

10-601

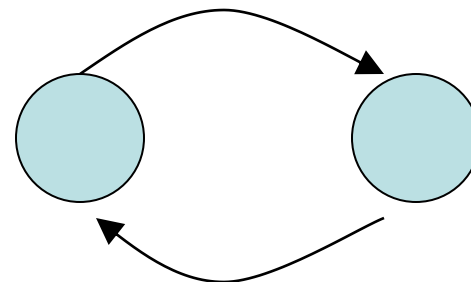
Machine Learning

Hidden Markov models (HMMs)

What's wrong with Bayesian networks

- Bayesian networks are very useful for modeling joint distributions
- But they have their limitations:
 - Cannot account for temporal / sequence models
 - DAG's (no self or any other loops)

This is not a valid Bayesian network!



Hidden Markov models

- Model a set of observation with a set of hidden states
 - Robot movement
 - Observations:** range sensor, visual sensor
 - Hidden states:** location (on a map)
 - Speech processing
 - Observations:** sound signals
 - Hidden states:** parts of speech, words
 - Biology
 - Observations:** DNA base pairs
 - Hidden states:** Genes

Hidden Markov models

- Model a set of observation with a set of hidden states
 - Robot movement

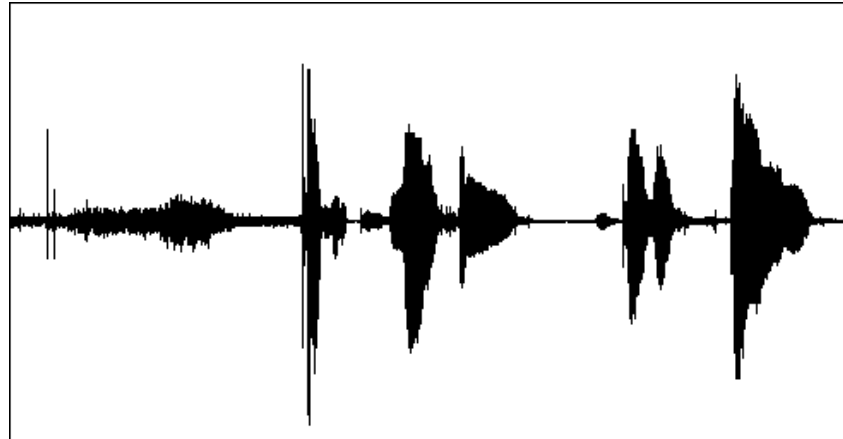
Observations: range sensor, visual sensor

Hidden states: location (on a map)



1. Hidden states generate observations
2. Hidden states transition to other hidden states

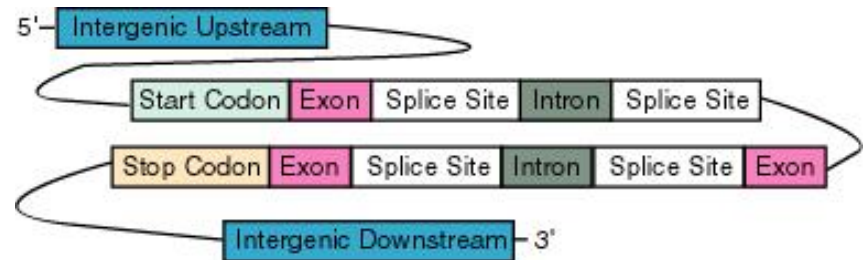
Examples: Speech processing



| | | | | | |
|-----|------|-------|-----|------|-----|
| sil | acht | negen | sil | drie | een |
|-----|------|-------|-----|------|-----|

| | | | | | |
|-----|-----|-----|-----|-----|-----|
| sil | spk | spk | sil | spk | spk |
|-----|-----|-----|-----|-----|-----|

Example: Biological data



```
ATGAAGCTACTGTCTTCTATCGAACAAGCATGCG
ATATTTGCCGACTTAAAAAGCTCAAG
TGCTCCAAAGAAAAACCGAAGTGCGCCAAGTGT
CTGAAGAACA ACTGGGAGTGTCGCTAC
TCTCCCAA AACCAAAGGTCTCCGCTGACTAGG
GCACATCTGACAGAAGTGGAATCAAGG
CTAGAAAGACTGGAACAGCTATTTCTACTGATTTT
TCCTCGAGAAGACCTTGACATGATT
```

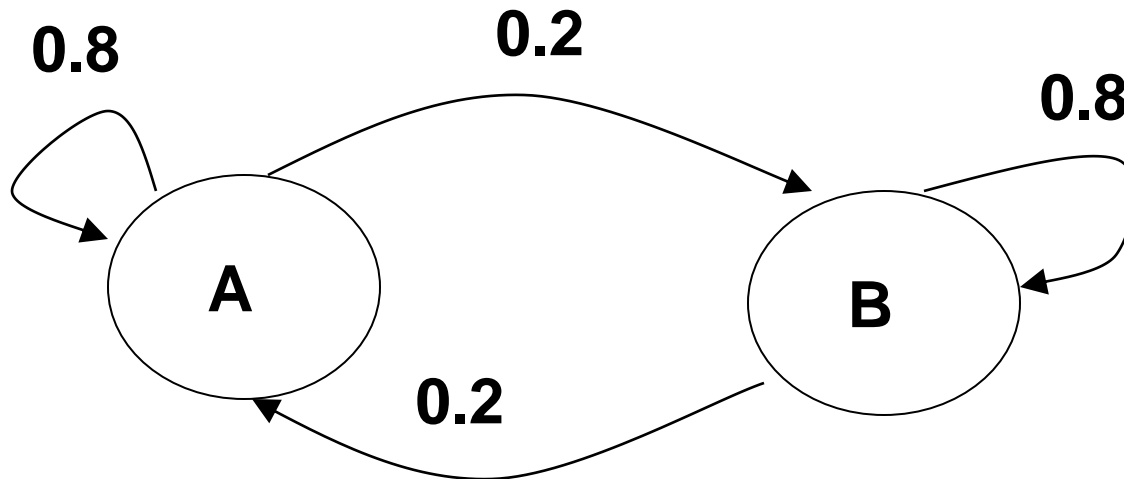
Contents

| | |
|--|----------------|
| <i>Preface</i> | <i>page ix</i> |
| 1 Introduction | 1 |
| 1.1 <i>Sequence similarity, homology, and alignment</i> | 2 |
| 1.2 <i>Overview of the book</i> | 2 |
| 1.3 <i>Probabilities and probabilistic models</i> | 4 |
| 1.4 <i>Further reading</i> | 10 |
| 2 Pairwise alignment | 12 |
| 2.1 <i>Introduction</i> | 12 |
| 2.2 <i>The scoring model</i> | 13 |
| 2.3 <i>Alignment algorithms</i> | 17 |
| 2.4 <i>Dynamic programming with more complex models</i> | 28 |
| 2.5 <i>Heuristic alignment algorithms</i> | 32 |
| 2.6 <i>Linear space alignments</i> | 34 |
| 2.7 <i>Significance of scores</i> | 36 |
| 2.8 <i>Deriving score parameters from alignment data</i> | 41 |
| 2.9 <i>Further reading</i> | 45 |
| 3 Markov chains and hidden Markov models | 46 |
| 3.1 <i>Markov chains</i> | 48 |
| 3.2 <i>Hidden Markov models</i> | 51 |
| 3.3 <i>Parameter estimation for HMMs</i> | 62 |
| 3.4 <i>HMM model structure</i> | 68 |
| 3.5 <i>More complex Markov chains</i> | 72 |
| 3.6 <i>Numerical stability of HMM algorithms</i> | 77 |
| 3.7 <i>Further reading</i> | 79 |
| 4 Pairwise alignment using HMMs | 80 |
| 4.1 <i>Pair HMMs</i> | 81 |
| 4.2 <i>The full probability of x and y, summing over all paths</i> | 87 |
| 4.3 <i>Suboptimal alignment</i> | 89 |
| 4.4 <i>The posterior probability that x_i is aligned to y_j</i> | 91 |
| 4.5 <i>Pair HMMs versus FSAs for searching</i> | 95 |

| | | |
|-----|--|-----|
| 4.6 | <i>Further reading</i> | 98 |
| 5 | Profile HMMs for sequence families | 100 |
| 5.1 | <i>Ungapped score matrices</i> | 102 |
| 5.2 | <i>Adding insert and delete states to obtain profile HMMs</i> | 102 |
| 5.3 | <i>Deriving profile HMMs from multiple alignments</i> | 105 |
| 5.4 | <i>Searching with profile HMMs</i> | 108 |
| 5.5 | <i>Profile HMM variants for non-global alignments</i> | 113 |
| 5.6 | <i>More on estimation of probabilities</i> | 115 |
| 5.7 | <i>Optimal model construction</i> | 122 |
| 5.8 | <i>Weighting training sequences</i> | 124 |
| 5.9 | <i>Further reading</i> | 132 |
| 6 | Multiple sequence alignment methods | 134 |
| 6.1 | <i>What a multiple alignment means</i> | 135 |
| 6.2 | <i>Scoring a multiple alignment</i> | 137 |
| 6.3 | <i>Multidimensional dynamic programming</i> | 141 |
| 6.4 | <i>Progressive alignment methods</i> | 143 |
| 6.5 | <i>Multiple alignment by profile HMM training</i> | 149 |
| 6.6 | <i>Further reading</i> | 159 |
| 7 | Building phylogenetic trees | 160 |
| 7.1 | <i>The tree of life</i> | 160 |
| 7.2 | <i>Background on trees</i> | 161 |
| 7.3 | <i>Making a tree from pairwise distances</i> | 165 |
| 7.4 | <i>Parsimony</i> | 173 |
| 7.5 | <i>Assessing the trees: the bootstrap</i> | 179 |
| 7.6 | <i>Simultaneous alignment and phylogeny</i> | 180 |
| 7.7 | <i>Further reading</i> | 188 |
| 7.8 | <i>Appendix: proof of neighbour-joining theorem</i> | 190 |
| 8 | Probabilistic approaches to phylogeny | 192 |
| 8.1 | <i>Introduction</i> | 192 |
| 8.2 | <i>Probabilistic models of evolution</i> | 193 |
| 8.3 | <i>Calculating the likelihood for ungapped alignments</i> | 197 |
| 8.4 | <i>Using the likelihood for inference</i> | 205 |
| 8.5 | <i>Towards more realistic evolutionary models</i> | 215 |
| 8.6 | <i>Comparison of probabilistic and non-probabilistic methods</i> | 224 |
| 8.7 | <i>Further reading</i> | 231 |

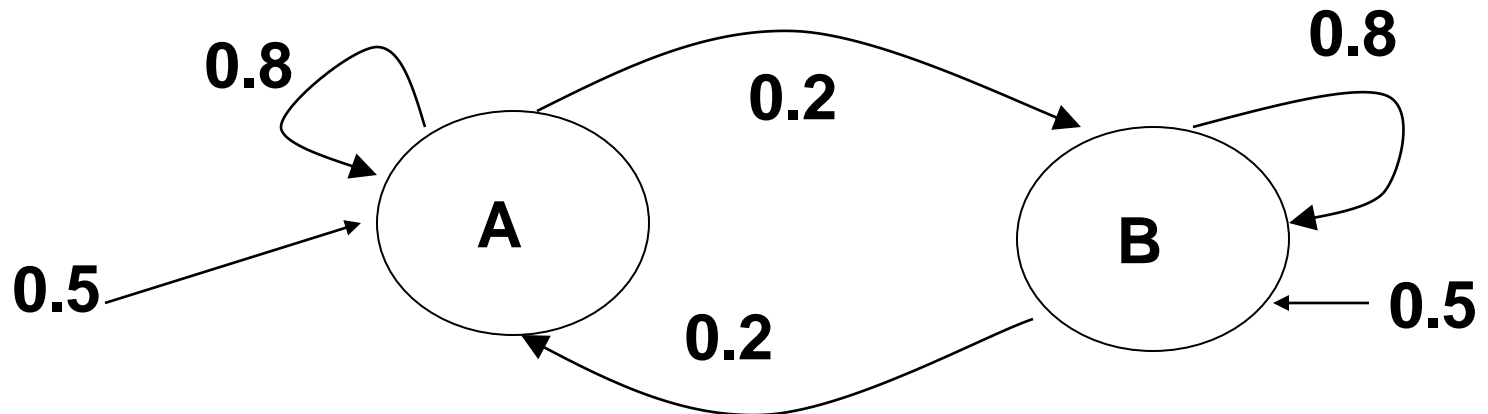
Example: Gambling on dice outcome

- Two dices, both skewed (output model).
- Can either stay with the same dice or switch to the second dice (transition mode).



A Hidden Markov model

- A set of states $\{s_1 \dots s_n\}$
 - In each time point we are in exactly one of these states denoted by q_t
- Π_i , the probability that we *start* at state s_i
- A transition probability model, $P(q_t = s_i \mid q_{t-1} = s_j)$
- A set of possible outputs Σ
 - At time t we emit a symbol $\sigma \in \Sigma$
- An emission probability model, $p(o_t = \sigma \mid s_i)$



The Markov property

- A set of states $\{s_1 \dots s_n\}$
 - In each time point we are in exactly one of these states denoted by q_t
- Π_j , the probability that we start at state s_j
- A transition probability model, $P(q_t = s_i | q_{t-1} = s_j)$

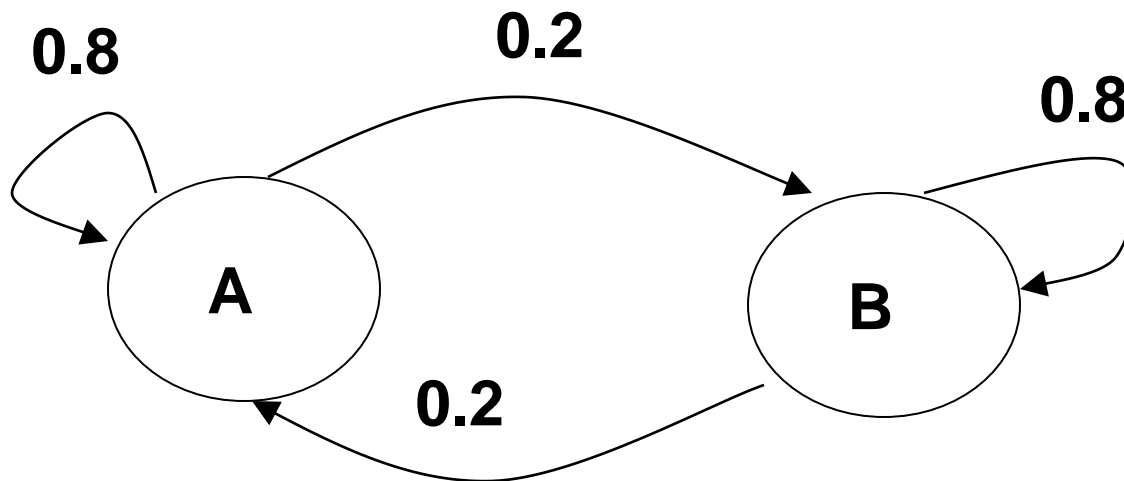
An important aspect of this definitions is the Markov property: q_{t+1} is conditionally independent of q_{t-1} (and any earlier time points) given q_t

More formally $P(q_{t+1} = s_i | q_t = s_j) = P(q_{t+1} = s_i | q_t = s_j, q_{t-1} = s_j)$

What can we ask when using a HMM?

A few examples:

- “What dice is currently being used?”
- “What is the probability of a 6 in the next role?”
- “What is the probability of 6 in any of the next 3 roles?”

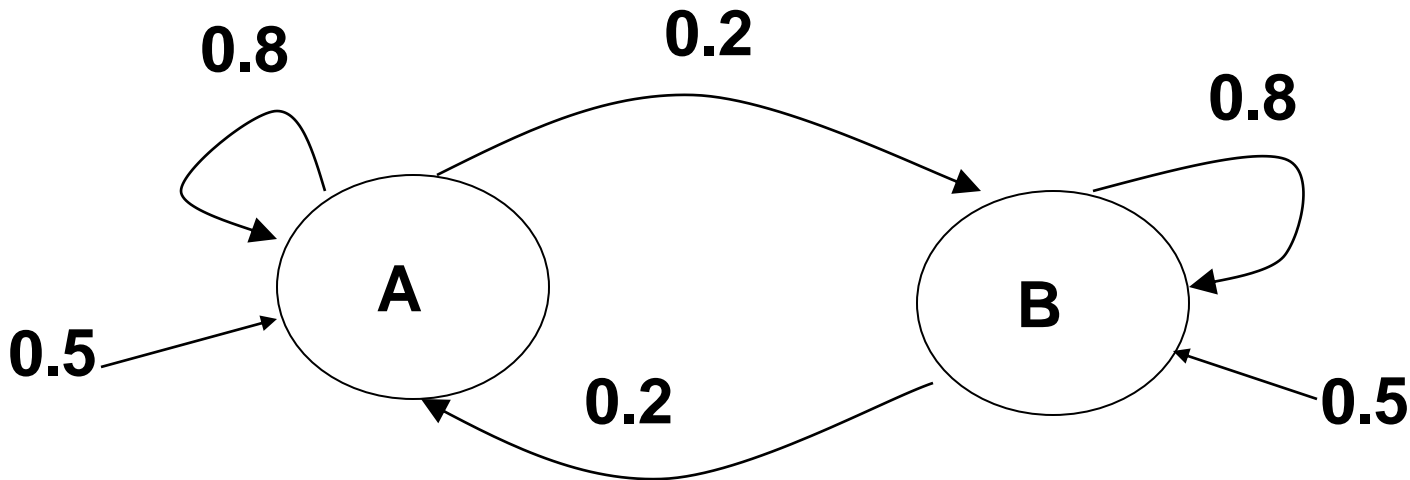


Inference in HMMs

- Computing $P(Q)$ and $P(q_t = s_i)$
 - If we cannot look at observations
- Computing $P(Q | O)$ and $P(q_t = s_i | O)$
 - When we have observation and care about the last state only
- Computing $\operatorname{argmax}_Q P(Q | O)$
 - When we care about the entire path

What dice is currently being used?

- We played t rounds so far
- We want to determine $P(q_t = A)$
- Lets assume for now that we cannot observe any outputs (we are blind folded)
- How can we compute this?



$P(q_t = A)?$

- Simple answer:

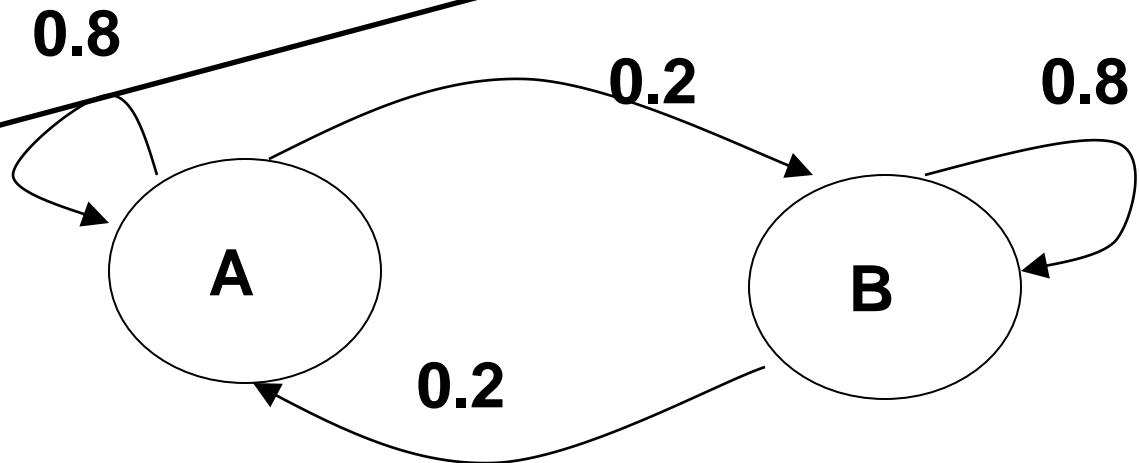
Lets determine $P(Q)$ where Q is any path that ends in A

$$Q = q_1, \dots, q_{t-1}, A$$

$$P(Q) = P(q_1, \dots, q_{t-1}, A) = P(A | q_1, \dots, q_{t-1}) P(q_1, \dots, q_{t-1}) =$$
$$P(A | q_{t-1}) P(q_1, \dots, q_{t-1}) = \dots = P(A | q_{t-1}) \dots P(q_2 | q_1) P(q_1)$$

Markov property!

Initial probability



$P(q_t = A)?$

- Simple answer:

1. Lets determine $P(Q)$ where Q is any path that ends in A

$$Q = q_1, \dots, q_{t-1}, A$$

$$\begin{aligned} P(Q) &= P(q_1, \dots, q_{t-1}, A) = P(A \mid q_1, \dots, q_{t-1}) P(q_1, \dots, q_{t-1}) = \\ &P(A \mid q_{t-1}) P(q_1, \dots, q_{t-1}) = \dots = P(A \mid q_{t-1}) \dots P(q_2 \mid q_1) P(q_1) \end{aligned}$$

2. $P(q_t = A) = \sum P(Q)$

where the sum is over all sets of t states that end in A

$P(q_t = A)$?

- Simple answer:

1. Lets determine $P(Q)$ where Q is any path that ends in A

$$Q = q_1, \dots, q_{t-1}, A$$

$$P(Q) = P(q_1, \dots, q_{t-1}, A) = P(A \mid q_1, \dots, q_{t-1}) P(q_1, \dots, q_{t-1}) = P(A \mid q_{t-1}) P(q_1, \dots, q_{t-1}) = \dots = P(A \mid q_{t-1}) \dots P(q_2 \mid q_1) P(q_1)$$

2. $P(q_t = A) = \sum P(Q)$

where the sum is over all sets of t states that end in A

Q: How many sets Q are there?

A: A lot! (2^{t-1})

Not a feasible solution

$P(q_t = A)$, the smart way

- Lets define $p_t(i)$ as the probability of being in state i at time t :
 $p_t(i) = p(q_t = s_i)$
- We can determine $p_t(i)$ by induction
 1. $p_1(i) = \Pi_i$
 2. $p_t(i) = ?$

$P(q_t = A)$, the smart way

- Lets define $p_t(i)$ = probability state i at time $t = p(q_t = s_i)$
- We can determine $p_t(i)$ by induction
 1. $p_1(i) = \Pi_i$
 2. $p_t(i) = \sum_j p(q_t = s_i \mid q_{t-1} = s_j)p_{t-1}(j)$

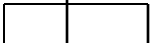
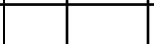

$P(q_t = A)$, the smart way

- Lets define $p_t(i) = \text{probability state } i \text{ at time } t = p(q_t = s_i)$
- We can determine $p_t(i)$ by induction
 1. $p_1(i) = \Pi_i$
 2. $p_t(i) = \sum_j p(q_t = s_i \mid q_{t-1} = s_j)p_{t-1}(j)$

This type of computation is called dynamic programming

Complexity: $O(n^2 \cdot t)$

Number of states in our HMM

| Time / state | t1 | t2 | t3 |
|--------------|----|--|---|
| s1 | .3 |  | |
| s2 | .7 |  |  |

Inference in HMMs

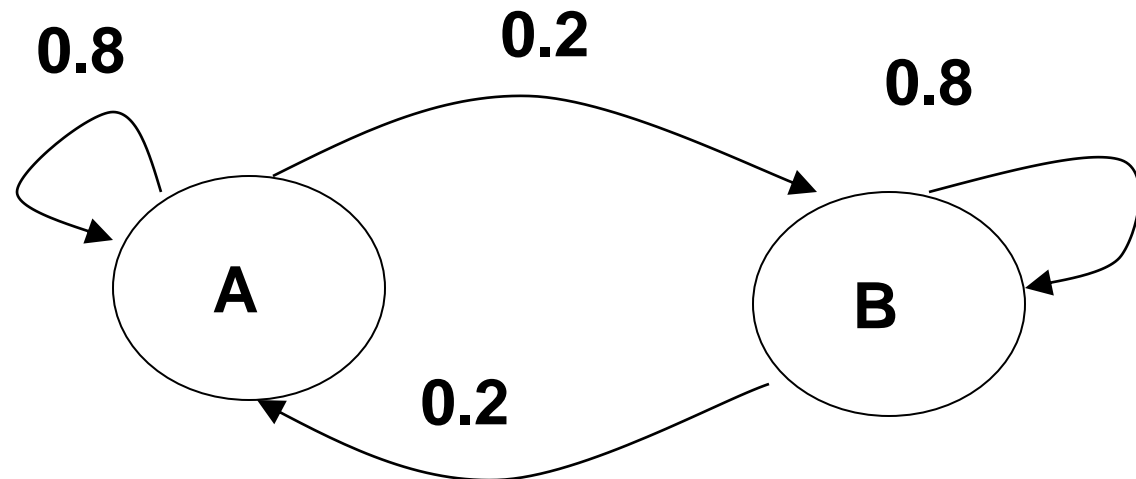
- Computing $P(Q)$ and $P(q_t = s_i)$ ✓
- Computing $P(Q | O)$ and $P(q_t = s_i | O)$
- Computing $\operatorname{argmax}_Q P(Q)$

But what if we observe outputs?

- So far, we assumed that we could not observe the outputs
- In reality, we almost always can.



| v | $P(v A)$ | $P(v B)$ |
|-----|------------|------------|
| 1 | .3 | .1 |
| 2 | .2 | .1 |
| 3 | .2 | .1 |
| 4 | .1 | .2 |
| 5 | .1 | .2 |
| 6 | .1 | .3 |



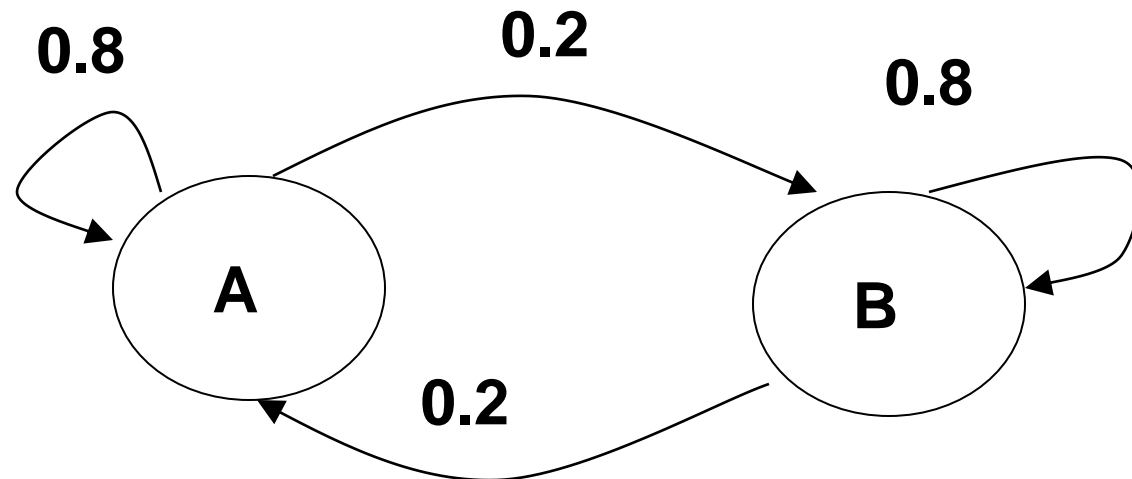
But what if we observe outputs?

- So far, we assumed that we could not observe the outputs
- In reality, we almost always do. Does observing the sequence

5, 6, 4, 5, 6, 6

Change our belief about the state?

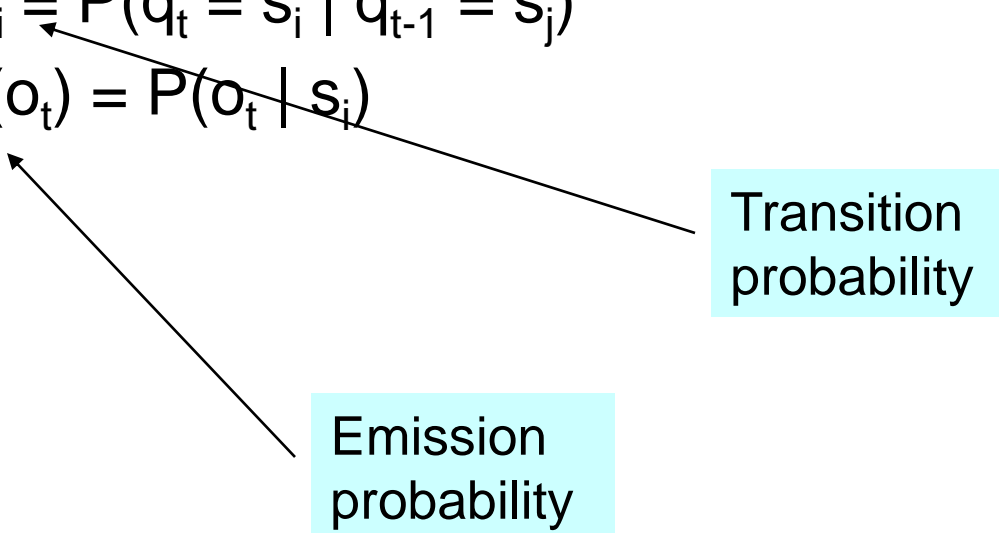
| v | $P(v A)$ | $P(v B)$ |
|---|------------|------------|
| 1 | .3 | .1 |
| 2 | .2 | .1 |
| 3 | .2 | .1 |
| 4 | .1 | .2 |
| 5 | .1 | .2 |
| 6 | .1 | .3 |



$P(q_t = A)$ when outputs are observed

- We want to compute $P(q_t = A \mid O_1 \dots O_t)$
- For ease of writing we will use the following notations (commonly used in the literature)
- $a_{j,i} = P(q_t = s_i \mid q_{t-1} = s_j)$
- $b_i(o_t) = P(o_t \mid s_i)$

Transition
probability



Emission
probability

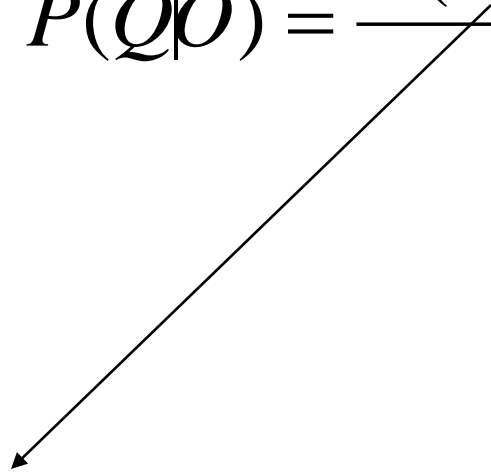
$P(q_t = A)$ when outputs are observed

- We want to compute $P(q_t = A \mid O_1 \dots O_t)$
- Lets start with a simpler question. Given a sequence of states Q , what is $P(Q \mid O_1 \dots O_t) = P(Q \mid O)$?
 - It is pretty simple to move from $P(Q)$ to $P(q_t = A)$
 - In some cases $P(Q)$ is the more important question
 - Speech processing
 - NLP

$P(Q | O)$

- We can use Bayes rule:

$$P(Q|O) = \frac{P(O | Q)P(Q)}{P(O)}$$



Easy, $P(O | Q) = P(o_1 | q_1) P(o_2 | q_2) \dots P(o_t | q_t)$

$P(Q | O)$

- We can use Bayes rule:

$$P(Q|O) = \frac{P(O | Q)P(Q)}{P(O)}$$

Easy, $P(Q) = P(q_1) P(q_2 | q_1) \dots P(q_t | q_{t-1})$

$P(Q | O)$

- We can use Bayes rule:

$$P(Q|O) = \frac{P(O | Q)P(Q)}{P(O)}$$



Hard!

P(O)

- What is the probability of seeing a set of observations:
 - An important question in it own rights, for example classification using two HMMs
- Define $\alpha_t(i) = P(o_1, o_2 \dots, o_t \wedge q_t = s_i)$
- $\alpha_t(i)$ is the probability that we:
 1. Observe $o_1, o_2 \dots, o_t$
 2. End up at state i

How do we compute $\alpha_t(i)$?

Computing $\alpha_t(i)$

- $\alpha_1(i) = P(o_1 \wedge q_t = i) = P(o_1 | q_t = s_i) \Pi_i$

We must be at a state in time t

$$\alpha_{t+1}(i) = P(O_1 \dots O_{t+1} \wedge q_{t+1} = s_i) =$$

chain rule

$$\sum_j P(O_1 \dots O_t \wedge q_t = s_j \wedge O_{t+1} \wedge q_{t+1} = s_i) =$$

$$\sum_j P(O_{t+1} \wedge q_{t+1} = s_i | O_1 \dots O_t \wedge q_t = s_j) P(O_1 \dots O_t \wedge q_t = s_j) =$$

Markov property

$$\sum_j P(O_{t+1} \wedge q_{t+1} = s_i | O_1 \dots O_t \wedge q_t = s_j) \alpha_t(j) =$$

$$\sum_j P(O_{t+1} | q_{t+1} = s_i) P(q_{t+1} = s_i | q_t = s_j) \alpha_t(j) =$$

$$\sum_j b_i(O_{t+1}) a_{j,i} \alpha_t(j)$$

Example: Computing $\alpha_3(B)$

- We observed 2,3,6

$$\alpha_1(A) = P(2 \wedge q_1 = A) = P(2 | q_1 = A)\Pi_A = .2 * .7 = .14, \alpha_1(B) = .1 * .3 = .03$$

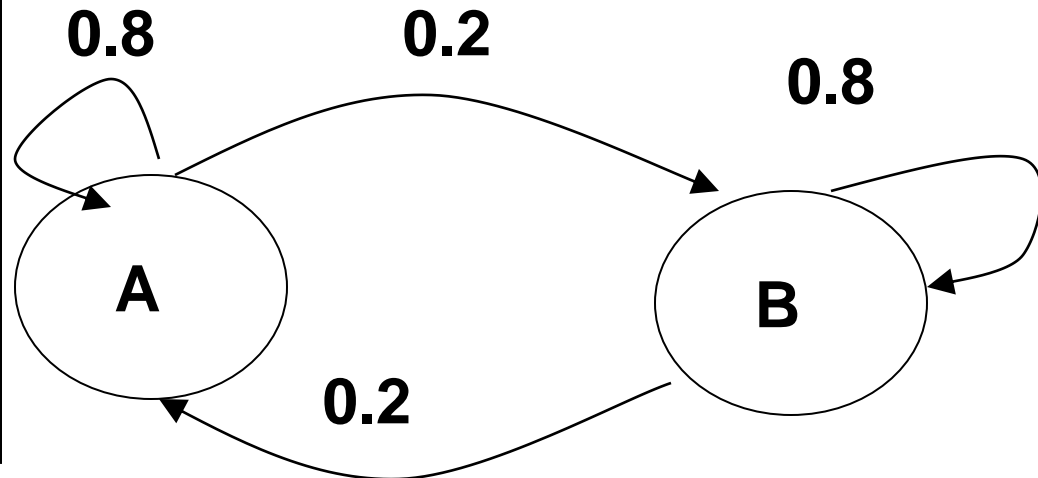
$$\alpha_2(A) = \sum_{j=A,B} b_A(3) a_{j,A} \alpha_1(j) = .2 * .8 * .14 + .2 * .2 * .03 = 0.0236, \alpha_2(B) = 0.0052$$

$$\alpha_3(B) = \sum_{j=A,B} b_B(6) a_{j,B} \alpha_2(j) = .3 * .2 * .0236 + .3 * .8 * .0052 = 0.00264$$

$$\Pi_A = 0.7$$

$$\Pi_B = 0.3$$

| v | P(v A) | P(v B) |
|---|----------|----------|
| 1 | .3 | .1 |
| 2 | .2 | .1 |
| 3 | .2 | .1 |
| 4 | .1 | .2 |
| 5 | .1 | .2 |
| 6 | .1 | .3 |



Where we are

- We want to compute $P(Q | O)$
- For this, we only need to compute $P(O)$
- We know how to compute $\alpha_t(i)$

From now its easy

$$\alpha_t(i) = P(o_1, o_2 \dots, o_t \wedge q_t = s_i)$$

so

$$P(O) = P(o_1, o_2 \dots, o_t) = \sum_i P(o_1, o_2 \dots, o_t \wedge q_t = s_i) = \sum_i \alpha_t(i)$$

note that

$$p(q_t=s_i | o_1, o_2 \dots, o_t) = \frac{\alpha_t(i)}{\sum_j \alpha_t(j)}$$

$$P(A | B) = P(A \wedge B) / P(B)$$

Complexity

- How long does it take to compute $P(Q | O)$?
- $P(Q)$: $O(n)$
- $P(O|Q)$: $O(n)$
- $P(O)$: $O(n^2t)$

Inference in HMMs

- Computing $P(Q)$ and $P(q_t = s_i)$ ✓
- Computing $P(Q | O)$ and $P(q_t = s_i | O)$ ✓
- Computing $\operatorname{argmax}_Q P(Q)$

Most probable path

- We are almost done ...
- One final question remains

How do we find the most probable path, that is Q^* such that

$$P(Q^* | O) = \operatorname{argmax}_Q P(Q|O)?$$

- This is an important path
 - The words in speech processing
 - The set of genes in the genome
 - etc.

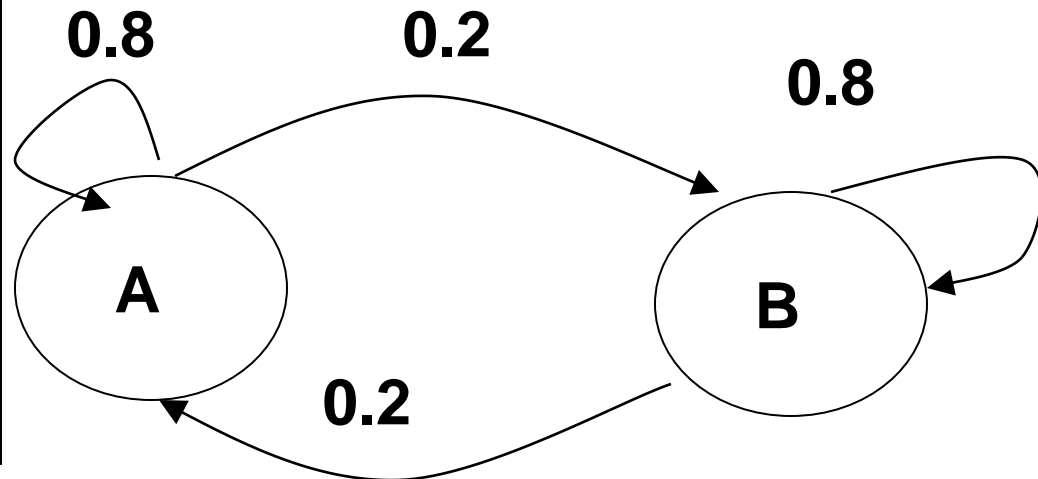
Example

- What is the most probable set of states leading to the sequence:

1,2,2,5,6,5,1,2,3 ?

$$\Pi_A=0.7$$
$$\Pi_b=0.3$$

| v | P(v A) | P(v B) |
|---|---------|---------|
| 1 | .3 | .1 |
| 2 | .2 | .1 |
| 3 | .2 | .1 |
| 4 | .1 | .2 |
| 5 | .1 | .2 |
| 6 | .1 | .3 |



Most probable path

$$\begin{aligned}\arg \max_Q P(Q | O) &= \arg \max_Q \frac{P(O | Q)P(Q)}{P(O)} \\ &= \arg \max_Q P(O | Q)P(Q)\end{aligned}$$

We will use the following definition:

$$\delta_t(i) = \max_{q_1 \dots q_{t-1}} p(q_1 \dots q_{t-1} \wedge q_t = s_i \wedge O_1 \dots O_t)$$

In other words we are interested in the most likely path from 1 to t that:

1. Ends in S_i
2. Produces outputs $O_1 \dots O_t$

Computing $\delta_t(i)$

$$\begin{aligned}\delta_1(i) &= p(q_1 = s_i \wedge O_1) \\ &= p(q_1 = s_i)p(O_1 | q_1 = s_i) \\ &= \pi_i b_i(O_1)\end{aligned}$$

$$\delta_t(i) = \max_{q_1 \dots q_{t-1}} p(q_1 \dots q_{t-1} \wedge q_t = s_i \wedge O_1 \dots O_t)$$

Q: Given $\delta_t(i)$, how can we compute $\delta_{t+1}(i)$?

A: To get from $\delta_t(i)$ to $\delta_{t+1}(i)$ we need to

1. Add an emission for time t+1 (O_{t+1})
2. Transition to state s_i

$$\begin{aligned}\delta_{t+1}(i) &= \max_{q_1 \dots q_t} p(q_1 \dots q_t \wedge q_{t+1} = s_i \wedge O_1 \dots O_{t+1}) \\ &= \max_j \delta_t(j) p(q_{t+1} = s_i | q_t = s_j) p(O_{t+1} | q_{t+1} = s_i) \\ &= \max_j \delta_t(j) a_{j,i} b_i(O_{t+1})\end{aligned}$$

The Viterbi algorithm

$$\begin{aligned}\delta_{t+1}(i) &= \max_{q_1 \dots q_t} p(q_1 \dots q_t \wedge q_{t+1} = s_i \wedge O_1 \dots O_{t+1}) \\ &= \max_j \delta_t(j) p(q_{t+1} = s_i | q_t = s_j) p(O_{t+1} | q_{t+1} = s_i) \\ &= \max_j \delta_t(j) a_{j,i} b_i(O_{t+1})\end{aligned}$$

- Once again we use dynamic programming for solving $\delta_t(i)$
- Once we have $\delta_t(i)$, we can solve for our $P(Q^*|O)$

By:

$$P(Q^* | O) = \operatorname{argmax}_Q P(Q|O) = P(Q^* | O) =$$

path defined by $\operatorname{argmax}_j \delta_t(j)$,

Inference in HMMs

- Computing $P(Q)$ and $P(q_t = s_i)$ ✓
- Computing $P(Q | O)$ and $P(q_t = s_i | O)$ ✓
- Computing $\operatorname{argmax}_Q P(Q)$ ✓

What you should know

- Why HMMs? Which applications are suitable?
- Inference in HMMs
 - No observations
 - Probability of next state w. observations
 - Maximum scoring path (Viterbi)