Hidden Markov Models: lecture 5

Parameter estimation

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:

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

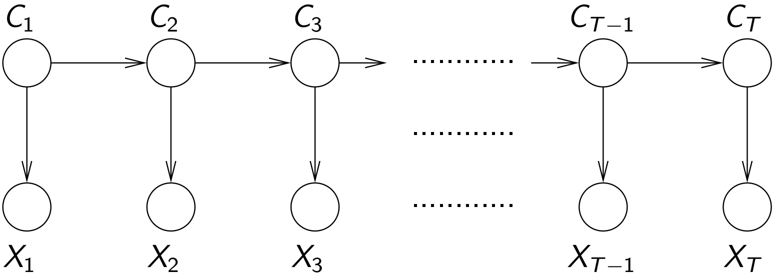
  - The emission probabilities $p_i(x)$ where i is a state and x is an emission:

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

- 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



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

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

Parameter estimation

- In the previous lectures, we discussed how to calculate the likelihood, and how to estimate the hidden states
- This assumed that we knew in advance the value of the transition matrix $\mathbf{\Gamma}$ and emission probabilities $p_i(x)$
- In this lecture we are concerned with estimation of these parameters θ
- We still consider that the structure of the HMM is known, especially the number of states
- This will be addressed in the next lecture

Notation

- The following notations are going to be convenient in this lecture:
- $u_j(t) = 1$ if and only if $c_t = j$ $(t = 1, 2, ..., T)$
- $v_{jk}(t) = 1$ if and only if $c_{t-1} = j$ and $c_t = k$ $(t = 2, 3, ..., T)$
- If we know the c_t then we can compute the $u_j(t)$ and $v_{jk}(t)$
- The likelihood can be rewritten using these notations

Using training data with known states

- Sometimes we have training data where the hidden states are known
- In machine learning terminology, this is supervised learning
- For example, in speech recognition, we may have a training dataset in which we know the spoken sentences
- In this case, we can count the number $f_{jk} = \sum_{t=2}^{T} v_{jk}(t)$ of transitions from state j to state k, and the transition probability can be estimated as:

$$\widehat{\gamma}_{jk} = \frac{f_{jk}}{\sum_{i=1}^{m} f_{ji}} \tag{1}$$

- This corresponds to the maximum likelihood estimate of γ_{jk} given $\boldsymbol{x}^{(T)}$ and $\boldsymbol{c}^{(T)}$

Using training data with known states

- The emission probabilities $p_k(x)$ can be estimated likewise via maximum likelihood for the positions where the state was k
- The exact form depends on the type of emission function $p_k(x)$
- For example if the emission distribution in state k is a Poisson distribution with mean λ_k (eg earthquake example), we have:

$$\widehat{\lambda}_k = \frac{\sum_{t=1}^{T} x_t u_k(t)}{\sum_{t=1}^{T} u_k(t)} \tag{2}$$

- If the emissions are discrete with $e_k(x)$ being the probability of emitting x in state k (eg casino example), then:

$$\widehat{e}_k(x) = \frac{\sum_{t=1}^{T} \mathbf{1}(x_t = x) u_k(t)}{\sum_{t=1}^{T} u_k(t)} \tag{3}$$

Maximising the likelihood

- In a previous lecture, we showed that the likelihood $L_T = p(\boldsymbol{X}^{(T)} = \boldsymbol{x}^{(T)})$ can be calculated using the forward algorithm
- We can therefore estimate the parameters using a standard numerical maximisation technique
- For example the R command `optim`
- Risk of numerical underflow if the likelihood is calculated without directly without scaling or transformation
- There are constraints to satisfy, especially the fact that the rows of $\boldsymbol{\Gamma}$ must add up to one
- Risk of convergence to a local rather than global maximum

Baum-Welch Algorithm

- The Baum-Welch algorithm is a very popular alternative approach to estimate the parameters, first described by Leonard Baum and colleagues in 1970
- It is a special case of the Expectation-Maximisation (EM) algorithm
- The Baum-Welch algorithm was described before the general EM algorithm in 1977
- First we will describe the general EM algorithm

Expectation-maximisation

- The EM algorithm is a general algorithm for maximum-likelihood estimation with missing data
- We want to estimate the parameters θ given some data x
- The likelihood $p(x|\theta)$ is complicated due to missing data
- But if there was no missing data, the likelihood would be a relatively simple function $f(\theta)$
- We initiate the EM algorithm with a starting value for θ
- The E step and M step are repeated until convergence is reached
- E step: Compute the expectation of the (functions of the) missing data that appear in $f(\theta)$
- M step: Find the θ that maximises $f(\theta)$ in which the (functions of the) missing data are replaced by their expectations computed in the E step.

Application of EM to the mixture of two Poisson

- We want to estimate the parameters $\theta = \{\lambda_1, \lambda_2, q\}$ given some data $x = (x_1, ..., x_T)$ independently identically distributed from a mixture of two Poissons:

$$p(x|\theta) = \prod_{t=1}^{T} q d_{\mathrm{Pois}}(x_t|\lambda_1) + (1-q) d_{\mathrm{Pois}}(x_t|\lambda_2)$$

- This can be seen as a problem of missing data. Let $c = (c_1, ..., c_T)$ denote from which Poisson each x_t was sampled, then we have:

$$p(x, c|\theta) = \prod_{t=1}^{T} \left(q d_{\mathrm{Pois}}(x_t|\lambda_1)\right)^{c_t} \left((1-q) d_{\mathrm{Pois}}(x_t|\lambda_2)\right)^{1-c_t}$$

Application of EM to the mixture of two Poisson

- E step. Compute the expectation of c given x and θ.

$$\widehat{c}_t = \frac{q d_{\mathrm{Pois}}(x_t|\lambda_1)}{q d_{\mathrm{Pois}}(x_t|\lambda_1) + (1-q) d_{\mathrm{Pois}}(x_t|\lambda_2)}$$

- M step. Find the θ that maximises the likelihood of c and x.

$$\widehat{\lambda}_1 = \frac{\sum_{i=1}^T x_t c_t}{\sum_{i=1}^T c_t}$$

$$\widehat{\lambda}_2 = \frac{\sum_{i=1}^T x_t (1 - c_t)}{\sum_{i=1}^T 1 - c_t}$$

$$\widehat{q} = \frac{\sum_{i=1}^T c_t}{T}$$

Application of EM to the mixture of two Poisson

- For example $\widehat{q}$ is derived as follows:

$$\log p(x, c|\theta) = \sum_{t=1}^{T} c_t \log(q) + (1 - c_t)\log(1 - q) + ...$$

$$\frac{\partial \log p(x, c|\theta)}{\partial q} = \sum_{t=1}^{T} \frac{c_t}{q} - \frac{1 - c_t}{1 - q}$$

$$\sum_{t=1}^{T}(c_t - \widehat{q}) = 0$$

$$\widehat{q} = \frac{\sum_{t-1}^{T} c_t}{T}$$

Baum-Welch Algorithm

- In the case of a HMM, the functions we need to estimate in the E step are the probability to be in a given state $u_j(t)$ and probability to transit from one step to another $v_{jk}(t)$
- Using local decoding and the forward-backward algorithm, we have already seen that:

$$\widehat{u}_j(t) = \alpha_t(j)\beta_t(j)/L_T$$

- Similarly we have:

$$\widehat{v}_{jk}(t) = \alpha_{t-1}(j)\gamma_{jk}p_k(x_t)\beta_t(k)/L_T$$

- In the E step, we compute $\widehat{u}_j(t)$ and $\widehat{v}_{jk}(t)$
- In the M step, we use Equations 1-3 to update θ, except that we replace $u_j(t)$ and $v_{jk}(t)$ with $\widehat{u}_j(t)$ and $\widehat{v}_{jk}(t)$, respectively.

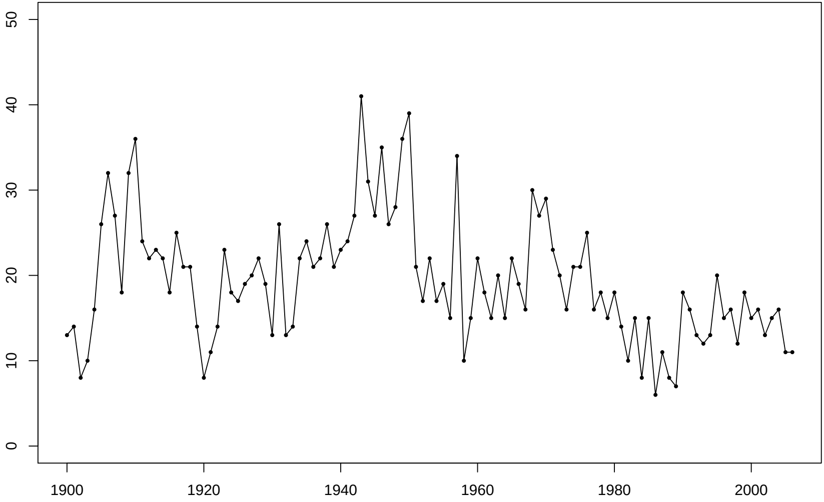
Viterbi training

- Start with some parameter values θ
- Find the hidden states using the Viterbi algorithm
- Estimate new parameter values as if the states were known to be the output of the Viterbi algorithm, ie using Equations 1-3
- Repeat until convergence

Viterbi training

- Viterbi training is less principled than the Baum-Welch algorithm
- It does not maximize the likelihood $p(\boldsymbol{x}^{(T)}|\theta)$
- Instead, it finds the value of θ that maximises $p(\boldsymbol{x}^{(T)}|\theta, \pi^*)$ ie the contribution to the likelihood from the most probable path π^*
- The Viterbi algorithm is faster than the Baum-Welch algorithm
- If the final aim is to produce good global decoding with the Viterbi algorithm, it can be argued that it makes sense to train using it

Earthquake example



Earthquake example with guessed parameters

```
library(depmixS4)
m=depmix(quakes~1,nstates=2,
         family=poisson(),ntimes=length(quakes))
m=setpars(m,c(0.5,0.5,0.9,0.1,0.1,0.9,log(15),log(25)))
cat(sprintf('logLik: %f\n',forwardbackward(m)$logLike))
summary(m)
```

Earthquake example with guessed parameters

```
## logLik: -343.011464
```

```
## Initial state probabilties model
## pr1 pr2
## 0.5 0.5
##
## Transition matrix
##        toS1 toS2
## fromS1  0.9  0.1
## fromS2  0.1  0.9
##
## Response parameters
## Resp 1 : poisson
##     Re1.(Intercept)
## St1           2.708
## St2           3.219
```

Earthquake example after Baum-Welch algorithm

```
library(depmixS4)
m=depmix(quakes~1,nstates=2,
         family=poisson(),ntimes=length(quakes))
m=fit(m,verbose=F)
cat(sprintf('logLik: %f\n',forwardbackward(m)$logLike))
summary(m)
```

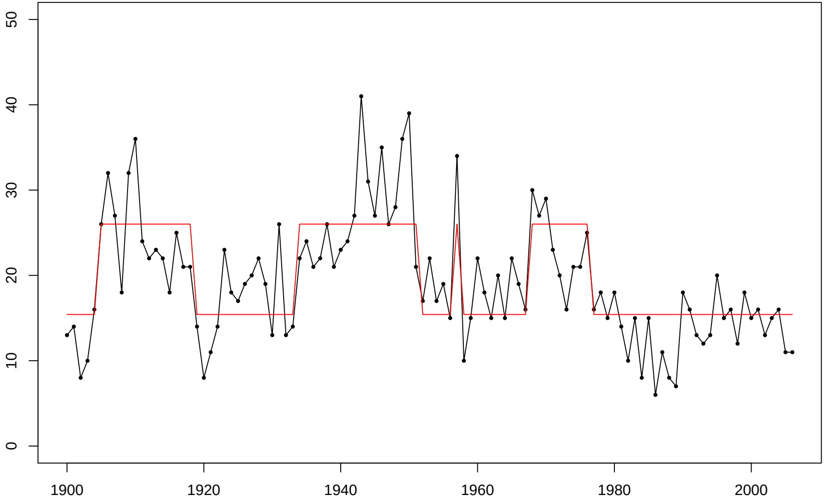
Earthquake example after Baum-Welch algorithm

```
## converged at iteration 28 with logLik: -341.8787

## logLik: -341.878704

## Initial state probabilties model
## pr1 pr2
##   1   0
##
## Transition matrix
##         toS1  toS2
## fromS1 0.928 0.072
## fromS2 0.119 0.881
##
## Response parameters
## Resp 1 : poisson
##     Re1.(Intercept)
## St1           2.736
## St2           3.259
```

Earthquake example: global decoding



Conclusions

- In previous lectures we have seen how to perform local or global decoding, ie estimating the hidden states
- In this lecture we have seen how to estimate the parameters of the HMM
- Supervised learning is easy
- The Baum-Welch algorithm is a special case of the EM algorithm
- A faster heuristic is given by Viterbi learning
- In the next lecture, we will explore how to select and check a model, and in particular how many hidden states to include