Hidden Markov Models

Gil McVean
Thursday 21st February 2008

The occasionally dishonest casino

- Suppose a casino typically uses a fair die, but every now and then switches to using one that biases to throwing 6s
- What we observe is the score from successive throws, but what we would like to make inferences about is when the casino is being dishonest

```
5 3 1 5 3 4 6 5 6 3 2 6 6 2 1 2 4 3 2 5 3 2 2 6
H H H H H H D D D D D D D H H H H H H H H H H H
```

- If the underlying states (honest, dishonest) evolve in a Markov fashion and the observed variables depend only on the underlying state in a simple probabilistic fashion the system is referred to as a **hidden Markov model**
  - Note that the observed variables are not Markov

The naïve approach

- We want to calculate the likelihood of the data
- The simple approach is to sum over all possible underlying states. Suppose I'd just seen the variables 1,6,4

Path	P(Path)	P(Data | Path)
HHH	π(H)q(HH)q(HH)	P(1|H)P(6|H)P(4|H)
HHD	π(H)q(HH)q(HD)	P(1|H)P(6|H)P(4|D)
HDH	π(H)q(HD)q(DH)	P(1|H)P(6|D)P(4|H)
DHH	π(D)q(DH)q(HH)	P(1|D)P(6|H)P(4|H)
HDD	π(H)q(HD)q(DD)	P(1|H)P(6|D)P(4|D)
DHD	π(D)q(DH)q(HD)	P(1|D)P(6|H)P(4|D)
DDH	π(D)q(DD)q(DH)	P(1|D)P(6|D)P(4|H)
DDD	π(D)q(DD)q(DD)	P(1|D)P(6|D)P(4|D)

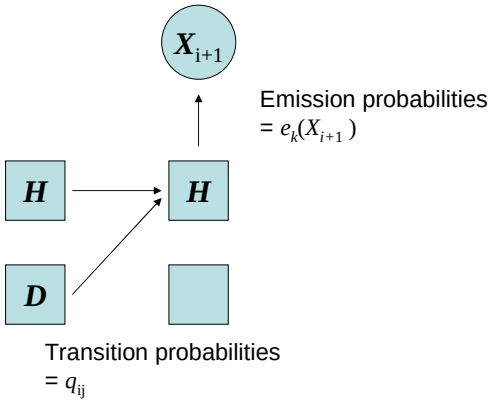
Transition probabilities and 'emission' probabilities

$$P(D \mid \theta) = \sum_{Paths} \Pr(Path \mid \theta) \Pr(D \mid \theta)$$

π(A) = Probability of starting in state A **q(AB) = Probability of transition from state A to state B**

Dynamic programming

- If I have x possible underlying states at each point and l points, there are x^l possible paths. This number rapidly gets unworkable.
- However, we can use a simple recursion to sum over all paths that requires only order xl operations
  - This is called **dynamic programming**
- The trick is to pretend we know how to calculate the likelihood conditional on being in state j at the ith position
- The likelihood conditional on being in state k at the i+1th position is the sum over all states (j) at position i of the conditional likelihood multiplied by the probability of going from state j to state k multiplied by the probability of observing the data at position i+1 given state k



$$f_H(i+1) = [f_H(i)q_{HH} + f_D(i)q_{DH}] \times e_H(X_{i+1})$$

Finishing the algorithm

- The total likelihood for a sequence of length l is

$$L = \sum_k f_k(l)$$

- The algorithm that works from left to right along the sequence is called the **forward algorithm**

$$f_j(i+1) = e_j(X_{i+1}) \sum_k f_k(i) q_{kj}$$

- There is an equivalent algorithm, called the **backwards algorithm**, that works from right to left

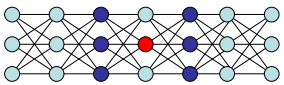
$$b_j(i) = \sum_k b_k(i+1) q_{jk} e_k(X_{i+1})$$

- The algorithms are initialised by setting $f(0) = b(\mathrm{L}) = 1$ and $q_{0j} = \pi_j, q_{j0} = 1$

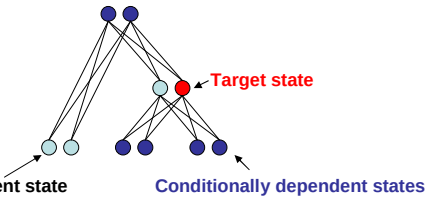
Why does it work?

- The dynamic programming algorithms exploit the 'conditional independence' inherent in the model structure
- This can be most clearly seen through graphical representations of models

Linear models



Branching models



Using the likelihood functions

- We can use the likelihood function to estimate parameters and test hypotheses
- HMMs are a special case of **missing data** problems, where the underlying 'hidden' states are the missing data
  - The **EM algorithm** can be used to estimate parameters by maximum likelihood
- Another common use of HMMs is to make statements about the missing data
  - What is the most probable sequence of the underlying state?
  - What is the probability that the underlying state at position i is k?

The Viterbi algorithm

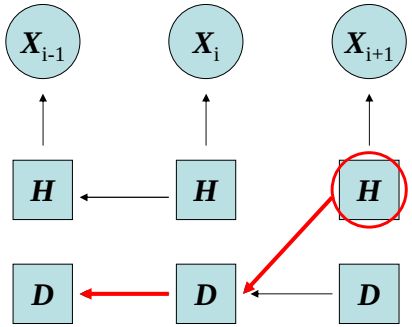
- An algorithm very similar to the forward algorithm can be used to give the most likely sequence of hidden states (the maximum likelihood sequence): the **Viterbi** algorithm
- Again, the trick is to say that we know the ML path up to position *i* that ends in state *j* (for all *j*). The ML path ending in state *k* at position *i*+1 is just the maximum over the paths ending in state *j* at position *i* multiplied by the transition probabilities

$$v_k(i+1) = e_k(X_{i+1}) \max_j \left[v_j(i) q_{jk} \right]$$

- A **traceback** matrix keeps track of which is the most likely path

$$t_k(i+1) = \arg\max_j \left[v_j(i) q_{jk} \right]$$

- Again, the ML path can be found from $\max_k [v_k(l)]$



$$v_k(i+1) = e_k(X_{i+1}) \max_j \left[v_j(i) q_{jk} \right]$$

$$t_k(i+1) = \arg\max_j \left[v_j(i) q_{jk} \right]$$

In practice

5 3 1 5 3 4 6 5 6 3 2 6 6 2 1 2 4 3 2 5 3 2 2 6

$e_H(1) = e_H(2) = e_H(3) = e_H(4) = e_H(5) = e_H(6) = 1/6$
$e_D(1) = e_D(2) = e_D(3) = e_D(4) = e_D(5) = 1/9 \quad e_D(6) = 4/9$
$q_{DH} = q_{HD} = 0.1$
$\pi_H = \pi_D = 0.5$

Most likely path

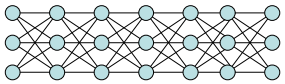
5 3 1 5 3 4 6 5 6 3 2 6 6 2 1 2 4 3 2 5 3 2 2 6
H H H H H H H H H H H H H H H H H H H H H H H H

But contribution of ML path to total likelihood = $\dfrac{\max_k [v_k(l)]}{L}$ **= 0.07**

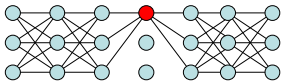
Posterior probabilities

- Instead of asking what the most likely path is, we can ask about the probability that the hidden chain had a particular state, *k*, at a particular step, *i*
- To do this we need to calculate the likelihood associated with all paths that go through state *k* at *i*, and normalise by the total likelihood

Paths in likelihood



Paths in conditional likelihood

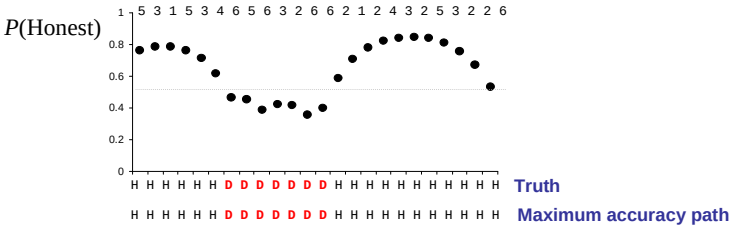


Calculating posterior probabilities

- To calculate this conditional probability we can make use of the two quantities we have already calculated using the forward and backward algorithms

$$P(X_i = k) = \frac{f_k(i)b_k(i)}{L}$$

- The path made from selecting the states with the highest posterior probability (marginally) is called the **maximum accuracy path**



Applications of HMMs

- Bioinformatics
  - Gene finding
  - Gene annotation
  - RNA structure prediction
  - Aligning divergent sequences
  - Modelling genetic variation
  - Inferring genealogical histories
- Automated speech recognition
- 'Segmenting' noisy measurements