Hidden Markov Models and You, Part Two

By Brian Grossmiller and Doug Norris

In the July edition of this newsletter, we introduced you to hidden Markov models (HMMs) by providing a brief introduction to this technique to tease out patterns within time series and including some discussion on how they can be used to solve actuarial problems. In a hidden Markov model, observed data are generated from one of multiple states that are hidden from view. In this second part, we will walk through the algorithm that is typically employed to evaluate HMMs, and also provide some more examples to spur your own efforts.

HOW ARE HIDDEN MARKOV MODELS BUILT?

A good first step toward adopting HMMs into your own practice is to understand how the evaluation algorithm works. We will be exploring an example developed in Excel. Naturally, in a production environment, we recommend an implementation of the algorithm in a more robust software package such as R (so "do as we say, not as we do").

The Excel workbook described in this article is available at *http://www.soa.org/news-and-publications/newsletters/forecasting-futurism/default.aspx*

One of the best-known algorithms for calibrating an HMM is the Baum-Welch, or expectation maximization (EM), algorithm. The EM algorithm is an iterative process, which recursively updates a set of HMM parameter estimates until they converge. The main four functions used in the Baum-Welch algorithm are commonly referred to by the first four Greek letters, as follows:

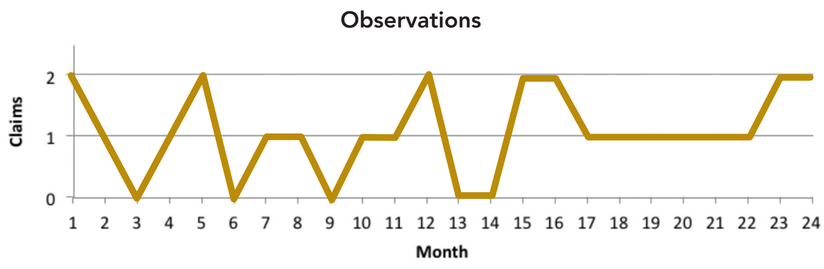
- Alpha (α): Forward probabilities, which are generated from an initial estimate of the hidden state at the first data observation, and calculated forward from there.
- Beta (β): Backward probabilities, which are computed as a conditional probability from the last (final) data observation.
- Gamma (γ): Combines the forward and backward probabilities into a probability estimate of the state transition at each data observation.
- Delta (δ): Sums the gamma function across all transitions, to provide an estimate of the hidden state at each data observation.

Once these four functions have been constructed, the HMM parameters can be re-estimated and the process repeated (until the parameters converge to a steady state). A brief example will better illustrate this process.

EXAMPLE: CHRONIC DISEASE FLARE-UP

Suppose that we have 24 months of observations for a patient with a chronic disease. The observations are the number of medical claims the patient had in each month: zero, one, or two or more (the latter shown as "2" in this example, as graphing numbers and text together can get messy). The observations are as in Figure 1.

Figure 1: 24 Months of Observed Claims Frequency



Based upon these observations, we might suspect that there are periods where the disease has flared up and produced more claims, and other periods where the disease is well managed (and perhaps the only claims in these periods are maintenance drugs). By fitting an HMM to these observations, we can obtain an estimate of the hidden state at each observation, and make an estimate of the number of claims we expect in the next month. Since we think that there are two states in this process (one in which the condition is well managed, and one in which the condition has flared up), we will model this data with a two-state HMM (it can be fun and educational to fit different-sized models to the same data).

CONTINUED ON **PAGE 12**

The Baum-Welch algorithm, as with many numerical algorithms, requires initial estimates for each of the model parameters. This algorithm produces a set of model parameters that maximize the likelihood of observing the given data; however, if we choose our parameters poorly, we may find a solution that is only a local maximum or saddle point (instead of a global maximum).

This algorithm has another thing in common with other numerical algorithms—choosing these initial parameter estimates is as much an art as it is a science. One way is to look at the data and hypothesize the state that each observation is in; from there, we can make the remaining parameter estimates.

For this example in particular, we require initial parameter estimates for:

- A probability distribution for each state of observing 0, 1 or 2+ claims.
- A guess as to which state we are in initially.
- Four transition probabilities (one each for State 1 to State 1, State 1 to State 2, State 2 to State 1, and State 2 to State 2).

Figure 2: Initial Parameter Estimates

Initial Two State Estimates	Pr(0)	Pr(1)	Pr(2)
Distribution of State 1	0.400	0.400	0.200
Distribution of State 2	0.200	0.200	0.600
Initial Pr(State 1)	0.500		
Initial Pr(State 2)	0.500		
Pr(State 1 -> State 1)	0.700		
Pr(State 1 -> State 2)	0.300		
Pr(State 2 -> State 1)	0.500		
Pr(State 2 -> State 2)	0.500		

Figure 2 shows some initial estimates for each of the parameters. These parameters, along with the data observations, allow us to compute each of the four functions required by the EM algorithm. Fortunately, the mathematics are fairly intuitive and simple, and we will tackle each in turn.

ALPHA (FORWARD PROBABILITIES)

Figure 3: Forward Probabilities

Alpha - Forward Probabilities

Month	Value	State 1	State 2
1	2	0.100000	0.300000
2	1	0.088000	0.036000
3	0	0.031840	0.008880
4	1	0.010691	0.002798
5	2	0.001777	0.002764
6	0	0.001050	0.000383
7	1	0.000371	0.000101
8	1	0.000124	0.000032
9	0	0.000041	0.000011
10	1	0.000014	0.000004
11	1	0.000005	0.000001
12	2	0.000001	0.000001
13	0	0.000000	0.000000
14	0	0.000000	0.000000
15	2	0.000000	0.000000
16	2	0.000000	0.000000
17	1	0.000000	0.000000
18	1	0.000000	0.000000
19	1	0.000000	0.000000
20	1	0.000000	0.000000
21	1	0.000000	0.000000
22	1	0.000000	0.000000
23	2	0.000000	0.000000
24	2	0.000000	0.000000
Total Probability		**0.000000**	

The forward probabilities (alpha values) represent the joint probability that, given the current HMM parameters, the model is in its current state and that each of the data points observed so far has happened. For instance, the first data point above is a "2." $\alpha_1(1)$ then represents the joint probability that the model is in State 1 in the first month, and that we have observed "2" claims.

The table of forward probabilities (shown in Figure 3) can look daunting at first, but it essentially comes down to two calculations. The first row is simply the initial state probability multiplied by the probability of the observation in that state. Using our initial parameter estimates, this comes down to:

$\alpha_1(1) = \text{Initial Pr(State 1)} * \text{Pr}_1(2) = 0.5 * 0.2 = 0.1$

The calculation changes for Month 2 and beyond. There, we take the forward probabilities in each state up to that point, multiply by the transition probability to the state in question, and multiply the sum of those by the probability of the observation. An example of the calculation of the forward probability at Month 2 for State 1 follows:

$\alpha1(2) = [\ \alpha_1(1) * \text{Pr(State 1} \rightarrow \text{State 1)} + \alpha_2(1) * \text{Pr(State 2} \rightarrow \text{State 1)}\ ] * \text{Pr}_1(1) = [\ 0.1 * 0.7 + 0.3 * 0.5\ ] * 0.4 = 0.088$

The remaining calculations follow this same formula. We also compute the total probability at the last step (simply the sum of the probabilities across all states in the last step). This is used in maximum likelihood estimation, and in both the gamma and delta functions (which we will get into later). As you can see, there are a few zeroes between the value and the decimal point, so taking the logarithm and maximizing that instead is often more convenient.

BETA (BACKWARD PROBABILITIES)

Figure 4: Backward Probabilities

Beta - Backward Probabilities

Month	Value	State 1	State 2
1	2	0.000000	0.000000
2	1	0.000000	0.000000
3	0	0.000000	0.000000
4	1	0.000000	0.000000
5	2	0.000000	0.000000
6	0	0.000000	0.000000
7	1	0.000000	0.000000
8	1	0.000000	0.000000
9	0	0.000000	0.000000
10	1	0.000000	0.000000
11	1	0.000001	0.000001
12	2	0.000002	0.000002
13	0	0.000006	0.000006
14	0	0.000018	0.000023
15	2	0.000050	0.000061
16	2	0.000170	0.000147
17	1	0.000513	0.000443
18	1	0.001546	0.001335
19	1	0.004660	0.004027
20	1	0.014024	0.012221
21	1	0.041824	0.038560
22	1	0.116800	0.152000
23	2	0.320000	0.400000
24	2	1.000000	1.000000

CONTINUED ON **PAGE 14**

The backward probabilities (beta values) represent a conditional probability that, given that the HMM is in state X at time t (and assuming the current HMM parameters), all of the observed data points from t+1 through the last month actually occurred. In our example, $\beta_{23}(1)$ represents the probability that, given that the HMM is in State 1 at time 23, we observed "2" claims at time 24.

These backward probabilities are calculated in reverse, starting from the last data observation. The beta values at the last data observation are defined to be one, because there is no data beyond this point (and so the probability of observing it is one).

Unfortunately the other calculations aren't as easy, but at least they are pretty much identical to one other. For each time step, the process is to calculate the sum of the probabilities of moving to each state, seeing the observation at the next time, and multiplying by the beta function back to that point. To demonstrate, let's take a look at the beta calculation at Month 22 in State 1:

$\beta_{22}(1) = \Pr(\text{State 1} \rightarrow \text{State 1}) * \Pr_1(2) * \beta_{23}(1) + \Pr(\text{State 1} \rightarrow \text{State 2}) * \Pr_2(2) * \beta_{23}(2) = 0.7 * 0.2 * 0.32 + 0.3 * 0.6 * 0.4 = 0.1168$

That one is a bit of a headache, but if you stare cross-eyed at the formula long enough you might see a 3D picture (also notice that, in each part of the calculation, the first two numbers are the same for each beta calculation). These backward probabilities are used in the calculation of the gamma function, so let's dive right in.

[THE BAUM-WELCH ALGORITHM] PRODUCES A SET OF MODEL PARAMETERS THAT MAXIMIZE THE LIKELIHOOD OF OBSERVING THE GIVEN DATA.

GAMMA (ESTIMATE OF STATE TRANSITIONS)

Figure 5: Estimate of State Transitions

Gamma - Estimate of State Transitions

		Transition			
Month	Value	1 to 1	1 to 2	2 to 1	2 to 2
1	2	0.234674	0.043742	0.502873	0.218711
2	1	0.617078	0.120469	0.180315	0.082138
3	0	0.632653	0.164741	0.126031	0.076575
4	1	0.359658	0.399025	0.067243	0.174073
5	2	0.360280	0.066621	0.400358	0.172740
6	0	0.641922	0.118717	0.167207	0.072154
7	1	0.682736	0.126392	0.133294	0.057578
8	1	0.687818	0.128212	0.128207	0.055763
9	0	0.682702	0.133323	0.126385	0.057590
10	1	0.641680	0.167407	0.118672	0.072241
11	1	0.358619	0.401734	0.066313	0.173334
12	2	0.354881	0.070051	0.393726	0.181342
13	0	0.585995	0.162612	0.152591	0.098802
14	0	0.288242	0.450344	0.056272	0.205142
15	2	0.163319	0.181195	0.182651	0.472835
16	2	0.291979	0.053991	0.456895	0.197135
17	1	0.631989	0.116885	0.175423	0.075703
18	1	0.681261	0.126151	0.134482	0.058106
19	1	0.687388	0.128355	0.128339	0.055917
20	1	0.681156	0.134571	0.126129	0.058143
21	1	0.631252	0.176034	0.116748	0.075966
22	1	0.286904	0.461096	0.053053	0.198948
23	2	0.148731	0.191226	0.165011	0.495032
24	2				

The forward (alpha) and backward (beta) probabilities are combined to produce the gamma function, which gives an estimate of the state transitions. This function is calculated separately for every possible state transition, and because (in this example) we have two states, there are four transitions. Note that we do not calculate the gamma function at the last observation; because this is the terminal state, there is no additional transition to estimate. In each calculation, four components are multiplied together, which are then divided by the total probability from the alpha function:

- The forward probability of the current time and state
- The transition probability to the state at the next time (for each column, this is one value)
- The probability of the observation at the next time and state
- The backward probability at the next time and state.

To give one example, the calculation at Month 1 for the transition from State 1 to State 1 is as follows (note that the beta and total probabilities are very small, and are presented in scientific notation for reading convenience):

[$\alpha_1(1)$ * Pr(State 1 $\rightarrow$ State 1) * $Pr_1(1)$ * $\beta_2(1)$] / Total Probability = [0.1 * 0.7 * 0.4 * 3.21E-11] / 3.83E-12 = 0.234674

Wasn't that fun? The good news is that the rest of the calculations are just like that (the other good news is that you don't have to do these calculations by hand). Anyhow, once the gamma function is built, the state transitions can be re-estimated for the next iteration of the HMM. The first column shown in Figure 5 contains all of the estimated probabilities for transitions from State 1 to State 1, while the first two columns contain all of the estimated probabilities for transitions originating in State 1. By adding up the first column, and dividing by the sum of the first and second columns, you have your new estimate of the transition probability from State 1 to State 1—finally some easy math. Speaking of easy math, next up is the delta function.

DELTA (ESTIMATE OF THE STATE AT EACH OBSERVATION)

Figure 6: Estimates of Hidden States

Delta - Estimate of Probability of Each State at Each Observation

Month	Value	State 1	State 2
1	2	0.278416	0.721584
2	1	0.737547	0.262453
3	0	0.797394	0.202606
4	1	0.758684	0.241316
5	2	0.426901	0.573099
6	0	0.760639	0.239361
7	1	0.809128	0.190872
8	1	0.816030	0.183970
9	0	0.816025	0.183975
10	1	0.809087	0.190913
11	1	0.760353	0.239647
12	2	0.424932	0.575068
13	0	0.748607	0.251393
14	0	0.738586	0.261414
15	2	0.344514	0.655486
16	2	0.345970	0.654030
17	1	0.748874	0.251126
18	1	0.807413	0.192587
19	1	0.815743	0.184257
20	1	0.815727	0.184273
21	1	0.807286	0.192714
22	1	0.748000	0.252000
23	2	0.339957	0.660043
24	2	0.313742	0.686258

CONTINUED ON **PAGE 16**

For our chronic disease example, Figure 6 shows the delta function for the first iteration of the HMM. All, except for the last step, are simply the sum of the gamma functions originating in that state. The first observation for State 1 is just:

$$\gamma_1\rightarrow_1(1) + \gamma_1\rightarrow_2(1) = 0.234674 + 0.043742 = 0.278416$$

The only exception is the last observation, which (fortunately) can be easily computed from the total probability determined in the alpha function. As you'll recall, that was just the sum of the probabilities across all states at the last observation. The delta function at the last step is the alpha function for the same state and step, divided by that total probability.

Figure 7: Re-estimated Parameters at the First Iteration

Initial Two State Estimates	Pr(0)	Pr(1)	Pr(2)
Distribution of State 1	0.245	0.598	0.157
Distribution of State 2	0.138	0.312	0.550
Initial Pr(State 1)	0.278		
Initial Pr(State 2)	0.722		
Pr(State 1 -> State 1)	0.733		
Pr(State 1 -> State 2)	0.267		
Pr(State 2 -> State 1)	0.551		
Pr(State 2 -> State 2)	0.449		

The delta function allows us to re-estimate all of the remaining HMM parameters. New initial state estimates are given by the delta function in the first row. Probability distributions can be derived by adding up the delta function in each state (for the observed 0, 1 or 2) and dividing by the total delta function. In our example, the resulting figures after the first iteration are given in Figure 7.

It can be educational to see how our parameters have changed from our initial assumptions. For both distributions, we had initially overestimated the probability of seeing zero claims (compared to what was actually observed in the data). We also increased the likelihood that we begin in the "condition flare-up" state, as the high number of claims present in the initial observation suggests.

One last interesting feature of the delta function is that it gives us an estimate of the state that the system is in at every observation. This tends to converge as we iterate the HMM, as shown in Figures 8, 9 and 10 for iterations 1, 10 and 100, respectively.

Figure 8: Hidden State Estimate at Iteration 1

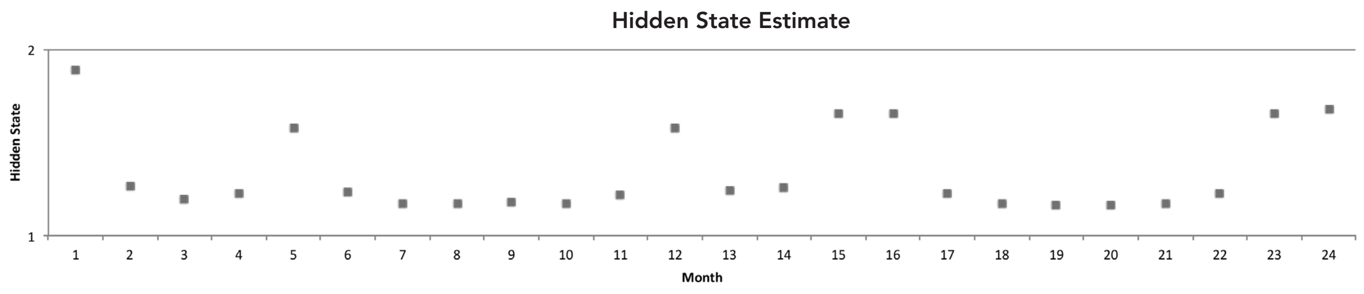


Figure 9: Hidden State Estimate at Iteration 10

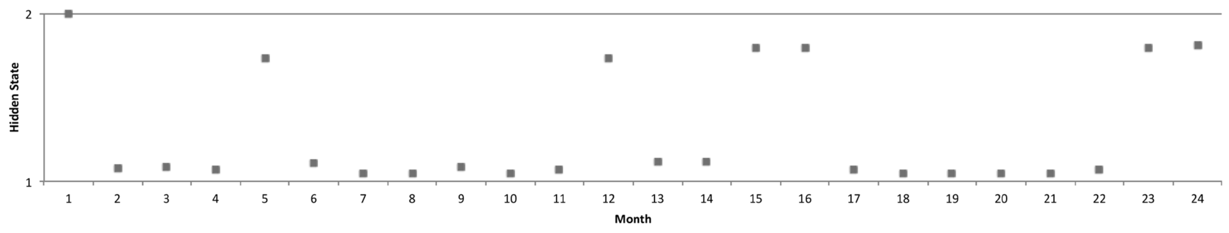
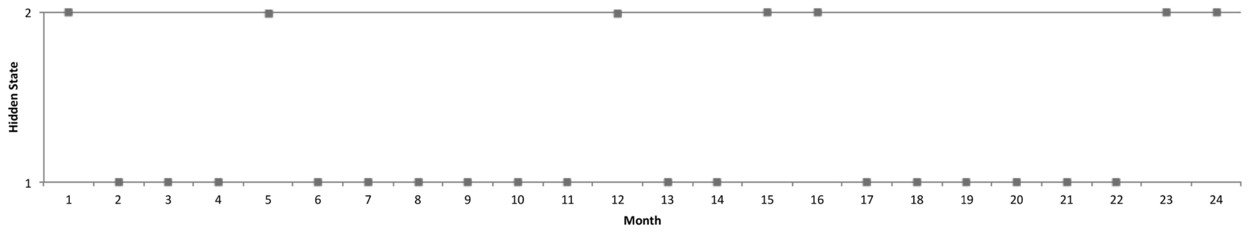


Figure 10: Hidden State Estimate at Iteration 100



NOW THAT WE HAVE OUR MODEL, WHAT DO WE DO WITH IT?

Now that we have gone through all that trouble of putting together an HMM, you might be wondering what comes next. This is the fun part, so let's put together an estimate of the number of claims that we could expect to see in Month 25. After 100 iterations, our parameters are as in Figure 11.

Figure 11: New Parameters after 100 Iterations

Initial Two State Estimates	Pr(0)	Pr(1)	Pr(2)
Distribution of State 1	0.293	0.704	0.002
Distribution of State 2	0.000	0.000	1.000
Initial Pr(State 1)	0.000		
Initial Pr(State 2)	1.000		
Pr(State 1 -> State 1)	0.766		
Pr(State 1 -> State 2)	0.234		
Pr(State 2 -> State 1)	0.668		
Pr(State 2 -> State 2)	0.332		

We also saw in Figure 10 that at Time 24 we are in State 2. We can use the state transitions alongside the expected values of each distribution to determine that:

- Expected Value Distribution 1: 0 * 0.293 + 1 * 0.704 + 2 * 0.002 = 0.708
- Expected Value Distribution 2: 0 * 0.0 + 1 * 0.0 + 2 * 1.0 = 2
- Pr(State 2 → State 1) * 0.708 + Pr(State 2 → State 2) * 2 = 0.668 * 0.708 + 0.332 * 2 = 1.132

A straight empirical distribution (average of the observed values) would provide an estimate of 1.0833; if the hidden state structure is a reasonable assumption for this particular disease, then we have a good case for using the HMM result instead. A great feature of HMMs is that they can run very quickly once implemented, and a more refined estimate across several thousand claimants can really add up.

Now that we've had a thorough grounding in the mechanics of hidden Markov models, let's look at a more realistic example.

CONTINUED ON **PAGE 18**

EXAMPLE: QUANTIFYING FUTURE RISK

As actuaries, we are oftentimes called upon to measure, manage and predict risk. For the truly gifted (such as ersatz hero George Costanza), becoming a risk management expert is as simple as listening to a few books on tape, but for the rest of us, it's not so easy. And with reimbursement on the line (as in the Medicare world, or in the commercial world under the Patient Protection and Affordable Care Act), it's important to gain insight on the future risk of a plan's members. The following rudimentary example explores the possibility of using hidden Markov models to estimate future risk.

Risk scores (which we can observe directly) are a function of an individual's health status (which we cannot observe directly). Suppose that we choose to model an individual's health as one of four states:

- Healthy
- Mildly sick
- Moderately sick
- Severely sick.

To build our hidden Markov model, actual risk scores are needed for individuals from one year to the next. The HMM package in R requires a string of data as input; if we used 1,000 data points to build our model, this would be representative of 1,000 years of a single individual's risk score. With lapse rates what they are, it is unlikely that a health plan has 1,000 years of consecutive data on a single individual. Moreover, it would be nice to build our model on more than the risk scores of one individual. An alternative is to string together multiple individuals' risk scores to produce a chain of data—suppose that we have four years of data for Person A: 1.08, 0.74, 1.42 and 1.20. We could then find a Person B, with risk scores of 1.20, 0.96, 1.77 and 0.80. Continuing this process, we could build a chain of data of arbitrary length on which to base our model. Of course, in a real model, you would probably also have to worry about other variables when sewing together members this way (such as age and gender). However, note that we do not need to know our members' health statuses in order to build the HMM.

For this exercise, we built fictional data, assuming that for each health status risk scores are distributed in a lognormal fashion (with mean risk scores getting progressively higher for less healthy statuses). For those who wish to follow along, here is some sample R code (for use with the hidden Markov model package found at the Comprehensive R Archive Network[1]):

- library(HiddenMarkov)
- delta<-c(1/4,1/4,1/4,1/4)
- Pi<-matrix(c(0.9,0.08,0.01,0.01,0.045,0.9,0.045,0.01,0.01,0.045,0.9,0.045,0.01,0.01,0.08,0.9),byrow=TRUE,nrow=4)
- x<-dthmm(NULL,Pi,delta,"lnorm",list(meanlog=c(log(0.15),log(0.3),log(0.55),log(1.4)),sdlog=c(log(5),log(5),log(5),log(5))),discrete=TRUE)
- x<-simulate(x,nsim=1000)

The last step of the code above is a random process, and so (if you are following along) you will have different sample data than we have used. Alternatively—and preferably—you could create a string of actual data from your own plan's experience.

Now we can create our four-stage HMM. As with most numerical methods, in order to build an HMM we must have an initial guess at the model structure. Suppose here that we believe that an individual stays at a given health level 70 percent of the time, switching to each of the other three health levels with 10 percent probability apiece. And suppose that we believe that, for each underlying health status, the risk score distributions are as follows:

- Healthy: Lognormal (parameters meanlog = log(0.15), sdlog = log(5))

- Mildly sick: Lognormal (parameters meanlog = log(0.3), sdlog = log(5))
- Moderately sick: Lognormal (parameters meanlog = log(0.55), sdlog = log(5))
- Severely sick: Lognormal (parameters meanlog = log(1.4), sdlog = log(5))

(The eagle-eyed among you will note that these parameters are precisely those used to produce the sample data above. If we aren't as lucky, we do run the risk that our model will converge to something other than the true optimal value.)

The following R code then builds a discrete-time HMM:

```
• Pi4<-matrix(c(0.7,0.1,0.1,0.1,0.1,0.7,0.1,0.1,0.1,0.1,0.7,0.1,0.1,0.1,0.1,0.7),byrow=TRUE,nrow=4)
• a4<-dthmm(NULL,Pi4,delta,"lnorm",list(meanlog=c(log(0.5),log(1.1),log(2),log(5)),sdlog=c(log(5),log(5),log(5),log(5))),discrete=TRUE)
• a4$x<-x$x
• y4<-BaumWelch(a4)
• print(summary(y4))
```

With our sample data, this code produces an HMM with transition matrix:

0.959	0.041	0.000	0.000
0.000	0.962	0.000	0.038
0.097	0.000	0.903	0.000
0.000	0.555	0.095	0.905

and state distributions of:

- Healthy: Lognormal (meanlog = log(0.23), sdlog = log(5.16)); expected value 0.87
- Mildly sick: Lognormal (meanlog = log(0.41), sdlog = log(4.80)); expected value 1.42
- Moderately sick: Lognormal (meanlog = log(1.27), sdlog = log(4.66)); expected value 4.17
- Severely sick: Lognormal (meanlog = log(1.42), sdlog = log(4.48)); expected value 4.38

(Note that we did not exactly replicate the parameters used to generate the initial random data, nor should we expect to have done so. The Baum-Welch algorithm develops the most likely HMM to produce the observed data, but there are other distributions that could produce the same set of random data. Just as we do not expect the most likely outcome of rolling a fair six-sided die to be a "6," we do expect some "6" values when we repeatedly roll a fair six-sided die.)

Suppose that we have an individual with a risk score of 0.9 in the most recent year, and we would like to estimate that person's risk score in the upcoming year. First, we would need to estimate the current health state—this would be done most accurately by using a maximum likelihood approach. However, in practice (and in particular with lognormal distributions), it is simpler merely to assume that the state is closest to the individual's observed risk score (in this case, a risk score of 0.9 would most closely correspond to the healthy state).

Using the transition matrix above, we would then estimate that, in the coming year, the individual will be healthy (with 95.9 percent probability) or mildly sick (with 4.1 percent probability). The distribution of the expected risk score would be a composite lognormal distribution, and we would predict the mean future risk score to be 0.89. (The composition of lognormal distributions is not easy to express in a closed form, but one could find the entire distribution using a numerical method, and could then compute likelihoods such as the probability that next year's risk score will exceed 10.)

CONTINUED ON **PAGE 20**

This is meant to be a (somewhat) simple example, and you are probably already coming up with improvements to the algorithm. For instance, one might expect that the most recent two years of risk scores would be a better predictor of next year's risk score than this year's risk score alone. This would require a larger HMM to implement (because of the need to track both last year's and this year's health status), but would be straightforward to accomplish.

How many states should one build into a model? That's more of the art of the HMM—similar to the question of which distribution best fits a set of data. One fun exercise (for the reader) is to take the example above (either with random data or your own actual data), and try fitting HMMs of different sizes (and with different assumptions). Larger HMMs, with more available parameters, may fit the data better, but one will ultimately reach a point of diminishing returns. As well, there is a risk of over-fitting (remember that a model's true ability lies in how it performs with unseen data, not in simply replicating the data used to build the model).

REFERENCES

Zucchini, Walter, and Iain MacDonald. 2009. *Hidden Markov Models for Time Series*. Upper Chapman & Hill/CRC. Print. ▼

ENDNOTES

[1] HiddenMarkov: Hidden Markov Models. Retrieved Sept. 20, 2013, from http://cran.r-project.org/web/packages/HiddenMarkov/index.html.

Doug Norris

Doug Norris, FSA, MAAA, PhD, is a consulting actuary at Milliman Inc. in Denver, Colo. You can reach him at *doug.norris@milliman.com*

Brian Grossmiller

Brian Grossmiller, FSA, FCA, MAAA, is consulting actuary with DeepView Solutions in Portland, Ore. He can be reached at *brian.grossmiller@deepviewsolutions.com*