Hidden Markov Models: lecture 3

Local decoding

Xavier Didelot

HMM definition

- A Hidden Markov Model (HMM) is a Markov chain in which the sequence of states $C_1, ..., C_T$ is not observed but hidden
- Instead of observing the sequence of states, we observe the emissions $X_1, ..., X_T$
- A HMM is defined by two quantities:
  - The transition matrix $\mathbf{\Gamma}$ of elements γ_{ij} where i and j are states:

    $$\gamma_{ij} = p(C_t = j | C_{t-1} = i)$$

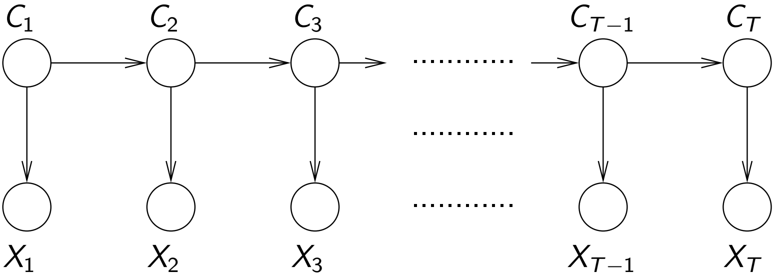
  - The emission probabilities $p_i(x)$ where i is a state and x is an emission:

    $$p_i(x) = p(X_t = x | C_t = i)$$

- The unconditional distribution at t is denoted $\boldsymbol{u}(t)$ and the initial distribution is $\boldsymbol{u}(1)$

$$\boldsymbol{u}(t) = (p(C_t = 1), p(C_t = 2), ..., p(C_t = m))$$

Dependency graph of a hidden Markov model



$$p(\boldsymbol{X}^{(T)}, \boldsymbol{C}^{(T)}) = p(C_1) \prod_{k=2}^{T} p(C_k | C_{k-1}) \prod_{k=1}^{T} p(X_k | C_k)$$

$$p(\boldsymbol{x}^{(T)}, \boldsymbol{c}^{(T)}) = u_{c_1}(1) \prod_{k=2}^{T} \gamma_{c_{k-1} c_k} \prod_{k=1}^{T} p_{c_k}(x_k)$$

Forward recursion

- Define the vector $\boldsymbol{\alpha}_t$ such that

$$\alpha_t(j) = p(\boldsymbol{X}^{(t)} = \boldsymbol{x}^{(t)}, C_t = j)$$

- In particular:

$$\alpha_1(j) = p(X_1 = x_1, C_1 = j) = u_j(1)p_j(x_1)$$

$$\boldsymbol{\alpha}_1 = \boldsymbol{u}(1)\boldsymbol{P}(x_1)$$

- We have the recursion:

$$\alpha_t(j) = \sum_{k=1}^{m} \alpha_{t-1}(k)\gamma_{kj}p_j(x_t)$$

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}\boldsymbol{\Gamma}\boldsymbol{P}(x_t)$$

- We can used the forward algorithm to calculate the values of $\boldsymbol{\alpha}_t$ iteratively, and to compute the likelihood $L_T = \boldsymbol{\alpha}_T\mathbf{1}'$
- Computational complexity is $O(Tm^2)$

Filtering and smoothing

- The distribution $p(C_T|\boldsymbol{X}^{(T)})$ of the hidden state at the end of the observed sequence is often of interest
- This problem is called **filtering** and can be solved using the forward algorithm
- By definition $\alpha_T(j) = p(\boldsymbol{X}^{(T)} = \boldsymbol{x}^{(T)}, C_T = j)$ and therefore:

$$p(C_T = j|\boldsymbol{X}^{(T)}) = \frac{\alpha_T(j)}{\sum_{i=1}^m \alpha_T(i)} = \frac{\alpha_T(j)}{L_T}$$

- More generally, we might want to know the distribution $p(C_t|\boldsymbol{X}^{(T)})$ of hidden state at some point in the observed sequence
- This problem is called **smoothing** but can't be solved just using the forward recursion on its own
- The forward recursion reveals $\alpha_t(j) = p(\boldsymbol{X}^{(t)} = \boldsymbol{x}^{(t)}, C_t = j)$ but we can not deduce directly $p(C_t|\boldsymbol{X}^{(T)})$, except for $t = T$

Decoding terminology

- The smoothing problem is also called **local decoding**
- Decoding means to find the values of the hidden states. Local decoding means we find the value at a given point, as opposed to **global decoding** where we try and find the joint distribution of hidden states at all timepoints
- Note that global decoding is not the same as applying local decoding to all timepoints one by one because the states are not independent
- Global decoding will be covered in the next lecture, for now we focus on local decoding, ie finding $p(C_t|\boldsymbol{X}^{(T)})$

Backward recursion

- Define the vector $\boldsymbol{\beta}_t$ such that

$$\begin{aligned}\beta_t(i) &= p(X_{t+1}=x_{t+1}, X_{t+2}=x_{t+2}, \dots, X_T=x_T | C_t=i) \\ &= p(\boldsymbol{X}_{t+1}^T = \boldsymbol{x}_{t+1}^T | C_t=i)\end{aligned}$$

- In particular:

$$\beta_{T-1}(i) = p(X_T=x_T | C_{T-1}=i) = \sum_{k=1}^m \gamma_{ik} p_k(x_T)$$

- In matrix format: $\boldsymbol{\beta}'_{T-1} = \boldsymbol{\Gamma P}(x_T)\mathbf{1}'$
- We have the recursion:

$$\begin{aligned}\beta_t(i) &= \sum_{k=1}^m p(X_{t+1}=x_{t+1}, \boldsymbol{X}_{t+2}^T=\boldsymbol{x}_{t+2}^T, C_{t+1}=k | C_t=i) \\ &= \sum_{k=1}^m \beta_{t+1}(k)\gamma_{ik}p_k(x_{t+1})\end{aligned}$$

- In matrix format: $\boldsymbol{\beta}'_t = \boldsymbol{\Gamma P}(x_{t+1})\boldsymbol{\beta}'_{t+1}$

Backward algorithm

- We deduce the **backward algorithm** which has similar properties to the forward algorithm from the previous lecture:
  - Set $\beta'_{T-1} = \mathbf{\Gamma} \boldsymbol{P}(x_T)\mathbf{1}'$
  - For t from $T-2$ down to 1, calculate $\beta'_t = \mathbf{\Gamma} \boldsymbol{P}(x_{t+1})\beta'_{t+1}$
  - Return the likelihood $L_T = \boldsymbol{u}(1)\boldsymbol{P}(x_1)\beta'_1$
- This algorithm calculates the likelihood in an alternative manner to the forward algorithm, which is redundant, but it is important because of the next slide

Combining forward and backward values

- We have:

$$\begin{aligned}\alpha_t(i)\beta_t(i) &= p(\boldsymbol{X}^{(t)}, C_t = i)p(\boldsymbol{X}_{t+1}^{T}|C_t = i) \\ &= p(C_t = i)p(\boldsymbol{X}^{(t)}|C_t = i)p(\boldsymbol{X}_{t+1}^{T}|C_t = i) \\ &= p(\boldsymbol{X}^{(T)} = \boldsymbol{x}^{(T)}, C_t = i)\end{aligned}$$

- Therefore:

$$\boldsymbol{\alpha}_t\boldsymbol{\beta}_t' = p(\boldsymbol{X}^{(T)} = \boldsymbol{x}^{(T)}) = L_T$$

- We now have T redundant ways of calculating L_T
- More importantly:

$$p(C_t = i|\boldsymbol{X}^{(T)} = \boldsymbol{x}^{(T)}) = \frac{p(C_t = i, \boldsymbol{X}^{(T)} = \boldsymbol{x}^{(T)})}{p(\boldsymbol{X}^{(T)} = \boldsymbol{x}^{(T)})} = \alpha_t(i)\beta_t(i)/L_T$$

- We can therefore combine the forward and backward values to perform local decoding

Forward-backward algorithm

- The forward-backward algorithm can be used to perform local decoding
- It is simply the combination of the forward and backward algorithms
- Firstly we run the forward algorithm to calculate the values of $\boldsymbol{\alpha}_t$ and the likelihood L_T
- Secondly we run the backward algorithm to calculate the values of $\boldsymbol{\beta}_t$
- Finally we compute the values of $p(C_t = i | \boldsymbol{X}^{(T)})$ using:

$$p(C_t = i | \boldsymbol{X}^{(T)} = \boldsymbol{x}^{(T)}) = \alpha_t(i)\beta_t(i)/L_T$$

- The path made from selecting the states with the highest marginal probability is called the **maximum accuracy path**

Example: the occasionally dishonest casino

- Suppose a casino typically uses a fair die, but every now and then switches to loaded one with increased probability of throwing 6s
- We observe the scores from successive throws
- We want to know when the casino is being dishonest

Example: the occasionally dishonest casino

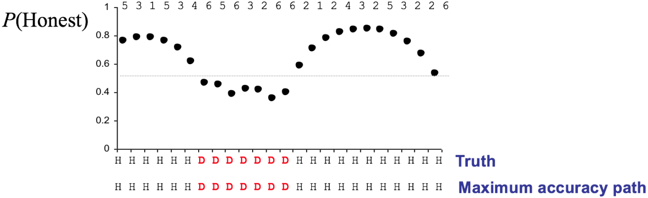
- State H is honest, state D is dishonest
- The emissions are categorical, so the model is a multinomial-HMM
- $p_H(1) = p_H(2) = p_H(3) = p_H(4) = p_H(5) = p_H(6) = 1/6$
- $p_D(1) = p_D(2) = p_D(3) = p_D(4) = p_D(5) = 1/9$ and $p_D(6) = 4/9$
- $\gamma_{HD} = \gamma_{DH} = 0.1$ and $\gamma_{HH} = \gamma_{DD} = 0.9$
- Observed values:

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

- True unobserved sequence of states:

  `H H H H H H D D D D D D D H H H H H H H H H H H`

Example: the occasionally dishonest casino



- In this case the maximum accuracy path is exactly equal to the true path!
- But it does not mean that it is the always the best path
- In fact, the maximum accuracy path can even end up having a probability of zero!
- To find the best path globally, we need to perform global decoding

Conclusions

- The forward algorithm can be used to calculate the likelihood in a HMM
- The backward algorithm can do the same thing and works in the same way, but goes backwards (from T to 1 rather than from 1 to T)
- Combining the forward and backward algorithm reveals the marginal probability of the hidden states, thus allowing to perform smoothing aka local decoding
- Next time we will ask ourselves how to perform global decoding