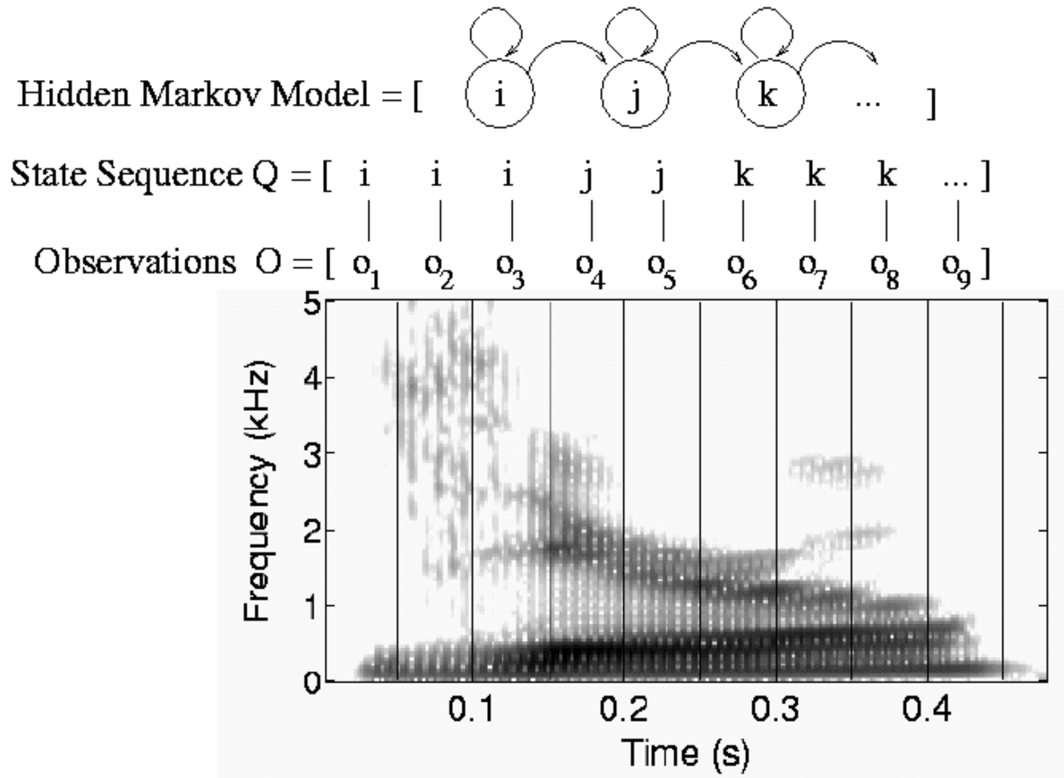
CS440/ECE448 Lecture 6: Hidden Markov Models

Mark Hasegawa-Johnson, 1/2024



Outline

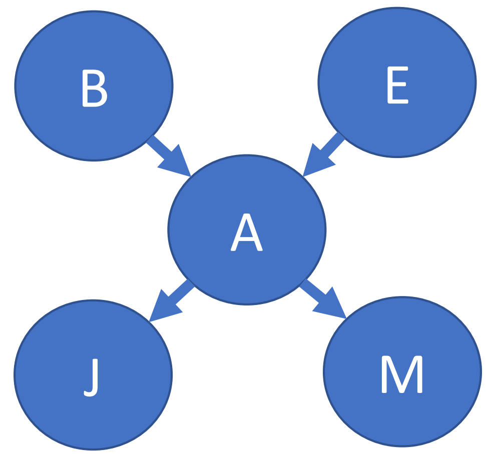
- Review: Bayesian classifier, Bayesian networks
- HMM: Probabilistic reasoning over time
- Viterbi algorithm

Review: Bayesian Classifier

- Class label $Y = y$, drawn from some set of labels
- Observation $X = x$, drawn from some set of features
- Bayesian classifier: choose the class label, y, that minimizes your probability of making a mistake:

$$f(x) = \underset{y}{\operatorname{argmax}}\, P(Y = y | X = x)$$

Bayesian network: A better way to represent knowledge



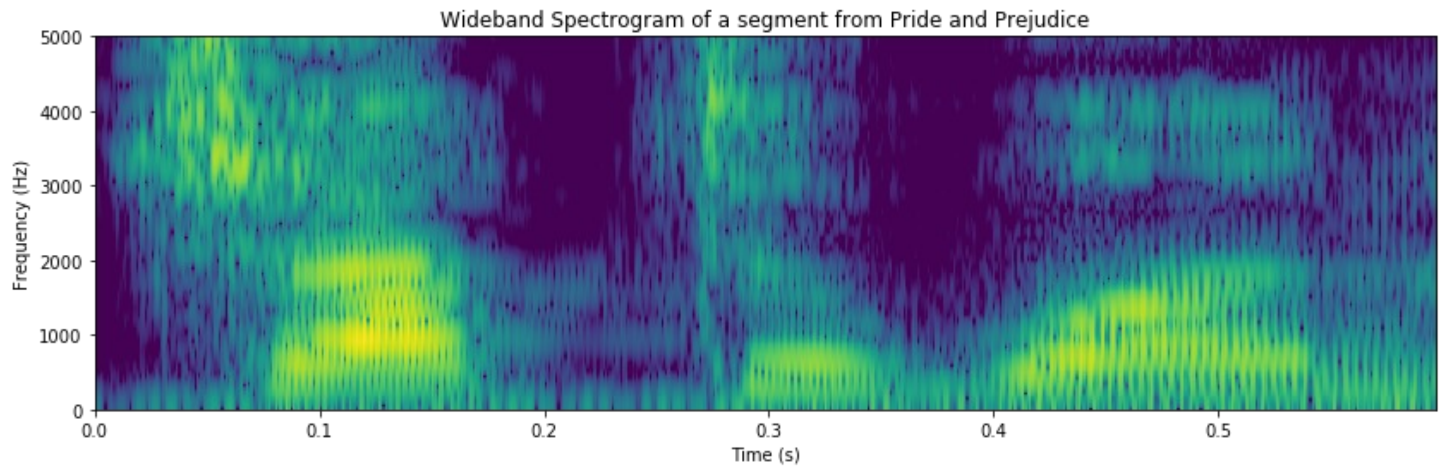
A Bayesian network is a graph in which:

- Each variable is a node.
- An arrow between two nodes means that the child depends on the parent.
- If the child has no direct dependence on the parent, then there is no arrow.

Outline

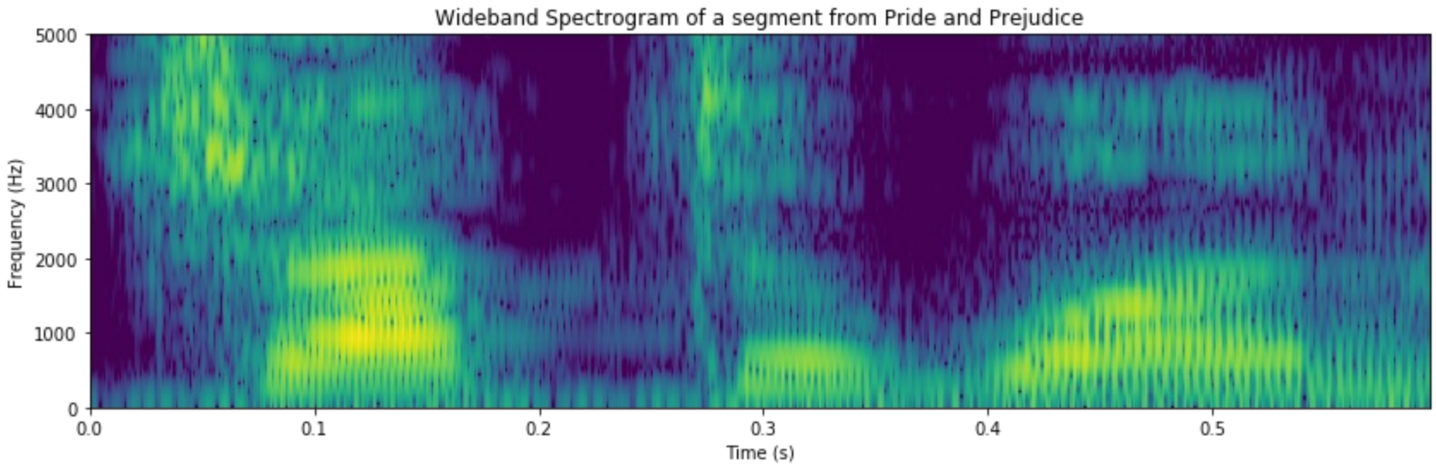
- Review: Bayesian classifer, Bayesian networks
- HMM: Probabilistic reasoning over time
- Viterbi algorithm

Example: Speech Recognition



- Here's a spectrogram of the utterance "chapter one."
- Each column is the Fourier transform of 0.02s of audio, spaced 0.01s apart. Let's call the spectral vector X_t, where t is time in centiseconds
- The speech sounds follow a sequence: silence for a while, then /sh/ for a while, then /ae/ for a while, then…. Let's denote the speech sound at time t as Y_t

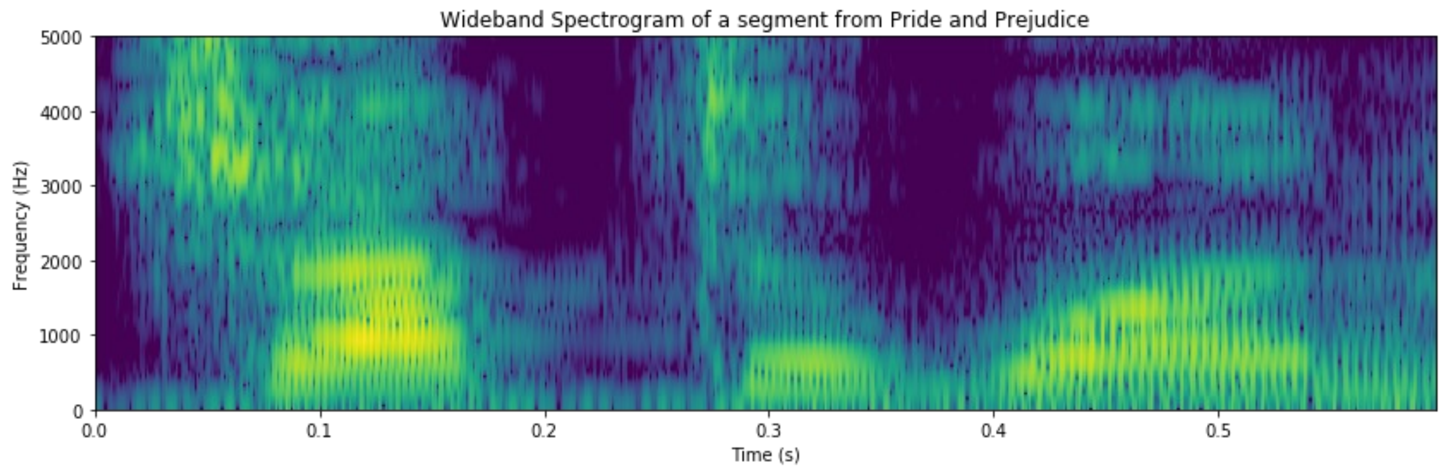
Hidden Markov Model

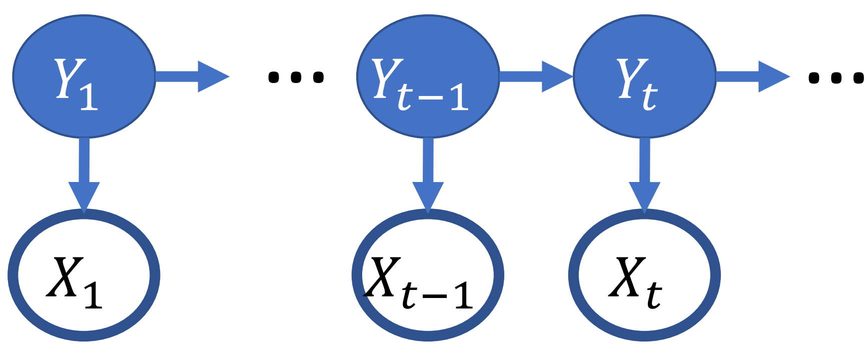


A Hidden Markov Model is a Bayes Network with these assumptions:

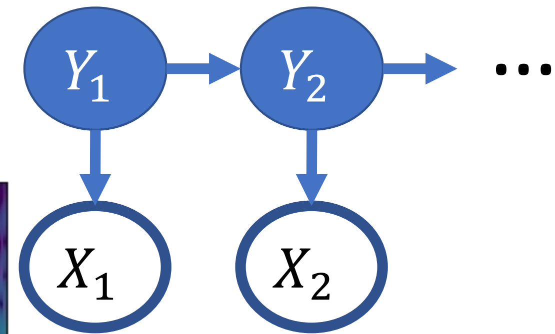
- Y_t depends only on Y_{t-1}
- X_t depends only on Y_t

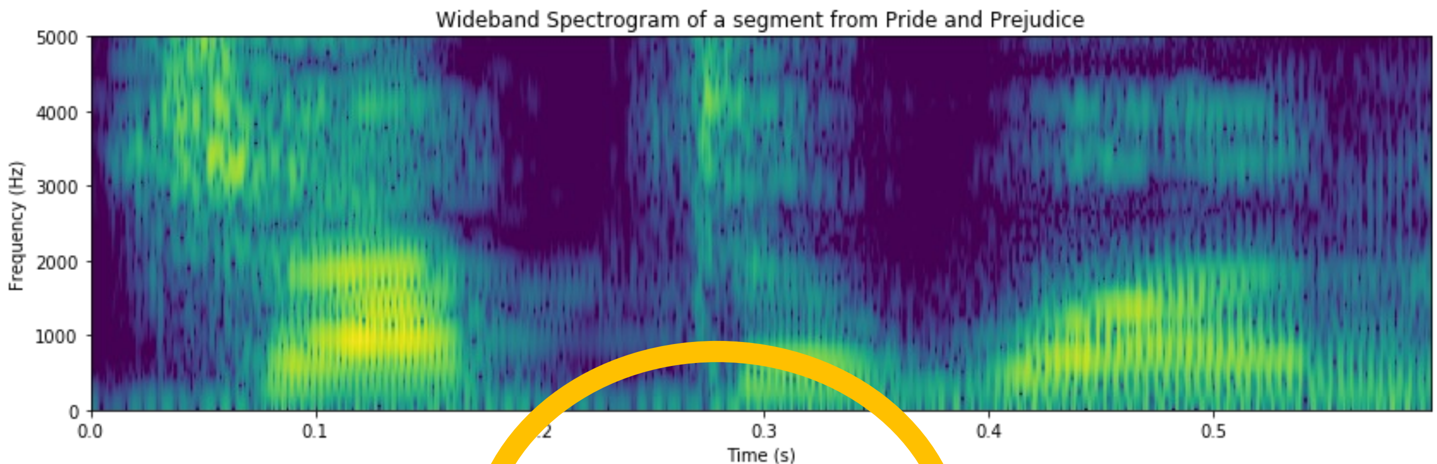
Hidden Markov Model





Hidden Markov Model

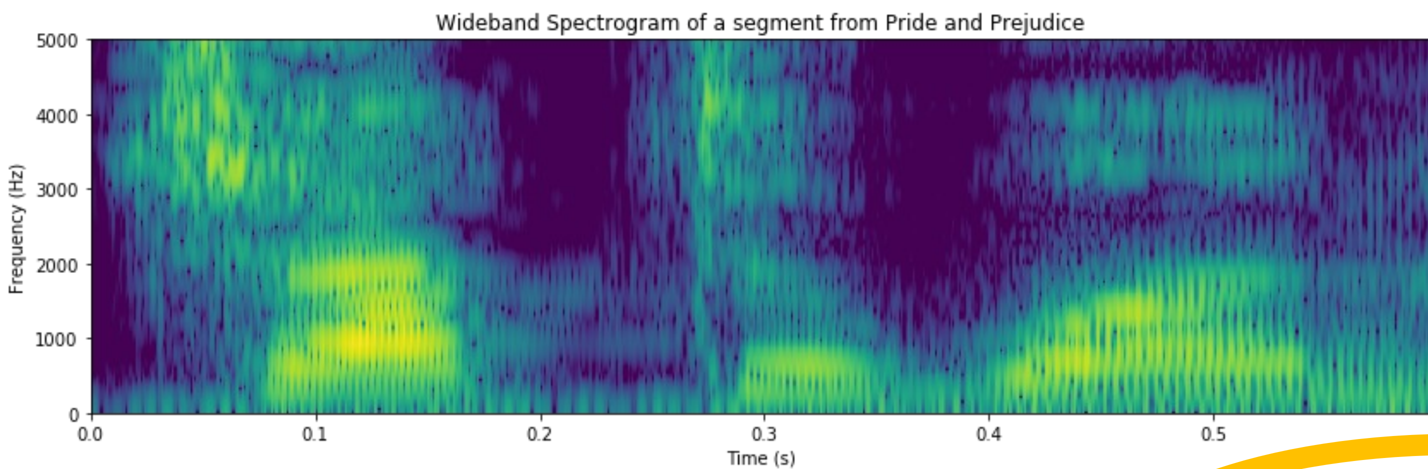


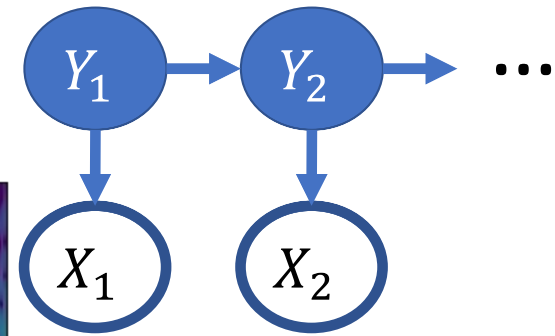


$$P(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_d} P(y_1) P(\boldsymbol{x}_1 | y_1) P(y_2 | y_1) P(\boldsymbol{x}_2 | y_2) P(y_3 | y_2) \cdots$$

$\mathcal{O}\{|\mathcal{Y}|^d\}$ terms in this summation. Does this mean time-complexity is $\mathcal{O}\{|\mathcal{Y}|^d\}$?

Hidden Markov Model





$$P(\boldsymbol{x}_1, \dots, \boldsymbol{x}_d) = \sum_{y_d} \cdots \sum_{y_2} P(y_3|y_2)P(\boldsymbol{x}_2|y_2) \sum_{y_1} P(y_2|y_1)P(\boldsymbol{x}_1|y_1)P(y_1)$$

- There are only $|\mathcal{Y}|$ terms in this summation $(0 \leq y_0 \leq |\mathcal{Y}| - 1)$.
- This summation needs to be computed $|\mathcal{Y}|$ times (once for each value y_1)
- Total complexity: $\mathcal{O}\{|\mathcal{Y}|^2\}$

Key advantage of a hidden Markov model: Polynomial-time complexity

- Suppose there are $|\mathcal{Y}|$ different speech sounds in English, and the length of the utterance is d centiseconds ($|\mathcal{Y}| \approx 50, d \approx 100$)
- Without the HMM assumptions, to compute $f(\boldsymbol{x}) = \text{argmax} P(y_1, \dots, y_d | \boldsymbol{x}_1, \dots, \boldsymbol{x}_d)$ requires a time complexity of $\mathcal{O}\{|\mathcal{Y}|^d\} \approx 50^{100}$
- With an HMM, each variable has only one parent, so inference is $\mathcal{O}\{|\mathcal{Y}|^2\} \approx 50^2$
- The computationally efficient algorithm that we use to compute $f(\boldsymbol{x}) = \text{argmax} P(y_1, \dots, y_d | \boldsymbol{x}_1, \dots, \boldsymbol{x}_d)$ is called the Viterbi algorithm, named after the electrical engineer who first applied it to error correction coding.

…and it works much better than naïve Bayes:

Claude Shannon (1948) gave these examples:

- Text generated by a naïve Bayes model (unigram model):

Representing and speedily is an good apt or come can different natural here he the a in came the to of to expert gray come to furnishes the line message had be these…

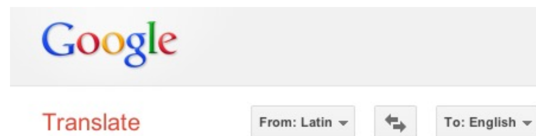
- Text generated by a HMM (bigram model):

The head and in frontal attack on an English writer that the character of this point is therefore another for the letters that the time of who ever told the problem for an unexpected…

Applications of HMMs

- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)

- Machine translation HMMs:
  - Observations are words (tens of thousands)
  - States are cross-lingual alignments



- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map

Source: Tamara Berg

Outline

- Review: Bayesian classifer, Bayesian networks
- HMM: Probabilistic reasoning over time
- Viterbi algorithm

Viterbi Algorithm

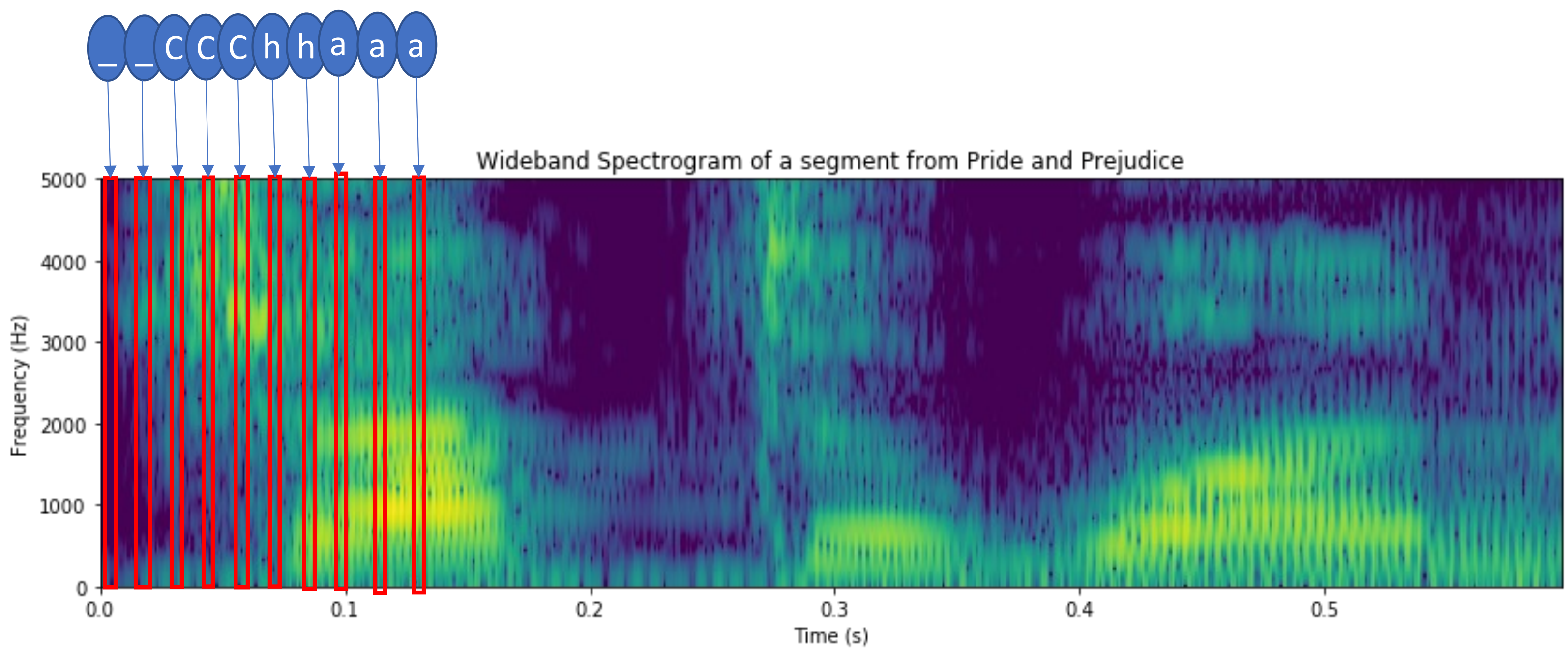
The Viterbi algorithm is a computationally efficient algorithm for computing the maximum *a posteriori* (MAP) state sequence,

$$f(\boldsymbol{x}) = \mathrm{argmax}_{y_1,\ldots,y_d} P(y_1, \ldots, y_d | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$$

Example: Speech Recognition

- Observations: X_t = spectrum of 25ms frame of the speech signal.
- State: Y_t = phoneme or letter being currently produced

The goal of speech recognition: find the most probable sequence of text characters $\{y_1, \ldots, y_T\}$ given observations $\{X_1 = x_1, \ldots, X_T = x_T\}$.

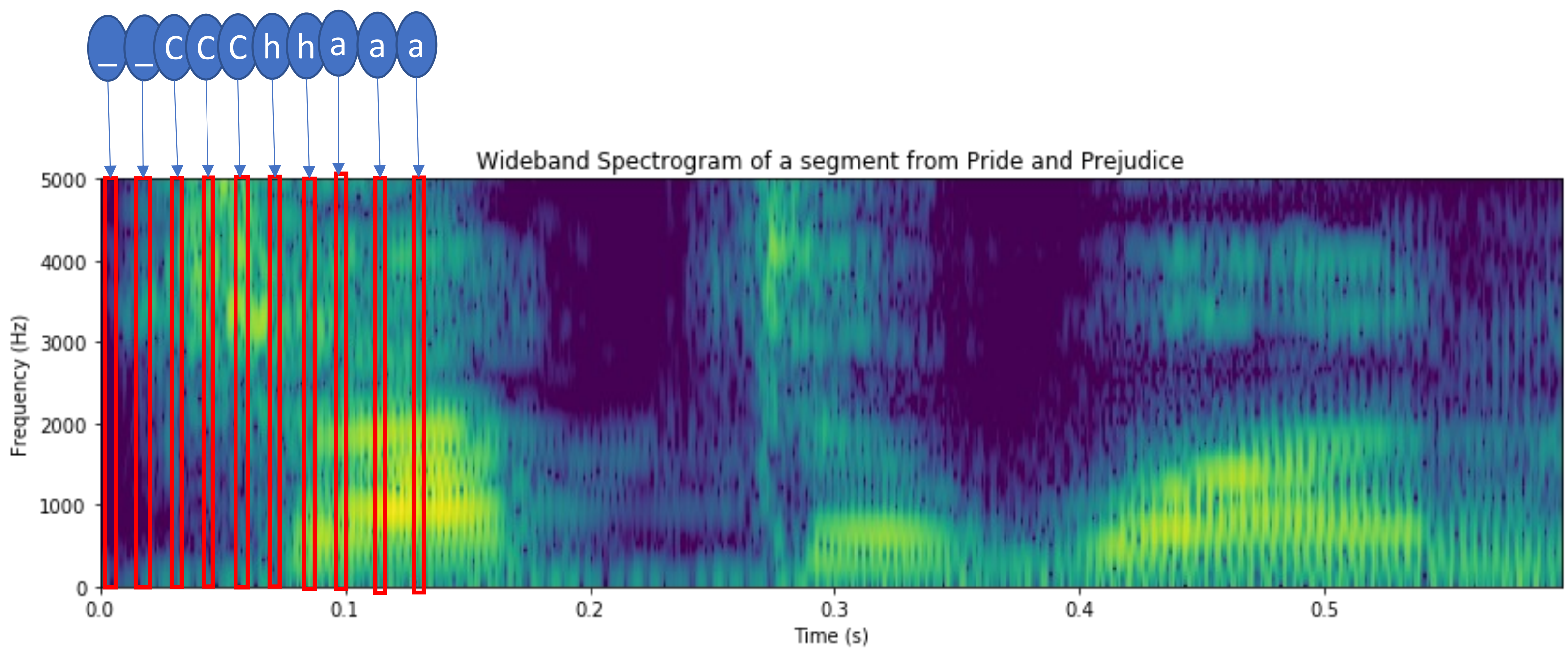


Viterbi Algorithm

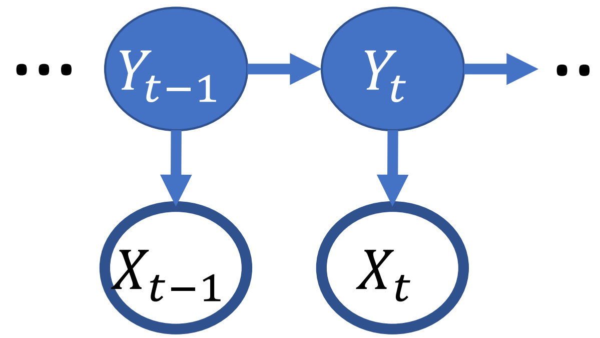
Basic concept:

$$\max P(y_1, \dots, y_d, \boldsymbol{x}_1, \dots, \boldsymbol{x}_d)$$

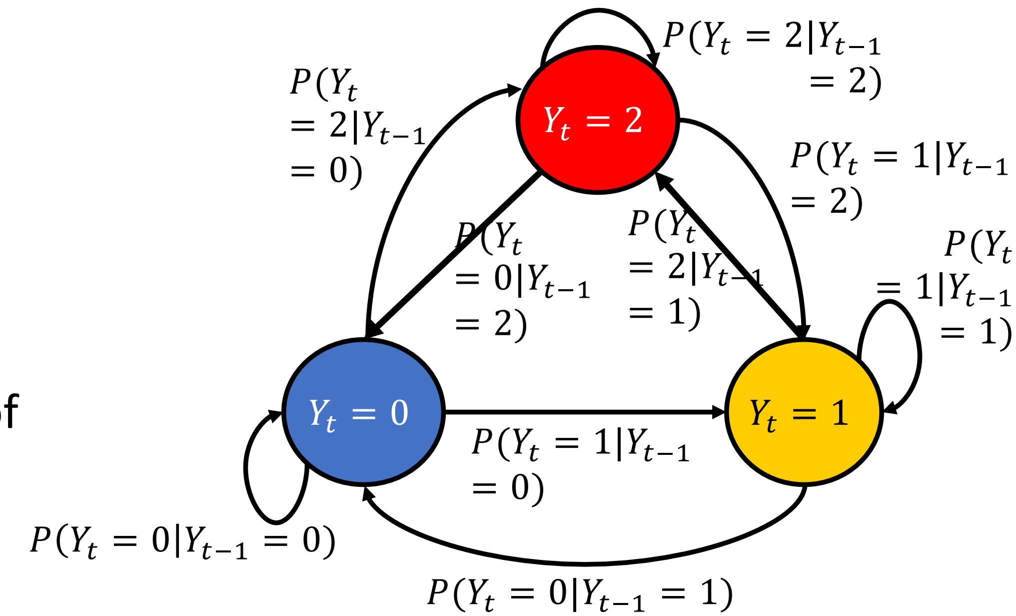
$$= \max_{y_d} \cdots \max_{y_2} P(y_3|y_2)P(\boldsymbol{x}_2|y_2) \max_{y_1} P(y_2|y_1)P(\boldsymbol{x}_1|y_1)P(y_1)$$



Two views of an HMM

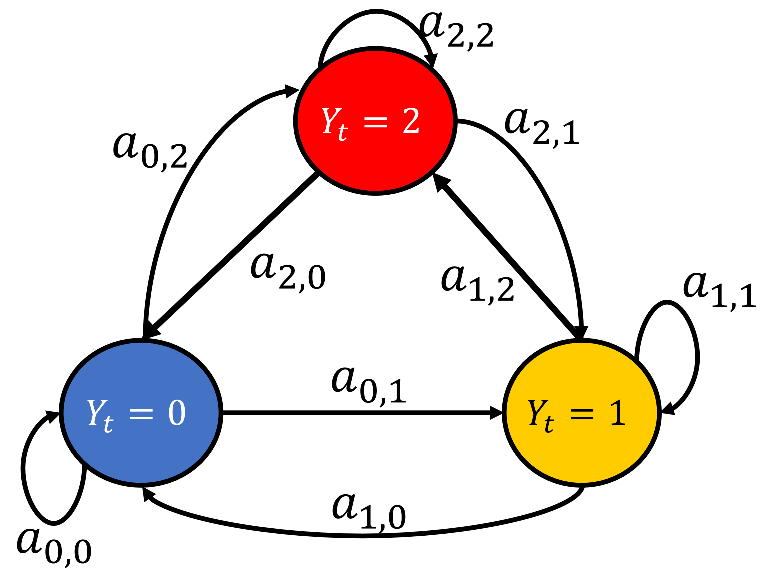


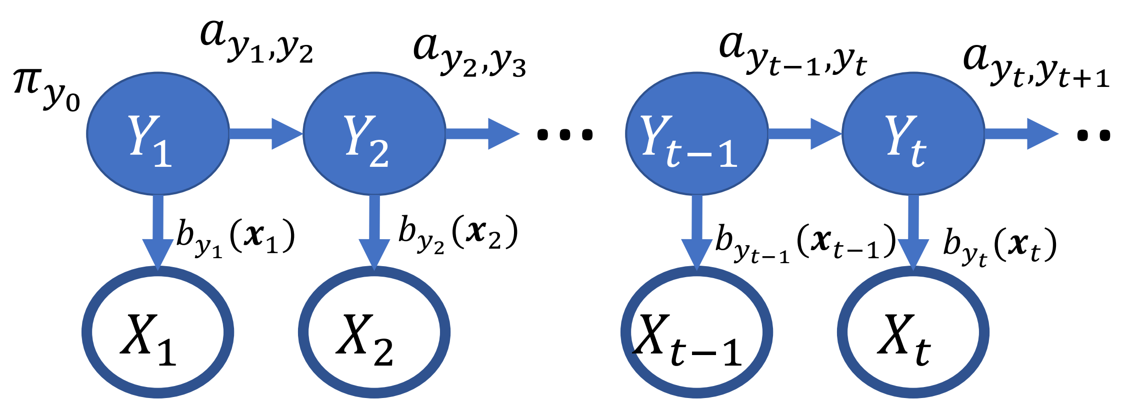
- Bayesian Network diagram shows:
  - Time
  - Y_t depends only on Y_{t-1}
  - X_t depends only on Y_t
- Finite State Machine diagram shows:
  - All of the possible values that Y_t can take
  - The time-complexity of inference is $\mathcal{O}\{|\mathcal{Y}|^2\}$ (because that's the number of edges in the FSM diagram)



Notation

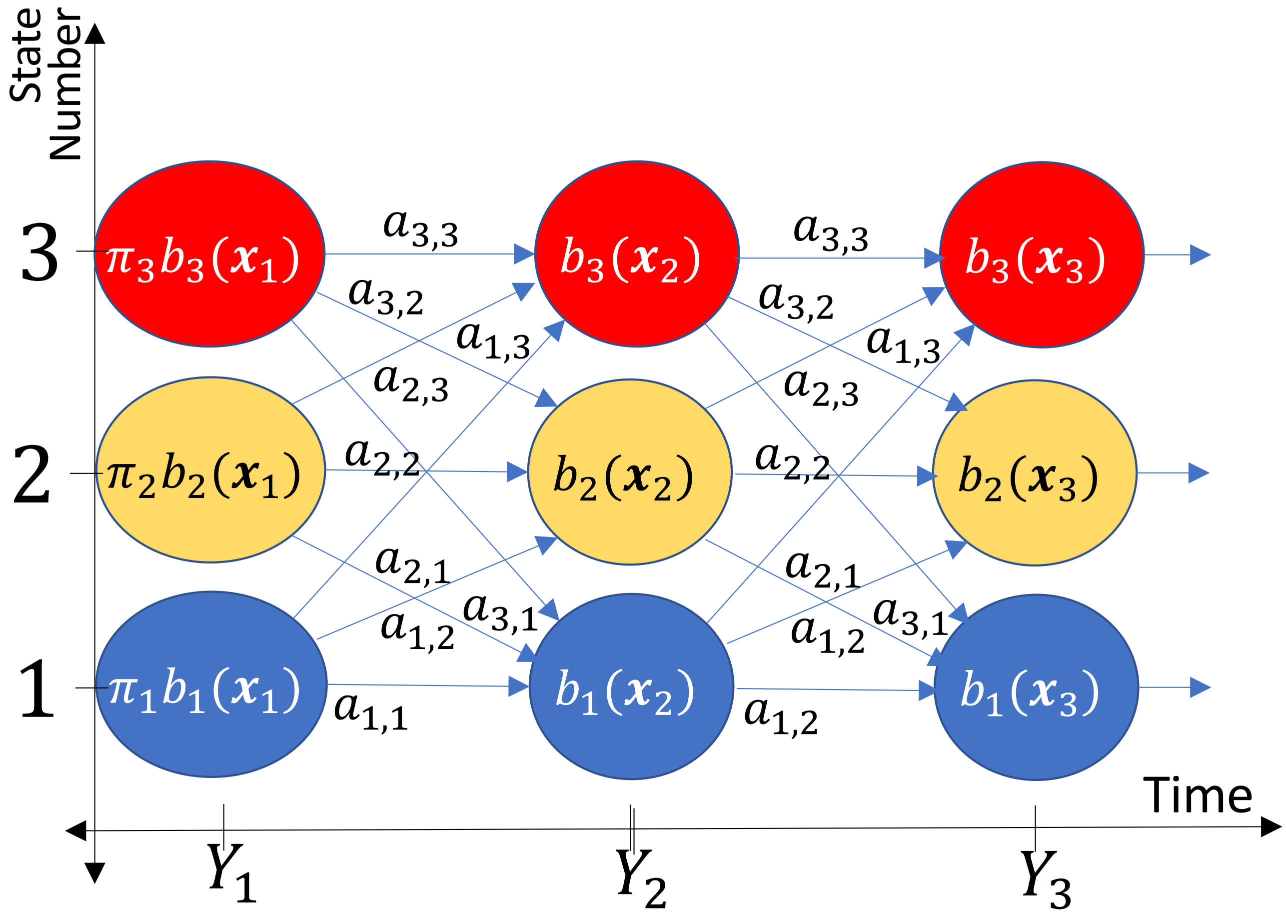
- Initial State Probability:
$$\pi_i = P(Y_1 = i)$$
- Transition Probabilities:
$$a_{i,j} = P(Y_t = j | Y_{t-1} = i)$$
- Observation Probabilities:
$$b_j(x_t) = P(X_t = x_t | Y_t = j)$$





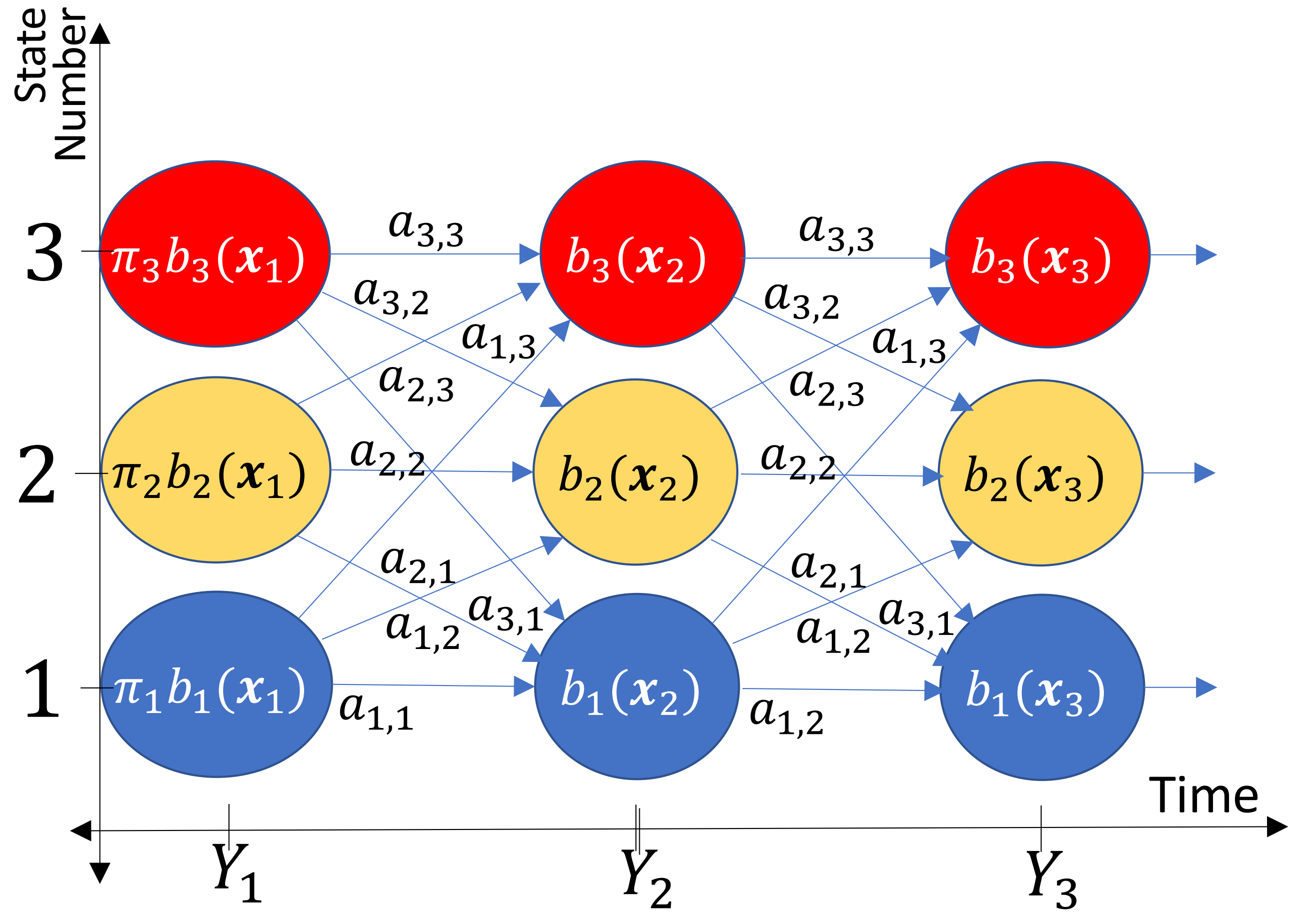
The Trellis

- Time is on the horizontal axis
- State number on the vertical axis:
  $$\pi_i = P(Y_1 = i)$$
- Edges show state transitions: $a_{i,j}$
- States show observation probabilities: $b_j(x_t)$



The Trellis

- A sequence of state variables is a path through the trellis.



For example:

$$P(Y_1 = 1, Y_2 = 3, Y_3 = 2, X_1 = \boldsymbol{x}_1, X_2 = \boldsymbol{x}_2, X_3 = \boldsymbol{x}_3) = \pi_1 b_1(\boldsymbol{x}_1) a_{1,3} b_3(\boldsymbol{x}_2) a_{3,2} b_2(\boldsymbol{x}_3)$$

Node probabilities and backpointers

- **Node Probability**: Probability of the best path until node j at time t

$$v_t(j) = \max_{y_1,\ldots,y_{t-1}} P(Y_1 = y_1 \ldots, Y_{t-1} = y_{t-1}, Y_t = j, X_0 = x_0, \ldots, X_t = x_t)$$

- **Backpointer**: which node precedes node j on the best path?

$$\psi_t(j) = \operatorname*{argmax}_{y_{t-1}} \max_{y_1,\ldots,y_{t-2}} P(Y_1 = y_1 \ldots, Y_{t-1} = y_{t-1}, Y_t = j, X_0 = x_0, \ldots, X_t = x_t)$$

- **B**

Viterbi Algorithm

- Initialization: for all states i:

$$v_1(i) = \pi_i b_i(\boldsymbol{x}_1)$$

- Iteration: for $2 \leq t \leq d$, for all states j:

$$v_t(j) = \max_i v_{t-1}(i)\, a_{i,j} b_j(\boldsymbol{x}_t)$$

$$\psi_t(j) = \operatorname*{argmax}_i v_{t-1}(i)\, a_{i,j} b_j(\boldsymbol{x}_t)$$

- Termination:

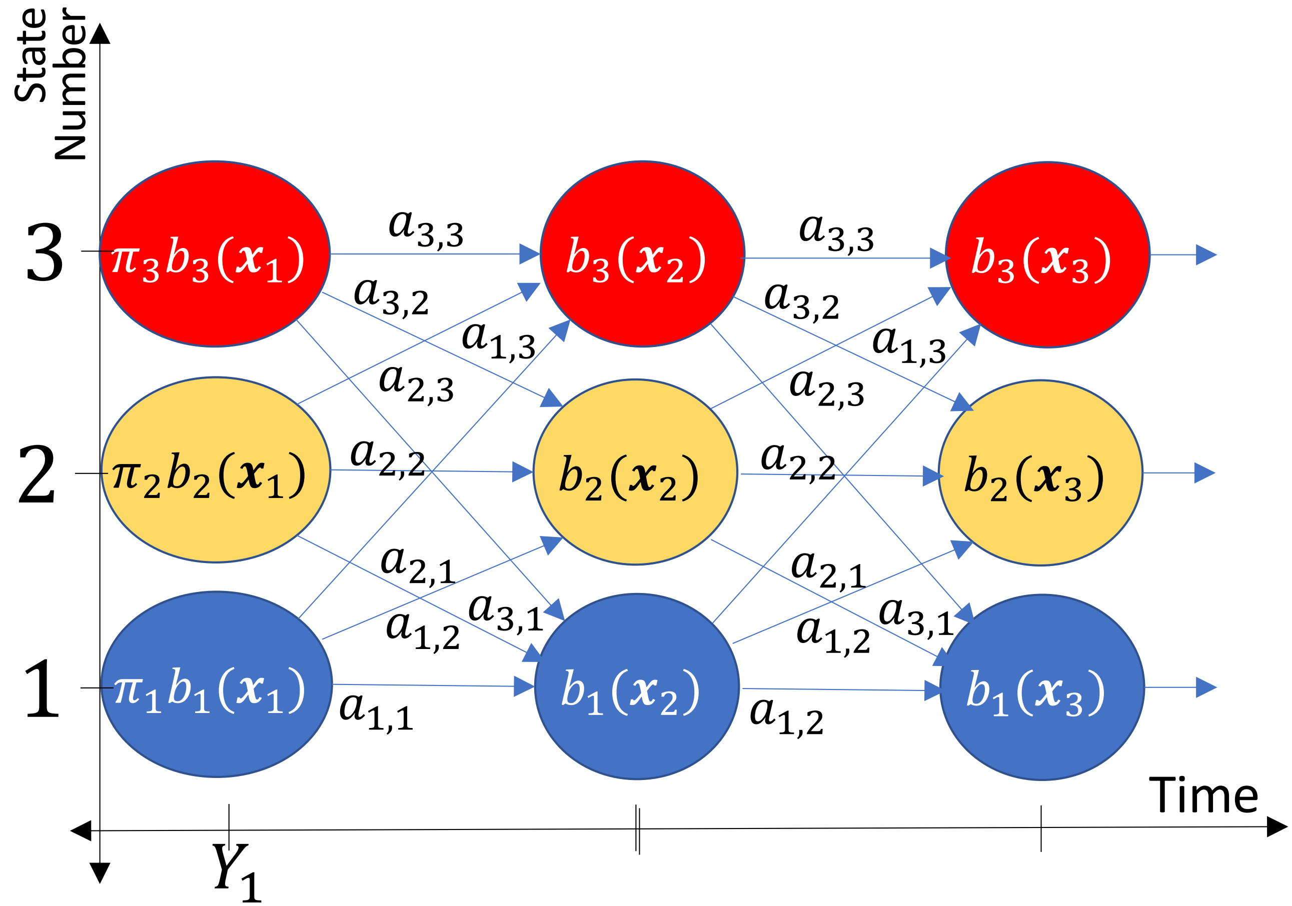
$$y_d = \operatorname*{argmax}_i v_d(i)$$

- Back-Trace:

$$y_t = \psi_{t+1}(y_{t+1})$$

Initialization

$$v_1(i) = \pi_i b_i(\boldsymbol{x}_1)$$



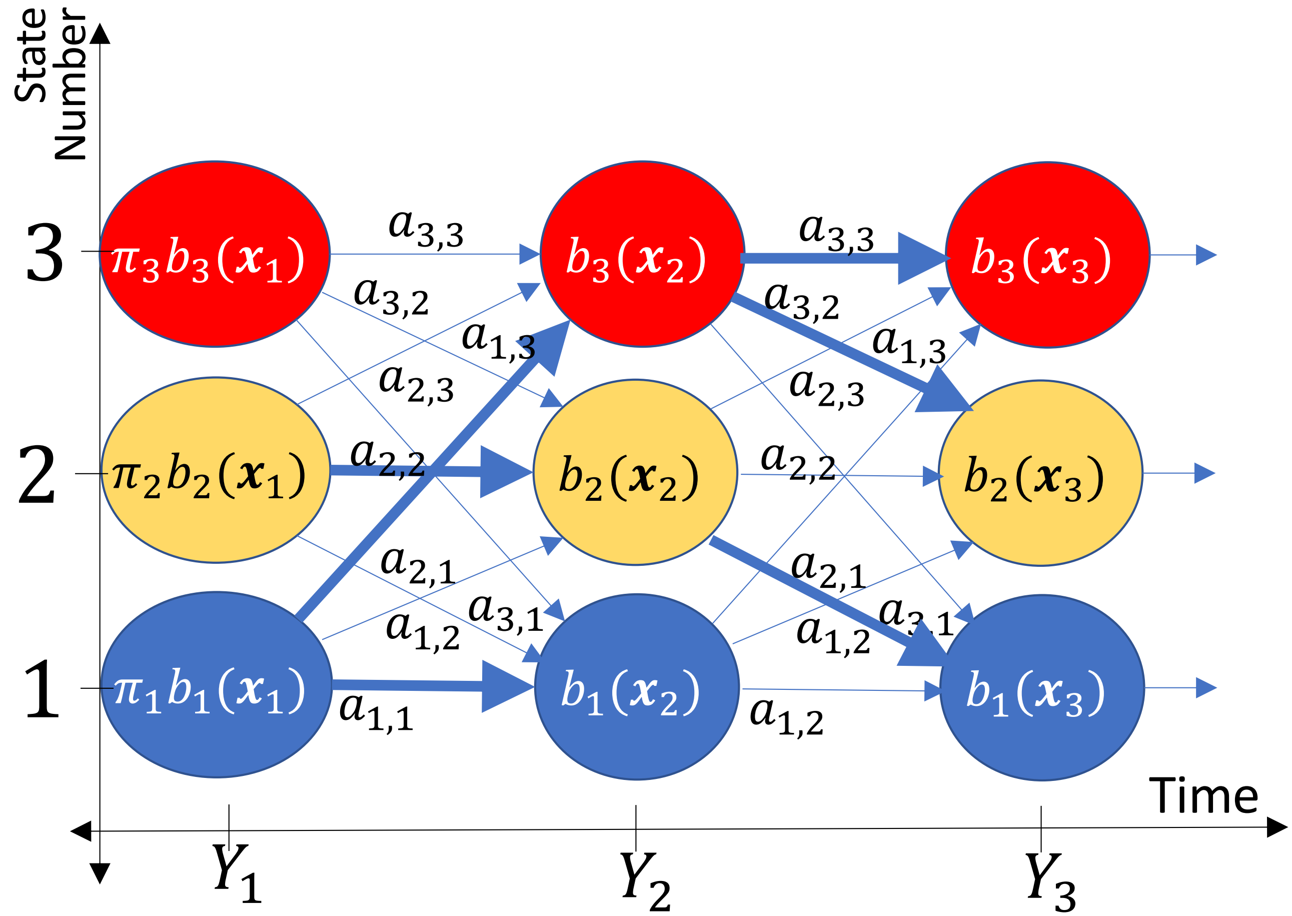
Iteration

Each node now has a value:

$$v_t(j) = \max_i v_{t-1}(i)\, a_{i,j} b_j(\boldsymbol{x}_t)$$

… and there is exactly one backpointer, from every node, to exactly one node in the previous time step:

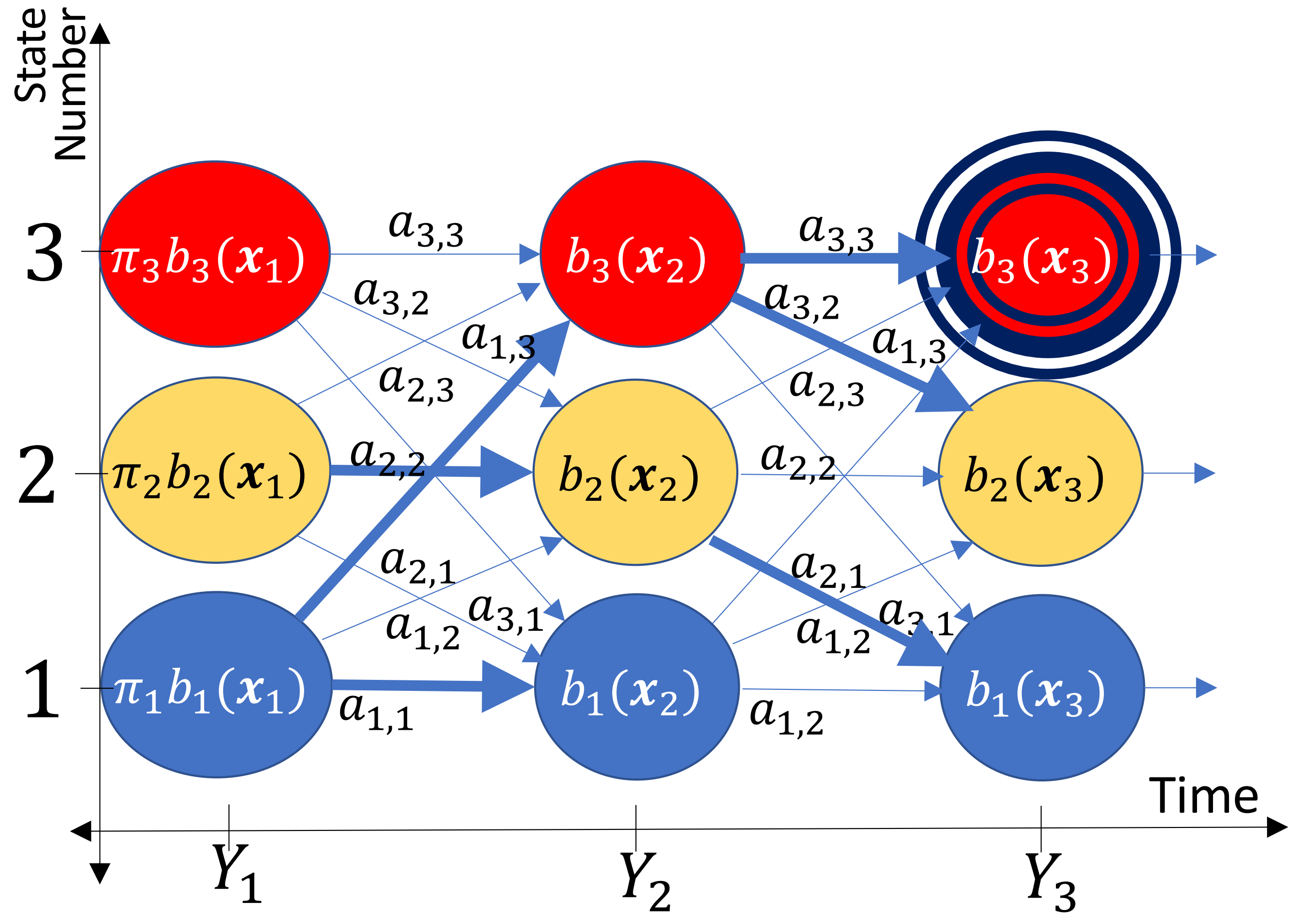
$$\psi_t(j) = \operatorname*{argmax}_i v_{t-1}(i)\, a_{i,j} b_j(\boldsymbol{x}_t)$$



Termination

The best path is the one that ends with the highest-value node:

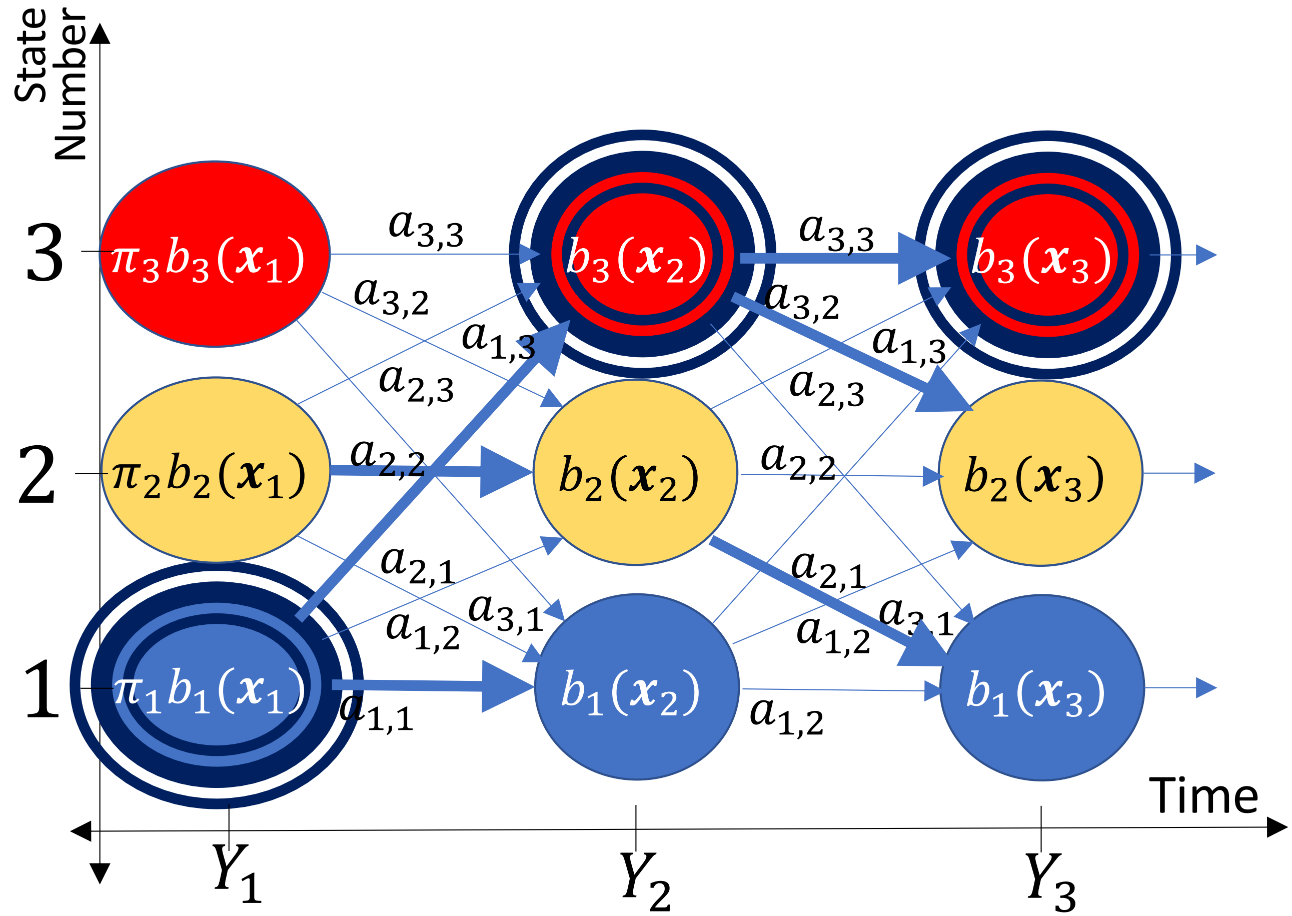
$$y_d = \underset{i}{\operatorname{argmax}}\, v_d(i)$$



Back-Tracing

The most likely state sequence is the one that ends with the highest-value node:

$$y_t = \psi_{t+1}(y_{t+1})$$



Viterbi Algorithm Computational Complexity

- Initialization: for $i \in \mathcal{Y}$:

$$v_1(i) = \pi_i b_i(\boldsymbol{x}_1)$$

$\mathcal{O}\{|\mathcal{Y}|\}$

- Iteration: for $2 \leq t \leq d$, for $j \in \mathcal{Y}$:

$$v_t(j) = \max_{i \in \mathcal{Y}} v_{t-1}(i)\, a_{i,j} b_j(x_t)$$

$$\psi_t(j) = \operatorname*{argmax}_{i \in \mathcal{Y}} v_{t-1}(i)\, a_{i,j} b_j(x_t)$$

$\mathcal{O}\{d|\mathcal{Y}|^2\}$

- Termination:

$$y_d = \operatorname*{argmax}_{i \in \mathcal{Y}} v_d(i)$$

$\mathcal{O}\{|\mathcal{Y}|\}$

- Back-Trace:

$$y_t = \psi_{t+1}(y_{t+1})$$

$\mathcal{O}\{d\}$

Total: $\mathcal{O}\{d|\mathcal{Y}|^2\}$

Try the quiz!

- Quiz: https://us.prairielearn.com/pl/course_instance/147925/assessment/2392750

Outline

- Review: Bayesian classifer, Bayesian networks

$$f(x) = \operatorname*{argmax}_{y} P(Y = y | X = x)$$

- HMM: Probabilistic reasoning over time

$$\pi_i = P(Y_1 = i)$$
$$a_{i,j} = P(Y_t = j | Y_{t-1} = i)$$
$$b_j(\boldsymbol{x}_t) = P(X_t = \boldsymbol{x}_t | Y_t = j)$$

- Viterbi algorithm

$$v_t(j) = \max_{i \in \mathcal{Y}} v_{t-1}(i)\, a_{i,j} b_j(\boldsymbol{x}_t)$$
$$\psi_t(j) = \operatorname*{argmax}_{i \in \mathcal{Y}} v_{t-1}(i)\, a_{i,j} b_j(\boldsymbol{x}_t)$$