README: Hidden Markov Models

September 30, 2015

1 Hidden Markov Models

Simply: A Hidden Markov Model (HMM) is a Markov Model (chain) where you observe some random outcome depending on the state, rather than the state as you would in a normal Markov Model.

1.1 Crooked Casino

e.g. If a casino was randomly switching out a crooked biased die for a fair die every so often then the observation would be the roll rather than which die it was.

1.2 Elements of HMM

HMM is made up of a few different vectors of values

1.2.1 State Vector

All possible states of the Markov Chain are given by

$$S = \{S_1, S_2, ..., S_N\} \tag{1}$$

and the state at time t is given by q_t

1.2.2 Observations

All possible observations are given by

$$\mathbf{V} = \{v_1, v_2, ..., v_M\} \tag{2}$$

and a sequence of observed values for $1 \leq t \leq T$ is

$$\mathbf{O} = \{O_1, O_2, ..., O_T\} \tag{3}$$

1.2.3 State Transition Probability Matrix

THe probability from going from state i to state j over a time step is a_{ij} and these form a matrix A. Technical definition of a_{ij} is

$$a_{ij} = P\left(q_{t+1} = S_j | q_t = S_i\right), \quad 1 \le i, j \le N \tag{4}$$

Note: this means going from row i to column j

1.2.4 Observation Probability Matrix

This denotes the probability of seeing the symbol v_k in state j. It's denoted in the matrix B as $B = \{b_j(k)\}$ where

$$b_j(k) = P\left(v_k \text{ at t } | q_t = S_j\right), \quad 1 \le j \le N, \quad 1 \le k \le M \tag{5}$$

1.2.5 Initial Probs

Set the probability of each state being chosen initially to

$$\pi = \{\pi_1, \pi_2, ...\pi_N\} \tag{6}$$

where π_i is the probability of the i^{th} state being the initial state

1.3 Coding

There's a simulation function written in R I did in $hiddenMarkovModelSimulations.R$ that should help. Theres two scenarios set up regarding a crooked casino and a DNA coding region thing. see $jono'sSlidesonHMM.pdf$ Coding Challenges for the scenario.

2 Forward Algorithm

The forward algorithm allows us to calculate the probability of seeing a set of observations **O** given a parameter set λ, $P(\mathbf{O}|\lambda)$. Define a forward variable

$$a_t(i) = P(O_1, ..., O_t, q_t = S_i | \lambda) \tag{7}$$

which is the probability of a sequence of observations up to O_t and the state being S_i at time t. Then we can calculate λ, $P(\mathbf{O}|\lambda)$ using a recursive algorithm as follows

2.1 Initialisation

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N \tag{8}$$

2.2 Induction

$$a_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \tag{9}$$

$$1 \le t \le T-1 \quad \text{and} \quad 1 \le j \le N \tag{10}$$

2.3 Termination

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{11}$$

3 Backward Algorithm

The backward algorithm lets you calculate the probabilities of a sequence from a certain time position onwards to the end. Quote: "the probability of the partial observation sequence from t+1 to the end, given state S_i at time t and the model λ". We do this by introducing a new variable

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, ..., O_T | q_t = S_i, \lambda) \tag{12}$$

3.1 Initialisation

$$\beta_T(i) = 1, \quad 1 \le i \le N \tag{13}$$

The end point are all 1 as with no observations afterwards they're all 100% likely

3.2 Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \tag{14}$$

$$1 \le i \le N, \quad \text{and} \quad t = (T-1), (T-2), ..., 1 \tag{15}$$

3.3 Is this useful

Not at the moment, apparently it applies in the third part of HMM which is adjusting the parameters $\lambda = (A, B, \pi)$

4 Viterbi Algorithm

Viterbi finds the sequence which has the highest probability of occuring given we know the parameter space $\lambda = \{A, B, \pi\}$ and the set of outcomes O.

It works recursively forward calculating these probabilities then at the end backtracks over the data and collects the highest prob sequence.

It calculates a value $\delta_t(i)$ which is the maximum probability of seeing a state $q_t = i$ by considering all possible sequences $q_1, ..., q_{t-1}$ that lead up to it. It's defined as

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P\left(q_1, q_2, ..., q_{t-1}, q_t = i, O_1, ..., O_t \middle| \lambda\right) \tag{16}$$

So we already know the observations $\mathbf{O}$ and we want to find the maximum Prob for $q_t = i$ by varying all possible sequences $q_1, ..., q_{t-1}$. We use a clever recursive algorithm to get away from the enormous amounts of calculations brute forcing this would take.

DEFINE: $\psi_t(i)$ is the state that maximised the probability of going from the previous time step to the current time step if $q_t = i$.

The algorithm works as follows

4.1 Initialisation

Set the initial values

$$\delta_1(i) = \pi_i b_i(O_1) \tag{17}$$

$$\psi_1(i) = 0 \tag{18}$$

$\psi_1(i)$ is never used so this is just to fill the vector, could be ignored easily

4.2 Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1} a_{ij}]\, b_i(O_t) \quad \text{where} \quad 2 \leq t \leq T \tag{19}$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_{t-1} a_{ij}] \quad \text{where} \quad 1 \leq j \leq N \tag{20}$$

Basically calculate all the probabilities of changing from all possible previous states to the current state with the observed observation. Also write down what state that was. Do this for all possible states at the current step.

4.3 Terminate the loop

Record the best prob as p^* and the best last state as q_T^*

$$p^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{21}$$

$$q_T^* = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_T(i)] \tag{22}$$

4.4 Backtrack the data and find the state sequence

Now we've recorded all the probabilities of each time step being at any state, we can backtrack over the data and find what the best one was at each point and we've got the best possible sequence

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad \text{for} \quad t = (T-1), (T-2), ..., 1 \tag{23}$$

Then the vector $\mathbf{q}^* = \{q_1^*, q_2^*, ..., q_T^*\}$ is the vector of the most probable states for each time step

4.5 Have I coded this anywhere?

YES! See *viterbi.R* and *testViterbi.R* where I've written the algorithm in R and a test case similar to the HMM lecture 2 given by Jono. Problems do exist:

- Don't use a lot of observations or the probabilities just round down to 0 in the computer and nothing useful happens
- The max probs can repeat and it hasn't been programmed to deal with that case, it'll just give a friendly error message and tell you to sod off.