The HMM fitter

Terry Therneau

July 24, 2017

1 HMM likelihood

This section is technical and many users may skip it.

For analysis of the Mayo Clinic Study of Aging (MCSA) data we need to use a Hidden Markov Model (HMM). In these models there are two processes. First is an underlying state space through which the patients progress. The covariates and parameters affect rates of progression from one state to another, and are the primary quantities of interest. The second is an observation process: at some set of times t one or more outcome variables y are measured which are linked to the true states via a probability model. This portion of the model will also have parameters. In many cases the observed states y will have the same labels as the true states, such duplicate labels mostly serve to generate confusion.

A primary reference for the material is Christopher Jackson's excellent manual for the msm package [1]. Others that I found particularly helpful were a web page by Nokoalai Shokiriv [4], Satten and Lingini [3] and computational details from Kalbfleisch and Lawless [2].

Per Shokhirev there are 3 canonical problems for HMM models.

1. Given the model parameters, compute the probability of a particular output sequence. This is solved by the Forward or Backwards algorithms.

2. Given the model parameters and an output sequence, compute the most likely path for the hidden states (the true but unobserved path). This is solved by the Viterbi algorithm.

3. Given an output sequence, find the most likely set of model parameters. This is solved by the Baum-Welch algorithm.

We are focused on problem 1: an underlying search algorithm will choose parameters β, which map to rates, which in turn lead to a likelihood value. The search algorithm uses this to find an MLE. Many references, e.g. Wikipedia, focus only on problem 2 the exclusion of the others. This led to much early confusion for this author, e.g., the Viterbi algorithm involves a forward recursion and earns a 'forward' label there. It might look like 3 is our target, but that would only be true if there were no covariates and we were directly estimating the rates.

Let S be the set of possible true states; 6 in our basic Alzheimer's model, with $s_1, s_2, \ldots, s_m$ a possible path through those true states for some subject who has m observations. Let (y_{i1}, t_{i1}), $(y_{i2}, t_{i2}), \ldots$ be the set of observations that we have made for subject i. Let $f(y|s)$ be a probability or density function for y given that the true state is s. This can be a general probability distribution, e.g. the forced expiratory volumen (FEV) for someone with lung disease is assumed

to follow a lognormal distribution with a certain mean and variance. When y is discrete f can be represented by an error matrix E which has a row for each true state and a colum for each outcome, a the function f simply returns the appropriate element.

Let R be the matrix of underlying rates for some interval, then the transition probability matrix is $P_{im} = \exp(Rt)$ for the transition from time m to time $m+1$, $t = t_{i,m+1} - t_{im}$ is the length of time for the interval and exp is a matrix exponential. The jk element of P is the probability that the subject will be in state k at the end of the interval given that they began the interval in state j. The off diagonal elements of R are our modeled rates, and each will depend on covariates and parameters via $r = \exp(X\beta)$, r being the element in question. The exponential guarrantees that elements are positive. Those elements of R that correspond to non-possible transitions are 0. The rows of R are constrained to sum to zero and this determines the diagonal elements. For the matrix as a whole β is an p by k matrix where p is the number of columns of X and k is the number of non-zero transitions.

Symbol matching. I'm a statistician and am attached to X for a predictor, y for the observed response, β for parameters, i for subject and t for time. Many texts illustrate a single subject and so can use i for something else, and almost all differ from my notational choices. (There is fairly broad agreement on s for the states, but not much else.)

- Shokhirev is written in discrete time, $i = 1, \ldots, N$ indexes time.
  - q_i is the path through the states, o_i are the observed outcomes.
  - B is the emission matrix with $b_{ij} = b_i(o_j)$ the probability that j will be observed if i is the true state. (Yes, he reuses i multiple times).
- Jackson's msm manual
  - Q is the rate matrix, P the probabilities, and E the error matrix.
  - Subject i progresses through true states s_{ij} with $j = 1, \ldots, n_i$ and has observed values y_{ij}.

The HMM probability is formally a very large sum:

$$P(y_i|\beta, \pi) = \sum_{s_1, s_2, \ldots} P(y_i|s)P(s; \beta, \pi) \tag{1}$$

The sum is over all possible paths through the states: if subject Jones had 7 observations and there are 6 states this would be a sum over $6^7 \approx 280,000$ terms. The vector π is the initial state distribution and β are the parameters of the transitions. For each of those possible paths we have the probability of that path, given the coefficients of the model, times the probability of observing the vector of states y_i for that true path.

A big simplification comes from two assumptions: first that y_{ij} depends only on the current true state s_{ij} (subject i, jth observation) and the Markov property of the underlying chain, which means that the current state depends only on the prior state. This allows the computation to

be written as a nested sum:

$$\begin{aligned} P(y_i|\beta,\pi) &= \sum_{s_1} [P(y_{i1}|s_1)P(s_1|\pi,\beta)B_2] \\ B_2 &= \sum_{s_2} [P(y_{i2}|s_2)P(s_2|s_1,\beta)B_3] \\ B_3 &= \sum_{s_3} [P(y_{i3}|s_3)P(s_3|s_2,\beta)B_4] \\ B_4 &= \ldots \end{aligned} \tag{2}$$

The references all now say "thus calculating the probability reduces to matrix multiplication", which was not at all obvious to me. Let's work it out.

Let P_{im} be the transition matrix for subject i from observation time $m-1$ to m for that subject. The jk element of the matrix is the probability that the subject will be in true state k at time m given that they were in state j at $m-1$. P depends only on the underlying rates and observation times. The matrix product $P_{i1}P_{i2}$ is the transition matrix from 0 to 2: the jk element of the matrix product is a sum over all the states that might have been visited at time 1 (write it out). This is a case where matrix multiplication happens to do exactly the right thing. Likewise $P_{i1}P_{i2}P_{i3}$ is the transition for 0 to 3, etc., and $\pi P_{i1}P_{i2}P_{i3}$ the probability vector for the states at time 3 where π is the initial probability distribution.

Let D_{im} be a diagonal matrix with jj element $Pr(y_{im}|s_m = j, \gamma)$, the probability of the observed y at time m for each possible true underlying state, possibly depending on some parameters γ. The jk element of $T_{im} = P_{im}D_{im}$ is the probability that subject i will be in true state k *and* observed status y_{im} at time m given that they were in true state j at time $m-1$. The matrix product $\alpha_{im} = \alpha_1 \prod_{k=2}^{m} T_{ik}$ is the vector containing for each true state the probability that subject i will will be in that state at observation m and have the sequence $y_{i1}, y_{i2}, \ldots y_{im}$ of observed states. The probability is then

$$\begin{aligned} \alpha_m &= \alpha_1 \prod_{k=2}^{m} T_{ik} \\ p &= \sum_{k=1}^{ns} \alpha_{mk} \end{aligned} \tag{3}$$

where m is the number of observations for that subject and ns is the number of states. This captures equation (2) above, and is identical to Jackson's definition of T, section 1.6.3 of the msm manual, and to the recursion formula of Shokhirev.

The very first step of the comptation requires a bit more thought. There are three approaches available. The first, used in the msm package, is to replace $Pr(s_1)$ with the initial state probability vector π, so that $\alpha_1 = \pi D_{i1}$. This is the initialize step in Shokhirev, and appears to be a standard formula. Each subject is randomly chosen from an underlying population with probability vector π, and we go forward from there. The `cav` data set in the msm package is an example where everyone's true state is known at enrollment (heart transplant). The `msm` package allows each subject to have a separate inititialer π.

Method 2 uses updated prevalence and is appropriate to the Alzheimer data, in which subjects are a population sample. It is essentially the use of an age specific π vector for each subject. If

someone were recruited at age 82, the overall population prevalence at that age is $\pi(50)P_i(82)$, $P_i(82)$ being the transition matrix from age 50 up to age 82 for the covariate set of subject i and $\pi(50)$ the initial values at age 50.[1] However, the study did not randomly select subjects from the distribution just computed — that would have meant enrolling a lot of dead people. The `enroll` argument is a vector with one element per state containing the probability of enrollment given that state. This is used to create a renormalized initial probability

$$\pi_k^* = \pi_k e_k / \left(\sum pi_k e_k\right)$$
$$\alpha_1 = \pi * D_{i1}$$

A non-renormalized value would discourage high death rates; the likelihood will rate it as an unlikey event to have enrolled several living 85 year olds but no dead ones if Pr(death) by age 85 is 70%. The enroll vector will often be 1 for all states except death.

Satten and Longini also discuss this issue and change the first line of equation (2) to

$$P(y_{i2}, y_{i3}, \ldots | y_{i1}, \beta, \gamma, \pi) = \sum_{x_1} [P(S = s_1 | y_{i1}, \gamma, \pi) B_2]$$

That is, α_1 becomes a posterior distribution over the states given the first observation. The distribution is allowed to depend on parameters of the misclassification process (γ) and on initial state, but not on the paramters of the transition probabilities β. They then choose a suitable prior for π and integrate it out. This has not yet been added to the code. For the MCSA our posterior *does* depend on β since the latter determines the population distribution at each age.

(A fourth approach is found in my earlier hmm routines and which I now think is incorrect. It renormalized after multiplication by D rather than before.)

An additional aspect is censoring, which happens in two ways. The first is when an observation is inserted into the data set solely to change the covariates; there was no observed y at that time. In that case it suffices to set $D = I$, the identity matrix. The transition matrix for the combined interval is $P_{i,m}P_{i,m+1}$ and insertion of the identity matix makes equation (3) match the correct expression. The second case is partial observation. Say that y is discrete and at some point we know that y lies in some subset of states. In that case the appropriate D matrix is the sum of matrices for the subset, e.g., the probability that y would be 1 or 3 given the true state is s is the sum of the probabilities that y will be a 1 and that it will be a 3. The msm package takes the second view while we take the first.

The likelihood contribution for exactly observed states such as death is slightly different. First, we assume that any such states are observed without error. An instant before the death (at observation m) the state probability is $\alpha_{m-1}T_{im}$, the subject is in one of the non-dead states, and this is followed immediately by a transition to death. This last transition has a rate given by the death column of $R_i(t)$, the rate matrix at the death time. Then $\alpha_m = \alpha_{m-1}TimD$ just as before, but instead of a diagonal matrix D we have a mostly zero one:

$$D_{ij} = \begin{cases} R_{ij} & j = d, i \neq d \\ 0 & \text{otherwise} \end{cases}$$

where d is the column of R corresponding to the death state. (Note: if an exact state also implies that this is the last observation for a subject, then D could again be diagonal since the next step will be a sum, and an exact state then simply replaces one diagonal matrix with another.)

[1] Questionable notation for P on this line of text.

The HMM can have multiple y values at each time: we might have an MRI scan, lab tests, etc. This leads to a multiplication by $D^{(1)}D^{(2)}\ldots$ at the step, i.e., that we observed *all* of those y values. There is a possibility for a user to mess up and declare a time point where the first y says that the subject must be in one set of states and the second y a different non-overlapping set, which of course leads to a likelihood of zero. The code does not yet check for this.

2 Design

Specification of an HMM model is complex, because the model is complex.

- The underlying Markov model: the states, state names, and the allowable transitions.
- Which covariates apply to each transition, and which of these transition/covariate pairs share a coefficient.
- For each result y the transmission function needs to be defined, which provides the distribution of y given state. Exact states do not need a transmission function, e.g., death.
- The initial probabilities, whether these should be estimated, and if so what covariates to use.
- Initial values for the coefficients and specification of which of them, if any, are fixed.

Another issue to consider is that msm and the survival functions treat the input data differently. In the survival functions each line of data is marked as an interval (t1, t2] and contains the covariates that apply over that interval along with the outcome at the end of the interval. The data sets for msm contain a single time value along with covariates and endpoints that were measured at that time. The msm setup is actually more natural in the sense that it mimics the way in which data is gathered. The hmm code follows the msm convention, with `cbind(time, y1, y2, ...)` or `hbind(time, y1, y2, ...)` as the left hand side of the formula. The hbind routine is local to hmm, and is a little more general wrt responses that are character or factors. (If one of the responses was character and another a numeric cbind will make them both character). When there are multiple responses the code allows some of them to be missing; for a death time we might have all of them missing.

3 Expansion

For the study that motivated hmm age has a dominating effect on the transition rates. This code takes the approach that data will be pre-processed such that "age" is a time dependent covariate, but with any inserted lines marked as 'censored': there is no observation of the subject's state on these inserted birthdays.

4 The hmm function

We have to inform the program of a long list of items in order for fitting to proceed. For an HMM these include

- The time, covariates, and subject identifiers.
- The set of true states and the allowed transitions.
- The error mapping from true states to observed states.
- Which covariates apply to which states, and possible initial values for them.
- The initial state distribution.

The data is specified by using standard R modeling language, illustrated in the call below.

```
> fit <- hmm(hbind(age, y) ~ iage + ns(iage,3) + male, data=data2,
+            subset=1:50, id=clinic, otype=otype,
+            qmatrix= qmat, qcoef=qcon, rcoef=rcon, pcoef=pcoef)
```

In this example `age` indicates the observation time at which the state was observed for the subject and `y` is a variable giving the *observed* outcome for each observation, as opposed to the true underlying state of the HMM. The iage variable is integer age: for HMM models a covariate must be constant over each time interval, so as a predictor iage moves forward in steps at each birthday. The otype and id variables are required, and will normally be found in the data set along with the formula variables. The data must have all of the observations for a particular id in contiguous rows, sorted by time within subject; the data does not need to be sorted by subject. The `obstype` variable is 0= censored, 1= usual, 2= death, 3= entry. The `data` and `subset` arguments work exactly as in other R modeling functions.

Having age as both our time scale and as a covariate can be confusing, and one needs to use two separate variable names. The variable on the left hand side encodes actual observation times and so will be continuous. The variable on the right hand side is a time dependent covariate and will normally be an integer, corresponding to the idea that there is a rate for age 50-51, another for age 51-52, ... and each rate applies over the whole age interval. This is similar to US death rate tables, but more importantly a cutpoint variable like this is the only way that the HMM model can handle continuous age. How many intervals to use is up to the user: single years of age is perhaps overkill (at least at the younger ages).

4.1 Transition rates

The matrix of transition rates is specified using the `qmatrix` argument, which has the same form as it does in the msm package. If there were 6 true states then Q will be 6 by 6, element i, j is 0 if transitions from state i to state j do not (directly) occur and a positive value if the transition is allowed. The row and/or column names of Q contain the names of the states, which will be attached to various parts of the printout; no dimnames corresponds to using 1, 2, 3, ... as the state names.

For our 6 state model of Alzheimers the Q matrix would be

```
> qmat <- matrix(c(0, 1, 1, 0, 0, 1,
+                  0, 0, 0, 1, 0, 1,
+                  0, 0, 0, 1, 1, 1,
+                  0, 0, 0, 0, 1, 1,
+                  0, 0, 0, 0, 1, 1,
```

```
+                   0, 0, 0, 0, 0, 1), ncol=6, byrow=TRUE)
> states <- c("A-N-", "A+N-", "A-N+", "A+N+", "demented", "dead")
> dimnames(qmat) <- list(states, states)
> qmat

          A-N- A+N- A-N+ A+N+ demented dead
A-N-         0    1    1    0        0    1
A+N-         0    0    0    1        0    1
A-N+         0    0    0    1        1    1
A+N+         0    0    0    0        1    1
demented     0    0    0    0        1    1
dead         0    0    0    0        0    1
```

The diagonal is ignored and can contain any value. (Subjects can always stay in the same state, and the program knows this.) Zeros correspond to transitions that cannot occur and non-zero to those that can. The non-zero values will normally be simple indicators as in the above example, with initial values for the transitions found in the `qcoef` argument. Alternately, the elements of `qmat` can contain initial values for the log of the transition rates. Any initial values in `qcoef` will override the values in `qmat`.

Covariates are controlled by the `qcoef` argument, which is a data frame containing variables `state1, state2, term, coef` and optionally `init`. The first three variables can be either numeric or character, depending on whether you want to refer to the states and/or terms by number or label. A term in the model formula such as `ns(x3, 3)` will expand into multiple coefficients but is thought of as one *term* by R, and entered as such in qcoef. Do not forget the intercept, which is implicit in most R models and can be referred to either as '(Intercept)' or as term number 0.

Any row in `qcoef` that has coef=0 causes that coefficient to not be optimized; it will be fixed at its initial value. All rows with `coef` set to the same nonzero value are constrained to share the same coefficient. The actual values used in the coef column for this purpose are of no consequence although one will normally use integers. Initial values for the intercept terms are are taken from the `qmatrix` argument or the `qcoef` data frame; if present the `qcoef` values override any in Q.

In the model given above, assume that we want a common death rate for the first 4 states, differing for males and females, and the spline term only for the A-N- to A+N- transition. Below is a sample qcoef argument. The program will automatically add the intercepts and their initial values. In order to be consistent with the msm package the current code expects the hazard (exp(coef)) in qmat and the coefficient itself in qcoef. I'm not sure if this is a good idea.

```
> states <- c("A-N-", "A+N-", "A-N+", "A+N+", "demented", "dead")
> qcon1 <- data.frame(state1 = states[c(1,3,1,2,3,4,5)],
+                     state2 = states[c(2,4,3,4,5,5,6)],
+                     term = rep(c("ns(iage, 3)", "iage"), c(1,6)),
+                     coef = 1:7)
> # force equal age coeffiecients for death
> qcon2 <- data.frame(state1 = states[rep(1:4, 3)],
+                     state2 = rep("dead", 12),
+                     term   = rep(c("(Intercept)", "iage","male"), c(4,4,4)),
```

```
+                     coef = rep(12:14, c(4,4,4)))
> rbind(qcon1, qcon2)

     state1   state2        term coef
1      A-N-     A+N- ns(iage, 3)    1
2      A-N+     A+N+        iage    2
3      A-N-     A-N+        iage    3
4      A+N-     A+N+        iage    4
5      A-N+ demented        iage    5
6      A+N+ demented        iage    6
7  demented     dead        iage    7
8      A-N-     dead (Intercept)   12
9      A+N-     dead (Intercept)   12
10     A-N+     dead (Intercept)   12
11     A+N+     dead (Intercept)   12
12     A-N-     dead        iage   13
13     A+N-     dead        iage   13
14     A-N+     dead        iage   13
15     A+N+     dead        iage   13
16     A-N-     dead        male   14
17     A+N-     dead        male   14
18     A-N+     dead        male   14
19     A+N+     dead        male   14
```

4.2 Response functions

Response functions are set up in a parallel way. The `rfun` argument contains the name of the response function, or a list of functions if there are multiple y values at each visit. The first response function goes with the first y value and etc. The `rcoef` argument is a data frame with names of response, lp, term, coef and optionally init; the first of these is not needed if there is a single response. A given response function can depend on multiple linear predictors. The term, coef, and init arguments are identical to those for qcoef.

A response function will be called with four arguments

y a vector of values y of length n

nstate number of states

eta a matrix of linear predictors with n rows.

derivative FALSE or TRUE, whether the function should also return derviatives of the result

The routine will return a matrix with *nstate* rows and n columns. Each column refers to a single observation, and contains $P(y_i|s_i = j)$, the probability of seeing the given y if the true state were j. The routine is not called for missing y values, the parent `hmm` routine has taken care of that. If derivatives are requested then the response will have a `gradient` attribute which is an array whose first two dimensions are identical to the return, and last dimension is the number of linear predictors. Each "slice" of the array contains the derivatives wrt a single linear predictor.

The code contains a response function `noerror.hmm` which treats every response as though it were observed without error. Using this yeilds the usual Markov model rather than a hidden Markov model. Say one has a response function with no optimized parameters, e.g., if error rates were known a-priori. One way to deal with this are to write a simple function with those rates baked in as constants; it depends on no parameters so the the `rcoef` argument would be empty. Another is to write one that depends on parameters but call it with those parameters fixed. An rcoef data frame for the latter would have one or more rows with lp= 1, 2, ..., term=0 (intercept), coef=0 (fixed value) and init = the desired value. In either test cases the parent routine is smart enough not to ask for derivatives.

4.3 Starting probabilities

The starting probability π_0 is defined in the same way as the response functions using `pfun` and `pcoef`. The `pfun` function has arguments of number of states, linear predictors, and derivative.

5 Optmization

Within the code there are a few key variables. First is the X matrix which has n rows and m columns, and is like any other X matrix in a modeling function. Second is β which has m rows and p columns, one for each linear predictor. There is a linear predictor for each allowed transition rate, along with those for the response functions and π. Some elements of β may be constant. The third is `cmap`, which has the same shape as β, and contains parameter numbers. A value of 0 means that the corresponding element of β is fixed (often at zero), a value of k means that the corresponding element of β is taken from the kth paramter. The maximum value of `cmap` is the number of estimated parameters, which is also the length of the `coefficient` vector. This is the vector that the optimization routine sees.

The `scale` option controls centering and scaling of the variables. If centered then the intercepts are with respect to the mean covariate values, scaling scales them all to a standard deviation of 1. Centering can be critical for convergence of the code, and scaling will often be helpful. The default for the routine is `c(TRUE, FALSE)` for centering but not scaling. The `scale` compontent of an hmm fit contains the vector of centering values and the vector of scale values. This can be used as the argument in other hmm fits to ensure the same center/scale from run to run of a set of models.

A strength of the code (but more work for the user) is that it allows the use of an optimizer of choice. The routine should have `par` and `fn` as the first arguments, and return a list containing elements `loglik` and coefficients. As an olive branch to the optim routine the results can also be labeled as `par` and `value`. The `mfun` argument to `hmm` should contain the name of the optimizer and `mpar` a list of any other parameters desired by the routine. When called `par` will contain the intial values of the parameters and `fn` will by default be 'hmmloglik'. However, any of 'hmmloglik', 'hmmgrad' or 'hmmfull', can be be used by adding this as the `fn` element of `mpar`. Each of these takes a single vector of parameter values as input.

- hmmloglik returns a single log-likelihood value
- hmmgrad returns a vector of first derivatives

- hmmfull returns vector of log-likehihood values, one per subject, along with a matrix of derivatives with one row per subject and one column per parameter.

References

[1] C. Jackson. *Multi-state modeling with R: the msm package*, 2016.

[2] J. D. Kalbfleisch and J. F. Lawless. The analysis of panel data under a Markov assumption. *J. Amer. Stat. Assoc.*, 80:863–871, 1985.

[3] G. A. Satten and Jr. I. M Longini. Markov chains with measurement error: Estimating the true course of a marker of the progression of human immunodeficiency virus disease. *Applied Stat.*, pages 275–309, 1996.

[4] N Shokhirev. *Hiden Markov models.*