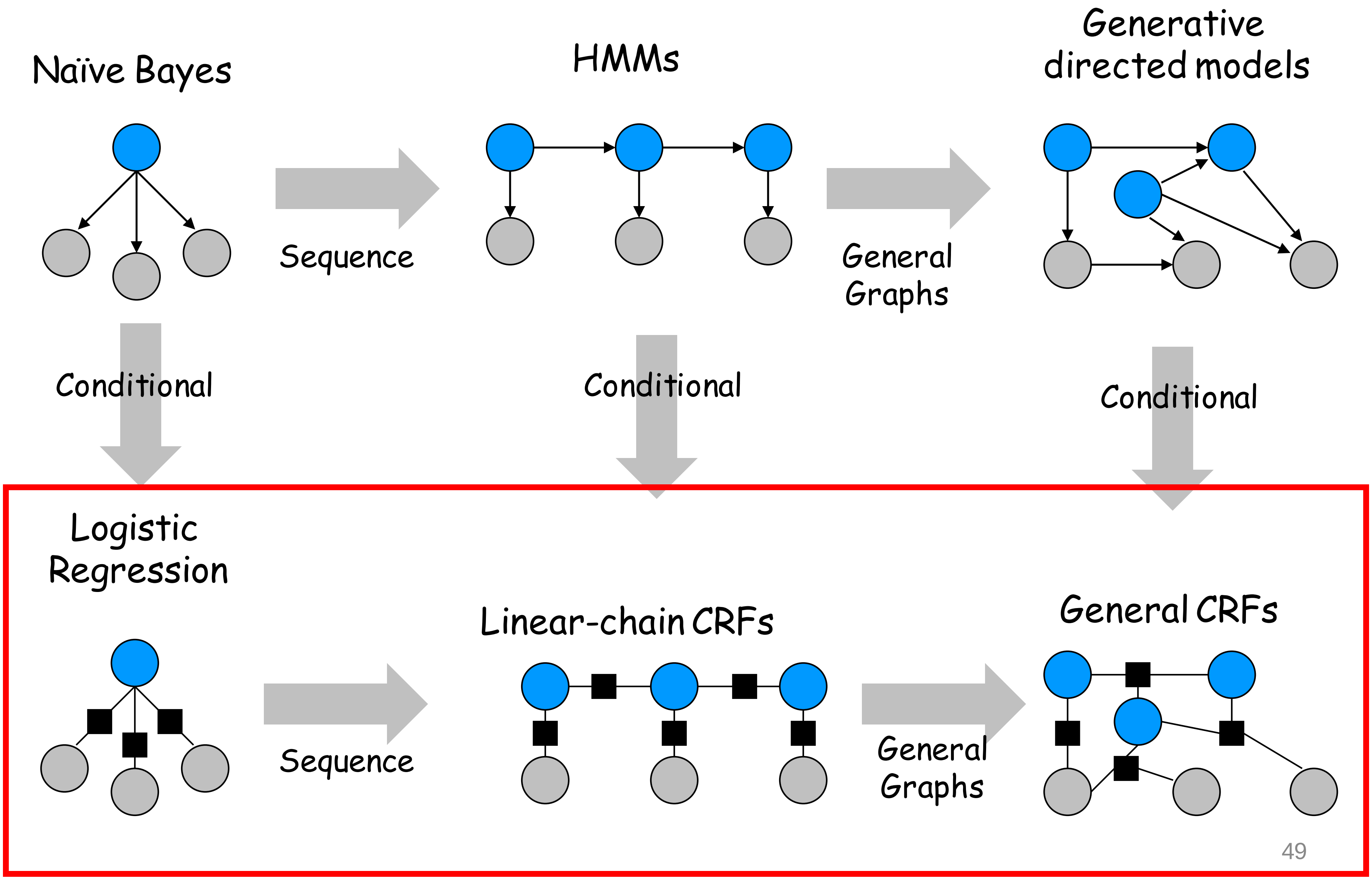
- We want more than an Atomic View of Words
- We want many arbitrary, overlapping features of words

identity of word
ends in "-ly", "-ed", "-ing"
is capitalized
appears in a name database/Wordnet
…



Use discriminative models instead of generative ones
(e.g., Conditional Random Fields)

Finite State Models



Temporal Models

- Full Bayesian Networks have dynamic versions too
  - Dynamic Bayesian Networks (Chapter 15.5)
  - HMM is a special case
- HMMs with continuous variables often useful for filtering (estimating current state)
  - Kalman filters (Chapter 15.4)