SOCIETY OF ACTUARIES Forecasting and Futurism Section

ISSUE 8 | DECEMBER 2013

Forecasting & Futurism

NEWSLETTER

Embrace the Future—But Beware the Smug

By Dave Snell

Actuaries
Risk is Opportunity.®

ISSUE 8 | DECEMBER 2013

Forecasting & Futurism

NEWSLETTER

Published by the Forecasting and Futurism Section Council of the Society of Actuaries

This newsletter is free to section members. Current issues are available on the SOA website *(www.soa.org)*.

To join the section, SOA members and non-members can locate a membership form on the Forecasting and Futurism Section Web page at *www.soa.org/forecasting-futurism*.

Embrace the Future—But Beware the Smug

By Dave Snell

When I mention *Back to the Future*, most readers will remember some favorite scenes from the movie. It was a refreshing comedy where a teenage boy, Marty (Michael J. Fox), travels back in time and meets his parents as teenagers. The movie came to mind after reading Alberto Abalo's excellent article, "The Future Ain't What It Used to Be," where he mentions his heightened appreciation, as a new parent, for life insurance. I also had flashbacks to the birth of our first child … and my associated purchase (like Alberto's) of my first life insurance policy. As he aptly summarized, we have a lot to be proud of in the Forecasting & Futurism (F&F) Section. One source of pride is that we are providing some state-of-the-art actuarial tools and techniques in our session presentations and our newsletter articles. Some of them are from other disciplines, and the jargon can make it more difficult to comprehend without specialized knowledge outside our usual actuarial education. Another reason for pride is that we are striving to make these topics readable and understandable and to show how they can help you.

That's one reason *Back to the Future* was popular. It took a difficult concept, such as time travel, and brought to light thought-provoking ideas (like what happens if you interfere with the courtship of your parents) that built upon the notions of many years ago. H.G. Wells published his science fiction novella, *The Time Machine*, in 1895; but *Back to the Future* added whimsical examples that gave the idea more appeal to an audience nearly a century later. Likewise, this issue is packed with new ideas that have origins from decades ago or even longer; but it provides new extensions, examples or insights that make them more relevant to your toolkit today.

Artificial society modeling goes back to at least 1996, when Robert Axtell wrote about Sugarscape; and the idea behind Sugarscape goes back even further—to Thomas Schelling's "Models of Segregation," written in 1969. Yet, for many years, Sugarscape was treated more like a recreational exercise. Ben Wolzenski, last year, modified the Sugarscape model to investigate insurance sales. This year, in his article

"A Return Visit to the Sugarscape," he revisits the use of agent-based modeling and investigates the impact on life insurance sales from factors such as increased unemployment, deferred household formation and increased productivity. It's a good read, where Ben explains what results he expected, how the results differed from what he expected, and the insights he gained from an analysis of the differences. He also discusses some interesting concepts such as "wasted productivity," which I found intriguing.

Markov models date back to 1906, and hidden Markov models (HMMs) to 1960, yet Brian Grossmiller and Doug Norris have given them new life and actuarial applicability in "Hidden Markov Models and You, Part Two" (a continuation of their HMM article in our July 2013 issue) quantifying the likely health claims that a particular individual will have over the next 24 months. It's a long article, and, frankly, I recommend that you download the Excel workbook they supplied (link shown in the article) and utilize it to better follow along as you read it. Additionally, you may wish to reread part one of their HMM article, "Hidden Mar-

CONTINUED ON **PAGE 4**

EMBRACE THE FUTURE … | FROM PAGE 3

kov Models and You," in the July 2013 issue of *Forecasting and Futurism Newsletter*. They also supply some R code for you. HMMs are an extension of Markov processes, which are currently on the actuarial syllabus. They allow you to infer the matrix of state transitions when it is not known. This is a powerful technique that you may be able to apply to many modeling situations where your data is affected by external conditions.

NeuroEvolution of Augmenting Topologies (NEAT) sounds brand new; but they were first described in 2002 as a variant of neural network theory, which goes back to the 1940s. Recent research on human brains suggests that the neocortex employs a very efficient neural network that can allow you to recognize a friend's face in under half a second even when she has a new hairstyle, makeup, contact lenses and she is not looking directly at you. Conversely, standard, if-then logic approaches that are based on many pre-defined rule-sets give disappointing results even with a supercomputer. The better recognition systems now use neural nets; but these are not intuitive to create. Jeff Heaton, in "A NEAT Approach to Neural Network Structure," explains to us what NEAT networks are, and how they address some of the drawbacks of more conventional neural networks, such as the tedium of setting appropriate weights for the connections. These advance the toolset we have for quick pattern recognition and classification problems.

A rapidly growing area of classification and regression techniques is that of predictive modeling (PM). Actuaries have made models for decades with the intent of predicting the financial impact of future risks; but the advent of big data is forcing us to make better use of our advanced statistical training. We think this is so germane to the actuarial profession that with this issue we are adding an ongoing PM column. Richard Xu starts us out with a strategy for success with PM in his article, "Modeling Process." In clear diagrams and associated explanations, he gives us a five-step process to help ensure that we maintain focus on the business problem we set out to solve, and that we have a clear path to solutions that show not just correlation between variables, but, more importantly, causal relationships. Quoting from Richard's article, "Statistical modeling is potentially a double-edged sword. If applied correctly, it is a very powerful and effective tool to discover knowledge in data, but in the wrong hands it can also be misused and generate absurd results."

The Oracle of Delphi takes us back to the eighth century B.C., and she might have been one of the first members of the F&F Section; but perhaps since the SOA did not exist back then, the Delphi method did not get much traction until its rebirth in 1944 for the Army Air Corps and then its more formal development in 1959 by the RAND think tank. The basic idea is that group opinions can be more accurate than individual opinions, and the Delphi approach facilitates the gathering of the collective wisdom without the biasing effect of hierarchical individual relationships. F&F has been a leader in the use of Delphi studies, and in this issue we discuss two of the more recent ones: Ben Wolzenski describes a joint F&F and Long Term Care (LTC) Section study in "Land This Plane—A Delphi Study about Long-Term Care in the United States" that garnered more than 100 pages of ideas from the diverse panel of 50 experts. Somewhat surprising to me, the overwhelming majority (95 percent) felt the need for an active government role to address the LTC issues and "promote the general welfare." The article title is from the code name for the study. "Land This Plane" denoted the lofty objective, which was "to create a vision for how America ought to deal with the impending long-term care crisis."

Our second Delphi article is a reprint (with permission) of an article by Paula Hodges, from the Product Development (PD) Section newsletter, *Product Matters!* In yet another outreach from our section, a joint study with the PD Section, F&F Council members Ben Wolzenski and Alberto Abalo, along with Paula Hodges, of the PD Section, conducted a real-time Delphi session at the 2013 Life & Annuity Symposium. Paula describes this session showing "how additional information and the anonymity of the experts influenced changes in the ultimate consensus of the group" in her article "Delphi Study in Real Time—Life & Annuity Products and Product Development."

The concept of genetic algorithms dates back to 1954 when Nils Aall Barricelli first began to simulate evolution on a computer; but the first book on genetic algorithms was by John Holland, in 1975. Of course, nature has been utilizing them since the beginnings of life as we know it. Unfortunately, that resulted in a lot of genetic jargon baggage in previous presentations (including my own) on genetic algorithms to solve insurance problems. In my article, "Genetic Algorithms Revisited—A Simplification and a Free Tool for Excel Users," I attempt to demystify them by breaking away from all the intimidating biological terms and just showing how they can be understood as simple processes. My goal is to teach how to make a genetic algorithm (for those who wish to know) and how to use one even if you don't care how the innards work. I created a general purpose Excel add-in that allows you to use genetic algorithms to solve some types of problems not easily solved by other methods.

Many thanks are due to Alberto Abalo and Doug Norris, our contest judges for the F&F genetic algorithm contest. They give us a succinct but highly informative summary of the contest results in "And the Winner Is …" where they announce the winning entry: "Diagnosing Breast Tumor Malignancy with a Genetic Algorithm and RBF Network."

Jeff Heaton was our winner, and he was awarded the prize, a new iPad, at the F&F breakfast meeting in San Diego during our SOA annual meeting. His entry impressed the judges as an excellent example of an actuarial application of genetic algorithms to help predict breast cancer. We are including his entry descriptive write-up in this issue. Jeff also includes a link to his program, in C# (pronounced see sharp), to solve this type of problem. In the coming issues, we hope to have more machine-learning articles from Jeff. We are happy to have him as an associate (non-actuary) member of F&F.

Throughout this introduction to the current issue I make note of the value-added benefit of the spreadsheets our authors have provided for you. Yet, we have included an article from an auditing firm: "Are Spreadsheets Sabotaging Your Accuracy?" by Steve Epner; and he seems to believe that "the continued use of spreadsheets to manage mission-critical functions is an unacceptable risk for 21st century firms."

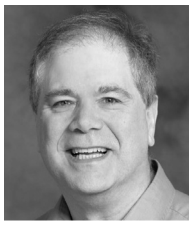
Why would we include this seemingly contrarian view?

Let me explain by quickly recapping an episode of *South Park* (season 10, episode 2) titled "Smug Alert":

> Stan sings the praises of hybrid cars and the whole town decides to drive them. The most popular model is the Toyonda Pious (Toyota Prius). Ranger McFriendly points out that even though hydrocarbon emission levels are down, the town now has a more serious problem—that the Pious owners (the owners, not the cars) emit "self-satisfied garbage" that has polluted the air far worse than smog. This environmental disaster is called "smug."

I have to confess. I drive a Prius; and it does tend to foster smug. It's a challenge to keep it under control. In F&F, we are touting a bunch of new technologies and techniques from the collection of complexity sciences that sometimes seem to diminish the value of techniques you learned in preparation for the actuarial exams. In some cases we are even directing you to spreadsheets (as well as other nontraditional tools and computer languages, such as R, that your IT area probably does not support). We promote these inferential, mostly inductive, methods as additional arrows for your quiver of tools. Personally, I think that spreadsheets can have a valid role in mission-critical applications for insurance companies; but that we have to be responsible in our usage of them.

The articles in this issue are to help you understand, explain, and to some extent sell your companies on the benefits of these newer technologies and techniques. In some respects, they are slick and new and look superior to "old" ways, just as the DeLorean time machine in *Back to the Future* may have looked superior to the H.G. Wells version. However, sometimes they can unknowingly attract smug. When I need to bring home 4 foot by 8 foot plywood sheets for my construction projects, I drive our old van, not the Prius. Embrace the future; but beware the smug! ▼


Dave Snell

Dave Snell, ASA, MAAA, is technology evangelist at RGA Reinsurance Company in Chesterfield, Mo. He can be reached at *dsnell@rgare.com.*

The Future Ain't What It Used to Be

By Alberto Abalo

On July 6, 2013, my way of thinking about the future changed. On that day I became a father.

As is likely true for most readers of this newsletter, I have always been captivated by questions about The Future and humankind's efforts to predict what comes next. *Blade Runner* and *2001: A Space Odyssey* were two of my favorite movies growing up, both feeding into my fascination with plausible future worlds, each making me ponder deep questions from an early age. What if machines could think? What if people could live forever? What does it mean to be human? The appeal of spending my days lost in thought over what-if scenarios played no small part in my choice of profession and in my participation in the affairs of this special interest section in particular.

Evelyn Luz Abalo was born July 6 at 3:54 a.m. Since that early morning (or was it late night?), my outlook changed suddenly and without the comfort of deliberate evaluation. Thinking about The Future, once the source of amusement, is now charged with a practical uncertainty that is sometimes too much to bear. What happens when the world's oil supply runs out? Where will humans live after the ice caps melt? How much will college tuition cost in 2021? (You don't want to know.) And the truly unthinkable: Who would take care of my daughter if I suddenly died?

I made a decision about the future and bought my first life insurance policy. The transaction taught me more about insurance and the importance of our profession than a decade of working in the industry did. (Forget Wal-Mart: Insurance should be sold at Babies R Us.) In an insurance transaction, real money is traded in return for a promise. Never has the implication of that hit me harder than when the money was coming out of my own checking account, and the promise was to protect the financial well-being of my family. Whether that money is taken responsibly is determined by the work of an actuary. What we do is important—not just for the insurance companies we work for, but for families likes yours and mine. Preserving the integrity of the insurance systems we help design requires careful and disciplined thought about the future and a constant evaluation of state-of-the-art practices in risk management, forecasting and assumption setting.

Alberto Abalo

Alberto Abalo, FSA, CERA, MAAA, is a principal at Oliver Wyman in Atlanta, Ga. He can be reached at alberto.abalo@oliverwyman.com.

A stated purpose of our section is to introduce our members to forecasting and futurism methods that are being successfully used in other fields and might serve as complements or alternatives to traditional actuarial techniques. At its core, our mission is to look for innovative ways to solve problems, and to share our knowledge with the profession at large. The newsletter you are now reading is a big part of that. In my opinion, our section's newsletter boasts some of the most stimulating material that can be found in any SOA publication. The proof is in the pudding: Our previous issue covered predictive modeling, Bayesian networks, hidden Markov models, Delphi studies and behavioral economics. It even contained references to *The Wizard of Oz* and Pink Floyd's "Dark Side of the Moon" (the authors claim it was a coincidence). I can't wait to read what Dave Snell and our thoughtful contributors have put together for the current issue.

As part of its educational mandate, the section maintains a presence at key SOA meetings, sponsoring sessions at the Life & Annuity Symposium, health meeting and annual meeting. Our docket of sessions this year included predictive modeling, real-time Delphi, agent-based models and genetic algorithms. In 2013 we also sponsored well-received webcasts on emerging risks and predictive modeling. The two webcasts were presented in collaboration with the Joint Risk Management and Reinsurance sections, respectively. We have found that our potential as a section, whose focus is on tools and techniques, is fully realized in collaboration with other sections whose purpose is more practical in nature.

Before signing off, I want to welcome the newest members of the Forecasting and Futurism Section Council: Brian Holland, Haofeng Yu, Vicki Zhang and Geof Hileman. I

look forward to working with each of you. I also want to thank the outgoing members of the council: Clark Ramsey, Brian Grossmiller, Jon Deuchler and Donald Krouse. Their service over the past three years (four in Jon's case) has led the section to where it is today. While their presence on the council will be missed, I hope to continue counting on them as friends of the section and will seek them out for always-thoughtful conversation at future SOA meetings.

On behalf of the section council, I welcome your thoughts and suggestions on how to improve the section. Feel free to contact me or any of the council members and let us know how we can make your section more valuable to you and the profession. While those of us in the section find The Future to be intrinsically interesting, I now recognize that for the tools and methods we study to be truly valuable we need to find practical applications for them in our profession. I invite everyone reading this to work with us in making this happen.

Enjoy the newsletter!

Regards,

Alberto Abalo ▼

A Return Visit to the Sugarscape

By Ben Wolzenski

ABSTRACT

This article describes the use of artificial society modeling to gauge the effect of insurance agent population and effectiveness on individual life insurance sales. This is part of an ongoing effort to extend the use of one type of agent-based modeling beyond health care, but is far from a practical at this point.

The December 2012 edition of this newsletter contained the article "Artificial Society Modeling with Sugarscape." In brief, the article described an artificial society as an agent-based model in which the user defines the rules for the agents ("citizens") and the environment. Sugarscape is an artificial society model described by Joshua Epstein and Robert Axtell in their pioneering book, *Growing Artificial Societies.* The article went on to describe how the online applet for Sugarscape could be adapted and interpreted to roughly model the effect of societal changes on future life insurance sales.

For example, a simulated increase in unemployment led to fewer life insurance purchases; delayed household formation produced fewer short-term but greater long-term insurance purchases; and combining increased unemployment, deferred household formation and increased productivity led to greater variability of results over multiple model simulations.

Ben Wolzenski

Additional modeling has explored the effect of changes in the relative population of insurance agents and their effectiveness. But first, let's review the Sugarscape model a bit. The Sugarscape model is a large grid, with an initial population of "citizens" who need to move about to gather goods they need to survive. In moving about, citizens can meet each other. When they meet, one of several things can happen. One citizen may cause the other to change to the "cultural group" of the first. Disease may be transmitted from one to the other. The two can mate, creating a child. Most frequently, the two can engage in trade. In trade, the two citizens exchange goods. In our customized version of Sugarscape, one such transaction is a life insurance purchase.

Ben Wolzenski, FSA, MAAA, is managing member at Actuarial Innovations, LLC in St. Louis, Mo. He can be reached at *bwolzenski@rgare.com.*

One of the features in the customized model is the random designation of a small percentage of citizens to be insurance agents. When one of the citizens is an insurance agent, there is a greater probability that the transaction is a life insurance purchase. What happens when the relative population of agents decreases? To what extent could this be offset by increased sales effectiveness of insurance agents? What outcomes would be produced by greater effectiveness of insurance sales not involving insurance agents?

The baseline Sugarscape model for exploring these questions is one in which the probability that a citizen will buy a life insurance policy in any year is close to recent U.S. experience (about 3.4 percent) and in which the probability that the purchase is from a life insurance agent is 90 percent.[1]

The first simple modification produced close to predicted results. What if there were 50 percent fewer agents, with no other changes? The total number of sales should drop to 56 percent of the previous level.[2] The actual average result was 58 percent of baseline, which was six-tenths of a standard deviation greater than expected.[3] The proportion produced by agents was 81 percent, as expected.

Similarly, would the algebraic prediction of the needed increase in frequency of non-agent sales produce the original number of sales? A 50 percent reduction in the 90 percent of baseline sales from agents equals 45 percent of baseline, so the remaining 55 percent would have to come from non-agents. Non-agents would have produced 10.6 percent of the baseline number without an increase in frequency,[2] so they would need a rate 5.2 times the baseline probability of sale by a non-agent. When that increased probability was tested in the model, the resulting total number of sales was 98 percent of the baseline, with 45 percent coming from agents, as expected.

With these results, I expected that an algebraically predicted increase in agent productivity would make up for the smaller number of agents. However, the model did not produce such a result. When productivity (probability of a sale upon contact with an insurance agent) was nearly doubled[4] to offset the 50 percent decrease in the number of agents, the mean number of sales was only 84 percent of the baseline amount, about three standard deviations lower than expected. The shortfall was entirely due to low agent production; non-agent production was actually slightly higher than predicted. My first reaction was that I must have coded the input incorrectly, so I would have to rerun all the simulations and re-record the results. But when I checked the input, it was correct, so something else was going on in Sugarscape.

In the model, whether any transaction is a life insurance sale is determined by whether a number, obtained by successive multiplication of probability factors, is greater or less than a random number between 0 and 1. The base probability starts out fairly low, but is adjusted based on the citizen's age, cultural group membership, wealth, number of children, and most significantly by whether the transaction is with an insurance agent. It turned out that the effect of successive multiplications with much higher agent productivity (probability of producing a sale) was to produce a significant number of comparison numbers greater than 1. (For example, the baseline probability of an insurance sale if the transaction is with an insurance agent is 0.5, and this was increased to 0.99 with higher productivity. However, if the citizen had three children, both probabilities would be increased 30 percent, to 0.65 and 1.29, respectively.) Any number greater than 1 is effectively wasted productivity, since the random number to which it is compared cannot exceed 1. Thus, doubling agent productivity did not double the number of sales. Of course this was just an idiosyncrasy of the model, but it suggests an analogy to a point of diminishing returns for agent productivity in the real world.

This suggests that it is worthwhile to make other tests about the boundaries on the interpretive use of the model. In one such test, I looked at factors that would influence the size of the population, which for our purposes must be relatively stable after an initial period. Tests showed that if the maximum vision of citizens (how far away they can see to find goods and other citizens) is too small, the population will die out unless the environment is richly endowed with goods, and even then the population may be highly unstable due to other random variables. On the other hand, the initial population density (the percentage of cells occupied by citizens at time equals zero) appears to have no effect on the ultimate level and stability of the population. Rather, it is determined by how other model variables are set. For example, an initial population density of 5 percent produces about the same ultimate population size and stability as an initial population density of 80 percent if all other model variables are the same. Clearly, I still have much to learn about the artificial world of Sugarscape.

WHAT HAPPENS WHEN THE RELATIVE POPULATION OF AGENTS DECREASES? TO WHAT EXTENT COULD THIS BE OFFSET BY INCREASED SALES EFFECTIVENESS OF INSURANCE AGENTS?

Readers' questions about the Sugarscape model and any suggestions as to future testing are welcome! ▼

ENDNOTES

1 A 2005 article in *The Actuary* reported: "In 2003, independent agents accounted for over half of sales, career agents sold about 40 percent and the remaining 10 percent came from a number of newer channels including brokers, web sales and banks. Newer channels are growing their market share…." While brokers would be included in my definition of agents, Web sales and banks would not. Ten percent is a rough estimate of what non-agent sales may have grown to 10 years later (2013 versus 2003). Better data from any reader would be welcome.

2 Agent sales would be 50 percent of 90 percent, which equals 45 percent of the baseline total. The 10 percent of non-agent sales would increase slightly to 10.6 percent of the baseline total. That is because with the same total population, a decrease in the number of agents would result in an increase in the number of non-agent citizens. The total is rounded to 56 percent in the text above so as not to overstate precision.

3 The results compared were 30 simulations of 1,000 generations each time. The standard deviation of the total number of sales was approximately 4 percent of the baseline number of sales.

4 The increase in the relative population of non-agents at baseline productivity only required 198.7 percent of baseline agent productivity for the algebraic prediction.

30 March to 4 April 2014

www.ICA2014.org

Join more than 2,000 actuaries from across the globe at the 30th International Congress of Actuaries!

- Earn up to 27 continuing education hours from sessions covering the latest global trends.
- Network with peers from around the world.
- Enjoy cultural and historical activities in and around Washington, D.C.

Register online today at www.ICA2014.org

Contact info@ica2014.org with any questions.

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:

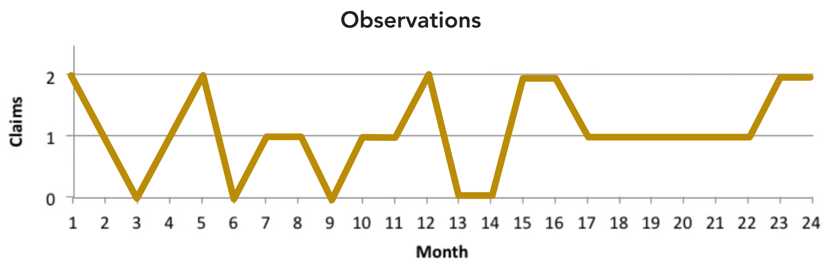
- Alpha (α): Forward probabilities, which are generated from an initial estimate of the hidden state at the first data observation, and calculated forward from there.
- Beta (β): Backward probabilities, which are computed as a conditional probability from the last (final) data observation.
- Gamma (γ): 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:

$\alpha_1(1)$ = Initial Pr(State 1) * $Pr_1(2)$ = 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:

$\alpha1(2)$ = [$\alpha_1(1)$ * Pr(State 1 → State 1) + $\alpha_2(1)$ * Pr(State 2 → State 1)] * $Pr_1(1)$ = [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.

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 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		

Figure 8: Hidden State Estimate at Iteration 1

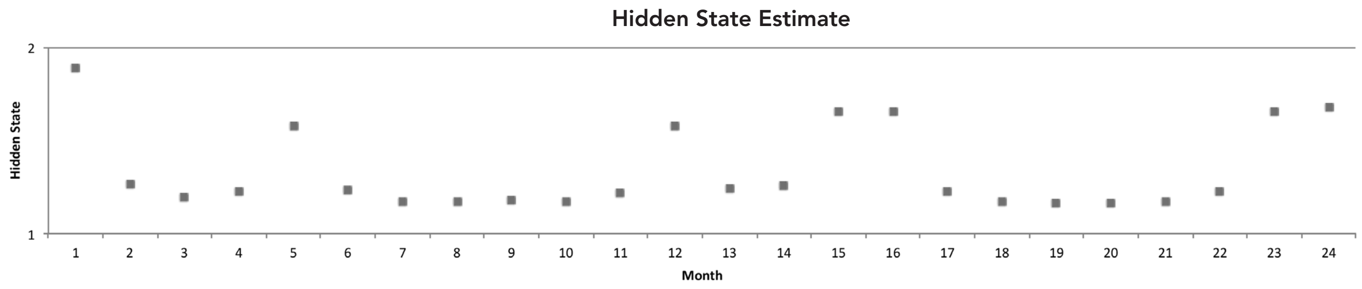


Figure 9: Hidden State Estimate at Iteration 10

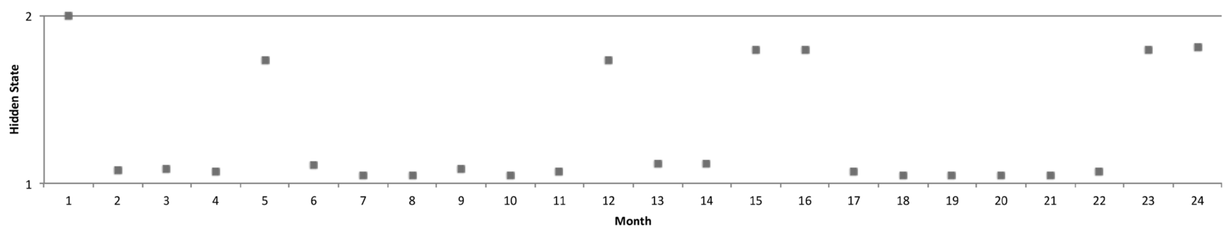
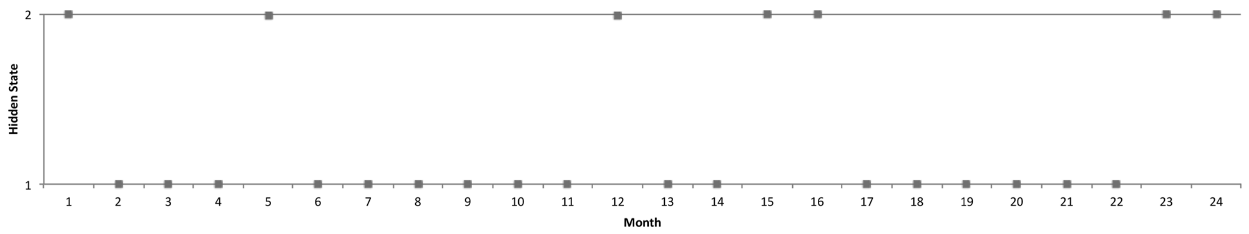


Figure 10: Hidden State Estimate at Iteration 100



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*

ATTEST TO YOUR CPD HOURS

It's time to attest to your CPD credits.

Visit **soa.org/cpd** to learn more.

A NEAT Approach to Neural Network Structure

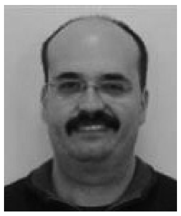
By Jeff Heaton

Neural networks are a mainstay of artificial intelligence. These machine-learning algorithms can be used for regression and classification. Typical feed forward neural networks require you to specify an exact structure of layers and neurons. This article will introduce you to three recent innovations in neural network design that alleviate you of some of the structural decisions typically required of the neural network practitioner. These three related neural networks are named NEAT, HyperNEAT and HyperNEAT-RS. Kenneth Stanley of Houston University is the primary researcher behind NEAT and many of its adaptations.

NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT)

NEAT is a neural network structure developed by Ken Stanley in 2002 while at the University of Texas at Austin. (Stanley 2002) NEAT uses a genetic algorithm to optimize both the structure and weights of a neural network. The input and output of a NEAT neural network are identical to a typical feed-forward neural network. For a review of feed-forward neural networks, refer to "Predictive Modeling" in the July 2013 issue of *Forecasting and Futurism Newsletter* (Xu 2013).

A NEAT network starts out with only a bias neuron, input neurons and output neurons. There are no initial connections between any of the neurons! Of course, such a completely unconnected network is useless. NEAT makes no assumptions. What if one of the input variables is statistically independent of the output? NEAT will often discover this by never evolving optimal genomes to connect that statistically independent input neuron to any other part of the network.


Jeff Heaton

Another very important difference between a NEAT network and an ordinary feed-forward neural network is that NEAT networks do not have clearly defined layers. NEAT networks do have a clearly defined input and output layer. However, the hidden neurons do not organize themselves into clearly delineated layers. A similarity between NEAT and feed-forward networks is that NEAT uses a sigmoid activation function, similar to feed-forward networks.

Jeff Heaton is EHR informatics scientist at RGA Reinsurance Company in Chesterfield, Mo. He can be reached at jheaton@rgare.com.

NEAT networks make extensive use of genetic algorithms. For a review of genetic algorithms see "Genetic Algorithms—Useful, Fun and Easy!" in the December 2012 issue of *Forecasting and Futurism Newsletter* (Snell 2012). NEAT uses a typical genetic algorithm that includes both crossover and mutation. Both mutation and crossover are operations that involve one or more parents to produce children. The crossover operation uses sexual reproduction to produce a child from two parents. The mutation operation uses asexual reproduction to produce a child from one parent.

NEAT Mutation

NEAT mutation consists of several mutation operations that can be performed on the parent genome. These operations are discussed here.

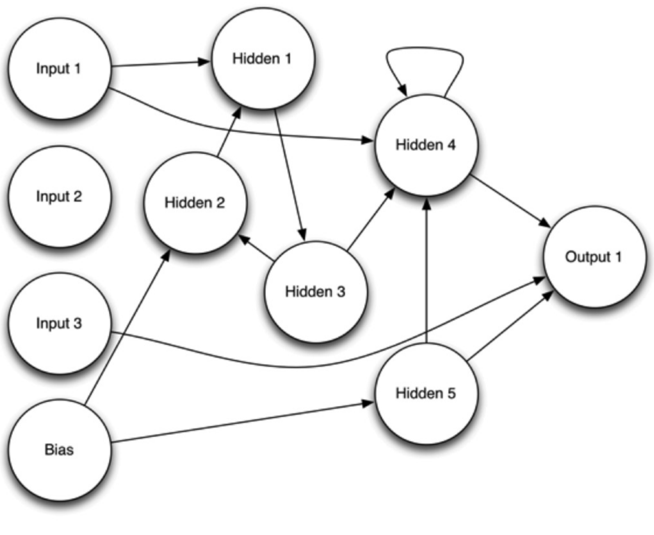
- **Add a neuron:** A neuron is added by first selecting a random link. This link is replaced by a new neuron and two links. The link is effectively split by the new neuron. The weights of each of the two new links are selected so as to provide nearly the same effective output as the link being replaced.

- **Add a link:** Two random neurons are chosen: a source and destination. The new link will be between these two neurons. Bias neurons can never be a destination. Output neurons cannot be a source. There will never be more than two links in the same direction between the same two neurons.

- **Remove a link:** Links can be randomly selected for removal. Hidden neurons can be removed if there are no remaining links interacting with them. A hidden neuron is any neuron that is not input, output or the single bias neuron.

- **Perturb a weight:** A random link is chosen. Its weight is then multiplied by a number from a normal random distribution with a gamma of one or lower. Smaller random numbers will usually cause a quicker convergence. A gamma value of one or lower will specify that a single standard deviation will sample a random number of one or lower.

The mutations are weighted so that the weight perturbation occurs the most frequently. This allows fit genomes to vary their weights and further adapt through their children. The structural mutations happen with much less frequency. The exact frequency of each operation can be adjusted by most NEAT implementations. The following diagram shows a typical NEAT genome.



You can see from the above that input 2 was disregarded. You can also see that the layers are not clearly defined. There are recurrent connections, and even connections that skip directly from input to output.

NEAT Crossover

NEAT crossover is more complex than many genetic algorithms. Most genetic algorithms assume that the number of genes is consistent across all genomes in the population. This is not the case with NEAT. The NEAT genome is an encoding of the neurons and connections that make up an individual genome. Child genomes that result from both mutation and crossover may have a different number of genes than their parents. This requires some ingenuity when implementing the NEAT crossover operation.

NEAT keeps a database of all the changes made to a genome through mutation. These changes are called innovations. Innovations are created to implement mutations. However, the relationship between innovations and mutations is not one to one. It can take several innovations to achieve one mutation. There are only two types of innovation: creating a neuron and a link between two neurons. One mutation might result from multiple innovations. A mutation might also have no innovations. Only mutations that add to the structure of the network will generate innovations. The following list summarizes the innovations potentially created by the previously mentioned mutation types.

- **Add a neuron:** One new neuron innovation and two new link innovations.
- **Add a link:** One new link innovation.
- **Remove a link:** No innovations.
- **Perturb a weight:** No innovations.

It is important to note that NEAT will not recreate innovation records if such an innovation has already been attempted. Additionally, innovations do not contain any weight information; innovations only contain structural information.

Innovations are numbered. This allows NEAT crossover to determine what pre-requisite innovations are needed for a later innovation. Crossover for two genomes occurs between innovations. This allows NEAT to ensure that all prerequisite innovations are also present. A naïve crossover, such as is used by many genetic algorithms, would potentially combine links with nonexistent neurons.

NEAT Speciation

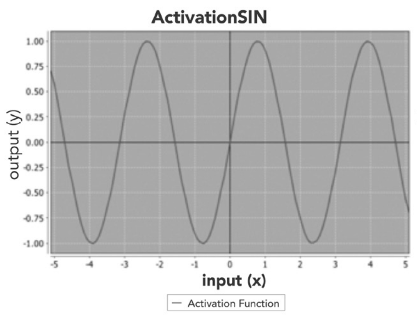
Crossover is a tricky proposition. In the real world crossover only occurs between members of the same species. This is actually a bit of a circular definition. Real-world species are defined as members of a population that can produce

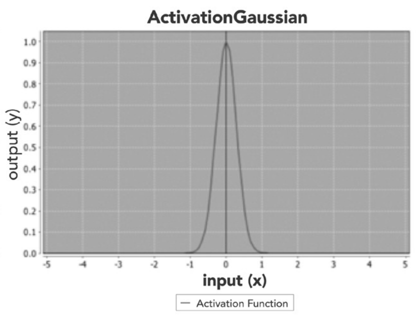
CONTINUED ON PAGE 24

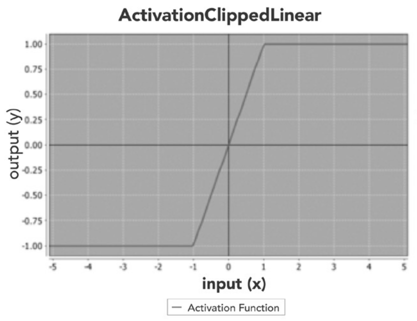
viable offspring. Attempting crossover between a horse and humming bird genome would be catastrophically unsuccessful. Yet a naïve genetic algorithm would certainly try!

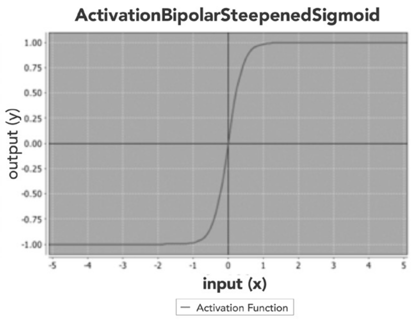
NEAT uses a type of k-means clustering to group the population into a predefined number of clusters. The relative fitness of each species is then determined. Each species is then given a percentage of the next generation's population count. The members of each species then compete in virtual tournaments to determine which members of the species will be involved in crossover and mutation for the next generation.

A tournament is an effective way to select parents from a species. A certain number of trials are performed. Typically we use five trials. For each trial two random genomes are selected from the species. The more fit of each genome advances to the next trial. This is very efficient for threading, and is also biologically plausible. You don't have to beat the best genome in your species, just the best genome you encounter in the trials! A tournament is run for each parent needed. One parent is needed for mutation and two for crossover.









In addition to the trials, several other factors determine the species members chosen for mutation and crossover.
One or more elite genomes are always carried directly to the next species. The number of elite genomes is configurable.
Younger genomes are given a bonus so they have a chance to try new innovations.

Interspecies crossover will occur with a very low probability.

All of these factors together make NEAT a very effective neural network type. NEAT alleviates you of the need to define a neural structure. Additionally, the non-level, recurrent nature of a NEAT network gives it some additional potential over regular feed-forward networks.

HYPERNEAT

Kenneth Stanley developed HyperNEAT in 2009. (Stanley 2009) HyperNEAT recognizes one very biologically plausible fact. In nature genotypes and phenotypes are not identical. What is the difference between a genotype and phenotype? The genotype is the DNA blueprint for an organism. The phenotype is what actually results from that plan.

HyperNEAT Genomes

A genome is the instructions for producing a much more complex phenotype. This is not the case with regular NEAT. The NEAT genome describes exactly, link for link, neuron for neuron, how to produce the phenotype. This is not the case with HyperNEAT. HyperNEAT creates a population of somewhat special NEAT neurons. These genomes are special in two ways. First, whereas regular NEAT always uses a sigmoid activation function, HyperNEAT can use any of the following activation functions:

- Clipped linear
- Bipolar steepened sigmoid
- Gaussian
- Sine

You can see these activation functions here.

The second difference is that these NEAT genomes are not the final product. They are not the phenotype. However, these NEAT genomes do know how to create final products. The end-result phenotype is a regular NEAT network with a sigmoid activation function. The above four activation functions are only used for the genomes. The ultimate phenotype always has a sigmoid activation function.

HyperNEAT Phenotype

You might be wondering how a neural network can be used to create another neural network. This requires one additional structure, called a substrate. The substrate defines the final structure of the phenotype. The genome neural network is then queried to determine the weight for each connection specified in the genome. There are many different substrate structures. Most HyperNEAT implementations allow you to construct substrate structures of your own. Substrate structures are very similar to traditional neural network structures in that the structure is fixed. The genome neural network's structure is dynamic, just as in NEAT. However, the phenotype structure is fixed. The genome neural network defines the weights.

The substrate is a 3D model contained in a cube. The bias and input neurons are typically embedded on one face of the cube. The output neurons are all embedded on the opposite face. Hidden neurons, if there are any, are placed in the 3D space between these two faces. Links are defined between these 3D neurons.

The substrate does not need to be a 3D cube. The substrate is actually a hypercube. In geometry, a hypercube is an n-dimensional analogue of a square (n = 2) and a cube (n = 3). Very often substrates consist of three dimensions. However it is also possible to use substrates of lower or higher dimensionality.

Why Use HyperNEAT?

You might be wondering why we would want to introduce the additional complexity of creating a genome neural network just to produce the resulting phenotype neural network. The reason is training time. When dealing with problems with a very large number of input and output neurons, training times can be very large. HyperNEAT networks are scalable.

Regular neural networks require a total retrain if you add or remove neurons. HyperNEAT allows you to change the number of neurons. One genome neural network can generate any number of phenotype neural networks of higher scale. This allows you to train at low scale and actually use the neural network at a higher scale. This allows the lower-scale training to progress much faster than the higher-scale neural network we plan to actually use.

HyperNEAT makes it possible to deal with neural networks that may ultimately have tens of thousands of input neurons. Such large neural networks could take an enormous amount of time to train. This is compounded by the fact that a population of such large neural networks would not fit into the memory of most computers.
For a real-world , consider an that generates trading signals for real-time financial data. Such a program may be ultimately used on minute or second HLOC bars. However, training can occur on larger time durations. This will speed training, but allow the neural network to take advantage of higher resolution data when back testing or actually in use.

ES HyperNEAT

NEAT's primary feature is that we no longer have to think about the structure of a neural network. HyperNEAT's primary feature is that we can quickly train with lower-scale test data and use the resulting neural network on higher-scale data. However, HyperNEAT takes away the primary advantage that NEAT gave us. With HyperNEAT we now have a substrate to architect. We are right back to designing neural network structure.

ES HyperNEAT was developed in 2010 to solve this issue. (Risi 2010) ES stands for "evolvable substrate." ES HyperNEAT is an extension of HyperNEAT that allows the substrate to evolve as well. This allows every aspect of the neural network to evolve. The end result is still a genome

CONTINUED ON **PAGE 26**

A NEAT APPROACH … | FROM PAGE 25

neural network that can create a regular NEAT network at any scale.

CONCLUSIONS

The three different types of NEAT network presented in this article represent some very recent research into neural networks. ES HyperNEAT is at the pinnacle of current NEAT research. ES HyperNEAT provides a practical way to model problems with a very large number of inputs. Kenneth Stanley is the primary researcher behind this type of neural network. To learn more about all three of these neural network types, you can visit his university home page at the following URL.

http://www.cs.ucf.edu/~kstanley/

CITATIONS

- Stanley, K., and R. Miikkulainen 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10(2): 99–127.

- Xu, R. 2013. Predictive Modeling. *Forecasting and Futurism Newsletter*. July, pp. 12–16.

- Snell, D. 2012. Genetic Algorithms—Useful, Fun and Easy! *Forecasting and Futurism Newsletter*. December, pp. 7–15.

- Stanley, K., D. D’Ambrosio and J. Gauci. 2009. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life* 15(2): 185–212.

- Risi, S., J. Lehman and K. Stanley. 2010. Evolving the Placement and Density of Neurons in the HyperNEAT Substrate. *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO-2010). New York, N.Y., pp. 563–570. ▼

PREDICTIVE MODELING IN INSURANCE

Modeling Process

By Richard Xu

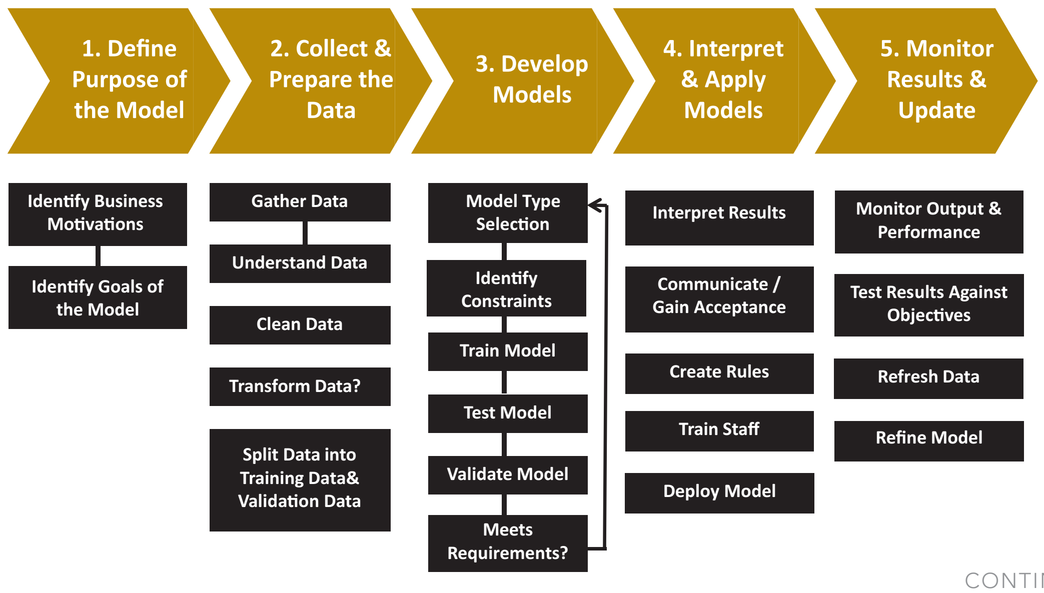
In the July 2013 issue of the Forecasting & Futurism Newsletter, we introduced basic techniques of statistical modeling that usually have applications in insurance. As there are increasing discussions about predictive modeling within the actuarial community, the Forecasting and Futurism Section Council members believe that we will only witness increasing applications of statistical modeling in the coming decade, and it will be beneficial to our readers to better understand predictive modeling and be well prepared for the modeling approach in every aspect of actuarial work. Starting from this volume, we decided to have a dedicated column on predictive modeling and discussion about various topics on how predictive modeling can be utilized to help actuaries to improve the effectiveness and efficiency of their work.

In this article, I will describe a general modeling procedure with an example. In the following newsletter, I plan to cover other topics, such as generalized linear models (GLMs) with their applications in insurance, classification and regression tree (CART) model with examples, applications of data clustering, other advanced models, data considerations, model considerations, etc. If there is any specific topic that our readers find of interest, we are happy to discuss it in this column.

To build a statistical model for in business is a very complicated process. It is nothing like an exercise in a classroom setting. There, a problem is usually well-defined, data are clean, and the focus is to learn modeling skills without consideration of practical implementation. In a real business world, actuaries may have to face many issues that have never been the topic in school. To achieve success in a modeling project, actuaries have to deal with these problems.

Although each modeling project is unique, there are certain features that are common to all projects. If a modeler can follow procedures that have been proven to be effective, some mistakes can be avoided and the project has more chance to succeed. Generally speaking, the following flowchart shows a best practice.

Modeling Procedure


1. Define Purpose of the Model
2. Collect & Prepare the Data
3. Develop Models
4. Interpret & Apply Models
5. Monitor Results & Update
Identify Business Motivations
Identify Goals of the Model
Gather Data
Understand Data
Clean Data
Transform Data?
Split Data into Training Data& Validation Data
Model Type Selection
Identify Constraints
Train Model
Test Model
Validate Model
Meets Requirements?
Interpret Results
Communicate / Gain Acceptance
Create Rules
Train Staff
Deploy Model
Monitor Output & Performance
Test Results Against Objectives
Refresh Data
Refine Model

CONTINUED ON PAGE 28

MODELING PROCESS … | FROM PAGE 27

Step 1: Define the Objectives

This may sound too naïve, but in reality there are many occasions in which a project starts without a clearly defined objective. As the project moves forward, the goal keeps changing and the team may lose focus and try to accomplish too much at one time. At the end, the project may fail to meet any business need.

Usually a project starts with business needs, where the objective may be vaguely known. Another possible scenario is the need to leverage the data that a company has already accumulated over time to gain a competitive advantage in the marketplace. In both cases, expertise is needed to define what could possibly be accomplished, from of the available modeling techniques and data sources. Without the needed experience, either the objective is not attainable by the current statistical model, the required data do not exist, or the model and data do not match.

Another important goal of this stage is to get the cooperation of the high-level management team. A successful predictive modeling project relies on the collaboration of many departments within an organization, and a change of company culture to a more data-driven approach is necessary. Without consensus from the management team, it is nearly impossible to have different divisions working together.

Step 2: Understanding Data and Business

Data is a critical component for an effective model. All relations and patterns that we can utilize for business have to come from data that a model is built on. It will never be overstated how important data is to statistical modeling. More often than not, a project could fail due to the lack of available data.

Generally speaking, an insurance company has accumulated a large quantity of data over the past decades. The data are usually good for the purpose of an experience study. However, current data usually lack the depth of information. Besides age, gender and location, insurers usually do not have much more information about their policyholders. This makes the modeling project nearly impossible for applications such as underwriting, fraud detection or a retention program. The solution is to find data from a third party, which may include credit score, financial information, motor vehicle records, etc. However, external data may bring up the privacy issue that many insurance companies try to avoid for risk of their reputation.

Equally important is the data quality. This could be another obstacle to a successful model. Actuaries are experts on data, and they will never be short of examples when there are many errors in data. In addition, data may come from different sources, and mismatched and missing values are always an issue with data.

It is always dangerous to build a model without full understanding of the data and the intended usage. That is often the difference between an experienced data scientist and brand new statistics graduate. It is important for the modeling team to work closely with the business and market experts to select a suitable model and variables, create derivative variables, or group data. All of these benefit from business understanding.

Step 3: Develop a Predictive Model

Rigorous mathematical training and decent business knowledge are prerequisites to being a good data scientist. On one hand, predictive modeling is built on statistical processes to find relationships in data, and solid understanding in modeling technique is a necessary component. On the other hand, the training from textbooks is far from enough to handle real projects. Thorough understanding of the insurance business and underlying business forces is equally important to building an effective model.

There are certain degrees of freedom to choosing types of models, but in reality the options are usually limited. Although there are many mathematic models available for potential in insurance (please see "Predictive Modeling" in last volume of Forecasting and Futurism Newsletter), you may find GLM discussed most frequently. One reason is its transparency and simplicity as it is a natural extension

of ordinary least squares (OLS) that all actuaries have had certain exposure to in their education. Other possible models include data clustering, CART model, etc. All of these have the one common feature, i.e., you can open the model and find business insights from the model. Another group of models—including neural network, random forest and support vector machine (SVM)—are black-box models. They are more powerful and effective in most cases, but lack of understanding leads to low acceptance in the . As actuaries are becoming more confident with modeling techniques, we may see more applications of these complicated models.

Once a model type is selected, to develop the model is more or less a process of selecting the right groups of variables, including their interactions, so that the model can best explain the observed data. To identify the most predictive variables and their combination is not a trivial task. It involves both statistical criteria and business sense. Certain statistical methods are available to select variables in a systematic and sequential manner, such as a stepwise procedure.

After a preliminary model is built, we would like to see its performance. There are statistical criteria for the purpose, such as deviance or "information criterion" metric, e.g., Akaike information criterion (AIC) or Bayesian information criterion (BIC). One large concern in modeling is so-called "fitting," where a model is so complicated relative to data that the model actually takes up noise in the data set rather than the actual underlying relationships. Validation can address both of these two issues. In the validation process, the developed model is applied to data that have not been used for model fitting. If the results deteriorate much from modeling results, fitting is most likely the reason. Also, validation results are close to the performance when the model is applied in real business.

These procedures are iterative in nature. The modeling results and validation results have always been compared to the desired performance. If results cannot meet requirements, the modeling team may have to go back to either improve the model or choose another type of model. This process may have to repeat several times until the model is effective enough to meet requirements.

Step 4: Interpret and Implement Model

After a predictive model is created, the understanding of the model is an important step. This includes interpretation, business insights and communication. It is dangerous to apply a model without fully understanding it. We can open up the model and interpret the relationship between variables, and check it against business intuition. This could be viewed as another layer of model validation. In addition, we may discover new relationships from the model that are subtle or hard to grasp by human intuition but could be found by statistical algorithm.

Communication is an intrinsic part of modeling. Without a proper understanding of the model, no one will feel comfortable to use the model even if it is from an expert, which makes implementation impossible. Detailed documentation without technical jargon is needed for all stakeholders. Presentations are very helpful. All these will help to ease the difficulties in applying the model in business procedures.

Implementation may involve many departments in a company, depending on the type of applications. For a large predictive modeling initiative, it may change the business routine of many divisions such as product development/pricing, underwriting, administration, IT, etc. It is collaborative among all parties, and the modeling team needs to work closely with experts from different fields and find an optimal way to deploy the model.

There might be other obstacles in implementation—for example, to understand and adjust for any regulatory or company constraints related to the model or the variables used

CONTINUED ON **PAGE 30**

MODELING PROCESS … | FROM PAGE 29

in the model during the model implementation as well as development phase. The considerations may include data limitations, company cultural barriers, privacy concerns, IT constraints, issues of indirect discrimination, and the company's reputational risk. Just because it is legal to use certain variables in the model does not necessarily mean it is ethical.

Step 5: Monitor Results and Update

As part of risk management, after a model is implemented, monitoring the performance is not just necessary, but required. The evaluation system may include early detection of possible error, unintended consequences of the model, anti-selection, or an impact on other product lines. A good quality assurance and feedback loop is also necessary to make sure the model is accomplishing the original objectives.

If new experience data are available, models must also be periodically updated or refined in order to stay current. The frequency of model updates will depend on the type of model and data being used.

These steps are general procedures that a good predictive modeling team needs to follow. For a specific project, certain steps may be more important than others. For example, for an experience study, development of a model may take more time than all other steps, as objectives and data are clearly defined—especially if a traditional experience study has already been in place. If it is an underwriting model, each of above steps is as important as others. Ignoring any step may lead to a failed project.

AN EXAMPLE

The applications of predictive modeling in insurance could cover nearly every aspect of the business, from product development/pricing, experience study, underwriting, to sales and marketing, administration, claim management. It will be helpful to have an example for readers to understand.

Here is the application of predictive modeling to an experience study on lapse rates in the level period of a life term product. As explained above, the goal is obvious, i.e., to use a statistical method to study lapse rates. Since it is a multivariate approach, bias from univariate in a traditional study could be avoided. In addition, some interaction terms could be included to address correlations between variables. The usage of data is much more efficient, and issues like low credible data can be handled

If a traditional experience study is already in place, data are readily available as the same data could be used for the predictive modeling. If not, the data understanding and cleaning are the same as a traditional study, and actuaries are experts in dealing with these issues, which are quite universal for actuarial study, not specific to statistical modeling.

A major task in the experience study is data modeling where statistical skills are heavily emphasized. From choice of model to variable selection, statistical training as well as experience is crucial. For this lapse rate study, a transparent model such as GLM is desirable as the model results can be compared to conventional wisdom. Naturally, a Poisson distribution with logarithm as the link function is used to model the lapse data. The variables available for modeling are limited to about a dozen variables. Statistical tools can be utilized to select which one would be in the model to explain lapse rate variance. However, in the process, business knowledge is equally important. This is where art and science come to play together. Statistical techniques alone may lead to a perfect model, but may have no value for business or have no connection to reality. The modeling process is iterative, and there might be back and forth discussion between the modeling team and other stakeholders, such as pricing, product development and valuation.

At the end, the lapse rate model will be in the following format:

$$R_{lapse} = R_{base} \times \prod_{i} F_i^S \times \prod_{i,j} F_{i,j}^C$$

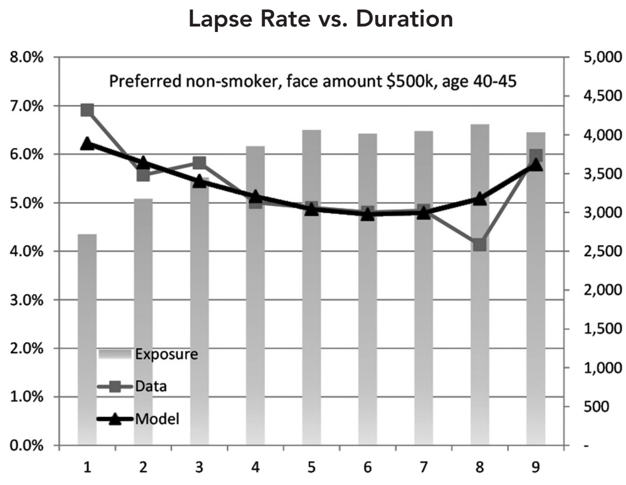
where R_{lapse} is the formula-based lapse rate predicted by model, R_{base} is the base lapse rate, F_i^S is factor for variable *i*, and $F_{i,j}^C$ is the interaction term for variable *i* and *j* if it is needed. The major variables that are included in the model include duration, underwriting class, issue age and face amount. And a few interaction terms between these variables are also incorporated in the model. The final format of the model is consistent with current actuarial practice with multiplicative formulas.

After the model is finished, it can be evaluated for its performance in addition to statistical assessment. One useful comparison is the lapse rates predicted by model vs. observed values. For example, we can test each cell to see how good the fitness is by plotting the predicted vs. observed values. An example is shown in the above plot. In addition, we may gain business insights from the model. The match of the model to business experience could also boost confidence of stakeholders in applying the model.

When the model is finally utilized in business, it is not the end of the project. Instead, monitoring the performance of the model in real life is an ongoing effort to ensure the model is as effective as the model in development. When new data is available or new experience is accessible, update of the model is required, which could lead to a new round of modeling efforts.

CONCLUSION

Statistical modeling is potentially a double-edged sword. If applied correctly, it is a very powerful and effective tool to discover knowledge in data, but in the wrong hands it can also be misused and generate absurd results. Here is a hypothetical example. One day, you are surprised to hear on the news that "visiting a gas station more than twice a week leads to 18 times higher mortality," which is based on statistical modeling on 47,000 data points. It is understandable that mortality will be higher if you drive more miles, but the mortality is out of proportion to the possible mileage. It may be hard to argue with this "modeler" as he has a strong statistical model and sizable data to support his re-



sults. However, there might have been a fundamental flaw in the process of modeling and model interpretation. Instead of the number of trips to a gas station, other factors may be much more important that lead to elevated mortality. For example, these who frequently visit gas stations may be buying cigarettes instead of filling gasoline, or they may live in low-income neighborhoods, relying on the gas station for daily needs and have limited access to the health system. Correlation does not mean causation, and statistical models can only be powerful when they are properly constructed and interpreted.

Recently, there was quite a lot of media exposure about causation of crime in large cities by lead, such as in Forbes and *The Washington Post*. The theory states that popular use of lead in gasoline and paint in the 1950s and 1960s caused the high crime rates in large cities in the 1970s and 1980s. Although it has been scientifically proven that lead has a severe impact on childhood development by both experience and historical data, there is no study to show the impact on crime. The correlation does exist in historical data, but the high crime rates in the 1970s and 1980s may also be ex-

CONTINUED ON **PAGE 32**

MODELING PROCESS … | FROM PAGE 31

plained by other factors, such as baby boomers in their 20s and 30s during that period, or the after-effects of the Vietnam War. There are thousands of social and economic metrics, and thousands of correlations with crime. You have a very good chance to find a very high correlation with crime rate that is purely accidental. I am not here to say the observation is wrong (which actually could lead to a scientific study of the causation between lead and crime), but to point out a possible misuse of modeling that relies on an accidental correlation.

Statistical modeling has broad applications in actuarial science as well as overall insurance business. To build an effective model, solid understanding of statistical knowledge is essential, but a good sense of business and enough data with high quality are also critical. Predictive modeling is about statistics, but more than that, it is about data and business. ▼

Richard Xu

Richard Xu, FSA, Ph.D., is senior data scientist and actuary at RGA Reinsurance Company in Chesterfield, Mo. He can be reached at *rxu@rgare.com.*

"Land This Plane"—A Delphi Study about Long-Term Care in the United States

By Ben Wolzenski

ABSTRACT

Many Americans will need long term care (LTC) in future years, yet only 10 percent of those 50 and over have LTC insurance (LTCI), and public programs are not funded to provide care for all who need it. The Long Term Care Section and the Forecasting and Futurism Section have co-sponsored a Delphi study,[1] code named "Land This Plane," with a lofty objective: to create a vision for how America ought to deal with the impending LTC crisis. This article describes the results of the study that were available at the time this article was written.

BACKGROUND

On Jan. 2, 2013, the "fiscal cliff" legislation formally repealed the Community Living Assistance Services and Supports (CLASS) Act and established a federal Commission on Long Term Care. The Sept. 18, 2013 pre-publication edition of that commission's report states the crisis. "A dramatic projected increase in the need for LTSS [long-term services and support] in the coming decades will confront significant constraints in the resources available to provide LTSS." Increasing numbers of elderly Americans who need care, combined with fewer caregivers and lower personal savings rates, will place even greater pressure on Medicaid and already stressed state and federal budgets.

On Jan. 4, 2013, members of the Long Term Care Think Tank invited Forecasting and Futurism section council members to join them in a discussion of a potential Delphi study.[2] The objective was no less than producing a consensus about how America should deal with the pending LTC crisis with a comprehensive, integrated solution. What would be the number and makeup of panelists, what would the questions cover, how would the logistics be handled, and could we move fast enough to provide input to the federal commission? A diverse panel of 50 experts was assembled: insurance executives and marketers; regulators and public policy advocates; and, of course, actuaries.

The questions were formulated, debated, finalized and sent to the panelists on Feb. 1 with a reply requested by Feb. 18. Replies were compiled, analyzed and discussed at the LTC Think Tank meeting in Dallas on March 3. The responses covered over 100 pages of text, and the work team concluded that the best way to conduct a second round was to consolidate the first round input into six major principles, under which specific questions were posed. The second round went out on May 15, with replies due in early June. The third round had a similar format and mostly the same questions, and was primarily aimed at giving panelists a chance to review their co-panelists' replies and give their final answers. It went out on Aug. 14 with an extended deadline for reply of Sept. 20. A report was presented at the Society of Actuaries Annual Meeting on Oct. 22. Along the way, however, interim results were shared with the Commission on Long Term Care, and it appears that some (but not all) of our conclusions found their way into the commission's report, although the commission may have reached the same conclusions independently.

Here, then, are the six principles drawn from responses to the study, the nearly complete tabulation of the extent of panelists' agreement with each principle, and some of the specific concepts underneath each principle. The full report of the study should be available online on the SOA website near the end of the first quarter of 2014.

PRINCIPLE 1: A ROBUST AND EFFICIENT LTC SYSTEM

All aspects of the LTC financing system need to incentivize family and household participation, responsible planning and behavior, and the most efficient use of LTC resources. An all-encompassing system should include incentives to plan for the future, purchase appropriate products, use appropriate care settings, and adopt healthy lifestyles to mitigate the need for LTC services.

Need a robust and efficient LTC system	**88 percent agreed**
Private insurance should be part of solution	**100 percent agreed**
System should incent:	
Responsible LTC planning	**100 percent agreed**

CONTINUED ON **PAGE 34**

"LAND THIS PLANE"… | FROM PAGE 33

Healthy lifestyles	**75 percent agreed**
Household and family participation	**84 percent agreed**

PRINCIPLE 2: SOCIAL INSURANCE

There is a need for the government to take an active role establishing or encouraging a limited LTC social insurance program to help finance care for people who can't purchase private LTCI due to either cost or underwriting issues. It will be open to all, but designed to meet the specific needs of the "middle class." It would be part of a public-private combination approach to LTC financing but not the single standalone solution**.**

Social insurance is a necessary part of the solution	**88 percent agreed**

PRINCIPLE 3: CHANGES TO MEDICAID

Medicaid needs to be changed to tighten eligibility by closing loopholes, strengthening eligibility requirements, and enforcing the rules strictly. At the same time, it also needs to be modernized to enable care on a national basis in the full range of settings. This includes home- and community-based care if appropriate and cost effective.

Need Medicaid reform—tighten eligibility	**79 percent agreed**
Need modernization—home- and community-based care	**83 percent agreed**

PRINCIPLE 4: CHANGES TO REGULATIONS AND LEGISLATION

In order to successfully promote the availability of LTCI in a robust and competitive market, regulations and legislation including the NAIC Model Act need to be substantially modified to take account of a new business paradigm for LTCI. The new LTCI business paradigm will entail re-engineering the overall product so that carriers will be able to balance acceptable risk levels with the need to offer meaningful consumer benefits at affordable premiums. The Model Act and other federal and state regulation and legislative revisions will need to take account of these new business realities while maintaining appropriate consumer protections.

Allow LTCI products with shorter benefit periods	**61 percent agreed**

Allow adult day care as option vs. required **68 percent agreed**

Agree with term plus side fund concept **45 percent agreed**

Principle 5: An Active Government Role

The government must take an active role developing and implementing the LTC financing solution. Federal and state governments should actively "promote the general welfare" for the benefit of their citizenry as well as their own fiscal health. They should do this by educating and influencing people to promote responsible planning and healthy behaviors related to their future LTC needs.

Need an active government role	**95 percent agreed**
Need government-sponsored public awareness	**92 percent agreed**
Less restrictive Partnership regulations	**85 percent agreed**
Tax incentives for LTC protection	**75 percent agreed**
Modify rules on tax-deferred savings (401(k), etc.)	**71 percent agreed**
National reinsurance plan	**59 percent agreed**

PRINCIPLE 6: IMPROVED MARKETING AND SALES

The way LTCI is marketed and sold needs to be improved by "mainstreaming the message" that LTC represents a significant and largely unplanned-for financial risk that needs to be addressed by consumers.

Improve LTCI training	**83 percent agreed**
LTCI knowledge should be core to CE designations	**75 percent agreed**

I have now participated on the work team of two completed research studies sponsored by the SOA using the Delphi technique, and I have studied three other SOA-sponsored Delphi studies. I believe that this Delphi study is a new high-water mark in the quality of the Delphi panel and in the potential impact of an SOA-sponsored Delphi study, and I look forward to their future use by the Forecasting and Futurism Section in collaboration with other SOA sections. ▼

END NOTES

1 For background on the Delphi technique, see "The Delphi Method" by Scott McInturff in the September 2009 issue of the Forecasting and Futurism Newsletter, available at http://www.soa.org/library/newsletters/forecasting-futurism/2009/september/ffn-2009-iss1-mcinturff.aspx.

2 The project team included Roger Loomis, Ron Hagelman, John O'Leary, Jason Bushey, John Cutler, Amy Pahl and Steve Schoonveld from the LTC Think Tank; Brian Grossmiller, Clark Ramsey and Ben Wolzenski from the Forecasting and Futurism Section Council; and Steve Siegel of the Society of Actuaries staff.

Ben Wolzenski

Ben Wolzenski, FSA, MAAA, is managing member at Actuarial Innovations, LLC in St. Louis, Mo. He can be reached at *bwolzenski@rgare.com.*

Delphi Study in Real Time—Life and Annuity Products and Product Development

By Paula Hodges

This article is reprinted with permission. It is an excerpt of an article that first appeared in the Oct. 2013 issue of *Product Matters!*

- Moderator and presenter: Paula Hodges (Ameritas)
- Presenter: Albert Abalo (Oliver Wyman)
- Presenter: Ben Wolzenski (Actuarial Innovations)

Session 86 at the 2013 Life & Annuity Symposium utilized the Delphi method to develop several predictions about developments in the life and annuity market over the next seven years. For those not familiar with the Delphi method, it is a process whereby a facilitator collects information from a group of experts on a particular subject matter. After collecting a first round of opinions, the facilitators share the aggregated results with the group. At that time, the group continues to participate anonymously, but with the benefit of the opinions, and sometimes commentary, from the other experts. Another round of polling takes place, and this continues until the results are stabilized.

This method has proven to be very predictive. In this session, the audience was utilized as the experts, and here are a few of the predictions made:

By the year 2020, U.S. and Canadian bond yields will be between 3 percent and 5 percent, but will remain relatively unchanged for the next three to five years. As this will challenge the spreads that insurance companies require, the burden will be passed along to consumers (higher prices), agents (lower commission), employees (lower wages and layoffs), and the company itself (lower profits). The group felt that a reasonable internal rate of return (IRR) expectation in this environment is less than 10 percent.

Paula Hodges

Paula Hodges, FSA, is second vice president, associate actuary for Ameritas Life Insurance Corp. She can be contacted at phodges@ameritas.com.

Life insurance products that will take off in the next few years are expected to be whole life and indexed universal life, while the indexed annuities will see the largest amount of growth in the annuity space.

With the aging of the current field force, alternative avenues will be sought by both consumers and insurance carriers. Therefore, marketing of life and annuity products is expected to shift to financial advisors for annuity sales and to the Internet for life products.

Not surprisingly, the biggest issues facing insurers over the next seven years are expected to be the low interest rate environment and the shifting demographics, impacting both the distribution force and the insured population.

This was a very interesting session, showing how additional information and the anonymity of the experts influenced changes in the ultimate consensus of the group. I look forward to the year 2020 when we can validate the opinions of the experts that were in the room for this enjoyable session. ▼

Genetic Algorithms Revisited—A Simplification and a Free Tool for Excel Users

By Dave Snell

One of the cool aspects of teaching is that over time I start to better understand the subject I am teaching. In order to clarify a technical concept for someone else, I often find myself background processing for days or months and suddenly seeing the more obvious points—the ones often obscured by the technical details when I am learning the topic.

Based on a few years of feedback now from giving presentations to many groups, writing articles in various publications and coding programs for diverse projects, I have come to the conclusion that a genetic algorithm is a very simple concept shrouded in too many intimidating biology terms. In this article I want to share my simplified view of what a genetic algorithm is, where you might use it, how you can build your own, and how you can use the one I built for you to solve your own problems.

Basically, a genetic algorithm is a set of simple rules to solve certain types of otherwise difficult problems. In order to apply a genetic algorithm, you need a problem that lends itself to this type of solution.

Criteria that make a problem suitable for a genetic algorithm:

1. The problem involves a lot of variables—to some extent, the more variables there are, the better this technique applies.
2. Each variable can take on potential values to produce different solutions.
3. We can substitute a value for each of the variables, and that particular combination of individual values can be thought of as a solution set.
4. The problem can be quantified in some manner so that any two solution sets can easily be compared to see which is better.

In the language of our children: OMG. Can it be that simple? What about DNA, genes, chromosomes, alleles, phenotypes, mitosis and meiosis, single nucleotide polymorphisms, etc.? Didn't this all start as an attempt to mimic the amazing role that genetics plays in natural selection (unfortunately dubbed evolution)? Yes, it did.

Humans fit this set of criteria quite well:

1. The blueprint for our cells is a long chain of pairs—over 3 billion pairs in a chain.
2. Each pair can be one of four values: A-T, T-A, C-G or G-C. In order to keep this simple, I am not even going to say what the letters mean. It does not matter for this discussion.
3. Every cell contains a chain made according to these pairs. Each person has a slightly different set of links (pairs) in their personal chain.
4. The unique combinations for two different chains result in two different persons; and you can compare them to see which one is taller, thinner, smarter or whatever, to infer which chain best met your goals.

That's the end of the biology lesson!

Let's consider some applications that you might find more relevant to actuarial work:

1. You have to choose which provider groups to include in a health insurance network. There are over 3,000 provider groups in your region and each group may offer from one to 100 specialty services. Each specialty (acupuncture, cardiology, oncology, etc.) has a relative cost, and each provider group has a relative cost (specialties, location, experience, etc.). Your challenge is to pick the combination of provider groups that minimizes cost while maintaining adequate coverage for each area

CONTINUED ON **PAGE 38**

GENETIC ALGORITHMS REVISITED … | FROM PAGE 37

of specialty (at least five otorhinolaryngologists, at least 50 pediatricians, etc.). Since each provider group will be "in" or "out" of the network, each solution set is 3,000 values (either 0 or 1) long and your potential number of solution sets is 2^{3000}—a very large number.

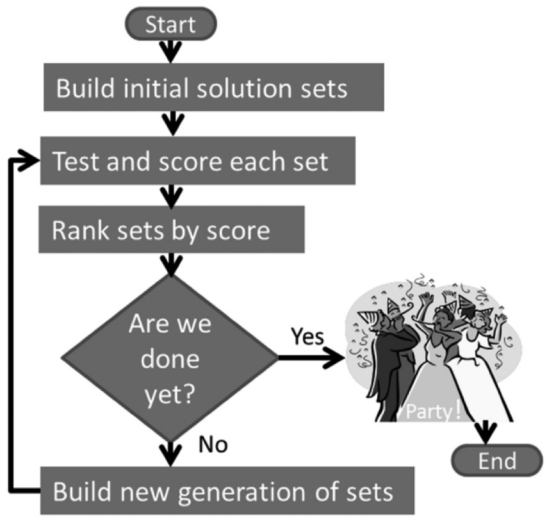
2. You have seven sales regions and 15 sales representatives. You wish to allocate them in a manner that adequately covers each region while respecting, to the extent practicable, the preference of each salesperson. Each solution set is 15 variables long, and each variable can be any of seven values so the number of solution sets possible is 7^{15} (more than 4.7 trillion).

3. You have a set of 50 equations with 100 unknowns, and the equations are not linear. Each unknown variable is a real number from the range -15 to +500. The number of potential solution sets is infinite. You want to minimize the sum of the output from each equation.

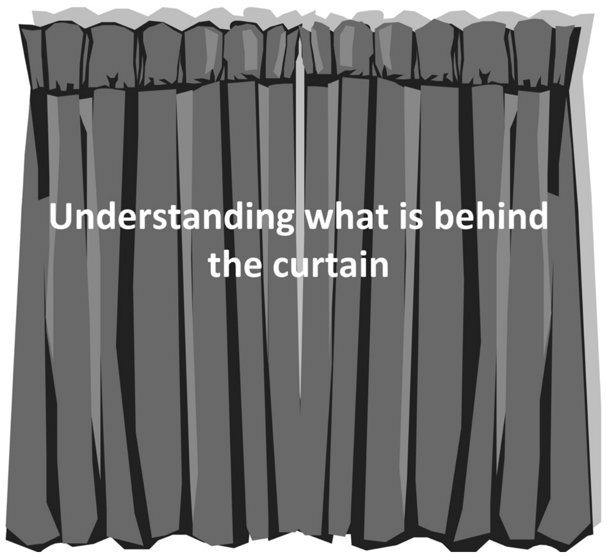
4. Your CEO asks you to do an enterprise risk management (ERM) analysis of a portfolio of business. She knows that a standard CTE (conditional tail expectation) with 10,000 stochastic runs will not surface the combination of enough tail events occurring together, yet she also knows that in the real world a tail event on one variable may trigger a domino effect where other tails are hit. Your task is to do a true worst-case type of scenario within the ranges of the several dozen key parameters.

How do we approach these problems? They all seem different; yet, they all seem to satisfy my stated criteria of applicability for a genetic algorithm.

First, I'll show you the answer:



If you do wish to learn what happens behind the curtain, please continue. Let's break our solution logic into some simple steps:



1. **Populate a collection of possible solution sets.** In previous talks and articles I have used the popular genetic algorithm term for this kind of collection as a generation. Whatever you wish to call it, the first collection of solution sets must be populated. This is one part of the process that used to prompt questions from my students. Unless you have special knowledge about the answers, it is customary to assign values to the variables of this first generation on a random basis. Most programming languages and spreadsheets offer a Random function.[1] Apply this appropriately to each variable position in your solution sets. How many solution sets should you use? Good question! If your solution set chains are very long, you may be able to fit only a smaller number of sets (perhaps 100) in memory at one time. If your chains are much shorter, you might wish to test 1,000 or more in one collection (generation). In the workbook code you can see how to do this in `PopulateInitialGeneration`.

2. **Test each set of the collection and save the scores obtained.** Whatever the nature of the problem, you need to decide how to judge the worth of your answer for a given solution set. That might be as simple as arranging your equations to end up with a single answer. Decide whether you want this answer to be minimized (for example, a cost) or maximized (a benefit). The workbook code for this is `TestSets.`

3. **Rank the scores.** Reorder the solution sets of the collection (generation) from best to worst. You can see this code in `RankTheScores`.

4. **Build the successive collection (generation) of solution sets.** This is the step that confuses the most people. If you just populate the next generation with random values, how is the method any different from a trial-and-error approach? The answer is that it is not, so we don't do it strictly randomly. Instead, we take advantage of the information from our previous collection of solution sets and results. You don't want the new collection of solution sets to ever be worse than the previous one. A way to guarantee that is to bring the top scoring sets over intact to the next collection. If you had 100 sets in your collection, you could bring over as few as one (the top scoring one) or as many as 100. There is not much point in bringing over 100 since that would really limit the improvement potential for the new collection.

If we bring over 20 solution sets from the previous generation, that means we need to create 80 more solution sets to get back up to 100 for this new generation. This is where we mimic, to some extent, biology; but again, do not let the terms of biology confuse you. We are going to build the new collection (generation) of solution sets by choosing values from the previous generation—notably the best scoring members of that generation. We might choose to pick from only the top five (those five sets with the best scores), the top 20, or even the top 50 as potential "parents" of the new generation. Let's say that each solution set chain is going to be 100 variables long. The source for variable #1 (the front of the chain) could be the variable #1 value from any solution set in our chosen group of parents from the previous generation. Often, it is most efficient to just randomly choose one of them. Likewise, the source for variable #2 in the new solution set could be variable #2 from any solution set in the parent pool. Again, just pick one at random.

"Wait a minute! Are you suggesting that a single solution set could be made from several different parent sources? Among humans, that is not allowable." You are correct! Among humans, it is not. But we were using biology only as a metaphor, so why limit ourselves to two parents per child (or two source solution sets per new solution set)?

Let's see how we could code that in Visual Basic (similar idea in most other languages):

CONTINUED ON **PAGE 40**

GENETIC ALGORITHMS REVISITED … | FROM PAGE 39

```
  Private Sub AddTheChildren()
  Dim parent As Integer, var As Long, child As Integer, children As Integer
1   children = setsPerGeneration - elites
2     For child = 1 To children
3       For var = 1 To setLength
4         parent = Int(parentPool * Rnd()) + 1
5         solutionSets(var, elites + child) = solutionSets(var, parent)
6       Next var
7     Next child
  End Sub 'AddTheChildren
```

	Detailed explanations for specific code lines
	`solutionSets(x,y)` = the value of the xth position variable in the chain for solution set y
1	For our example, we want 100 solution sets per collection (`setsPerGeneration`). We have decided to carry over intact the 20 top scoring sets (`elites`) so we need to generate 80 child (next generation aka next collection) solution sets (`children`).
2	`child` is the specific solution set you are creating
3	`var` is the variable number in the solution set of variables. `setLength` is the total number of variables in a solution set.
4	`Parent` is the specific source solution set from the previous generation. `Rnd` is a random number generator that returns a number from 0 to 1. `parentPool` is your choice for the number of parent sets. If you set `parentPool` to 50 then any of the top-scoring 50 sets could be a potential source for a `child` solution set.
5	Copy the value for variable number `var` from solution set `parent` to the new solution set `elites + child`. Again, from our example, if `child` = 1 and `parent` = 5, then copy the value from `var` 17 of old set 5 to `var` 17 of new set 21 (20+1).
6	Continue until all variables in new solution set `child` have been assigned a value.
7	Continue until all the new solution sets have been created and populated.

This gives you a new generation of solution sets. However, we can still improve our results further by randomly changing some of the child solution set values. This is called mutation. My subroutine `AddMutations` shows how to accomplish this.

Once you have a new collection (generation) of solution sets, go back and repeat steps 2, 3 and 4. Eventually, either your top-scoring solution set of step 3 is going to satisfy your goal; or it will start repeating itself. If it repeats itself for too many cycles, then it probably won't get any better so you might as well stop. If that happens before you are satisfied with the answer, then change some of the parameters: the number of solution sets in each generation, the number of elites (immortal sets), the number of parents allowed as sources, the number of generations requested, or the number of mutations allowed per set. Then, rerun the tool, choosing "from previous run" so you can continue to improve. The code `TestForCompletion` checks for a suitable end condition.

FREE TOOL

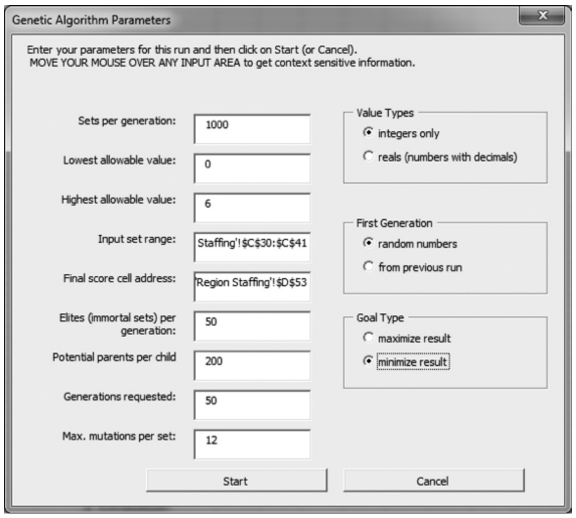
Sometimes, you don't really care how the car engine works; but you need to be able to drive it. Here is the quick way to accomplish that.

Start by downloading my general purpose genetic algorithm tool for Excel workbook from *http://www.soa.org/news-and-publications/newsletters/forecasting-futurism/default.aspx.* Save it to some folder on your PC, bring it up, and enable macros.

I won't go into a lot of detail here because the workbook has instructions built into it. The workbook also contains a couple of sample problems that you can solve to get a quick feel for how to structure your own workbook. This workbook and this article are a response to requests about how a person might adapt my earlier code to their workbook problems, I wrote this generalized genetic algorithm routine for you to be able to use it without having to learn to program. Alternatively, you can easily modify the program to extend the built-in features.

All you have to do is arrange your spreadsheets in any way that works from a given solution set (arranged in a column) and that assigns a score in some cell. On the parameters screen of the tool (see figure below), you will fill these into the "Input set range" and the "Final score cell address" (note that if you move your mouse over any input item, the program will show you context-sensitive help for that item). Then, fill in your choices for how many generations to run, how many sets per generation, how many mutations are allowed per new set, etc., and you can run your own genetic algorithm solutions.

Genetic algorithms provide you with a powerful tool for many types of problems that are very difficult to solve by other means. Recently, I discovered that Microsoft has added an "evolutionary" solution method to its excellent Excel Solver add-in. I almost stopped coding my add-in when I saw that; but then discovered that it is limited to at most 200 variables in the solution set, and it's still a black box that you cannot modify. Now that you know how to fish evolve (or, switching metaphors, now that you know what goes on behind the curtain), you are not limited to what they or I have built for you. Enjoy the free tool; and then enjoy the power of your new skill set! ▼


Genetic Algorithm Parameters

Enter your parameters for this run and then click on Start (or Cancel).
MOVE YOUR MOUSE OVER ANY INPUT AREA to get context sensitive information.

Sets per generation:	1000
Lowest allowable value:	0
Highest allowable value:	6
Input set range:	Staffing'!$C$30:$C$41
Final score cell address:	'Region Staffing'!$D$53
Elites (immortal sets) per generation:	50
Potential parents per child	200
Generations requested:	50
Max. mutations per set:	12

Value Types: [✓ integers only] reals (numbers with decimals)

First Generation: [✓ random numbers] from previous run

Goal Type: maximize result [✓ minimize result]

ENDNOTES

1 I describe this example (from Brian Grossmiller) in detail in my article "Genetic Algorithms—Useful, Fun and Easy" in the December 2012 issue of *Forecasting & Futurism Newsletter.*

2 Brian Grossmiller and I discussed this problem in our workshop at the 2013 SOA Annual Meeting.

3 I am purposely assuming here that the random functions are good ones. In most applications, that is not the case; but a discussion of how you generate randomness is beyond the scope of this article. By applying it appropriately, I mean to restrict your outcomes to the range of values (either real, or integers) acceptable for that variable.

Dave Snell

Dave Snell, ASA, MAAA, is technology evangelist at RGA Reinsurance Company in Chesterfield, Mo. He can be reached at *dsnell@rgare.com.*

And the Winner Is …

By Alberto Abalo and Doug Norris

The winner of this year's Forecasting and Futurism iPad contest is Jeff Heaton.

When we decided that this year's challenge would be to build a genetic algorithm from scratch, we understood we were asking for a lot. We were asking our members to make a time commitment that previous contests had not required, and we were taking a gamble on the interest in genetic algorithms in general. While the number of entries was understandably limited, it goes without saying that we were delighted by the quality of those we did receive. In all cases, entries were thoughtful, clearly took effort to put together, and served our ultimate goal of adding to the intellectual capital of our section.

Jeff's entry, "Diagnosing Breast Tumor Malignancy with a Genetic Algorithm and RBF Network," is reprinted here in its entirety. We encourage all readers to download the algorithm itself from our website (*www.soa.org/forecasting-futurism*) and follow the steps laid out in the article as they read along. We hope you'll agree that this application of genetic algorithms is both practical and relevant to the profession, and we look forward to hearing your thoughts on how it could be extended to solve other actuarial problems. Congratulations, Jeff!

P.S.: We felt the following entries in particular were worthy of special mention, and hope to publish summaries in future editions of the newsletter:

- Mark Bergstrom's capital optimization algorithm
- Richard Xu and Yuanjin Liu's fund mapping analysis algorithm
- Dave Snell and Brian Grossmiller's staffing algorithm. ▼

Alberto Abalo

Alberto Abalo, FSA, CERA, MAAA, is a principal at Oliver Wyman in Atlanta, Ga. He can be reached at alberto.abalo@oliverwyman.com.

Doug Norris

Doug Norris, FSA, MAAA, Ph.D., is a consulting actuary at Milliman Inc. in Denver, Colo. He can be reached at doug.norris@milliman.com.

Diagnosing Breast Tumor Malignancy with a Genetic Algorithm and RBF Network

By Jeff Heaton

In this article I demonstrate how to use a genetic algorithm to devise a radial basis function (RBF) network to predict the malignancy of breast tumors. The genetic algorithm is implemented in Microsoft Visual C#. This determination is made using the following nine attributes collected from a tumor.

- Clump thickness
- Uniformity of cell size
- Uniformity of cell shape
- Marginal adhesion
- Single epithelial cell size
- Bare nuclei
- Bland chromatin
- Normal nucleoli
- Mitoses.

I used training obtained by Dr. William H. Wolberg, University of Wisconsin Hospitals.[1] Using these attributes I constructed an RBF network to perform malignancy classification into two classes. These two classes were either malignant or benign.

This program is meant to be very versatile and extendable. This same program can be used to perform either classification or regression on a wide array of data sets. If you wish to use an RBF network, only changes to the script.xml file are necessary. If you would like to create your own model, you can easily add a model class to the C# source code. I tested this program on several data sets from the University of California at Irvine Machine Learning Dataset Repository.[2]

USING THE PROGRAM

This program can be downloaded from the SOA Forecasting & Futurism site at the following link: http://www.soa.org/news-and-publications/newsletters/forecasting-futurism/default.aspx.

I included both the source and compiled forms of this program. Additionally, I included the data file for the breast cancer research. This program operates on Excel .XLSX files. To take the program through its paces, use the following steps. The following files are included with the submission.

Script File: script.xml
Training data: breast-cancer-.xlsx
Evaluation data: breast-cancer--eval.xlsx
My Sample Output: output.xlsx
My Sample Population: population.ser

The included script file will automatically use the above filenames. The training data and evaluation data both contain non-overlapping samples from the original data. This allows you to evaluate with non-trained data. The training data has considerably more rows than the evaluation. It included two sample output files. However, you will overwrite them once you complete the steps below.

Use the following steps to run the .

1. Launch the GeneticAlgorithmUtil.exe .
2. Fill in the location of the script.xml file that you would like to use. If you are running from the folder I sent, it will most likely find the included script.xml file.
3. Click **Load** to load the script.
4. Click **Generate** to generate and score an initial random population.
5. Click **Train** to train the population. The program will show the number of child genomes created and the current best score. Stop training when the score goes to near 0.01. This can take a few minutes, depending on your computer's speed. It usually takes me around 70k genomes to get to 0.01.
6. At this point you have a trained population; you can **save** it if you wish.

CONTINUED ON **PAGE 44**

7. Now let's evaluate it on data not part of the training set. Click **Evaluate**.

8. A file named Output.xlsx will be generated. You can see the predictions from the program. It should be very accurate.

When you run the "Evaluate" option you will see the program attempt to classify data that it was not trained on. It is always important to hold back some of your training data for evaluation. This ensures that the model has actually learned and not simply memorized (over fit) the training data.

RADIAL BASIS FUNCTION NETWORK DESCRIPTION

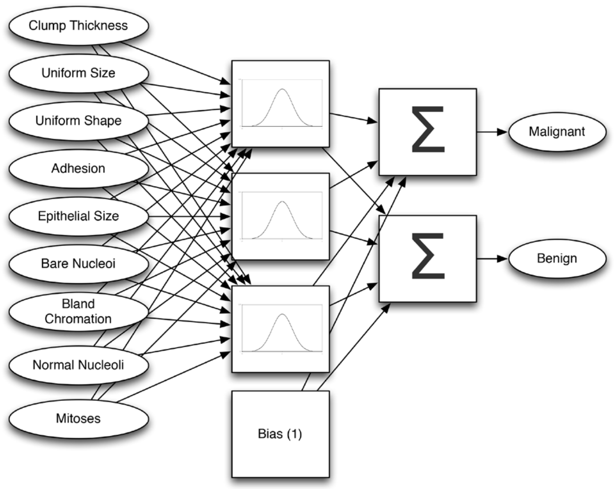
An RBF network is a statistical model that can be used for both classification and regression. It provides a weighted summation of one or more RBFs, each of which receives the weighted input attributes used to predict. The following equation summarizes an RBF network.

$$f(X) = \sum_{i=1}^{N} a_i p(\| b_i X - c_i \|)$$

Where **X** is the input vector of attributes, **c** is the vector center of the RBF, **p** is the chosen RBF (i.e., Gaussian), **a** is the vector coefficient (or weight) for each RBF, and **b** species the vector coefficient to weight the input attributes.

RBF NETWORKS CAN BE ADAPTED TO BOTH CLASSIFICATION AND REGRESSION PROBLEMS.

Graphically this can be viewed as follows.



Arrows represent all coefficients from the equation. The arrows between the input attributes and RBFs are represented by **b**. Similarly, the arrows between the RBF's and the summation are represented by **a**. You will also note that there is a bias box. This is a synthetic function that always returns 1. Because the bias function's output is constant, no inputs are required to it. The weights from the bias to the summation function vary similarly to the vector intercept in linear regression.

Because there are multiple summations, you can see that this is a classification problem. The highest summation specifies the predicted class. If this were a regression problem, there would be single output. This single output would represent the predicted output for regression.

GENETIC ALGORITHM DESCRIPTION

A genetic algorithm typically evolves a population of potential solutions to a problem. Each potential solution is typically called a genome or chromosome. Each potential solution is represented as a "DNA strand. This DNA strand is an array of numbers. To evolve an RBF, I treat each RBF as an array of numbers. These arrays contain the following values:

- Input coefficients
- Output/summation coefficients
- RBF width scalars (same width in all dimensions)
- RBF center vectors.

RBF networks are typically trained either using gradient descent or matrix transformations. However, such approaches only adjust the coefficients. This leaves the RBF parameters to manual selection.

Using a genetic algorithm allows me to evolve all parameters to the RBF network. The only aspect that I am not evolving is the number of RBF functions to use. This must be defined in the script.xml file. This is because most genetic algorithms use a fixed DNA size. This is biologically plausible because mating only occurs through genetically similar genomes (i.e., those of the same species).

Genetic algorithms require a score function to determine the superior genomes. The scoring function for this program is very simple. The training data is used to see how accurately a given genome (RBF network) can predict a given malignant or benign status.

This genetic algorithm employs many advanced techniques from current research.

- **Fully multithreaded**. This program makes use of the C# Parallel class to ensure that all available processors and cores are fully utilized. This results in very fast training on modern core CPUs.

- **Non-epoch based**. Unlike many genetic algorithms, the population is not rebuilt each epoch. Rather, parents are chosen from the population and resulting children replace weaker genomes. This is very efficient for threading. It is also biologically plausible. We do not have clear-cut lines (epochs) of when each generation begins and ends in the real world.

- **Tournament selection**. When a parent must be chosen, for mutation or crossover, we choose a random member of the population and then try 10 more random members to get a more suited potential parent. When unfit genomes must be removed, this process is run in reverse. This is very efficient for threading, and is also biologically plausible. You don't have to outrun the fastest tiger on earth, just the tigers you randomly encounter on a given day! This also removes the need for a common genetic algorithm technique known as elitism.

- **More than two parent crossover**. Why not have more than two parents? A child can be created from several optimal parents. This is a very interesting technique that I first learned about from a *Forecasting & Futurism Newsletter* article (December 2012) by Dave Snell.[3]

EXTENDING THE PROGRAM

A genetic algorithm can be used to optimize DNA for more than just RBF networks. To facilitate other models simply create a C# class that implements the following methods via the **IGAModel** interface.

void Init(ConfigScript script, string config);

The Init method simply gives you access to the config.xml script, as well as an optimal configuration string passed to your program. For the RBF network model this is the number of RBF functions.

Genome GenerateRandomGenome(Random rnd, ConfigScript script);

The GenerateRandomGenome method is called to generate a new random genome. This is typically performed as part of the initial population generation.

double[] Compute(double[] input, Genome genome);

CONTINUED ON **PAGE 46**

The compute method computes the genome's output based on the given input.

double Score(Genome testSubject, double[][] training, double[][] ideal);

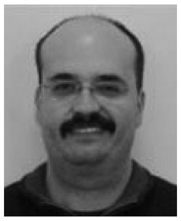
The score method determines how fit the genome is. Optional training data can be passed as well.

Genome Mutate(Random rnd, Genome genome);

Mutate the specified genome and return a child. The original genome is not changed.

Genome[] Crossover(Random rnd, Genome[] parents);

Perform a genetic cross over, based on the parents. The children are returned. ▼


Jeff Heaton

Jeff Heaton is EHR informatics scientist at RGA Reinsurance Company in Chesterfield**,** Mo. He can be reached at jheaton@rgare.com.

ENDNOTES

[1] Mangasarian, O.L., and W.H. Wolberg. 1990. Cancer Diagnosis via Linear Programming. SIAM News 23(5), September, pp. 1 & 18.

[2] Bache, K., and M. Lichman. 2013. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, Calif.: University of California, School of Information and Computer Science.

[3] Snell, D. 2012. Genetic Algorithms—Useful, Fun and Easy! *Forecasting & Futurism Newsletter.* December, pp. 7–15.

Are Spreadsheets Sabotaging Your Accuracy?

By Steve Epner

Many firms are addicted to using spreadsheets for many applications, including forecasting, trend analysis and other actuarial requirements. They are afraid to let go. Spreadsheets are ubiquitous, relatively easy to use, and can be very sophisticated, but they are often uncontrolled, poorly designed and inadequately maintained. There are real risks associated with the use of spreadsheets. By following a few practical suggestions spreadsheet use can be made safer.

THE EVOLUTION OF THE SPREADSHEET

In 1979, Dan Bricklin and Bob Frankston published the first modern PC-based electronic spreadsheet called VisiCalc. While previous row/column programs existed, the modern WYSIWYG ("what you see is what you get") interface in VisiCalc created a user-friendly, functionally rich solution that some credit with launching the PC business revolution.

Lotus 1-2-3 took over the spreadsheet lead in the early 1980s with additional innovation and functionality. The current spreadsheet leader, Microsoft Excel, was introduced in 1985 and was originally developed to support the Apple Macintosh. In 1987, Excel was introduced for the "IBM PC," and by the early 1990s Excel had surpassed Lotus 1-2-3 in both feature/function and sales.

Spreadsheets might have begun as basic row/column calculators, but they quickly matured into feature-rich software. They include sophisticated, built-in mathematical functions and programming language capabilities.

Power users began to use spreadsheets for larger scale business solutions; they were tackling tasks as diverse as financial statement analysis and actuarial projections. As the problems being addressed by spreadsheets grew, so did the spreadsheets themselves. It is not uncommon to find spreadsheets with thousands or even tens of thousands of cells. Today, the complexity of some spreadsheets rivals or even exceeds that of applications created in standard programming languages.

RISKY BUSINESS

Spreadsheets are the ultimate "end-user" business application. They are typically built by individuals or groups, and not information technology professionals. As such, it is unusual to find a spreadsheet that has been designed, developed and tested using the rigorous methods in use by professional software engineers.

Today, many mission-critical functions are being supported by spreadsheets that have been developed without formal methodologies. The business risk is not fully understood by most corporations. As the functions being supported by spreadsheets become more critical, so does the urgency to manage the development so that the decision makers can rely upon the results generated by those complex programs.

SPREADSHEET LIMITATIONS

The use of spreadsheets to perform complex business functions exposes a business to a number of risks and limitations.

Raymond Panko at the University of Hawaii proved that spreadsheet errors are common and result in meaningful, harmful impacts. His studies of spreadsheets in use by companies of many different sizes have found error rates from 24 percent to over 85 percent. Error levels of this magnitude can clearly have a measurable and non-trivial impact on decisions that are based on the inaccurate results produced by those spreadsheets.

He defined two categories of errors. Quantitative errors produce incorrect values elsewhere in the spreadsheet. Qualitative errors are flaws of design that may later cause errors through incorrect input or modifications that do not maintain the integrity of the original spreadsheet.

Steve Epner

Steve Epner is founder of the Brown Smith Wallace Consulting Group. He can be contacted at *sepner@bswc.com* or 314.983.1214.

ARE SPREADSHEETS SABOTAGING … | FROM PAGE 47

SPREADSHEETS MAY BE EVERYWHERE, BUT WE MUST LEARN TO CONTROL AND MANAGE THEM LIKE ANY OTHER TECHNOLOGY ASSET. ANYTHING LESS WILL INVITE PROBLEMS AND ERRORS IN OUR DATA AND DECISIONS.

Quantitative errors were categorized into three main types: mechanical (incorrectly keyed data, formulas, or pointing to an incorrect cell), omission (something important is left out) and logic (incorrect formulas due to errors in reasoning). In addition, there are life cycle errors that occur as spreadsheets are updated, modified and enhanced. These errors can be introduced long after the spreadsheet was designed and the original testing was completed.

The fact that these types of errors exist in spreadsheets is not surprising given the ad hoc nature in which most spreadsheets are created and maintained. Issues such as security, documentation, version control and validation are neglected or not even considered. Also, without a formal testing/feedback system, spreadsheet end users might not realize the extent to which output data is inaccurate.

DIFFICULT TO MAINTAIN

Even if the initial version of a spreadsheet is created successfully, it often will not remain that way through future revisions, iterations and/or enhancements.

Two important elements are the lack of documentation and the inevitable migration of employees to new job duties or even companies. Professional software developers budget for and invest significant time in the documentation of their systems. This is a necessary prerequisite to allow the system to be maintained and to allow ongoing support even if the original solution's authors are no longer available. Spreadsheet documentation is a rarity, and training replacements for developmental personnel is almost nonexistent.

Furthermore, spreadsheet programs are almost exclusively built based on the education, experience and expertise of a single user, department or firm. Compare this to more mature business applications, built over many years by professional software firms. Such applications include the "best practices" of hundreds or even thousands of end users. No in-house development can match that level of input.

Data control is the final area of concern for most organizations. Few spreadsheets identify the source information (when created, using what data, from what time period, and on what version of the spreadsheet) on all reports. In all cases, it must be possible to replicate the exact results, or the system will be suspect. And in most cases it is.

WHAT CAN WE DO?

The very first step is to determine if the spreadsheet is really needed. Many early sheets were created because the actuarial systems of that day were not adequate to meet the needs of decision makers. Now that has changed, review every spreadsheet with your software vendors and see which ones might be replaced using standard software that is properly tested and maintained.

If you must continue to use the spreadsheet, carefully test how well it works. Create a simple, but complete set of input data that will test all of the assumptions built into the spreadsheet. Then predict the expected results. Run the data through the spreadsheet and then reconcile the output. Any errors must be traced down and corrected.

This process may take a number of iterations. When a properly operating spreadsheet is developed, give it a version/release number and then "lock it down." Do not let anyone change anything without permission. Even when a change is necessary, until it is able to run the test data accurately (enhanced to account for any new functionality or changes to the spreadsheet), it cannot be used by others. Once approved, it receives the new version/release number and the old version is taken off the system.

Second, make sure that every spreadsheet report shows the version/release number on the top of every page. It should also be required to report the source of the input data and what periods it covers (including months, days and years).

CONCLUSION

Based on the limitations of spreadsheets, and the ongoing potential problems from undocumented, untested and uncontrolled spreadsheets, the continued use of spreadsheets to manage mission-critical functions is an unacceptable risk for 21st century firms. In too many cases, spreadsheets may be sabotaging their accuracy.

The stakeholders of every organization should insist that corporate executives take greater responsibility for the output and use of spreadsheets that impact client decisions. Now that we know the old methods are broken, we accept the risk if we do not correct them. ▼

Investment
SYMPOSIUM

MARCH 13-14, 2014
NEW YORK MARRIOTT MARQUIS
NEW YORK, NY

Investment strategies for challenging times

The Investment Symposium is developed and produced by industry experts for investment professionals working with pensions and insurance.

Get perspective on the industry's most pressing challenges and learn valuable strategies for navigating the future from trusted leaders and the best minds in the field.

Learn. Overcome. Grow. Register today for the Investment Symposium. Learn more information by visiting **InvestmentSymposium.org**

Sponsored by:

SOCIETY OF ACTUARIES

PRMIA
The Professional Risk Managers' International Association

SOCIETY OF ACTUARIES Forecasting and Futurism Section

Forecasting & Futurism

N E W S L E T T E R

475 N. Martingale Road, Suite 600
Schaumburg, Illinois 60173
p: 847.706.3500 f: 847.706.3599
w: www.soa.org