



SOCIETY OF
ACTUARIES

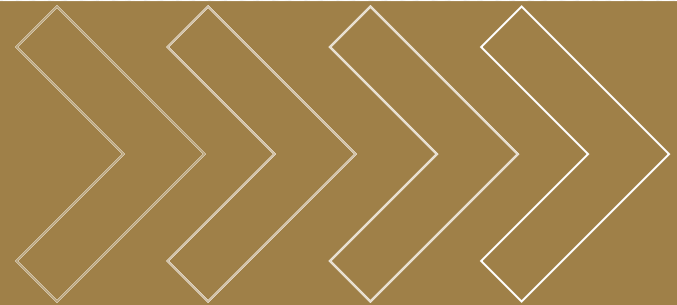
PREDICTIVE
ANALYTICS
AND FUTURISM
SECTION

Predictive Analytics and Futurism

ISSUE 14 • DECEMBER 2016

Making Predictive Analytics Our Own

By Joan C. Barrett



- 3 **From the Editor: Insights from a Dead Salmon!**
By Dave Snell and Kevin Jones
- 4 **Chairperson's Corner: On Volunteering, Learning, and a Sense of Community**
By Ricky Trachtman
- 6 **Looking Back and Ahead**
By Brian Holland
- 8 **Making Predictive Analytics Our Own**
By Joan C. Barrett
- 10 **Deciding What to Research: How to Spot and Avoid Bias**
By Kurt Wrobel
- 14 **Five Myths and Facts about Artificial Intelligence**
By Dr. Anand S. Rao
- 18 **Abstractions & Working Effectively Alongside Artificial Intellects**
By Dodzi Attimu and Bryon Robidoux
- 24 **Machine Learning: An Analytical Invitation to Actuaries**
By Syed Danish Ali
- 28 **Use Tree-based Algorithm for Predictive Modeling in Insurance**
By Dihui Lai, Bingfeng Lu
- 32 **Creating a Useful Training Data Set for Predictive Modeling**
By Anders Larson
- 35 **The random GLM Algorithm: A Better Ensemble?**
By Michael Niemerg
- 38 **Collaborative Filtering for Medical Conditions**
By Shea Parkes and Ben Copeland
- 42 **Getting Started with Deep Learning and TensorFlow**
By Jeff Heaton
- 46 **Guide to Deep Learning**
By Syed Danish Ali
- 48 **Introduction to Using Graphical Processing Units for Variable Annuity Guarantee Modeling**
By Bryon Robidoux

Predictive Analytics and Futurism

Issue Number 14 • DECEMBER 2016

Published by The Predictive Analytics and Futurism
Section Council of the Society of Actuaries

475 N. Martingale Road, Suite 600
Schaumburg, Ill 60173-2226
Phone: 847.706.3500 Fax: 847.706.3599

SOA.ORG

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 Predictive Analytics and
Futurism Section Web page at [http://
www.soa.org/predictive-analytics-and-
futurism/](http://www.soa.org/predictive-analytics-and-futurism/).

This publication is provided for
informational and educational purposes
only. Neither the Society of Actuaries
nor the respective authors' employers
make any endorsement, representation
or guarantee with regard to any content,
and disclaim any liability in connection
with the use or misuse of any information
provided herein. This publication should
not be construed as professional or
financial advice. Statements of fact
and opinions expressed herein are
those of the individual authors and are
not necessarily those of the Society of
Actuaries or the respective authors'
employers.

Copyright © 2016 Society of Actuaries.
All rights reserved.

2016 SECTION LEADERSHIP

Officers

Chairperson
Brian Holland, FSA, MAAA
brian.holland@aig.com

Vice Chairperson
Ricky Trachtman, FSA, MAAA
ricardo.trachtman@milliman.com

Secretary/Treasurer
Anders Larson, FSA, MAAA
anders.larson@milliman.com

Council Members

Vincent J. Granieri, FSA, EA, MAAA
vgranieri@predictiveresources.com

Geoffrey Hileman, FSA, MAAA
ghileman@kennellinc.com

Sheamus Kee Parkes, FSA, MAAA
sheamus.parkes@milliman.com

Bryon Robidoux, FSA, MAAA
bryon.robidoux@aig.com

Qiang Wu, ASA, Ph.D.
qiang.wu@mtsu.edu

Haofeng Yu, FSA, CERA, Ph.D.
haofeng.yu@aig.com

Newsletter Editors

David Snell, ASA, MAAA
dave@ActuariesAndTechnology.com

Kevin Jones, FSA, CERA
Associate Editor
kevin.jones@milliman.com

Board Partner

Joan Barrett, FSA, MAAA
joan.barrett@axenehp.com

SOA Staff

Andrew J. Peterson, FSA, EA, FCA, MAAA
Staff Partner
apeterson@soa.org

Jessica Boyke, Section Specialist
jboyke@soa.org

Julia Anderson Bauer, Publications Manager
jandersonbauer@soa.org

Sam Phillips, Staff Editor
sphillips@soa.org

Erin Pierce, Graphic Designer
epierce@soa.org

From the Editor: Insights from a Dead Salmon!

By Dave Snell and Kevin Jones

Our SOA Cultivate Opportunities Team, and the PAF section have been saying for a long time that actuaries provide some valuable insurance and risk management knowledge that some of the other quants (MBAs, CFAs, CPAs, ... substitute you're favorite xxAs), competing on the predictive analytics fronts with us cannot contribute. We feel that this is an important differentiator.

At last, the media has described the folly of analytics without subject matter expertise: Apparently, tens of thousands of studies have been published that cite fMRIs (functional MRI scans) as their justification; yet the underlying software for analysis of fMRIs had a software glitch that may have created false positives up to 70 percent of the time.

“A graduate student conducted an fM.R.I. scan of a dead salmon and found neural activity in its brain when it was shown photographs of humans in social situations. Again, it was a salmon. And it was dead.”

http://www.nytimes.com/2016/08/28/opinion/sunday/do-you-believe-in-god-or-is-that-a-software-glitch.html?action=click&pgty=Homepage&clickSource=story-heading&module=opinion-c-col-right-region®ion=opinion-c-col-right-region&WT.nav=opinion-c-col-right-region&_r=0

This issue offers you a collection of basic and leading edge predictive analytics. It also gives you insurance applications and some insights into how to employ the analytics in a more useful manner—one that minimizes the chance of dead salmon results.

“Chairperson’s Corner: On Volunteering, Learning, and a Sense of Community,” by Ricky Trachtman: Our incoming chairperson, Ricky, writes about a common theme throughout the PAF section: community. This may be viewed as a temporal update to the “it takes a village” concept. Ricky tells how the PAF community (then F&F—forecasting & futurism) helped him get involved, and how the volunteer spirit of our community becomes contagious—in a very good way. Yes, you put in a lot of time without direct compensation; but the result is highly rewarding!



“Looking Back and Ahead,” by Brian Holland: Brian gives us a reminder of the many cool accomplishments of our section just in the last year. It is an impressive list of results from our tireless and enthusiastic volunteers. It is extra work to volunteer your time and expertise; but the rewards are well worth it for others and for yourself. As Brian says, “A lot of the SOA is what we make of it, and the sections are a great way to pitch in and make the SOA what you want it to be.”

“Making Predictive Analytics Our Own,” by Joan C. Barrett: The Society of Actuaries is embracing predictive analytics enthusiastically now; and our board partner, Joan, provides sage advice on how we can take back much of the predictive analytics space in the financial services industries. She tells us how she personally shows that actuaries add value beyond the mathematical mechanics of modeling. She stresses monitoring experience (with several cogent examples) and employing behavioral economics (again, with examples) to check the sensibility of model results, and improve the models.

“Deciding What to Research: How to Spot and Avoid Bias,” by Kurt Wrobel: Kurt provides an article without even one mathematical formula in it; yet it contains essential advice for any actuary involved in predictive modeling. He explains how studies can become biased; and how you can watch for and avoid or minimize

CONTINUED ON PAGE 5

Chairperson's Corner: On Volunteering, Learning, and a Sense of Community

By Ricky Trachtman

A few years ago, I started on a path of volunteering to try to give back to an organization that had given much to me. By “much,” I am not talking about the anxiety, sleepless nights, stress, heartburn, and coffee addiction that are the oh-so-common side effects of the exam process. Nor do I mean the joy that all those confused faces bring me after mentioning in a conversation that I am an actuary, which is then followed by, “You’re a what?” by “much,” I mean, all the knowledge, perseverance, and self-learning skills that I have acquired through the SOA, and most of all how it has given me my profession. This path of volunteering eventually brought me to Ben Wolzenski, who asked me to put my name down to become part of the Forecasting and Futurism Council. I had always been interested in forecasting techniques, so I agreed. Soon after, I was elected as a council member and became the secretary/treasurer. I did not know exactly what to expect of the council and the section as a whole, but I can tell you that what I have experienced during the last couple of years being part of the council, is something I never expected. I was completely surprised.

What I found is a group of very involved individuals who care deeply about learning and innovating. They are passionate about predictive analytics and the technology that surrounds it. These are people who have created an environment that welcomed questions, focused on sharing their knowledge, and inspired others to do the same. This is a group of people who are willing to provide their time, effort, and knowledge to the rest of us. While I knew that we had more than a council and more than a section, I did not know how to describe us. So I did what any reasonable member of the predictive analytics and futurism (PAF) section would do, I used technology to find the answer. I opened my browser and started to type words into a google search, words such as group, innovation, members, learning, teaching, welcoming of questions, and volunteering. Eventually, a common word keep coming up and that word was “community.”

I then typed “community” into my google search and the first thing google brought back was references to the sitcom,



“Community,” which is a funny show, but not what I was looking for. Isn’t Google supposed to know what I want to know all the time? I continued scrolling down the result page some more until I found definitions of the word community that confirmed my suspicions. The PAF is more than just a section. It is a community that supports and nurtures knowledge.

As such I want to thank the members of the council, the friends of the council, the section volunteers, our section specialist Jessica Boyke, our SOA staff partner Andy Peterson, and our board partner Joan Barrett, for making this community what it currently is. While being part of the council I have had the privilege to work with two great chairs, Doug Norris and Brian Holland. Thank you both for all your great work and encouragement.

This sense of community has made volunteering a rewarding activity from which I have received more than I can possibly give back. I have made friends, learned many things and gained experience in leadership that I can bring back to my day-to-day work. I hope this short note inspires and encourages you to volunteer and continue to help our section be the great success that it already is.

I would like to give a special thanks to our great newsletter editors, Dave Snell and Kevin Jones, for all the hard work they do so we can all enjoy this publication. ■



Ricky Trachtman, FSA, MAAA, is a principal and consulting actuary at Milliman. He is currently the chairperson of the SOA Predictive Analytics and Futurism section council. He can be reached at ricardo.trachtman@milliman.com.

bias in your work. His tips can serve as a guideline for making sure you have objectivity, quality, and relevance in your modeling work.

“Five Myths and Facts about Artificial Intelligence,” by **Anand Rao:** Movies and television have perpetuated many misconceptions about Artificial Intelligence in the name of drama, but Anand is stepping forward to clear up a few of these. In order to see its future in the insurance industry, one has to understand what AI is, what it can do, and what its relationship will be to human beings. It may not produce magical solutions to all problems, but has great potential to enhance decisions.

“Abstractions and Working Effectively Alongside Artificial Intellects,” by **Dodzi Attimu and Bryon Robidoux:** In any technical field, abstraction is necessary to communicate ideas without getting bogged down in details. Dodzi and Bryon outline different types of abstraction and explore their uses in a software setting. They push this even further into layers of abstraction in machine learning. Remarkably, the human brain’s approach of Sparse Distributed Representations (SDRs) can be used to create a robust learning solution.

“Machine Learning: An Analytical Invitation to Actuaries,” by **Syed Danish Ali:** In ratemaking, do we trade specific risk for systematic risk? Machine learning can give new understanding in that question. Danish shows how it can be applied in exploring data, predictive modeling, and unstructured data mining. Moreover, it’s important in understanding big data and its effect on the insurance industry.

“Use Tree-based Algorithm for Predictive Modeling in Insurance,” by **Dihui Lai and Bingfeng Lu:** They provide a clear explanation of tree-based models, starting from a simple binary tree, through CART (classification and regression tree), to an ensemble of trees in a random forest, or boosted tree. In addition, they give advice on when to use which type, what their particular strengths are, and when a tree is not the best choice for your model.

“Creating a Useful Training Data Set for Predictive Modeling,” by **Anders Larson:** Data scientists (and actuaries involved in predictive analytics) rant about the value of clean data of sufficient quality and quantity for statistical significance. As Anders says, “This goes beyond the simple ‘garbage in, garbage out’ principle,” but what is actually meant by good data, and how do you obtain it? Read here the key terms that define a good training set; and how to create and use training sets that will provide insights on situations involving transactional data, and expanding populations.

“The Random GLM Algorithm: A Better Ensemble?” by **Michael Niernerg:** What if you could combine the benefits of a generalized linear model (GLM) with those of a random forest? Michael describes a lesser-known technique that attempts to do this—the random generalized linear model (RGLM). It uses randomization and bagging (often associated with random forests) and applies them to a GLM. He compares the RGLM

to random forest results, and provides overall observations about the strengths and weaknesses of this method.

“Collaborative Filtering for Medical Conditions,” by **Shea Parkes and Ben Copeland:** Collaborative filtering systems, known to some of us as recommender systems, are used by Amazon, Netflix, Google, and many others to determine what you might wish to see or buy based on what others like you have seen or bought, and on what other items are similar to ones you already saw or bought. Shea and Ben take us beyond the simple discovery of frequent item sets, and introduce ratings based upon estimated latent factors. Their example for a diabetes patient shows how this approach can improve the coding of patient comorbidities.

“Getting Started with Deep Learning and TensorFlow,” by **Jeff Heaton:** Deep Learning is the machine learning technique of choice for most image recognition and time series predictive analytics models. Jeff names the four powerhouse open source toolkits from Amazon, Baidu, Google, and Microsoft; and then he describes the Google product, TensorFlow, in more depth. TensorFlow, was introduced in 2015. It is new, and hot! Jeff shows us how to use it—with code examples to get started. Follow Jeff’s article, based on the graduate course he is teaching at Washington University.

“Guide to Deep Learning,” by **Syed Danish Ali:** Unstructured data presents data scientists with many challenges, but Deep Learning can be a valuable tool for insight here. Here, Danish gives an overview of the purpose and structure of Deep Learning, as well as the challenges. There are also many variants on deep architectures outlined.

“Introduction to Using Graphical Processing Units for Variable Annuity Guarantee Modeling,” by **Bryon Robidoux:** The need for faster processing has run into a bottleneck with traditional central processing units (CPUs) and the graphical processing unit (GPU), where dozens or even thousands of special purpose calculation engines can work in parallel, offers the potential for breakthrough speed improvements. However, it is not as simple as buying and inserting a high-end GPU card. Bryon takes us through a reality check on how, and when, to use these powerful processors effectively. ■



Dave Snell, ASA, MAAA, is technology evangelist at RGA Reinsurance Co. in Chesterfield, Mo. He can be reached at dave@ActuariesAndTechnology.com.



Kevin Jones, FSA, CERA, is associate actuary at Milliman in Buffalo Grove, Ill. He can be reached at Kevin.Jones@Milliman.com.

Looking Back and Ahead

By Brian Holland

It has been a privilege and an honor to serve on the PAF Section Council and as chair for the last year. It has been a delight to work with the other councilors whom you have elected and also with our quite active community of Friends of the Council. From my corner it looks like a particularly productive group. In 2016 we have a few accomplishments:

- Podcasts on special topics related to predictive analytics (find them on our SOA page);
- The first Practical Predictive Analytics Seminar, which followed the Life and Annuity Symposium;
- Two nice, thick newsletters;
- Virtual Open Forums, facilitating connections between members and newsletter contributors;
- A record number of sponsored sessions at SOA meetings, including two at ValAct this year; and
- Some exciting webcasts in the works.

Friends of the council are playing a major role:

- Dorothy Andrews—serving as ValAct coordinator
- Richard Xu—leading Annual Meeting coordination
- Dave Snell, Kevin Jones—editing the newsletter
- Eileen Burns, Matthias Kullowatz—presenting a seminar, volunteering much time and creative energy on this effort
- Ben Wolzenski—organizing volunteer activities

I have to give Satadru Sengupta a shout-out as well for presenting at the seminar. He is a data scientist, not an actuary. Reaching out to other experts is important to us, and we were glad he could present.

Changing our name has communicated our work to more actuaries and membership has shot up. We have even more members to serve. Our business is so complex and everyone is so busy, there is surely time for you to pitch in! Please see Doug Norris'



excellent article in Issue 12 for more. I've been very encouraged that he has taken time this year to call-in every month and be there as a sounding board for me—that's one more friend of the council who has been active.

I look forward to seeing what the section does in 2017. With the experience and interests on the council and your interests, with Ricky's leadership, it will remain a fun and collegial group. I'll remind you what I'm reminding myself: there is room to pitch in and support your interests while making friends. Do you want to see a webcast on a particular topic, or research done in a particular direction? The odds are, you're not alone. I encourage you to speak up and connect with others, maybe even organize some content or guide us to it. A lot of the SOA is what we make of it, and the sections are a great way to pitch in and make the SOA what you want it to be. ■



Brian D. Holland is the outgoing chairperson of the PAF council. He is director and actuary at AIG Life & Retirement in Atlanta, Ga. He can be reached at Brian.Holland@aig.com.

SOA Explorer Tool

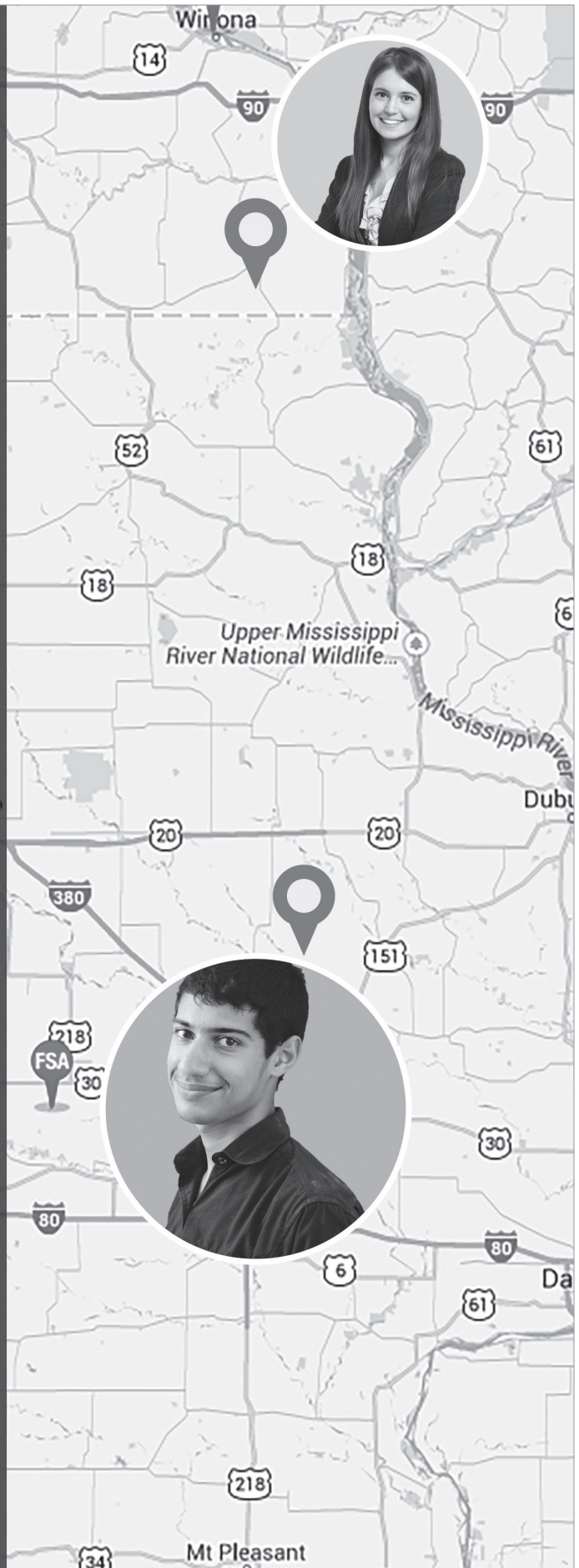
Find Fellow Actuaries
Around the Block or
Around the Globe

The SOA Explorer Tool is a global map showing locations of fellow SOA members and their employers, as well as actuarial universities and clubs.

Explorer.SOA.org



**SOCIETY OF
ACTUARIES**



Making Predictive Analytics Our Own

By Joan C. Barrett

Over the past few years, many quantitative roles in insurance companies have been filled by data scientists, economists and other near-professions rather than actuaries. In my area of practice, health care, the focus of these roles has been to produce studies that determine whether a recommendation to reduce costs or increase quality, such as a disease management program or an employee wellness program, is effective or not. With health care at 17 percent of the GDP, well-controlled, reliable statistical studies provide critical insights and background information for clinicians and business leaders alike. As an actuary, I regularly review every study I can get my hands on as a starting point for pricing a new product or for evaluating the impact that a change in technology may have on future health care costs. While I find these studies to be a useful starting point, most of them are just too specific, too complicated and too dated to use directly in my work. Instead, I devise something that is simple to apply and easy to explain. Of course, I caveat my work, describe the risks, monitor the results and update as needed.

Each time I go through this process, I ask myself the question “Is there any way to apply the power of predictive analytics in this process?” I have concluded that the answer to this question is yes and that the result will be the next generation of predictive analytics. But what do we need to know or learn to make this happen? Here are some examples.

Monitoring Experience. As actuaries, we routinely monitor experience for almost all our work, especially repeatable tasks like pricing and reserving. The most difficult part of that work is often deciding what to do once the results are tallied: Should we lower the rates? Should we raise the reserves? What happens if we wait for more data? The big fear for every actuary, of course, is that we will take some kind of action based on recent experience, only to find out later that the original projection was correct all along. If the original projection was based on a statistical model, then we can answer the question, “What are the chances I would see these results if my original projection is correct?” The answer to that question can provide valuable guidance in the decision-making process.

If a simple linear regression analysis was used for the original projection, the math is easy. We can use the variance of the re-

siduals to calculate the probabilities. Of course, as the projection gets more complicated, so does the probability calculations.

Pricing Risk. We have all studied risk analysis as part of the exam process. In most cases, the underlying theory is based on the assumption that we have a single distribution that defines the “total risk” and all the risk-related calculations are based on that distribution. In reality, the hardest part of what we do is to determine the total risk. As a health actuary, determining the trend is almost always the key pricing assumption. A health actuary will have a real problem if pricing is based on the assumption that the average health care costs will increase by 5 percent next year only to find out later that they increased by 10 percent. Given this sensitivity to trends, the questions I am asked most often include:

- If we add a 1 percent margin to our best estimate, what are the chances we will lose money anyway?
- If we cut rates by 2 percent in order to be competitive, then how much can we expect to lose?
- How comfortable are you really with your best estimate?
- What are the chances we will lose more than \$1 million?

To answer these questions, we really need to think of risk in two components: A pricing risk and a random variation risk. The random variation risk is the risk associated with fluctuations if the overall pricing assumptions were exactly right. If a projection is based on a simple linear regression, then the random variation risk is the risk as calculated using the variance of the residuals. Suppose, for example, that an insurer used a simple linear regression to determine that their best estimate of claims costs was \$1,000 per life with a standard deviation of \$50. To be conservative, they added a \$50 provision for adverse deviation for a total of \$1,050. The expected gain is the margin, \$50. But, since the margin and the standard deviation are the same, then the probability of a loss is about 16 percent. Is this sufficient to meet the concerns of the insurer or is it too conservative for marketing purposes?

The pricing risk is the risk, or opportunity, that happens if the overall claims are missed either intentionally or not intentionally. In the prior example, suppose the original projection was wrong and the true best estimate is \$1,030, then the expected net gain is now only \$20, but the probability of losing money has gone up to 35 percent. Is this safe enough? Too conservative? And what is the probability of missing the trend by 3 percent anyway? In this case, the pricing risk for the scenario is \$30, the value of the miss. The overall impact of this scenario is \$30 x the probability of a \$30 miss. If a simple linear regression is used, then that probability can be determined using the variance of the slope estimator. The total pricing risk is the sum of the pricing risk over all scenarios.



Admittedly, the examples above are simple and lead to an obvious question: Why do we need to do this when in all likelihood the chances of a 3 percent understatement are the same as a 3 percent overstatement and it will eventually come down to the best estimate scenario anyway? So, even if the pricing scenarios are symmetrical, which is a big if, the underlying risk may or may not be symmetrical, especially if there is any type of reinsurance or policy limits involved.

Behavioral Economics. Actuaries have been talking about the impact of consumer behavior on experience for some time now. Historically, the emphasis has been mostly on avoiding anti-selection, but more recently there has been more and more discussion on the impact of consumer behavior on insurance products. Life actuaries, for example, are looking at ways to use information about whether or not a prospect goes to the doctor regularly as an underwriting tool. Health actuaries often look at the effectiveness of using financial and other incentives in encouraging consumers to participate in well programs and other efforts to reduce cost or increase quality.

Well-controlled studies are key to measuring the impact of consumer behavior on past experience. The major question is, however, “How can we design a new program or product to get the optimal impact?” This is where behavioral finance comes in. Behavioral economics is a relatively new field that looks at the effects that psychological, social and emotional factors have on how consumers make financial decisions. This concept has been popularized in books like “Nudge” and “Predictably Irrational.” One of the major take-aways from this field is that consumers tend to be somewhat irrational and are often influenced by the

way alternatives are presented. Understanding this concept more fully can have a major impact on product development, especially if products can be designed in a way that reduces anti-selection.

Currently, most of the quantitative work in this field has been based on theoretical experiments: Is a consumer more likely to trade a sure \$100,000 for a 10 percent chance for \$1,000,000? If you give someone a free candy bar, will they give it away for free or charge for it? Although these experiments are fun and somewhat useful to an actuary, they do not reflect how consumers will react in a real situation where the health and financial security of their family may be at stake. Since actuaries have access to a considerable amount of data and a keen knowledge of the underlying business context, actuaries are in a unique position to capitalize on this as an area for personal growth and the growth for the profession.

So, can actuaries really make predictive analytics their own? Of course we can. Many actuaries are already using the techniques described above in their work. We can expect these actuaries to begin sharing their work and their findings more and more through the SOA continuing education and research infrastructure. ■



Joan C. Barrett, FSA, MAAA, is consulting actuary at Axene Health Partners, in Tolland, Conn. She can be reached at joan.barrett@axenehp.com.

Deciding What to Research: How to Spot and Avoid Bias

By Kurt Wrobel

Our profession provides advice for the most visible and important public programs and insurance products, including health insurance, life insurance, and pensions. With significant policy changes occurring in these fields, our advice has the potential to have an even more meaningful impact on the long-term sustainability of these programs. This environment has given our profession the unique opportunity to work on a wide variety of research topics that have a profound impact on people's lives—whether the research involves the changes brought about by the Affordable Care Act or policies regarding the funding of pension liabilities.

Beyond our familiarity with the technical features of insurance products and regulation, we have an opportunity to contribute to improving policies because of our reputation for providing impartial advice based on facts and reliable data. As we consid-

Beyond our familiarity with the technical features of insurance products and regulation, we have an opportunity to contribute to improving policies. ...

er our broader role in informing public policy, I also think it's important to carefully choose our research focus—particularly with the wide range of opportunities available to our profession.

In picking our spots, we need to be very careful in performing research in topical areas that are the most likely to result in biased research, including the exclusion of inconvenient data, conclusions drawn from only a subset of results, and extrapolation to a desired result. Even if our own research is conducted without bias, the results have the greatest potential to be dismissed along with much of the other research in a topical area, as simply confirming an already held political position.

This article will focus on characteristics that are most likely to lead to a biased research study—or a perception of such bias—and offer two case studies where an impartial analysis with meaningful results would be difficult if not impossible to accomplish.

HOW TO SPOT A RESEARCH TOPIC WITH THE GREATEST POTENTIAL FOR BIAS

As we consider research areas that could produce a biased analysis, several characteristics should raise concern about our involvement. Although any individual characteristic may not inherently lead to a biased research study, in many cases, a combination of these factors is much more likely to become the foundation for a biased study.

Disparate data sources with an inconsistent data collection process: Because accurate data will ultimately provide the foundation of any objective analysis, we should be careful to insist on meaningful and accurate data before going to the next step of analyzing this information.

Limited data: In addition to accurate data, a sufficient amount of information needs to be available to draw robust conclusions. With insufficient information, an analysis is little more than a guess with little actual value. In addition, a lack of information is much more likely to lead a researcher to substitute preconceived opinions to fill in the gaps of missing data.

Attempts to explain the expected outcome of a complex system over an extended period of time: To the extent a system has multiple causal variables that can affect the broader system and other causal variables, we should be very careful about assessing future events in these research areas. In these cases, a detailed analytic review will provide little additional insight into explaining the system and could provide unwarranted confidence in predicting the underlying system—particularly if the estimates are presented as a single-point estimate rather than a range of potential outcomes.

A politically charged question where a definitive answer to the research will not ever be known with any degree of accuracy: These research questions are best suited for those who have an interest in the outcome of the research question and not for unbiased truth-seekers. In addition, the lack of a definitive policy conclusion makes research in this area much less meaningful.

The challenge, of course, is that we often encounter gray areas where an analysis may have aspects that are far less than ideal, including imperfect data or the necessity to make projections of a very complex underlying model. In highlighting these limitations, I'm not suggesting that we never conduct an analysis that has some of these limitations, but rather that these limitations be weighed holistically as we consider whether a project warrants an actuarial review.



The above criteria are also very much consistent with the Society of Actuaries Public Policy Research and Analysis Statement and the goals of the Project Oversight Groups that help guide the research by the Society of Actuaries. The attached sidebar includes an excerpt from this statement that highlight the goals of SOA research.

HEALTH SYSTEM COMPARISONS ACROSS COUNTRIES

To better highlight the problem, the following discussion highlights an area of research that best exemplifies this problem with biased analysis—comparisons of country-specific health systems based on health outcomes. This research has long interested economists and other researchers who seek to explain differences in health outcomes among different countries.

While the data and methods vary, the research usually involves comparing an outcome (infant mortality, for example) over several countries with several variables that could explain the outcome without assuming a specific treatment that could be driving the result. In much of this research, the results will highlight the United States as an outlier with greater expenditure (as a percentage of GDP) and worse results (higher infant mortality, for example) and then suggest various policy solutions to help improve its position.

This research is instructive because it highlights all four elements of a research study that should be avoided:

- The data often comes from disparate sources with an inconsistent data collection process: As highlighted in many research studies, the data collection methods, the definition of specific outcomes, and the measurement of such outcomes can vary widely among different countries. Instead of using data reported by health care professionals with strict definitions in a consistent manner, some countries use surveys and family-reported data with definitions that are not uniformly applied across all countries. The data can be further affected by the extent of the medical treatment, with those countries with aggressive medical practices for costly conditions reporting results differently than other countries.
- Limited data: The research is often focused on a limited number of actual data points to perform the actual analysis. In many cases, the research focus is largely dependent on outcomes from the United States using fewer than 50 data points.
- Attempts to explain the expected outcome of a complex system over an extended period of time: The causal factors contributing to a health outcome could include diet, lifestyle choices, genetic factors, income, education, culture, and the

SOCIETY OF ACTUARIES PUBLIC POLICY RESEARCH AND ANALYSIS STATEMENT (APPENDIX 1)

The following excerpt highlights the Society of Actuaries stated goal regarding research:

The SOA has a history of working with public policymakers and regulators in developing historical experience studies and projection techniques as well as individual reports on health care, retirement and other topics. The SOA's research is intended to aid the work of policymakers and regulators and follow certain core principles:

Objectivity: The SOA's research informs and provides analysis that can be relied upon by other individuals or organizations involved in public policy discussions. The SOA avoids taking advocacy positions or lobbying specific policy proposals. (This objectivity is emphasized in the selection of the Project Oversight Group participants.)

Quality: The SOA aspires to the highest ethical and quality standards in all of its research and analysis. Our research is overseen by experienced actuaries and non-actuaries from a range of industry sectors and organizations. A rigorous peer review process ensures the quality and integrity of our work.

Relevance: The SOA provides timely research on public policy issues. Our research advances actuarial knowledge while providing critical insights on key policy issues, and thereby proving value to stakeholders and decision makers.

Quantification: The SOA leverages the diverse skill sets of actuaries to provide research and findings that are driven by the best available data and methods. Actuaries use detailed modeling to analyze financial risk and provide distinct insight and quantification. Further, actuarial standards require transparency and the disclosure of assumptions and analytic approach underlying the work.

country's health system and financing. Although some of these factors can be controlled for in the research, it remains extremely difficult to reliably control for these factors over several countries and account for all the factors that could contribute to a particular outcome.

- A politically charged question where a definitive answer to the research will not ever be known with any degree of accuracy: Because an experiment cannot be developed to directly compare one health care system with a population from an-

other country, a true definitive answer to the research is simply not possible. Without a clear, definitive conclusion, this research has limited use and it is much more likely to lead the researcher to develop a conclusion consistent with his or her political beliefs or a preconceived expectation.

And finally, in this macro-level research, we are less likely to have the opportunity to use our knowledge of the regulatory systems or health system specific information to make an evaluation of the differences.

While this research focus—comparing health outcomes across various national systems—may not be appropriate for our profession, I am confident that many in the actuarial profession would find the topic interesting. The differences in how care is delivered and financed and how it impacts outcomes can make for an interesting philosophical discussion. However, this philosophical interest should not lead our profession to engage in research that has the potential to impact our reputation and has little policy importance.

CLIMATE CHANGE

Similar to the cross-country comparative research, the climate change debate has many attributes that have the potential to ultimately lead to a biased analysis or less-than-meaningful results.

- The data used are collected from disparate sources and are subject to significant error. The historical temperature record has been obtained from a wide variety of sources with differing data quality, including Victorian-era sailors dragging thermometers behind their ships, ocean buoys, readings on land with readings between different sites taken at different intervals in different times of the day, and satellites measuring surface and lower troposphere temperatures.
- The data is limited, particularly in unpopulated and pre-development sections of the globe. As a result, much of the record prior to widespread distribution of the thermometer is based on proxy determinations, such as tree rings. Even in the more recent historical era, the raw data is then extensively modified in an attempt to homogenize it, such as filling in gaps for missed readings, adjusting results to estimate the effect of different recording processes, and estimating the effect of urban heat islands.
- The underlying system explaining global warming is very complex and dependent on many causal variables that could impact global temperature, and excludes many causal variables that likely affect global temperatures but are too complex to model, such as the effect of clouds. This complexity ultimately makes any modeling effort subject to significant error and leads to widely divergent expected results among researchers.

- The extent of global warming will not likely have a definitive conclusion for a long period of time, and the expected impact differs widely among researchers. Similarly, the attribution of any effects of the changing climate between natural versus manmade sources will not be known—nor are these effects reliably predictable—making it difficult to provide objective advice to policymakers.

And importantly, much of the research is already being performed by scientists and those with the expertise in climate change who are much better positioned to answer this question than actuaries.

In saying this, I'm not advocating a position, suggesting that this topic is not important, or that actuaries should not make some consideration of the potential for climate change or variability in our future estimates using studies from other disciplines. Instead, I believe the characteristics of the climate change question naturally lend themselves to areas of expertise outside the actuarial profession, and any conclusions and recommendations made by actuaries will not benefit our profession—particularly as we attempt to expand our influence in other areas of research that are more closely linked to our expertise.


CONCLUSION

Our profession has built a reputation as unbiased truth-seekers focused on questions that are important to the financial security of individuals, companies and governments. In building this reputation, we have focused on the aspects of our experience that are most important in developing a well-reasoned policy decision, including our technical skills, knowledge of detailed regulatory rules, and our access to important real-time information. As we look to expand our influence in a wide range of research areas with growing importance, I also believe we need to proceed carefully in areas that have historically produced biased analysis and are unlikely to produce meaningful results.

Instead of benefiting our profession, these research areas have the potential to lead us away from the work that built our reputation and toward advocacy positions that have done little to expand the influence of other professions. ■



Kurt J. Wrobel, FSA, MAAA, is chief financial officer and chief actuary of Geisinger Health Plan in Danville, Penn. He can be reached at kjwrobel@thehealthplan.com.



**SOCIETY OF
ACTUARIES**

Knowledge on the go

Insightful podcasts are available
to listen from anywhere!

The Society of Actuaries offers topical podcasts for those interested in insight and perspectives from fellow members. The podcasts are free to download and can be listened to from your computer or any portable audio device. Check back often as new podcasts are released.

SOA.org/Podcast

Five Myths and Facts about Artificial Intelligence

By Dr. Anand S. Rao

Artificial intelligence (AI) is splashed throughout the headlines these days. AI recently beat the human Go champion, autonomous cars can drive themselves, social bots can mimic human interactions and converse with others in social networks, and AI totally manages some hedge funds.¹

However, countering all of this exciting news is fear about the potential for AI to take the place of and even pose an existential threat to humans. In a seminal paper published in 2013, Frey and Osborne² estimate that nearly 47 percent of U.S. employment is susceptible to computerization and automation. Moreover, insurance agents, sales, underwriters, and claims adjusters are some of the jobs that have a high potential for automation.

AI will play—and arguably is already playing—a prominent role in insurance.³ In personal insurance,⁴ commercial insurance,⁵ and life insurance,⁶ AI has already had an impact on how insurers 1) underwrite, price, market, and manage coverage, 2) target and manage their customers, and 3) distribute products. And, while AI will automate within 10 years some of the routine tasks insurance agents, underwriters, actuaries,⁷ and claims adjusters perform today, we believe that it offers a great opportunity for insurers to transform these roles. Far from just automating certain tasks, AI will augment human expertise with more sophisticated data, tools and algorithms. This will enable insurers to make faster, better and cheaper strategic and operational decisions. At the same time, human experts embody a wealth of knowledge of and experience with the economy, markets, customers' needs and products and can play a critical role in training, testing and refining AI.

However, to better explain what AI may mean for the future of the industry, we need to debunk some of the myths about AI and explain realities.

MYTH #1: ARTIFICIAL INTELLIGENCE IS A DISTINCT AND MONOLITHIC AREA OF STUDY.

The term “artificial intelligence” was coined only 60 years ago. It has been subject to periods of excitement (often called “AI spring”) followed by ones of despair (often called “AI winter”).



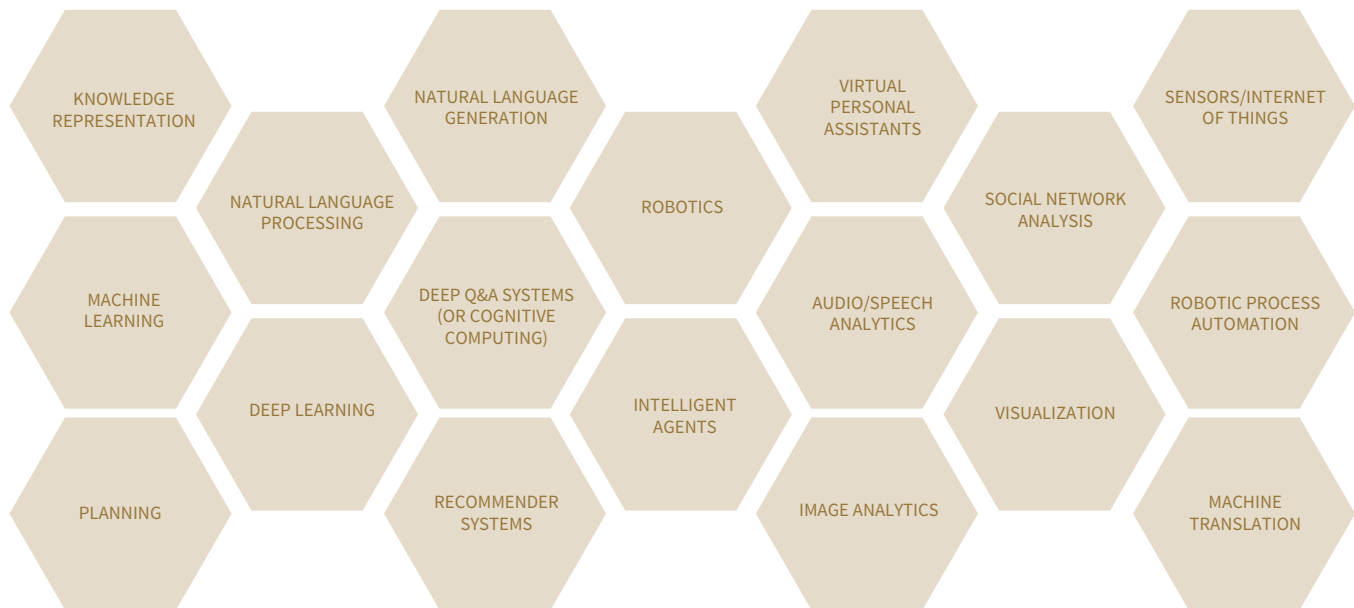
We are currently in a new AI spring, in which hype about the “triumph of AI” and the “fall of yet another human activity to AI” is incessant. While some of the latter claims are realistic and do reflect significant advancements in AI, most are hyperbole. The media, in its quest to simplify things for the layman, tends to equate AI with a specific subfield of AI to the complete exclusion of everything else.

For example, in light of the important advances taking place in machine learning and deep learning, many articles that address advances in these areas equate AI only with those two areas of AI. Similarly, an article on robotics will claim that AI is simply robotics or robotic process automation or conversational agents. This gives the reader—including business executives—a very limited perspective and typically leaves him wondering just what AI is and what it is not.

FACT #1: ARTIFICIAL INTELLIGENCE IS AN INTER-DISCIPLINARY AREA WITH MANY DISTINCT SUBFIELDS.

In the words of John McCarthy, one of the founding fathers of artificial intelligence, “AI is the science and engineering of making computers intelligent.” Given the lack of consensus on what is “intelligent,” AI researchers have pursued a number of areas of human intelligence, including pattern recognition, understanding, learning, problem solving, reasoning, and decision making. With a view towards unifying these disparate areas, Russell and Norvig,⁸ in their classic text define AI as “the study and design of intelligent agents where an intelligent agent is a system that perceives its environment and takes actions which maximizes its chances of success.”

FIGURE 1: ILLUSTRATIVE SUBFIELDS OF ARTIFICIAL INTELLIGENCE



As of yet, there is no single unifying theory or practical solution to implement intelligent agents that can perceive and act in all environments as well as humans. AI researchers have typically adopted a “divide-and-conquer” approach that incorporates techniques from a variety of scientific disciplines, including computer science, statistics, mathematics, physics, biology, philosophy and logic. In addition, whenever any so called “intelligent” problem is solved by AI, it ceases to be called AI. For example, handwriting recognition, speech synthesis, or voice recognition were all considered AI in the 1980s, but as they became part of everyday solutions (e.g., handwriting recognition is now available on tablets, speech synthesis and voice recognition are now available on smartphones), they moved out of the AI realm.

Figure 1 shows a number of subfields that come under the banner of AI. Note that these topics are not mutually exclusive and collectively exhaustive (MECE), but are researchers and businesses are actively pursuing them; moreover, they are mature enough to offer viable solutions to some practical problems.

MYTH #2: ALL TYPES OF PROBLEMS CAN BE SOLVED BY A SINGLE AI SOLUTION.

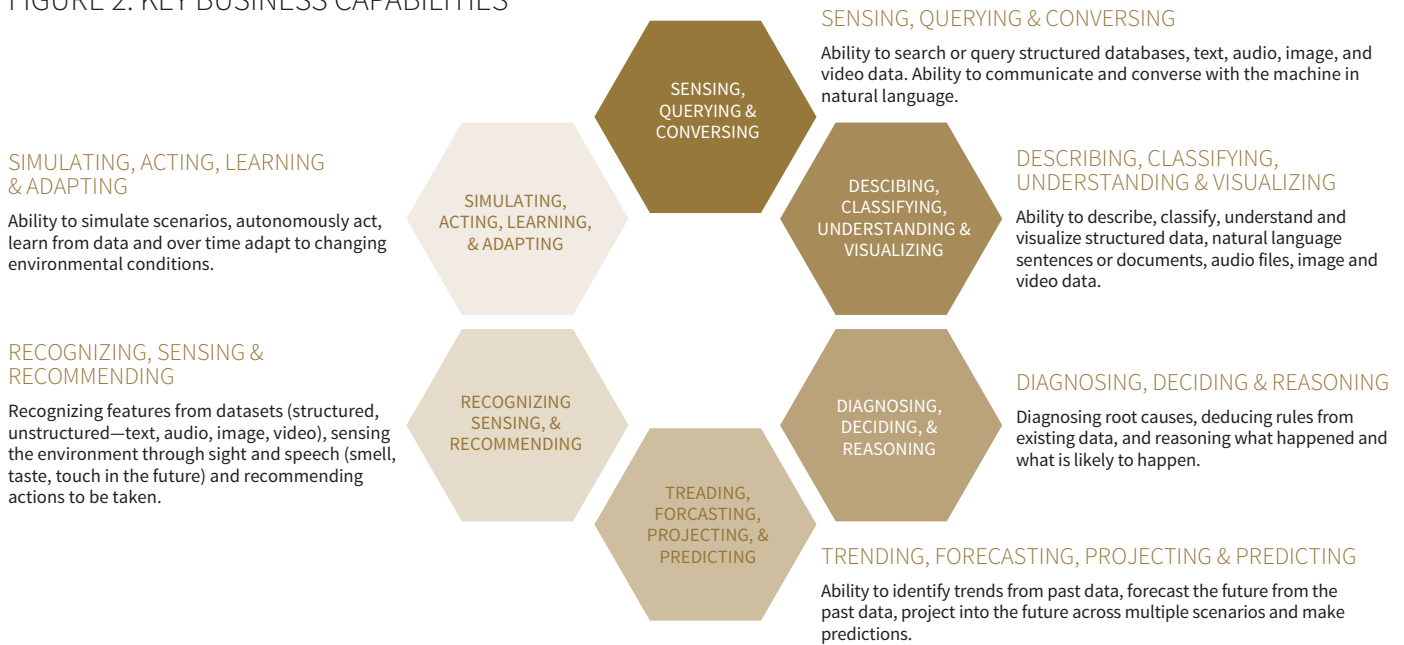
A more serious consequence of Myth #1 is that the media, business executives, AI solution providers, and in some cases even AI researchers truly believe that a single AI subfield or a solution based on that subfield can solve *any* business problem. Part of the myth is due to a genuine ignorance of all AI’s different subfields and what they can or cannot accomplish. Also, given the amount of venture capital and corporate funding that is going

into AI, there is an increasing tendency to make claims that a particular AI solution is relevant to “all” business problems. This could have negative consequences for insurance companies and other organizations that are trying to pilot AI solutions. High hopes that AI solutions can be panaceas may well lead to disillusionment when producers and users don’t change the world in one fell swoop. This could have ripple effects, such as failing start-ups and decreasing funds for AI (which, in turn, could lead to the next AI winter).

FACT #2: DIFFERENT TYPES OF PROBLEMS REQUIRE DIFFERENT TYPES OF AI TECHNIQUES.

The repertoire of human activity and intelligence is so vast that claiming a single AI solution—or even a combination of solutions—can match, much less surpass them, seriously underestimates the complexity of human cognition. Figure 2 outlines the key capabilities necessary for solving business problems. In spite of all the recent advances in AI, the world is still nowhere close to developing a unified theory or a single solution that incorporates all of the capabilities we outline. While the particular business problem a given researcher or company is trying to solve may not require all of those capabilities, it is very likely to require at least some of them. Even if a business problem is narrow enough to require limited capabilities, an AI solution may not be able to fully address it. For example, a natural language question answering system may be suitable for holding a conversation with the customer in order to solve a service issue, but may not be able to simulate multiple strategic options and thereby help management identify an optimum business strategy.

FIGURE 2: KEY BUSINESS CAPABILITIES



MYTH #3: MACHINE LEARNING AUTOMATICALLY LEARNS FROM DATA WITHOUT ANY HUMAN INTERVENTION.

In their enthusiasm to promote machine learning (and deep learning in particular), as well as to contrast it with other techniques, researchers, companies, and media often portray machine learning as something magical; if one feeds data into the system, then it will learn all the patterns and be able to answer any future queries. This perception is not only inaccurate, but also can be counter-productive for companies adopting AI. We have seen companies suddenly shift focus from building robust data collection platforms to trying to build machine learning or deep learning systems. In our view, organizations should have a basic level of “data hygiene” and the foundational AI skills and culture before embarking on advanced analytics, machine learning and deep learning.

FACT #3: MACHINE LEARNING REQUIRES A LABORIOUS PROCESS OF ACQUIRING AND CLEANSING LARGE AMOUNTS OF DATA, AND SELECTING, TRAINING AND GUIDING THE ALGORITHM.

Herbert Simon, a leading AI researcher, defined machine learning as “computers that automatically improve their performance through experience.” The three key phrases here are “automatically,” “experience” and “performance.” Unlike other types of AI that are programmed, machine learning learns patterns from the data it is given in order to improve its performance according to a specific criteria. Once the machine learning system has been

“trained” to recognize certain patterns, it can be provided with new data that it should be able to recognize or classify.

Machine learning systems vary by their representation (e.g., neural networks, rules), how they navigate through the search space of all possible solutions, and optimization criteria. Most machine learning systems require large quantities of training data. In addition, based on the type of learning (e.g., supervised learning) many of them also require humans to provide labeled data (i.e., training data requires the “right” answers). Given the number of different machine learning techniques, human expertise and judgement is critical in selecting the right techniques and fine-tuning relevant parameters. Deep learning, a specific type of machine learning based on multi-layered neural networks, has performed well on a number of image, video, audio and natural language processing tasks. However, the success of deep learning algorithms has depended largely on human ingenuity, notably fine tuning and modularizing the algorithms and collecting and preparing the right labeled data to train the system. In one particular implementation,⁸ humans spend more than 70,000 hours collecting, analyzing, and labelling more than 328,000 photos.

MYTH #4: IN THE WORLD OF AI, THERE IS NO ROLE FOR HUMAN EXPERTISE AND INTELLIGENCE.

Another common myth is that AI will replace humans and take away their jobs, leaving no role for human expertise and intelligence in the future. Some data science and machine learning advocates claim that all they need is data, not human domain

expertise or intelligence. While it is true data scientists who have no domain expertise in a given problem area have won data science competitions, humans have had to frame problems, state hypotheses, set criteria for success, and finally evaluate the solutions.

FACT #4: AUGMENTING HUMAN INTELLIGENCE WITH ARTIFICIAL INTELLIGENCE WILL LEAD TO BETTER RESULTS AND DECISIONS THAN EITHER ONE COULD ACHIEVE ON ITS OWN.

Artificial Intelligence has matured to the point that AI software can perform some common tasks and decisions (e.g., recognizing faces or landmarks in a photograph and tagging them appropriately). However, there are a number of more complex tasks and decisions in business and everyday life that still require human skills, expertise and ingenuity.

Accordingly, AI can play an important role in helping humans make better decisions. But, for example, evaluating a number of strategic options for an insurance company entering the “connected home” insurance market is not something that AI software can do autonomously today. However, we can build sophisticated AI models that capture the dynamics of consumer adoption, competitive dynamics and technology trends to postulate strategic options that AI software can evaluate. In such systems, based on criteria humans provide, AI software can evaluate and optimize a multitude of strategic scenarios—something that is impossible for human brains to do. As a result, augmented intelligence, where AI systems are initially based on human knowledge and then inform humans in an endless feedback loop, will become the norm. (In fact, there are a number of evolving AI systems that approach this level of sophistication today.)

MYTH #5: AI POSES A LOOMING, EXISTENTIAL THREAT TO HUMANITY.

This myth would have not been part of this list unless some well-known academics and business leaders publically announced that it's a potentially serious issue. In his book *On Superintelligence*, Nick Bostrom⁹ argues that an AI system which is able to create better and better versions of itself could very quickly surpass current levels of human intelligence. If this were to occur, he claims that we will be unable to predict or guarantee the values of this superintelligence. Stephen Hawking, Elon Musk, and Bill Gates¹⁰ also have recently echoed these concerns.

FACT #5: SUPERINTELLIGENCE IS NOT A TECHNICAL REALITY. AT LEAST NOT YET.

Superintelligence is a fascinating concept, but is decades away from potential realization. Almost all of AI today fails to approach even general intelligence and is good for addressing only very specific problems (see Myth #2). Even the field of Artificial General Intelligence, which aims to build general purpose intel-

ligence, is still in its infancy. Andrew Ng,¹¹ a leading AI researcher, puts it succinctly: “*The reason I say that I don't worry about AI turning evil is the same reason I don't worry about overpopulation on Mars. Hundreds of years from now I hope we've colonized Mars. But we've never set foot on the planet so how can we productively worry about this problem now?*” Nevertheless, because of concerns that AI may go rogue at some point in the future, there are a number of researchers who are addressing how to build AI systems with sufficient safeguards and ethics.

CONCLUSION

As insurance companies evaluate the use of artificial intelligence in their organizations, they need to understand the true potential and limitations of today's AI technology. While AI has matured substantially over the past couple of decades, it is still a long way from being a “silver bullet” for solving most problems, and it is incapable of completely replacing human labor and decisions. As a result, any deployment of AI needs to start with an understanding of the types of business problems AI can actually address and the best AI subfield(s) and techniques for addressing them. ■



Anand S. Rao, PhD, is Partner, Innovation Lead, PwC Data and Analytics at PricewaterhouseCoopers in Boston, Ma. He can be reached at anand.s.rao@pwc.com

ENDNOTES

- 1 Artificial Intelligence: March of the Machines. The Economist, June 25, 2016.
- 2 The Future of Employment: How Susceptible are Jobs to Computerization? by Frey, and Osborne, September 17, 2013.
- 3 AI in Insurance: Hype or Reality. Top Issues, Insurance, PwC White Paper, March 2016.
- 4 Artificial Intelligence in Personal Insurance: Chatbot agents, RPA ‘No Shoring’ AdminBots and more by Anand Rao. Carrier Management, August 3, 2016.
- 5 Artificial Intelligence in Commercial Insurance by Anand Rao and Joseph Calandro. Carrier Management, August 21, 2016.
- 6 Why Artificial Intelligence will not replace Insurance Agents by Anand Rao. Independent Agent, July 20, 2016.
- 7 2036: An Actuarial Odyssey with AI by Dodzi Attimu and Bryon Robidoux. Predictive Analytics and Futurism, published by Society of Actuaries, Predictive Analytics and Futurism Section, Issue 13, July 2016.
- 8 Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig. Prentice Hall, 1994.
- 9 Microsoft COCO: Common Objects in Context. Lin et al., <http://arxiv.org/pdf/1405.0312v3.pdf>, February 21, 2015.
- 10 Superintelligence: Paths, Dangers, Strategies. Nick Bostrom, Oxford University Press, Oxford, July 3, 2014.
- 11 Stephen Hawking, Elon Musk, and Bill Gates Warn About Artificial Intelligence by Michael Sainato. Observer, August 19, 2015.
- 12 Andrew Ng: Why ‘Deep Learning’ is a mandate for Humans, not just machines. Caleb Garling, Wired.

Abstractions & Working Effectively Alongside Artificial Intellects

By Dodzi Attimu and Bryon Robidoux

In the article “2036: An Actuarial Odyssey with AI” that appeared in the July 2016 issue of Predictive Analytics and Futurism, we explored the impact of artificial intelligence (AI) on actuarial work in particular and white collar work in general. Though there is the tendency to sensationalize the apocalyptic scenarios vis-à-vis, an “AI-calyipse,” there is still a possibility (even if small) that net outcomes could be, well, apocalyptic (e.g., drastic reduction in employment in traditional jobs without others springing up, leading to social upheavals). Although the full implications cannot be forecast with certainty, we can say with certainty that the humans will increasingly continue to work alongside machines (artificial intellects¹). Without a good framework to conceptualize and implement the partnership between humans and machines, very suboptimal utilization of technology can occur. This article specifically is about how abstraction, an important software development concept, can help in this regard. In addition, while there will be more emphasis on abstractions in the framework of software, the concept of abstraction is not limited to that domain.

ABSTRACTION

The concept of abstraction is ubiquitous. In everyday communication, it is summed up in the notion of communication based on one’s audience. Another use of this notion is captured in the phrase “keeping information at a high-level.” One definition of the word abstraction (See [3]) is: “The process of formulating generalized ideas or concepts by extracting common qualities from specific examples.” Informally, abstraction can be said to be a way of specifying the “what” rather than the “how.” In software development circles, abstractions are a means of managing complexity by thinking of software in terms of levels where each level has the right amount of information with more detailed information residing in the lower levels of the hierarchy. Abstractions can serve two related purposes:

1 Generalization—The purpose here is to focus attention on relevant components in a given layer. By abstracting away the lower level details, one can focus on the key components in a given layer. As an example, a high-level programming lan-

guage (e.g., C++, Java, C#) is an abstraction of a lower level language (assembly language). The user at the higher level language layer doesn’t have to worry about the low level assembly language layer. Domain specific languages (DSLs) are also higher levels of abstraction of lower level languages that process them. In the field of artificial intelligence, the highest level of abstraction would be natural language.

2 Flexible Implementation—A direct consequence of (i) above is the opportunity to carve out the specific implementation details and appropriately deal with them in a lower layer. In a solution that outputs data for example, one could have an abstract representation of the data and deal with the details of the various physical output formats like, Excel, JSON, HTML, etc. The flexibility stems from the fact that different implementations can be built for the same general purpose.

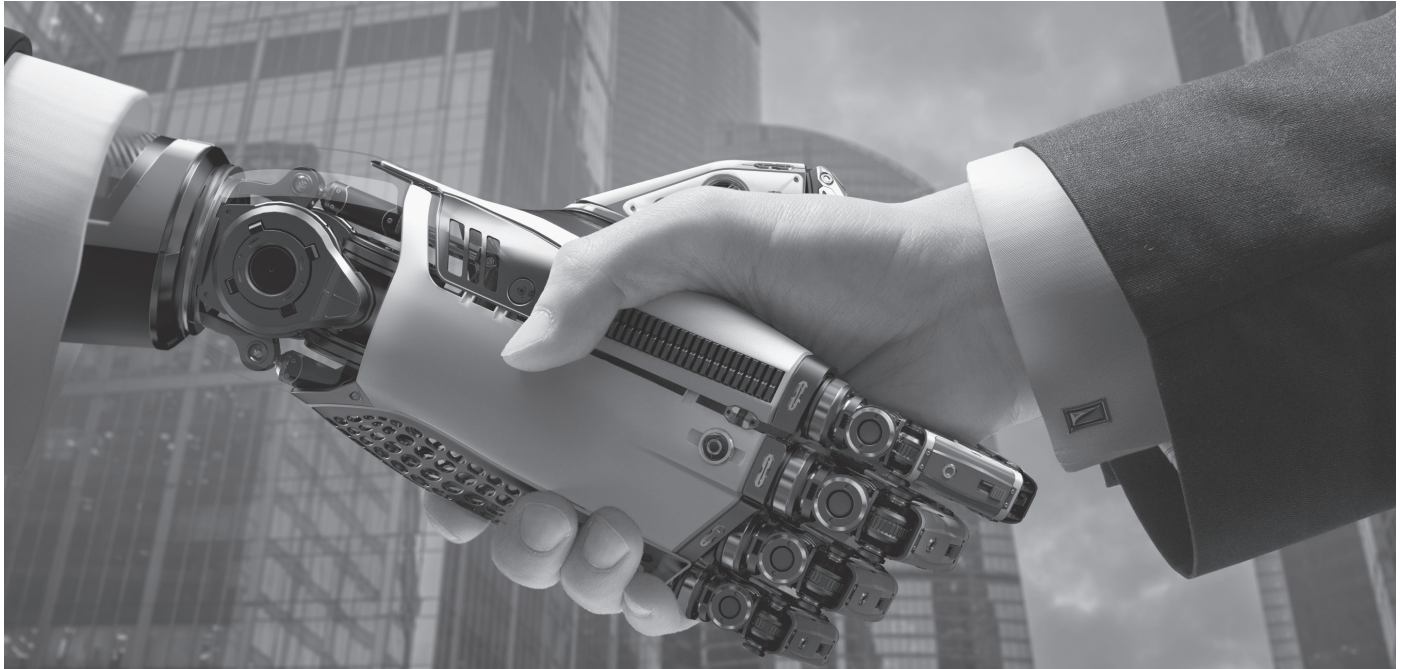
Different layers of abstraction could be identified for a given context/scenario. Typically, (i) would represent a higher level of abstraction and (ii) a lower level of abstraction. But it may be necessary, to further carve out a lower level of abstraction from (ii) and so on. In addition, in the programming language example noted in (i), an assembly language is a level of abstraction above the machine language (a language whose syntax consists of 0’s and 1’s).

In software development, one can identify three types of abstractions: data abstraction, procedural abstraction and configuration abstraction. Data abstraction is concerned with unifying different input sources and coming up with a simplified and generic representation. Procedure abstraction is concerned with defining different types of functionality in generic ways without specifying the details. For example, if the actuary wants to value a future, swap or option, they define a common way of valuing a derivative. The details of valuing each individual are ignored at this level (of abstraction). This allows the ability to design functionality without getting overwhelmed in details and allows for easier extension to other types of derivatives in the future. Configuration abstraction deals with changing the behavior of the software without requiring more code modifications. At its simplest level, this minimizes or in some cases avoids completely the coding of any details needed for the model to run.

FIXED AND VARIABLE PARTS OF SOFTWARE SOLUTION

One can classify AI into the categories of classical, machine learning and machine intelligence. At the highest level of abstraction, AI is software. In that regard, let’s assume we have a system’s logic as $\text{Sys}^{\text{Logic}}$, and the input (structure) that goes with the logic, $\text{Sys}^{\text{Input}2}$. That will constitute a logic-input pair, i.e., $(\text{Sys}^{\text{Logic}}, \text{Sys}^{\text{Input}})$. For a given software system, Sys, we can consider the input structure as a function of the logic, i.e.,

$$\text{Sys}^{\text{Input}} = \tau(\text{Sys}^{\text{Logic}})$$



where τ is the function that translates the logic of system to its input structure.

Given two software designs A and B to solve a problem, we would have their mathematical representation as

$$\begin{aligned} & (\text{Sys}^{\text{Logic}^A}, \text{Sys}^{\text{Input}^A} = \tau(\text{Sys}^{\text{Logic}^A})) \\ & \quad \text{and} \\ & (\text{Sys}^{\text{Logic}^B}, \text{Sys}^{\text{Input}^B} = \tau(\text{Sys}^{\text{Logic}^B})) \end{aligned}$$

respectively. The importance of the representation is that for a given system, there is a correspondence between the software code and the input structure.

Conversely, observe that given an input structure, there is a logical model specification that works with the structure to meet system requirements. In other words, given an input structure $\text{Sys}^{\text{Input}}$, one can obtain the corresponding logic, via the inverse transformation $\text{Sys}^{\text{Logic}} = \tau^{\leftarrow}(\text{Sys}^{\text{Input}})$. This begs the question whether there is a better starting point viz τ^{\leftarrow} and τ . Using a user experience (UX) paradigm (See [2]), the input design/structure should come first. In our context, it is the input structure that should be mapped first, i.e., τ^{\leftarrow} , should be the first focus as it maps the input to the logic. In addition, the exact details of how the input is structured can be abstracted away as well into another layer where emphasis is placed first on what data is required before getting to how (where) it is stored (e.g., txt file, xml, etc). We will consider the coded logic as the fixed part of the system and the input as the variable part. Designing a system where different behavior can be achieved via changes to input enhances flexibility and transparency in the use

of the system. In model building projects for example, there is potential to get unnecessarily held up over choice of methodology but with the appropriate abstraction, the system can be designed to support alternative approaches via the inputs. This effectively defers and delegates the decision on the choice of methodology to the end user.

CLASSICAL AI—ABSTRACTION IN MODEL DESIGN

Designing flexible models is an imperative in the fast-paced world of actuaries these days. This is an area where effective abstractions can be used to enhance flexibility of the system. Another important corollary of the pace of modeling requirements and ERM best practices is the uniformity of models across the enterprise. To achieve this, models should be designed leveraging abstractions that support flexibility³ for different uses/purposes. We illustrate with a relatively simple example. Consider a model that at any point in time evaluates a call option on an index.⁴ Mathematically, the formula for a European call on an underlying S with strike K in the generalized Black-Scholes⁵ model is $C(t, T;)$ given by:

$$C(t, T; \Sigma_{t,T}, K) = S(t)N(d_1) - Ke^{-(T-t)Y(t,T)}N(d_2), \quad (1A)$$

$$\text{where } d_1 = \frac{\ln\left(\frac{S(t)}{K}\right) + \left(Y(t,T) + \frac{1}{2}\Sigma_{t,T}^2\right)(T-t)}{\Sigma_{t,T}\sqrt{T-t}}, \quad d_2 = d_1 - \Sigma_{t,T}\sqrt{T-t} \quad (1B)$$

In the formulae above, $Y(t, T)$ is the yield from time t to T , and $\Sigma_{t,T}$ is the “(implied) volatility” of the forward price of the index. The forward price of an index (underlying) S , is defined as

$$F^S(t, T) := \frac{S(t)}{P(t, T)}$$

where $P(t, T)$ is the price at time t of a zero-coupon bond paying a unit amount of currency at time T defined by $P(t, T) = e^{-(T-t)Y(t, T)}$. This general model is actually more ideal for modeling purposes as the yield curve at any projection time step, t ,

$$Y(t, t+T) \quad T \text{ in } \{T_i\} \text{ i in set of points on yield curve}$$

is typically either an input or internally generated in actuarial models.

Under the classical Black-Scholes model, the formula for a call on the a stock index S is given by

$$C(t, T; \sigma_{t, T}) = S(t)N(d_1) - e^{-(T-t)r}N(d_2) \quad (2A)$$

$$\text{where } d_1 = \frac{\ln\left(\frac{S(t)}{K}\right) + (r_t + \frac{1}{2}\sigma_{t, T}^2)(T-t)}{\sigma_{t, T}\sqrt{T-t}}, \quad d_2 = d_1 - \sigma_{t, T}\sqrt{T-t} \quad (2B)$$

Where the interest rate between t and T is assumed to be the constant short rate, r_t , and the (implied) volatility of the stock index S between t and T is $\sigma_{t, T}$.

Though both models are idealizations of reality, the modeling needn't be held-up over uncertainties/differences of opinion

In model building projects for example, there is potential to get unnecessarily held up over choice of methodology but with the appropriate abstraction, the system can be designed to support alternative approaches. ...

about what to implement. This is because one can find an abstraction that can handle both approaches making any debate essentially irrelevant to model development. To see this, it suffices to note the relationship between the two approaches (or formulae). By inspection of the formulae in (2) and (1), the following relationships can be inferred:

- $Y(t, T) = r_t$ —That is, the generalized model uses the yield between projection time step and maturity, whereas the classic model uses a constant short rate (could be proxied by the short end of the yield curve for example)
- $\Sigma_{t, T} = \sigma_{t, T}$ —That is, for the generalized model, $\Sigma_{t, T}$ is the “implied volatility” of the forward price of the underlying index, (i.e., the variable

$$\frac{S(t)}{P(t, T)}$$

which incorporates the volatility in interest rates), whereas for the classical model, $\sigma_{t, T}$ is the “implied volatility” of the underlying index, $S(t)$.

Consequently, building model components that utilize the following input:

- Projection time, t ;
- Time of call expiry, T ;
- The value of stock index at time projection time step, $S(t)$;
- The strike price, K ;
- Implied volatility parameter—this would be $\Sigma_{t, T}$ for the generalized pricing formula and $\sigma_{t, T}$ for the classical pricing formula; and
- Interest rate parameter—this would be $Y(t, T)$ for the general case and r_t for the classical case,

should be able to accommodate either approach based on the input structure (focus will be on the raw input structure here):

- Have a volatility surface which is a two dimensional matrix structure, $\Sigma\left(\frac{K}{S}, \tau\right)$ where $\frac{K}{S}$ is the money-ness parameter and τ is the time to maturity; and
- At any projection time step have a parameter setting procedure that determines how to source the values:
 - In the case of classical Black-Scholes approach, choose the point on yield curve that will be used as short rate (default to the three-month rate, for example), otherwise, for the generalized case, choose the yield with tenor equal to maturity. If interpolation of the yield curve is required, utilize an interpolation function to do so.
 - For volatility, we would expect the user to enter the correct projected “implied volatility surface” corresponding to the approach desired, i.e., to use the classical paradigm, the input would be the projected surface for the stock index, whereas in the generalized case, it would be that of the forward price of the index.
 - This structure naturally handles instances where the surface is flat along one or both dimensions of (money-ness) and τ (term structure).

From a model configuration perspective, we will expose a configuration/input to the user, e.g., whether to use classical or generalized formula and in the case of the classical, which point on the yield curve to use as proxy for the “constant replicating short rate.” The above approach unifies both methodologies giving the user the flexibility to ultimately develop their assumptions

and corresponding inputs (hence methodology).⁶ A very unproductive approach in our opinion is to create a tailored solution for one case only to later have to “change” it to another case. Consequently, by pushing the decision to the user (via inputs), time as well as energy is saved.

Finally, this example also illustrates how a modeling functionality in particular and models in general can support sundry modeling uses, e.g., pricing, valuation, risk management, etc. In particular, more sophisticated volatility assumptions may be utilized for pricing purposes compared to for valuation purposes and the abstraction handles each approach in the same generic way.

MACHINE LEARNING SYSTEMS

What is machine learning? At the highest level of abstraction, this is a mechanism of creating systems that performs a task through processing of data without being explicitly programmed. The concept is aptly summarized in Mitchell T (1997): “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.”

The experience relates to a data processing step which starts with a data abstraction using a generic vector $x=(x^1, \dots, x^{N_{\text{feat}}})$ where N_{feat} represents the number of features for each data point. For example, to approximate the rate of inflation as a function of the 90-day and the one-year treasury rates, we have $x=(x^1, x^2)$, where x^1, x^2 represents the 90-day and one-year rates respectively.

Next, a helpful abstraction is to consider the different observations of x, with the processing of each observation constituting the experience E. Using subscripts to denote the observation number so that

$$\mathbf{x}_k=(x_k^1, \dots, x_k^{N_{\text{feat}}})$$

represents the kth input observation, an abstract representation of input data to a machine learning system is a matrix (or a table),

$$\mathbf{X}=\begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix}=\begin{pmatrix} x_1^1 & \dots & x_1^{N_{\text{feat}}} \\ \vdots & \ddots & \vdots \\ \dots & \dots & \dots \end{pmatrix}$$

In a supervised machine learning context, another input is the actual values corresponding to each of the input data observations. In our inflation prediction problem, these would correspond to the inflation corresponding to each 90-day and one-year treasury rate observation. We can represent the output a matrix, Y, with N_{obs} rows, each row corresponding to a data observation, i.e.,

$$\mathbf{Y}=\begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{N_{\text{obs}}} \end{pmatrix}=\begin{pmatrix} y_1^1 & \dots & y_1^{N_{\text{out}}} \\ \vdots & \ddots & \vdots \\ y_{N_{\text{obs}}}^1 & \dots & y_{N_{\text{obs}}}^{N_{\text{out}}} \end{pmatrix}$$

where N_{out} is the number of components of the output. In our inflation prediction case, $N_{\text{out}}=1$ and Y is a column vector.

Though machine learning encompasses more than artificial neural networks (ANNs), we will focus on ANNs as it is the approach to AI outside the classical methodology that is inspired (albeit in a very simplified way) on the working of the brain. In that regard, there are different artificial neural network architectures, with a common architecture being the feed-forward architecture. The machine learning problem reduces to finding an approximating function that performs the task T.⁷ There are many libraries that provide implementations for the actual training step which is the iterative estimation of parameters (weights of the neurons in network) of the approximation function. In these settings, the user needs to specify the number of layers in the neural network as well as the number of neurons per layer.⁸

This suggests a data structure of a vector

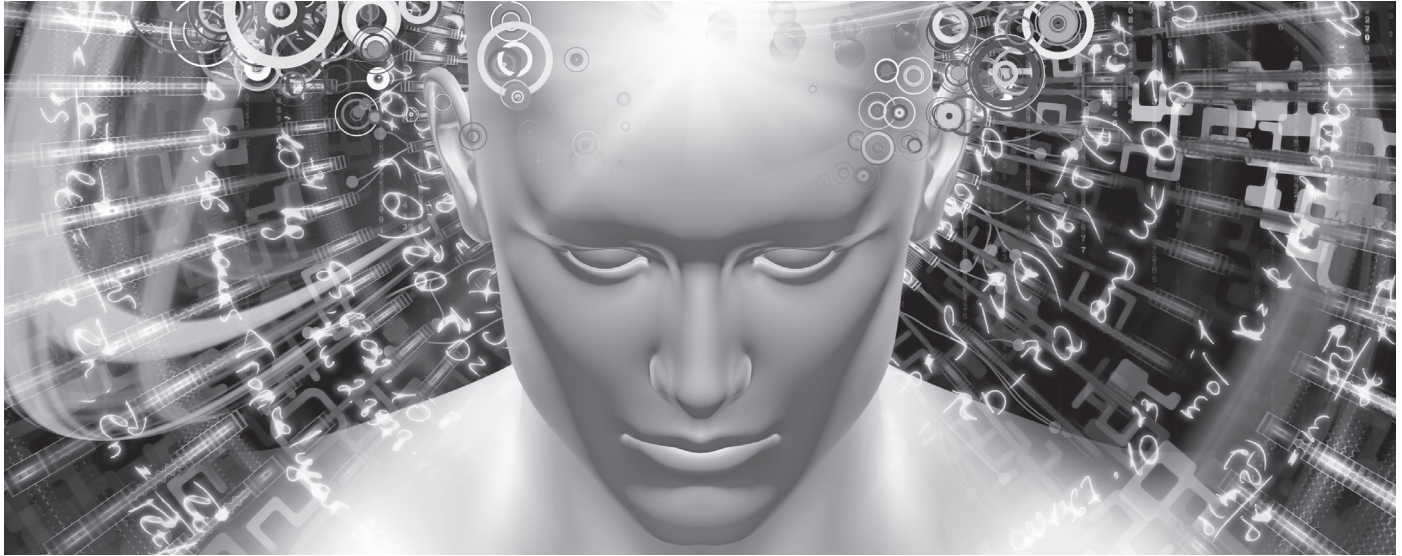
$$\mathbf{n}=(n^1, \dots, n^{N_{\text{layer}}})$$

where n^k represents the number of neurons in the k-th layer and N_{layer} represents the number of layers in the network. With this abstraction we have a blueprint for engineering a neural network system whose configuration is driven by inputs including X, Y and n. In so doing, we have abstracted away the low level details of the heavy lifting that would be carried out by a machine learning engine (e.g., an R package like neuralnet, Python package like scikit-learn, or first principle implementation) and all a user needs to utilize the system would be the data input data structure.

BIOLOGICAL NEURAL NETWORKS/ MACHINE INTELLIGENCE

One observation from the previous section is that abstractions play a key role in the world of traditional ANNs from the generic input structure to the generic processing of each input/observation. In this section we point out the fact that the brain itself is a big abstraction leveraging mechanism.

As described in the article “2036: An Actuarial Odyssey with AI” by Attimu and Robidoux (*Predictive Analytics and Futurism*, July 2016), Machine Intelligence systems attempt to model how the brain works with the Hierarchical Temporal Memory (HTM) framework developed by Jeff Hawkins of numenta.org. As noted in Hawkins, et. al. (2016), Classic AI and ANNs are designed to solve specific problems, e.g., the model component and ANN structure illustrated earlier. The biology of the neocortex, which occupies about 75 percent¹⁰ of the brain’s volume, is the basis of (HTM). Though one could be tempted to think that the neo cortex of the brain has very different algorithms for hearing, vision, touch, and other senses, this is not the case. The brain utilizes common algorithms for vision, hearing, touch, language and behavior.¹¹ Within the context of



this article, we can infer that though the former approaches (classical AI and ANNs) do admit abstractions, these abstractions are not powerful enough to generalize the cognitive processes of the brain. In fact, the brain’s function is probably the best example of the use of abstraction to create a generalized computing framework.

Knowledge representation (representing facts and relationships in the world) is difficult using traditional computing approaches. The brain’s approach to knowledge representation utilizes a data structure called Sparse Distributed Representations (SDRs). The SDR is the perfect example of the brain using data abstraction to abstract different sensory inputs into a common data structure. Just like a computer word, an SDR is made up of 0 and 1 bits. Unlike computer information, which could be represented using 8, 32, or 64 bits and for which semantic meaning of the information is captured in the bit representation as a whole, an SDR is made up of thousands of bits and they are sparsely activated (i.e., a small fraction of the bits are 1s) and each bit contributes to the semantic meaning of the representation. The SDR representation has some very powerful and useful properties including being robust to noise.

To illustrate the difference between sparse and dense representations, consider the ASCII code for the letter ‘x’ which is 01111000. When we flip the 4th digit, we obtain the representation 01101000 which corresponds to the letter ‘h.’ This illustrates the fact that there is no semantic meaning inherent in the individual bits, but in the collection of all the bits. This representation is not robust to noise. On the other hand, consider an SDR representation scheme which consists of 1000 bits of which only 1 percent are 1s. The bits of SDRs carry semantic information. For example, the positions in the SDR could represent different characteristics of class of data represented. To illustrate, consider sound data where bits would capture pitch, amplitude, etc. Furthermore, two SDR’s that

are semantically similar will have overlaps in their “on” (“1”) bits. Consider the information encapsulated by two SDRs shown below:

$$\begin{aligned} \text{SDR}_x &= \underbrace{011101011100010001 \dots 000001}_{1000 \text{ bits}} \\ \text{SDR}_y &= \underbrace{011101011100010000 \dots 100010}_{1000 \text{ bits}} \end{aligned}$$

There is an overlap in position of 80 percent of the “1” bits. Since the individual bits in an SDR have meaning, the x and y are closer semantically than x and another data point, z, for example, whose “1” bit positions overlap with 50 percent of those of x. In fact, SDRs have very important mathematical properties that traditional data structures lack and which make them a particularly powerful abstraction of information for modeling cognitive processes. One important property is their robustness to noise. Indeed, a subset of the on (“1”) bit positions can be used to identify an SDR with high accuracy.¹² For details we refer the reader to Hawkins, et. al. (2016).

We revisit our earlier point about the cognitive (computational) processes in the neocortex being homogenous. The key to learning via the neocortex of the brain is that every sense is responsible for putting its information into a sparse distributed representation (SDR). The SDR must capture the important characteristics for the task. At this point, the brain doesn’t have to worry about what created the data (SDR). It only has to concern itself with recognizing patterns. In effect, the details of the specific types of information input are abstracted away via SDRs. Consequently, a general algorithm can work on different types of cognitive processes e.g., smell, sight, touch, etc. A great example that illustrates this idea is the Brainport which is a sensor that sits upon the top of the tongue and allows blind individuals to “see” using the tongue.¹³

The pattern recognition and learning is done within Hierarchical Temporal Memory (HTM). This is a perfect example of function abstraction in the brain. Each level of the hierarchy works with the SDR data structure and **performs the same learning algorithm**. The difference is the level of the information. Imagine learning a language, you start by learning letters. You then learn words and then finally sentences. Each one of the learning tasks would be at each level in the hierarchy. Abstracting the learning in this way creates a generalized algorithm which reduces the training time and decreases the memory usage compared to using traditional methods of data processing.

The pattern recognition is done by first learning spatial patterns, which constitutes learning bits that often appear together. The temporal patterns learn how the spatial patterns appear throughout time. After these patterns have been learned it can start using them to make inference on new data. These inferences can be used to predict what is likely to occur next. The nice part of this design is that you don't have to separate learning from inference. They feed off of each other with this design. In the Numenta Platform for Intelligent Computing (NUPIC) library, encoders are used to change your data into an SDR. You feed the SDR to the Spatial Pooler and Temporal Pooler to start the learning process.

CONCLUSION

Abstractions are not only a means to create flexible and robust systems; they also help our understanding of concepts and how they relate to each other. Designing software solutions requires the use of appropriate abstractions to make systems both manageable and easy to use. From classical software to more modern AI oriented software, thinking in terms of appropriate abstractions helps engineer more effective solutions to improve the human-machine collaboration we will increasingly see in white-collar work in general and in actuarial work in particular. ■



Dodzi Attimu, FSA, CERA, CFA, MAAA, Ph.D., is director and actuary at Prudential Financial Inc. He can be contacted at dodzi.attimu@prudential.com.



Bryon Robidoux, FSA, is director and actuary, at AIG in Chesterfield, Mo. He can be reached at Bryon.Robidoux@aig.com.

REFERENCES

- 1 Kerievsky J, Refactoring to Patterns, Addison Wesley, 2005
- 2 Esposito D and Saltarello A, Microsoft.NET: architecting Applications for the Enterprise, 2nd Ed, Microsoft Press, 2014.
- 3 <http://dictionary.reference.com/browse/abstraction>
- 4 [https://en.wikipedia.org/wiki/Abstraction_\(computer_science\)](https://en.wikipedia.org/wiki/Abstraction_(computer_science))
- 5 Bjork T, Arbitrage Theory in Continuous Time, 3rd Ed, Oxford Finance, 2009
- 6 Jeff Hawkins, On Intelligence, Times Books, 2005
- 7 Kaplan J, 2015. Humans Need Not Apply: A Guide to Wealth and Work in the Age of Artificial Intelligence, New Haven: Yale University Press
- 8 Awad M and Khana R, Efficient Learning Machines, Apress Open, 2016
- 9 <https://en.wikipedia.org/wiki/Brainport>
- 10 <http://numenta.com/assets/pdf/whitepapers/hierarchical-temporal-memory-cortical-learning-algorithm-0.2.1-en.pdf>
- 11 Mitchell, T. Machine Learning, McGraw Hill, 1997
- 12 Attimu D and Robidoux B, PAF Newsletter, Issue 13, July 2016
- 13 Hawkins, J. et al. 2016. Biological and Machine Intelligence. Release 0.4. Accessed at <http://numenta.com/biological-and-machine-intelligence>
- 14 K. Hornik et al, Multilayer feedforward networks are universal approximators. Neural Networks, 2(5): 359-366, 1989.
- 15 Hagan T et al, Neural Network Design, 2nd Ed, 2016.

ENDNOTES

- 1 The term is borrowed from Kaplan(2015)
- 2 In this article, we consider configurations as inputs
- 3 This is related to the transformation τ^* or τ discussed earlier
- 4 This could be part of an Equity-Indexed Annuity projection engine
- 5 Unlike the classical Black-Scholes model that assumes constant (deterministic) interest rates, the generalized model dispenses of that restriction, being valid under stochastic interest rate assumption. See for example, pages 406-409 in [5]
- 6 It is well known that both approaches are simplifications and may not be appropriate for some modeling situations
- 7 See Hornik et al (1989)
- 8 For an introduction to neural networks, see Hagan et al (2016).
- 9 Other things that might be exposed to input include the performance measure P and other lower level implementation choices that are part of the core machine learning engine API employed
- 10 See Hawkins et al (2016)
- 11 This was first proposed by Vernon Mountcastle in 1979 (See [13])
- 12 In fact the human brain seems to identify entities with just a subset of information e.g. One may be able to identify another's voice even if the voice is in a noisy background.
- 13 See [9]. Note that this example does not explicitly rely on the abstractions of HTM per say, it and is evidence of generality of cognitive algorithms utilized by the brain.

Machine Learning: An Analytical Invitation to Actuaries

By Syed Danish Ali

This post highlights the various value-additions that machine learning can provide to actuaries in their analytical work for insurance companies. As such, a key problem of swapping specific risk for systematic risk in general insurance ratemaking is highlighted along with key solutions and applications of machine learning algorithms to various insurance analytical problems.

“In pricing, are we swapping specific risk for systematic risk?”¹

The hypothesis is that in normal market conditions, premiums are kept at low levels to increase revenues and market share. The traditional approach requires precise figures (point estimates) and so leads to understatement of uncertainty. This keeps a comfort level for us, but the hidden risk of underpricing in our premium estimates is hardly given the attention it merits. This crops up

from the rug it was shrugged in during stressed market conditions when high loss ratios then systematically prove the premium rates to be underpriced and unsustainable.

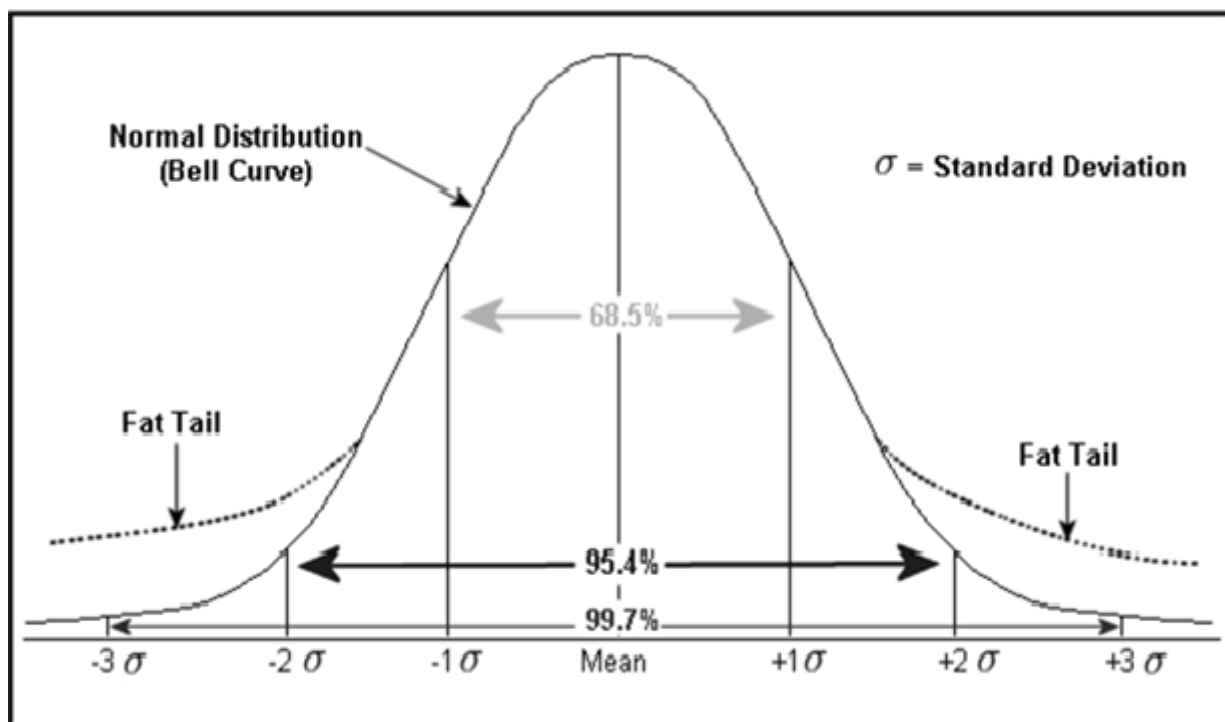
In other words, are we causing the fat tail problem² by our practices? Even if not, what can be done to reduce the fatness of such tails and bring the hidden uncertainties onto the surface explicitly?³

A fat tail exhibits large skew and kurtosis and so there is a higher probability for large losses compared to other distributions like normal distributions as shown by the diagram below. This higher loss tendency remains hidden under normal market conditions only to resurface in times of higher volatility. Complexity scientists call fat tails the signature of unrecognized correlations. Fat tails are an indicator that cascading risks are influencing the probability distribution.

While our discussion does not provide an exhaustive guide to the machine learning tools and algorithms available to the actuary, it provides an outline of them while supplying a context for them in the ratemaking process.

We argue that what was perceived as uncertain can now be made less uncertain with machine learning. Also the uncertainty should be captured from where it was partly generated like risky classes were underwritten which later lead to greater pricing uncertainty and so on.

Machine learning has brought about an explosion of algorithms in recent times. As actuaries are not traditionally trained for



Source: MachineLearningMastery.com



machine learning, and because there are so many algorithms, it can lead to ‘paralysis through analysis’ where one is confounded by so many choices (R’s Caret package of machine learning has 147+models) and decides instead to do nothing but follow previous precedent. The mindmap above, still not exhaustive, made by Jason Brownlee at Machine Learning Mastery highlights a number of diverse classes and subclasses of algorithms and approaches applied in Machine Learning:⁴

Each of these models has a different bias, and hence its own strengths and weaknesses relative to other algorithms and areas of application. It is certainly not possible to discuss many of these algorithms so we will try to stick to “actionable insights” produced from focusing on a small number of relevant algorithms.

With regards to pricing uncertainty and ratemaking applications generally, machine learning can be applied in ratemaking in a number of ways:

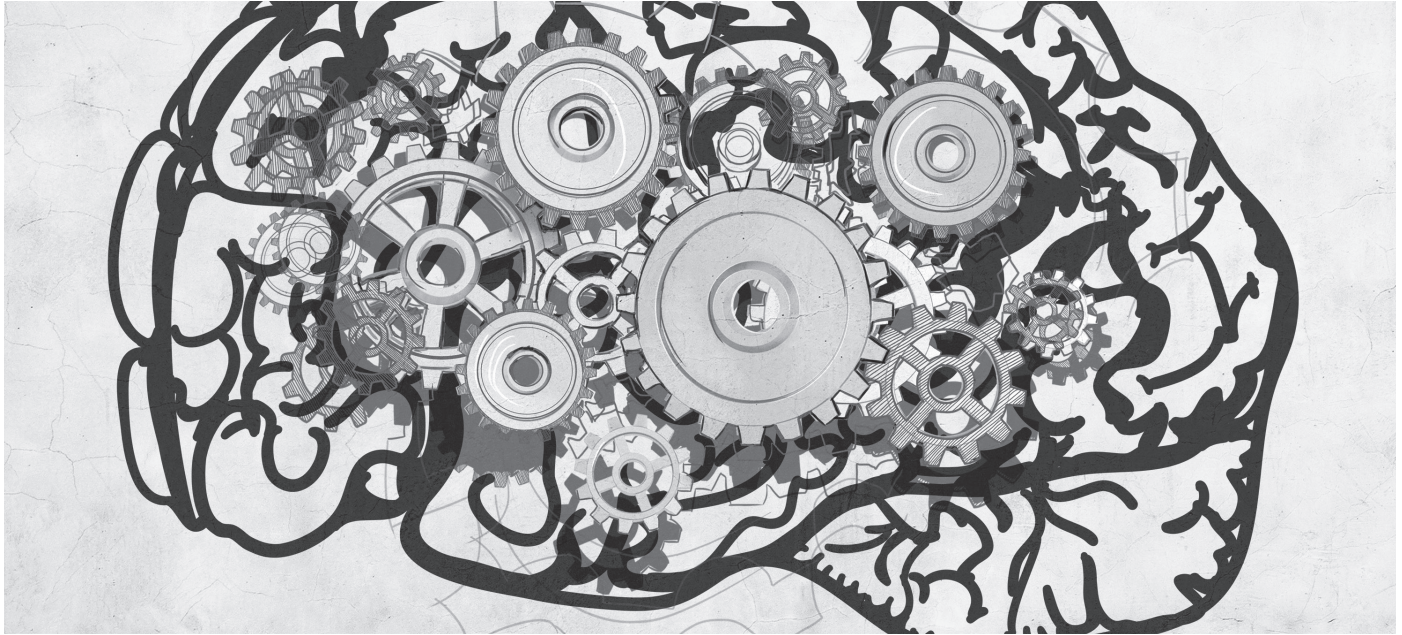
- Exploring our data;
- Predictive modeling; and
- Unstructured data mining and text analytics.

EXPLORING OUR DATA

Decision trees such as hidden decision trees or random forests can allow us to see the map and the critical paths upon which the data is proceeding. Thus, the trend and nature of even huge datasets can be understood through decision trees.⁵ Decision trees are unsupervised methods of learning which means that they expose the trends within the data without relying only on what the analyst is interested in querying.

Clustering, especially K-means clustering, is an imperative algorithm that exposes different clusters operating within a given data.⁶ This can tell us the groupings within claim registers and premium registers like one cluster can be that bodily injuries are associated with third parties that are associated with non-luxury vehicles that are commercial and so on.

For time series decomposition, there are R codes available for running this decomposition algorithm. Basically, decomposition of time series takes a real-data time series and breaks it down into: 1) trend (long term), 2) seasonal (medium term); and 3) random movements.⁷ Such decomposition can have huge potential in understanding trends in data. For instance, claims data have trends



that follow an underwriting cycle and mimic the economic cycle closely. An instance for a seasonal trend can be higher sales of travel insurance in spring break and summer breaks and so on.

PREDICTIVE MODELING

Aside from exploring the data, various uncertain elements of risks can be captured by predictive modeling as well.

Generalized Linear Models (GLMs) can be applied to arrive at a distribution of frequency and severity of claims. Mostly Gamma or Lognormal distributions are fitted to severity data and Poisson or Negative Binomial to frequency. Another approach is to directly apply Tweedie distribution on pure premium.

GLMM is a natural extension to GLM models as the linear predictor now contains random effects as well to incorporate fuzziness and give a stochastic feel for enhanced pricing.⁸

Predictive modeling using GLMs and GLMMs can also be assigned to categorize a particular policy into its proper risk category, like into predictive risk for claim likelihood for a particular policy and so on (unacceptable risk, high risk, medium risk, low risk, etc.). Separate modeling can then be done for each major risk category so as to expose greater insight into the ratemaking process.⁹ The results from the separate models can act as a feedback loop to the risk and underwriting categories of how valid and reliable these categories are, and promote greater cooperation between underwriting function and the claim/reserving function, which is vital to generating adequate risk-adjusted premiums.

While it is important to select optimum risks for predictive modeling and ratemaking on a broad level, it is also vital to take the notion of fairness into account. There have been a couple of backlashes around ratemaking such as a law not allowing the use of gender to quote prices, controversial social images of using credit scores to quote premiums and most recently, pricing optimization where customers and regulators have pointed out that simply market dynamics like price elasticity and consumer preferences should not lead to different premiums and that only risk factors (and not market factors) should lead to premium differentiation.¹⁰

Complexity scientists also favor use of power law distributions, like the Pareto-Levy distribution, for any modeling purpose. This should be tried by the actuary to apply it on severity data and compare its results with other distributions to see if any improvements have been achieved.¹¹

UNSTRUCTURED DATA AND TEXT MINING

It is well known that 80 percent of data is unstructured. Unstructured data is the messy stuff every quantitative analyst tries to traditionally stay away from. It can include images of accidents, text notes of loss adjusters, social media comments, claim documents and review of medical doctors, etc. Unstructured data has massive potential, but has never been traditionally considered as a source of insight before. Deep learning is becoming the method of choice for its exceptional accuracy and capturing capacity for unstructured data. The traditional relational databases use rows and columns in handling data, but NoSQL (Not-Only-SQL) uses a number of other components such as giving unique key or hash tagging to every item in the data. Insurance companies can utilize NoSQL databases like MongoDB, Cloudera and Hadoop

because they capture so many elements of reserving that were deemed belonging to the domain of uncertainty before, as they were too messy and qualitative.¹²

Text mining utilizes a number of algorithms to make linguistic and contextual sense of the data. The usual techniques are text parsing, tagging, flagging and natural language processing.¹³ There is a correlation between unstructured data and text mining as many unstructured data is qualitative free text like loss adjusters' notes, notes in medical claims, underwriters' notes, and critical remarks by claim administration on particular claims and so on. For instance, a sudden surge in homeowners' claims in a particular area might remain a mystery, but through text analytics, it can be seen that they are due to rapid growth in mold in those areas. Another useful instance is utilizing text analytics when lines have little data or are newly introduced, which is our research aim here.¹⁴

Sentiment analysis/opinion mining over expert judgment on level of uncertainty in reserves can also prove fruitful. Natural Language Processing (such as in Stanford 'CoreNLP' software available free for download¹⁵) is a powerful source of making sense out of the texts.

Claim professionals often have more difficulty in assessing loss values associated with claims that are commonly referred to as "creeping cats."¹⁶

These losses typically involve minor soft tissue injuries, which are reserved and handled as such. Initially, these soft tissue claims are viewed as routine. Over time, however, they develop negatively. For example, return-to-work dates get pushed back, stronger pain medication is prescribed, and surgery may take place down the road. Losses initially reserved at \$8,000–\$10,000 then become claims costing \$200,000–\$300,000 or more. Since these claims may develop over an extended time period, they can be difficult to identify. Creeping cat is a big problem for emerging liabilities because mostly, we do not fully know what we are dealing with. Emerging risks like cyber-attacks, terrorism, etc., have shown to have huge creeping cat potential.

As discussed, predictive models can review simulated claim data from agent-based modeling, network theory and other methods mentioned in this report for similarities and other factors shared by such losses, thereby alerting the claims professional to emerging risks that may have creeping cat potential. With this information, strategies and resources can be applied at a point in time where they can be most effective in an effort to achieve the best possible outcome and control cost escalation. Additional loading on premiums can also be given on areas with higher creeping cat potential.

In conclusion, by measuring and exposing areas of uncertainty that are traditionally not considered, we can reduce our chances of swapping specific risk for systematic risk in our ratemaking

procedures and lessen fatness of the tails and handle emerging liabilities in a more resilient manner.

Moving these data collection policies and the uses of this data from the subconscious to our consciousness is a first step in the process of potentially applying big data in a business context. The use of big data and analytics has rapidly evolved from a back-room niche to a strategic core competency.¹⁷

In conclusion, actuaries will have to understand and appreciate the growing use of big data and the potential disruptive impacts on the insurance industry. Actuaries will also need to become more proficient with the underlying technology and tools required to use big data in business processes.¹⁸ ■



Syed Danish Ali is a senior consultant at SIR consultants, a leading actuarial consultancy in the Middle East and South Asia. He can be reached at sd.ali90@gmail.com.

ENDNOTES

- 1 Idea adapted from The Economist "In Plato's Cave"; January 2009.
- 2 Fat Tail: Lexicon of financial times
- 3 Image from advisoranalyst.com available here (<http://advisoranalyst.advisoranalystgr.netdna-cdn.com/wp-content/up...>)
- 4 Jason Brownlee at Machine Learning Mastery; Mindmap of machine learning algorithms
- 5 HR Varian, 2014; The Journal of Economic Perspectives. "Big Data: New Tricks for Econometrics."
- 6 Liu, D. R., Shih, Y.Y., 2005; The Journal of Systems and Software, 77 (2005) 181–191. "Hybrid Approaches to Product Recommendation Based on Customer Lifetime Value and Purchase Preferences"
- 7 Zucchini and Nenadic; R Vignette: Time Series analysis with R—Part I
- 8 University College London; Introduction to GLMM
- 9 Breton and Moore; SOA 14; Predictive Modeling for Actuaries: Predictive Modeling Techniques in Insurance
- 10 CAS Task Force (November 2015): Price Optimization White Paper.
- 11 Smith, L. and Tossani, L.S. CAS; "Applications of Advanced Science in the New Era of Risk Modeling"
- 12 IBM White paper 2013: "Harnessing the Power of Big Data and Analytics for Insurance"
- 13 Stanford Natural Language Processing Group; available at: <http://nlp.stanford.edu/software/>
- 14 CAS Ellingsworth and Balakrishnan: 2008. "Practical Text Mining in Insurance"
- 15 Stanford Natural Language Processing Group; available at: <http://nlp.stanford.edu/software/>
- 16 Lentz, W. GenRe Research (November 2013); "Predictive Modeling—An Overview of Analytics in Claims Management"
- 17 SOA; The Actuary Magazine, December 2013/January 2014 – Volume 10, Issue 6, Ferris, et. al. "Big Data."
- 18 Ibid

Use Tree-based Algorithm for Predictive Modeling in Insurance

By Dihui Lai, Bingfeng Lu

Artificial intelligence (AI) has captured the attention of a broad audience recently. A creative use of an AI algorithm with “big data” could potentially bring revolutionary benefits to many industries. Deep learning, for example, has brought great success in areas like auto-drive, voice/face recognition and the ancient game Go.

Among all the AI algorithms, the decision tree has been widely used for supervised learning and it shows great capability in solving classification and regression problems. The inherent structure of a tree algorithm makes it good for addressing rule-based problems, determining similarities among the objects and classifying groups. In this article, we will overview some popular tree-based models and understand how such models may be applied to the insurance industry.

WHAT IS A TREE-BASED MODEL?

The essences of a tree model, as one may have expected, are roots, branches and leaves. A tree model always starts with a root and grows into branches. Unlike a real tree, a model tree usually only has two splits on each branch. This is also known as a binary tree. The binary structure is powerful yet easy to describe mathematically. A binary tree keeps growing through a series of yes/no questions until the leaves are reached.

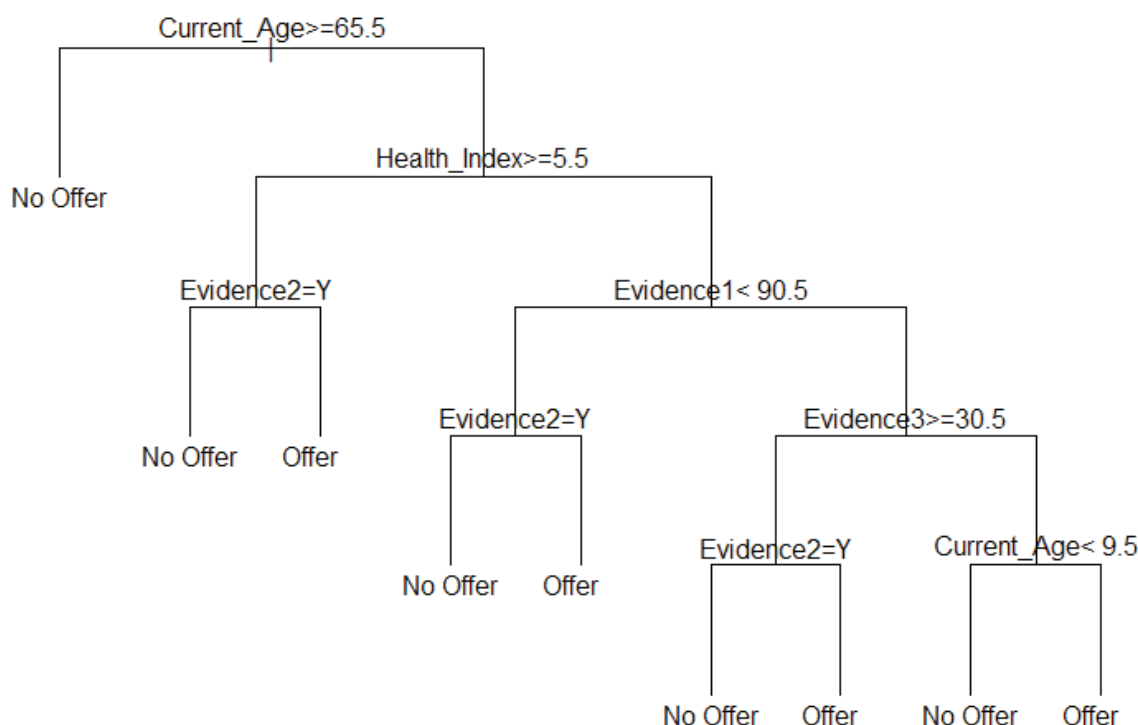
A tree-based model may contain multiple trees and form a tree ensemble. The techniques that are used for growing a tree ensemble include bagging, random forest, gradient boosting, etc.

CLASSIFICATION AND REGRESSION TREE (CART)

The best way to understand a tree algorithm is to begin with the classification and regression tree (CART) model. Not surprisingly, the model is commonly used to solve classification and regression problems. To grow a CART tree, an algorithm automatically figures out at each branch the split that minimizes the overall impurities in the child nodes. The trees keep branching until the leaves are reached. Growing a tree without proper bounding will potentially lead to over-fitting. A technique called pruning is always used to prevent over-fitting.

To understand the power of a CART model, let us look at an upsell problem that a health insurance company is trying to

FIGURE 1





solve. In the scenario, the company tries to offer a certain new insurance product to their existing customers. To determine if an offer can be made, the underwriters have to look through a series of rules and assess the health risks of their customers. For the purpose of demonstration, we built a CART model based on existing customers with decisions made by underwriters and show that the model reproduces the underwriting decision with great accuracy.

Here we only consider a subset of all the rules in the underwriting manual, including only five variables (i.e., age, health index and three other evidence variables). The target variable is the underwriting decision, which can be either “offer” or “no offer.” Without understanding the medical knowledge behind the underwriting rules, the CART model (Figure 1) successfully reproduces the underwriting decisions with an accuracy greater than 99.9 percent.

Looking into the splits that the tree model learned, we find them matching the underwriting rules very well. For example, the top splits correctly state the fact that this product only targets people younger than 66. Besides the age restriction, the model also correctly learned more complex rules that are combinations of age, health index and a series of other evidences.

Figure 1: A CART tree for predicting the underwriting decision of a health product, based on age, health index and three other evidence variables. For each split, the model will go to the left branch if the label is true and to the right branch otherwise. For a certain input, the model

tree keeps making decisions upon splits until a leaf, either “Offer” or “No Offer,” is reached.

As a comparison, we also built a logistic regression model for the same data set. The regression model has a slightly worse prediction accuracy of 95 percent. It is not surprising as the underwriting decision is made by answering a series of yes or no rules, which fits into the inherent structure of a tree model better than a regression model.

THE LIMIT OF CART MODEL

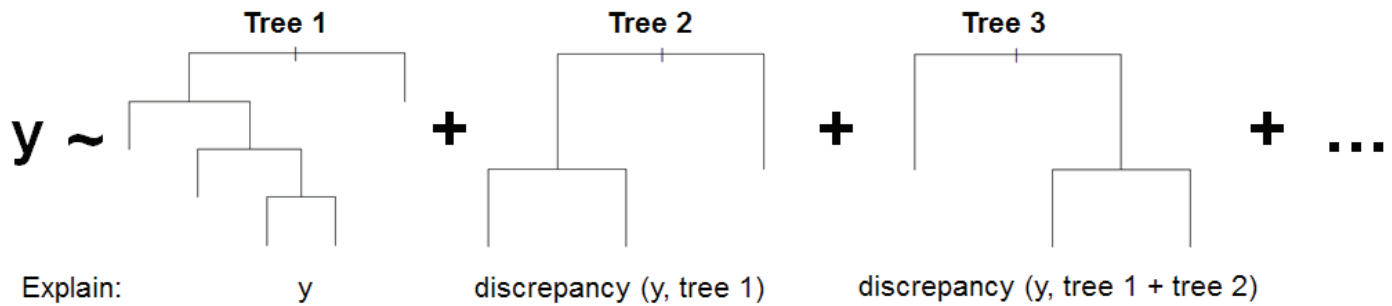
Apparently, the example above is oversimplified. A real underwriting manual considers far more than just five factors. Moreover, there could be hundreds of factors that determine a person’s health situation and a lot of them may not even be known to us. For example, the contributing factors to cancer could be aging, tobacco, sun exposure, radiation exposure, chemicals, viruses, bacteria, etc. However, there is still a 10 percent to 30 percent chance¹ that a person developed cancer due to “bad luck” (the unknown). Moreover, the occurrence of an event might result from complex intermingles of various factors that do not fit into a yes/no structure.

In facing complex problems, the performance of a CART model is not always satisfactory. More sophisticated approaches are often needed to tackle a real world problem.

TREE ENSEMBLE

Random Forest: The occurrence of an event might be due to numerous factors in a complex way. If one single tree cannot

FIGURE 2



handle the complexity, would a number of them do? The answer is often yes. In a random forest algorithm, all trees are grown in parallel to predict the same target, but each tree is only provided with a subset of all information available. It may sound a little counter-intuitive that the algorithm is trying to build a better model with less information. In fact, although each single tree grown this way is a weaker predictor by itself, the final decision that is made through a voting mechanism from the tree crowd normally ends up better.²

Boosted Tree is another popular way of growing a tree ensemble. Unlike Random Forest where all trees are grown to predict the same target, boosting algorithms approach the target sequentially. Specifically, the algorithm starts by growing a simple base tree that tries to make a good approximation of the target function. It is fine if the base tree cannot make accurate predictions as a subsequent tree will grow to make improvement on top of the existing one. The variation that cannot be explained by the base tree will be the target of the second tree (Figure 2). If discrepancy between the trees' prediction and the target remains, a third tree will grow. The process continues iteratively until a converging point is reached. Loosely speaking, the algorithm grows trees one-by-one and each tree is grown to correct the error that results from its precedents.

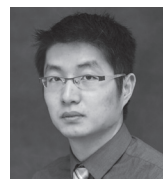
Boosted Tree often has a great performance accuracy³ as it can capture complex nonlinear patterns effectively. However, an intuitive interpretation of the trees grown from boosting algorithms is usually hard because the trees (except the base tree) are not actually predicting the target directly.

Figure 2: A schematic draw for Boosted Tree Algorithm. A base tree (Tree 1) is grown targeting on the objective function. A following tree (Tree 2, Tree 3) is then grown to explain the discrepancy between the target and the existing trees. The process continues until a converging point is reached.

WHICH ALGORITHM TO CHOOSE?

With all the tree and forest descriptions, it is probably a good

time to make a decision on which algorithm to pick. Should one use a single tree, a forest of trees, or other AIs? As we have demonstrated in our CART model, a tree algorithm is generally good at reproducing a system that is designed upon discrete rules (e.g., underwriting decisions), especially if the rules follow a binary structure. The tree algorithm naturally puts data into blocks and is therefore ideal for business problems like segmentations, categorization, etc. When you wish to solve problems that consist mainly of continuous changes (e.g., mortality risks, claim incident), algorithms like GLM (generalized linear model), or survival analysis might be better choices. Neural network based algorithms like deep learning are normally good at solving problems that involve image processing, handwriting recognition, face recognition, etc., which are not often confronted in insurance. ■



Dihui Lai, Ph.D., is a data scientist at RGA Reinsurance Company in Chesterfield, Mo. He can be reached at dlai@rgare.com.



Bingfeng Lu is an assistant data scientist at RGA Reinsurance Company in Chesterfield, Mo. He can be reached at blu@rgare.com.

REFERENCES

1. Song Wu, Scott Powers, Wei Zhu and Yusuf A. Hannun, "Substantial contribution of extrinsic risk factors to cancer development," *Nature*, 529, no. 7584 (2016).
2. Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, "An Introduction to Statistical Learning with Applications in R," Springer, (2015), pgs. 319-20.
3. Trevor Hastie, Robert Tibshirani and Jerome Friedman, "The Elements of Statistical Learning Data Mining, Inference, and Prediction," Springer, (2013), pgs. 589-91

SOA Professionalism Ready-to-Go Kit

Host a Top-Notch Professionalism Workshop for Your Employees (Without Leaving the Office)

Check out the Professionalism Ready-to-Go Kit:

- Includes a facilitator guide, logistics guide, slide presentation and participant guide
- Participants may attain Continuing Professional Development credits
- Uses real-life scenarios and provides opportunities for group discussion
- Ideal for 20-30 employees of any level
- At \$500, it is an excellent value

Learn more at [SOA.org/ReadyToGoKit](https://www.soa.org/ReadyToGoKit).



Creating a Useful Training Data Set for Predictive Modeling

By Anders Larson

In the past decade, the capabilities of predictive analytics have improved dramatically thanks to greater availability of large data sources, increased computing power, and innovation from the statisticians, data scientists, and actuaries at the forefront of the field. As a result, there has been more and more interest from companies across nearly every industry to harness the power of machine learning and other advanced predictive modeling techniques.

For all the advancements that have been made, the ability to produce useful and accurate results with any of these techniques is still ultimately reliant on one thing: robust and appropriate data with which to train the model. This goes beyond the simple “garbage in, garbage out” principle. There’s no doubt that data with blatantly incorrect or sparsely populated information won’t do us much good in building a predictive model. It should go without saying that data cleaning is an essential step in the model building process.

We could train a model with an immaculately clean data set with 500 million records and 100 variables, then use that model to make predictions using an equally clean data set with the exact same set of 100 variables, and we could still end up with awful predictions if the model is based on faulty assumptions. In fact, this is perhaps one of the most dangerous situations, when it seems for all the world like we have a model we can trust, and so we do trust it, until it’s too late, when it becomes clear that our predictions were just ... bad.

One of the most important elements of a useful training data set is that it is a reasonable representation of the data we’ll be using to make predictions about the future. In general, we want the same data generating process underlying the training data to plausibly apply to any new data fed to the model when making predictions. Even with clean data, there are often subtle biases in training data that can cause us to build a model that is inappropriate to apply to new data. To help provide more clarity, I’m going to describe a few specific examples in more detail. What I hope to accomplish here is to heighten awareness, so that, the next time you begin building a training data set, you can be on the lookout for the dangers that might be hiding in the data. I

apologize in advance for my own not-so-subtle bias toward examples from the health care world.

LIMITATIONS ON PATIENTS WITH CLAIMS HISTORIES

Accountable care organizations (ACOs) participating in the Medicare Shared Savings Program (MSSP) receive full claims detail from the Centers for Medicare and Medicaid Services (CMS) on all of the patients assigned to it. In an ACO’s first year, CMS provides claims histories for all currently assigned patients, extending back one year prior to the start of the ACO. The data is clean, consistent, and reliable, and given that all ACOs should have at least 5,000 patients, there are plenty of observations with which to train a predictive model. However, there is one catch: CMS does not provide any claims history for patients who died prior to the start of the ACO’s first year (decedents).

For most Medicare ACOs, approximately 5 percent of patients alive at the start of the year will die by the end of the year, and

DEFINING KEY TERMS

Training data set—The set of records used to calibrate a predictive model and determine relationships between characteristics (features) and a particular outcome (response). The data should be divided into two subsets, often based on a time or date variable. The two subsets are:

- **Feature:** The data used to gather characteristics that will be used as features in the model. For instance, in a model that predicts health care costs, this time period would be used to determine things like clinical conditions, historical costs, and historical utilization of services. In a standard, prospective predictive model, this data would come from an earlier time period than the response set described below.
- **Response:** The data used to observe the outcome you are looking to predict. For purposes of calibrating the model, this data set should not be used to determine any of the characteristics that will be used as features. In a standard, prospective predictive model, this data would come from a later time period than the feature set described above.

Prediction data set: The set of records used to make new predictions using the model calibrated on the training data set. This data set is similar to the training feature data, but likely for a more recent time period (often the most recent time period). It is used to gather the same characteristics as the feature portion of the training data above. ■



these patients generally incur very high costs in the last few months of life.¹ When constructing a training data set, it would make sense to remove patients who died during the feature period, because there would be no need to predict their future costs. However, you would want to include patients who died during the response period, because it is likely that some patients for whom you will make predictions will die in the predictions period.

Unfortunately, in this situation we have no decedents available in the training response period, which creates a bias in the training data set. The patients selected for inclusion in the training data set are, on average, healthier and lower-cost than the patients for whom you will be making predictions. This is true even after accounting for other patient characteristics, such as the presence of chronic conditions. The average patient with congestive heart failure who does not die in the next six months is still much less costly than the average patient with congestive heart failure who does die in the next six months.

As a result, the predicted costs will be understated for a model trained on this data set. Communicating the limitations of the model to the end user will be particularly important in this situation (and complicated). The predictive models can still be quite useful, as long as the focus is more on the rank order of the predicted costs rather than the specific level of predicted costs. Conversely, using this model to predict population-level costs would likely be inappropriate. Ultimately, as time passes, the ACO will receive enough data on patients who die after the start of the performance year, and a more appropriate model can be retrained.

DIFFICULTIES WITH NEW AND EXPANDING POPULATIONS

The ever-evolving health care landscape in the United States presents opportunities for predictive modelers, but not without additional challenges. One situation that can be particularly tricky is when a new class of patient is introduced into a population. The expansion of the Medicaid program in many states is a particularly instructive example.

In most states, the Medicaid population prior to 2014 was comprised of disabled adults and low-income families and children. Under the Patient Protection and Affordable Care Act (ACA), states are encouraged to extend eligibility to low-income adults who did not otherwise qualify. The morbidity levels of these newly eligible patients was a huge unknown prior to the start of the program because many of these patients had been previously uninsured.

But let's take a step forward and look at the situation even after a year has passed since the expansion of Medicaid in a particular state. Assume you are constructing a predictive model to predict individual and population-level costs for a managed Medicaid plan. You can use the past year of history to build a training data set, but that training data set may have biases built into it. All of the patients who were newly eligible for Medicaid did not enroll immediately, and those who do enroll right away may not be representative of the type of patients who will ultimately enroll (for instance, the early enrollees may have higher morbidity on average or have pent-up demand for services).

Compounding this problem is the fact that there was a relatively small number of these patients in the training data set, and yet

The ability to produce useful and accurate results with any of these techniques is still ultimately reliant on one thing: robust and appropriate data with which to train the model. This goes beyond the simple “garbage in, garbage out” principle.

the population now represents a much larger portion of the prediction data set. The predictive model you construct is going to extrapolate the learnings from that small sample to make predictions for a much larger group, which will exacerbate any biases you have in your training data set. As actuaries, exercising judgment in this situation is essential. Placing too much trust in a model that is not necessarily aware of outside influences, such as pent-up demand, is a serious risk. One option in this case would be to look at outcomes in other states that expanded Medicaid previously. When viewed at a population level, do the results of your predictive model look reasonable compared with experience elsewhere?

CHALLENGES WITH TRANSACTIONAL DATA

The challenge of creating a training data set becomes more complicated when dealing with transactional data. In the examples above, we have a way to measure the number of patients (or more generally, the exposure units) that are associated with the claims or other utilization measures that occur. In some cases, data is simply provided about transactions (claims, services, payments, etc.) as they occur, with no corresponding information about which patients were eligible to have these transactions.

Using enrollment or exposure information, rather than claims information, to select patients for the training data set generally makes the most sense. This enables an understanding of the difference between patients who could have used services and patients who did not use any services because they were not eligible (or were not included in the original data). With transactional data, such as an electronic health record, the predictive modeler often has to make an attempt to infer some type of exposure metric. One option is to look for the first date that a patient appeared in the electronic health record and assume that the patient was “eligible” to receive services from that point forward.

These inferred enrollment estimates will also be needed to select patients for inclusion in the training data set. In these situations, particular caution must be used to avoid biasing the training data set. In general, it is dangerous to use anything learned in the training response period to determine which records to include

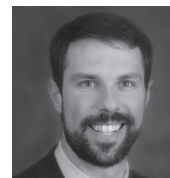
in the training data set. Rather, pretend that your response period is truly unknown, just like the future you’re trying to predict. In this example, our best approach would be to include patients whose inferred enrollment began prior to the start of the training response period and were therefore “eligible” to receive services in the training response period.

This approach will still yield less-than-perfect results. For instance, assume the data provided to you includes all services in the past 24 months. Then assume there is a patient who had only one service in the data, and it occurred three months ago, but this patient also had a service 30 months ago, which is not in the data. Let’s say you are training the model to predict services over a 12-month span, so you set the training response period to begin 12 months ago. This patient would be excluded from the training data set because it was assumed that person was not eligible to receive a service. Had the data been cut six months earlier, you would have observed that initial visit 30 months ago, and the patient would have been included in the training data set with no services in the training response period.

Unfortunately, there is no magic bullet for handling transactional data with no exposure information. Getting a thorough understanding of the data-generating process underlying the data will help, but it is critical to be aware of the limitations and potential uncertainty of a model built on this type of data.

CONCLUSION

The examples in this article are by no means exhaustive. Every predictive modeling scenario has its own unique challenges, and arguably it’s never possible to put together a training data set that is a perfect representation of the prediction data set. But taking care to create a useful and appropriate training data set is an often underappreciated step in the predictive modeling process. There’s no question that expertise in selecting and calibrating the model itself is a vital skill, as is the ability to communicate and interpret the results, but any model will be imperiled from the start without a solid understanding of the data used to train it. ■



Anders Larson, FSA, MAAA is an actuary at Milliman in Indianapolis. He can be reached at Anders.larson@milliman.com.

ENDNOTES

- 1 Riley, G.F. & Lubitz, J.D. (April 2010). Long-Term Trends in Medicare Payments in the Last Year of Life. Health Services Research. Retrieved September 2, 2016, from <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2838161/>.

The Random GLM Algorithm: A Better Ensemble?

By Michael Niemerg

The random generalized linear model (RGLM) is a predictive algorithm based upon the idea of putting linear models in an ensemble. It does this by taking some of the features of random forests—randomization, bagging—and applies them, as the name implies, to generalized linear models. This is a seductive premise, but does it make for a competitive algorithm?

Before introducing the RGLM algorithm in more depth, let's talk about its two closely related algorithms: linear models and random forests. This will give us some background to understand both how the RGLM is put together and give us some intuition on how the algorithm might or might not be a good predictor. I will also use random forest as a basis of comparison when I test the model out later on some sample datasets.

For the most part, RGLM is using the linear regression we all know and love (more on how it does this later). It also allows generalized linear models of the logistic, multinomial, and Poisson variety. These allow us to model binary classification, classification, and counts in linear regression form respectively. Linear models have many advantages including ease of interpretability and use, fast training time, and overall versatility. For linear models, variable selection, interactions, and higher-order effects should be considered in the model-building process. Two possible ways to do this could be manually by looking at regression statistics and using good judgement or through stepwise selection procedures that attempt to do so in a more automated fashion through an iterative approach of adding and/or removing variables depending on how they improve a statistical measure. As we will see later, stepwise selection procedures will prove foundational to how RGLM is constructed.

Next up, a quick overview of random forests. Random forests are an ensemble model based upon decision trees. The random forest algorithm involves growing a “forest” of many independent decision trees where each decision tree is based upon bootstrapping (independent sampling with replacement) the dataset being modeled. Additionally, when building the decision trees, each candidate split is based upon a random subset of predictors. Once all the decision trees are created, they are then aggregated via majority voting (classification) or averaged (regression) to get a single prediction.



Random forests have many advantages as an algorithm: they can handle both classification and regression, can be trained rapidly, require modest amounts of model tuning, and do a good job of handling nonlinear interactions. Overall, the combination of these characteristics of random forests make it a powerful algorithm. One of the biggest drawbacks is that while an individual decision tree is easy to interpret, when you aggregate many of them in a random forest you lose that interpretability. However, random forests are still able to give you some insight into their inner workings through variable importance measures.

Now, let's shift focus to RGLM itself. In a sense, RGLM is a cross-breed between GLMs and random forests. Like random forests, the ultimate model is an ensemble. However, it's trying to take the advantages of random forests and apply them using linear models. That is, each base learner that makes up the ensemble in RGLM is a regression, not a decision tree. Like a random forest, however, RGLM still builds its component models from bootstrapping and by using a randomized subset of features in each base learner.

Each one of the base learners is built as follows: 1) A bootstrap sample is selected from the dataset; 2) A randomized set of predictors get selected; 3) The predictors get ranked according to their association with the variable being predicted; 4) The highest ranked predictors become candidates for selection in a regression model; and 5) Stepwise regression (specifically, forward selection) is applied to create a linear model.

For a given model, somewhere between a dozen and several hundred base learners are built. Since each one is built on a different sampling of the data and using a different set of candidate predictors, each one will be unique. Once each of the base models is created, they are then combined into the final ensemble model used for prediction using either averaging for regression or majority voting for classification.

The intuition here is that by randomly sampling both the datasets and the predictors, the ensemble model is more powerful than a single model could be. Ideally, an ensemble allows for a good deal of flexibility in the model it creates, but avoids overfitting since the noise tends to get washed out among the different models. However, RGLM is going to have a challenge in that an ensemble can only be as flexible as its base learner and linear regression base learners can only take this so far.

The reason for this limitation is that the best model involves a trade-off between bias and variability and linear regression is a high bias, low variance procedure whereas ensembles benefit most from base learners that are the opposite. We need low bias base learners in our ensemble so that the resulting model has low bias. We need the “flexibility” that often comes from high variance base learners to ensure that the ensemble is capturing all the signal in the data. Any amount of overfitting caused by high variance within the base learners gets muffled by averaging them, so that the final model will actually have low variance.

Because RGLM has a linear regression base learner, extreme outliers could still dominate even an ensemble while “wigglyness” and complex relationships in the data could be hard for RGLM to capture. See, for example, the following charts which show a dataset that is very linear except with a highly nonlinear subregion (Figure 1). A fitted linear regression produced by RGLM (which is virtually identical to the regression model that would be produced with an ordinary GLM) is shown in orange while the random forest is shown in gray (Figure 2). While this example is obviously contrived, it conveys the difference in flexibility between the two models. The random forest is able to identify the anomalous subregion and is able to average out the behavior in that vicinity without much ado while the entire intercept of the linear regression gets thrown off by those points (see how its prediction line hovers above most of the data points). The predictions are simply too high everywhere, except in the anomalous region where the model prediction is too low.

FIGURE 1: TEST DATASET

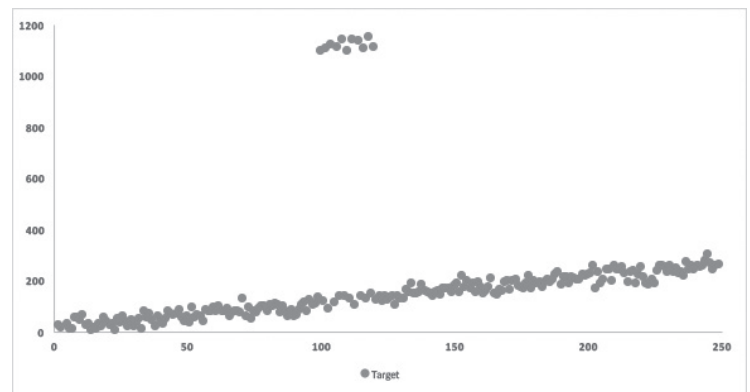
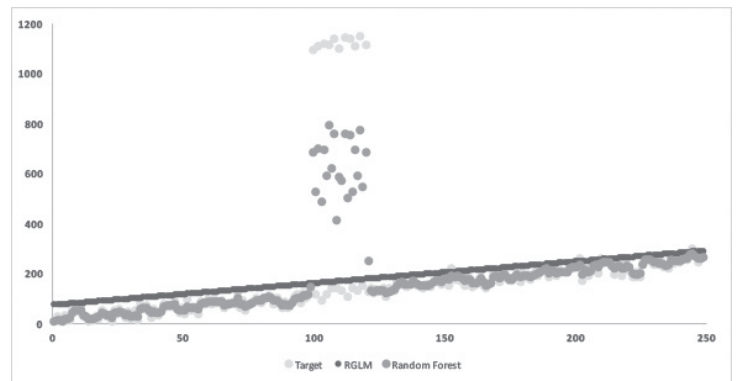


FIGURE 2: TEST DATASET WITH RGLM & RANDOM FOREST PREDICTION



All right, now that we’ve built some intuition on how RGLM works, let’s take it out for a test drive. I chose ten datasets: four suited for logistic regression and six for linear regression.

To summarize the predictive accuracy for logistic regression, I used area under the curve (AUC). AUC is a common measurement when comparing binary classification models. For any model that can return probabilistic output, a receiver operating characteristic (ROC) curve can be constructed. An ROC curve graphs the true positive rate vs. the false positive rate at the possible thresholds for classifying an observation. The AUC is then a measurement of the area under this curve. The higher the AUC the better. To compare predictive accuracy for linear regression datasets, R-squared was used. R-squared is a measure of the strength of linear association between two variables, again the higher the better. I will be calculating these statistics for each dataset on a testing set that was withheld from the model fitting process so that we can get a good sense of model generalizability on new data.

It turns out that random forest outperforms RGLM in the datasets I chose, sometimes by a wide margin. The difference varies

The results, as well as a description of the datasets used, are shown in the following table:

Dataset	Model Type	Observations	Predictors	RANDOM FOREST		RGLM	
				RunTime (seconds)	Evaluation Metric*	RunTime (seconds)	Evaluation Metric*
1	Classification	45000	16	8.15	92.4%	402.22	81.2%
2	Classification	27000	10	2.17	87.1%	142.64	79.6%
3	Classification	27000	100	19.92	88.5%	4463.66	76.2%
4	Classification	800	8	0.05	80.1%	44.98	79.5%
5	Regression	1500	5	0.16	76.9%	5.12	56.9%
6	Regression	9000	12	4.83	99.8%	55.96	93.7%
7	Regression	6000	19	0.64	69.7%	23.18	14.8%
8	Regression	1000	8	0.11	87.4%	5.91	51.0%
9	Regression	10000	4	1.58	99.3%	18.57	100.0%
10	Regression	5000	11	1.61	48.7%	27.39	19.2%

* The evaluation Metric is AUC for classification and R-squared for regression

by dataset with RGLM coming in very competitively for some of the datasets and random forest coming in as an easy winner on others. One RGLM, trained on dataset 9, just barely beat the random forest, and its accuracy rounded up to 100 percent. In terms of computation time, random forest was a clear winner across the board, being an order of magnitude faster to train.

One thing to note is that I tried various configurations of RGLM. The above chart doesn't represent the absolute best I was able to get out of RGLM, but involves a compromise of predictive accuracy and runtime (the parameters are 50 base learners, interactions up to level two, and with each model considering half of its predictors in the base learner as a candidate for forward selection). Some further optimization is able to close some of the gap—and RGLM was able to ever so slightly outperform random forest on another dataset with some additional model tuning—but the overall gap remains. I wasn't able to come up with any configuration that made RGLM the clear winner over random forest. Meanwhile, I didn't do any tuning or optimization to the random forest (I simply started with 40 trees and default settings). Furthermore, parameter tuning can only get you so far with RGLM. There aren't that many parameters to tune to begin with. At the end of the day, I think the true drawback is that RGLM is still a linear model which means that it is somewhat limited in its ability to capture highly nonlinear interactions, as mentioned earlier.

To elaborate on that point some more: ultimately, any RGLM ensemble is really just a linear model itself (a combination of linear models is itself a linear model) and so it is inherently limited by all the things that linear models are limited by. In fact, RGLM is probably even more limited to the extent that it is simpler to add higher-order terms and transformations onto a typical linear model than to an RGLM.

One caveat I feel compelled to mention: a truly fair and robust comparison would require a larger sample of datasets. In fact, a much more robust comparison of RGLM to other methods was performed in the paper by Song and Horvath this article was based upon (see the references at the end). In their results, the creators of RGLM were able to get superior performance on RGLM even when comparing it to many of the most common algorithms used today in predictive analytics.

Overall, I haven't seen RGLM used much in practice. Based on Song and Horvath, it seems it can offer superior performance on some datasets, but I'm skeptical of its ability to do so reliably on a wide range of applications. Also, due to its rather lengthy computation time, I'd be hard-pressed to recommend it as an all-purpose algorithm. It's an interesting concept, but I can't help but think it needs some alterations—some way to be a little more like a random forest, some way to alter its base learners to add flexibility, capture nonlinear features, and better benefit from the ensemble approach—before being able to be a top tier contender. ■



Michael Niemerg, FSA, MAAA, is an actuary at Milliman in Chicago. He can be reached at michael.niemerg@milliman.com.

REFERENCES:

Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning* (2nd ed.). Springer. ISBN 0-387-95284-5.

Song L, Langfelder P, Horvath S (2013) Random generalized linear model: a highly accurate and interpretable ensemble predictor. *BMC Bioinformatics* 14:5 PMID: 23323760 DOI: 10.1186/1471-2105-14-5.

Collaborative Filtering for Medical Conditions

By Shea Parkes and Ben Copeland

Recent trends in health care legislation have led to a rise in risk-bearing health care provider organizations, such as accountable care organizations (ACOs). Entrusted with the care of thousands of patients, these organizations must leverage data-driven approaches to population health management in order to improve quality of care and reduce costs.

One area of concern for data-driven analysis involves the accuracy of a patient's clinical documentation. Efforts to improve accuracy in a population's clinical records are often referred to as clinical documentation improvement or coding improvement. From a clinical standpoint, the benefit from coding improvement is obvious. A patient record that contains the entirety of the patient's illnesses will result in a more appropriate treatment plan.

However, there can be financial incentives in coding improvement. Alternative payment models often account for the health status of a patient population, through the use of risk scores, when reimbursing a health care provider for services. A more accurate clinical record ensures that risk-bearing health care providers are appropriately compensated when they care for sicker or healthier populations.

Coding improvement initiatives often start by looking through a given patient's records for explicit evidence of conditions that did not make it into the official diagnosis information: conditions coded on claims in prior years, or mentioned in the unstructured text of an electronic medical record. After these explicit sources of coding improvement are exhausted, more analytical methods can evaluate a patient for comorbidities to consider adding (or removing). One approach is to find explicit evidence of missed codings in large reference data sets and train predictive models that can be then be applied to other, potentially slimmer sources. This can work well for predicting specific chronic conditions in a population, even when only a short claims history is available.

Collaborative filtering can provide a different approach to identifying uncoded conditions by identifying common clinical patterns among patients in a population. Analysts can then make patient-level lists of conditions to review based upon comorbidities experienced by similar patients. The collaborative filtering



approach works well at giving personalized lists of potential comorbid conditions from the patient perspective.

WHAT IS A COLLABORATIVE FILTERING SYSTEM?

If you have ever viewed a product on Amazon or watched a show on Netflix, then you have been a part of a collaborative filtering system, also known as a recommender system. Collaborative filtering systems are commonly applied to help users identify potentially interesting products among a large list of options, through the use of historical viewing or rating information. For example, Netflix will recommend certain shows to you based on your previous viewings. These recommendations are built using viewing or rating data from other users who have viewed the same shows as you.

Collaborative filtering often takes three forms: user-based, item-based, or matrix factorization. User-based collaborative filtering seeks to find users that have rated items similarly, and predict preferences for other items that similar users liked. Item-based collaborative filtering seeks to find similarities among items themselves, and then suggest items that are similar to a user's highly rated items. Matrix factorization estimates latent factors for each user and item and then uses these latent factors to find items that hopefully align with a user's preferences.

For an illustration of collaborative filtering in a clinical setting, consider the hypothetical patient panel in Figure 1.

FIGURE 1: EXAMPLE PATIENT PANEL

Condition	Patient 1	Patient 2	Patient 3	Patient 4
Diabetes	X	X		
Hypertension	X	X		
Coronary Artery Disease			X	
Hyperlipidemia		X	X	X
COPD			X	X

Patient 1 appears to be most similar to Patient 2. Thus, for Patient 1, hyperlipidemia might be considered as a potential comorbidity. Likewise, Patient 4 is most similar to Patient 3, so coronary artery disease might be considered as a potential comorbidity.

FIGURE 2: EXAMPLE PATIENT PANEL, CONDITIONS TO CONSIDER

Condition	Patient 1	Patient 2	Patient 3	Patient 4
Diabetes	X	X		
Hypertension	X	X		
Coronary Artery Disease			X	O
Hyperlipidemia	O	X	X	X
COPD			X	X

The preference inputs in collaborative filtering may take two forms: explicit ratings or implicit ratings. Explicit ratings are generated when the users themselves identify their preferences, such as giving a rating to a movie or a product. While explicit ratings carry a higher level of confidence for a user’s preference, they are often not available. More commonly, implicit ratings are inferred from a user’s actions, such as viewing a movie or buying a product.

The implementation explored in this article utilizes an implicit rating, matrix-factorization model to identify relative likelihood ratings for uncoded conditions. Each patient is a “user,” with potential comorbid conditions being suggested as the “items.” Implicit condition confidence values, or ratings, are inferred

FIGURE 3: CONDITION RATINGS BASED ON ESTIMATED LATENT FACTORS

Latent Factor	Patient	Diabetes	Hypertension	COPD	Menopause
1	0.8	0.2	0.3	0.1	-1.0
2	0.4	0.6	0.8	0.1	0.1
3	-0.5	0.1	-0.1	-0.1	0.1
4	0.6	-0.2	0.2	0.5	-0.1
Patient Rating	---	0.23	0.73	0.47	-0.87

from the medical history of each patient in a population. These patient, condition, and confidence inputs are processed to generate latent factors for each patient and condition. These latent factors, an abstract representation of similarities among patients and conditions, can be combined to generate a rating for each patient-condition pairing.

The hypothetical example in Figure 3 illustrates using the estimated latent factors to generate condition ratings for a single patient.

A condition’s rating for a given patient is calculated as the dot product of the patient’s latent factors and the respective condition’s latent factors (e.g., Diabetes Rating = $0.8 \times 0.2 + 0.4 \times 0.6 + -0.5 \times 0.1 + 0.6 \times -0.2$). Here, hypertension would be identified as the most likely potential comorbidity to consider. While latent factors are not easily interpretable, one could roughly associate each latent factor with a patient characteristic. Latent factor 1 could be gender-related because it has a strong coefficient for menopause. Latent factor 2 may be related to blood pressure, considering the high coefficients of both diabetes and hypertension, while latent factor 4 may be related to lung issues. Most real matrix factorization models use so many latent factors it would not be reasonable to try to actually attach interpretations to them.

A matrix factorization approach provides some useful benefits. The model is fast and simple to train, and thus can realistically be tuned to find unique relationships for each patient population. There are implementations available in cluster computing frameworks that gain additional speed by distributing the calculations (e.g., Apache Spark). Matrix factorization works well with the sparse nature of patient condition information (e.g., most patients only have a handful of conditions). Finally, the comorbid nature of many conditions can be naturally expressed via latent factors (e.g., a latent factor related to cardiovascular disease can usefully explain many conditions).

FEATURE ENGINEERING

There are two important considerations for generating useful input data: which features will be used, and how will confidence values for these features be determined. The features chosen here are a combination of historical condition information and demographic information. These features and their confidence values are generated from a patient population’s clinical history.

For condition features, diagnoses in a patient’s clinical history are grouped into clinically meaningful categories, or conditions, using the Clinical Classifications Software (CCS) of the Agency for Healthcare Research and Quality (AHRQ). Pa-



tients who are seen for the same condition multiple times are given a higher confidence value. More confidence is given for conditions that have been coded more recently. Additionally, more confidence is given for conditions that were coded in an inpatient setting rather than an outpatient setting.

The two main demographic features are age and gender. Unlike condition features, demographic features are given the same confidence level across all patients. The confidence value is determined such that demographic importance does not overpower condition information. However, these confidence values must also be large enough that gender-specific and age-specific conditions are modeled appropriately.

FITTING THE MODEL

The two most important hyper-parameters are lambda, the regularization parameter, and rank, the number of latent factors. Lambda should be strong enough to avoid overfitting in the training data, while also still allowing for meaningful personalization in predictions. Rank must be high enough to allow for meaningful groupings in latent factors, while avoiding the computational burden of higher rank models.

The goal is to identify the hyper-parameter values that are most useful for identifying uncoded comorbidities. To accomplish this, a tuning data set that excludes the most recent months of data is created. The hold-out data is analyzed to find conditions coded for the first time in a patient’s medical history. A variety of models are trained on the tuning dataset with different hyper-parameter values. For each model, the hold-out data is used to calculate the percentage of newly coded conditions appearing

in each patient’s 10 highest-rated uncoded conditions. Using the best performing hyper-parameter values, a final model is trained with all of the available data to make up-to-date patient-level lists of the highest-rated conditions.

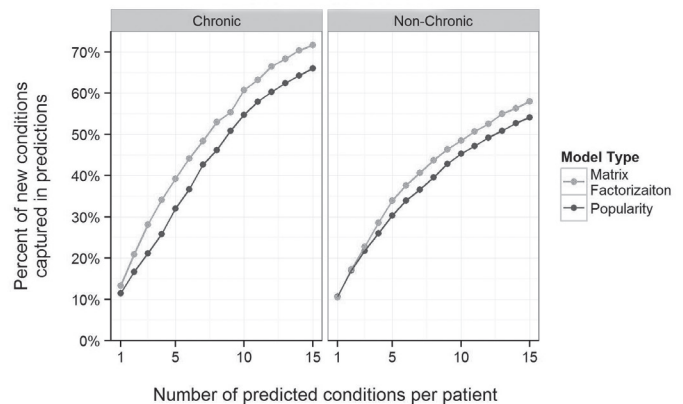
This whole tuning process is fast enough to calibrate hyper-parameters for each unique patient population.

MODEL PERFORMANCE

When using any advanced analytics, it is always important to have a useful baseline model to compare against. For a collaborative filtering model, the most basic reference model would be a simple **popularity** model that identifies the population’s most common conditions, excluding conditions that have already been coded for a patient. For example, a popularity model would identify the most common condition, such as hypertension, as the highest-rated condition to consider for all patients that do not already have hypertension coded.

The illustration in Figure 4 compares model accuracy on a sample population for the collaborative filtering model (Matrix Factorization) versus the simpler Popularity model. The vertical axis shows the estimate of accuracy discussed above: the percentage of newly coded conditions from the hold-out set that were among the predicted conditions for each patient. The horizontal axis displays accuracy for different numbers of predicted condi

FIGURE 4: MODEL ACCURACY ON TWO-MONTH HOLD-OUT



The left side focuses on chronic conditions, which are more likely to go uncoded if they are not the primary reason that a patient seeks care. The right side focuses on non-chronic conditions. Because of the higher intensity level required in care, non-chronic conditions are more likely to be coded at the time the illnesses arise. For both the chronic and non-chronic conditions, the matrix factorization model consistently outperforms the popularity model.

CASE STUDY

This case study will examine model inputs and model results for a sample patient with diabetes. For this patient, the input features, the top 10 highest-rated conditions, and a breakdown of the contribution towards the highest-rated condition will be explored.

DIABETES PATIENT

The table in Figure 5 shows the input features and their respective confidence values. Demographic features are given a constant confidence value, whereas the confidence values for condition features are a factor of the patient medical history.

FIGURE 5: INPUT FEATURES AND CONFIDENCE VALUES

FEATURE	CONFIDENCE
Age- 45	
Gender- Male	
Subscriber Relationship- Policyholder	
Diabetes mellitus with complications- Chronic	
Essential hypertension	
Disorders of lipid metabolism	
Other nutritional; endocrine; and metabolic disorders- Chronic	
Diabetes mellitus without complication- Chronic	

The table in Figure 6 shows the top 10 highest-rated conditions and their relative ratings for this patient. The ratings are determined through a recombination of latent factors for the patient and the respective condition.

FIGURE 6: HIGHEST-RATED CONDITIONS

HIGHEST-RATED CONDITIONS	RATING
Thyroid disorders- Chronic	
Mood disorders- Chronic	
Anxiety disorders- Chronic	
Other upper respiratory disease- Chronic	
Esophageal disorders- Chronic	
Nutritional deficiencies- Chronic	
Other nervous system disorders- Chronic	
Osteoarthritis- Chronic	
Spondylosis; intervertebral disc disorders; other back problems- Chronic	
Asthma	

The table in Figure 7 breaks down the relative contribution for the highest-rated condition, thyroid disorders. A condition's rating can be decomposed into contributions from each of the input features, based on the feature's confidence value and latent factors.

FIGURE 7: CONTRIBUTING FEATURES, THYROID DISORDERS

THYROID DISORDERS - CONTRIBUTING FEATURES	CONTRIBUTION
Subscriber Relationship- Policyholder	
Essential hypertension	
Diabetes mellitus with complications- Chronic	
Age- 45	
Disorders of lipid metabolism	
Other nutritional; endocrine; and metabolic disorders- Chronic	
Diabetes mellitus without complication- Chronic	
Gender- Male	

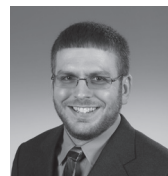
The demographic features have a high contribution to the rating, which is partially due to the high confidence value associated with these features. Hypertension and diabetes are other strong contributing factors. Male gender appears to be slightly negatively associated with thyroid disorders.

CONCLUSION

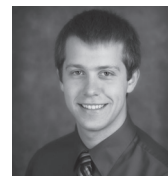
These lists of potential comorbid conditions can be used in a number of work-flows. Most importantly, these condition lists could be used to remind clinicians of common comorbidities to consider coding at the time of service.

In addition to identifying new conditions, the same model can be used to identify potential outliers in the conditions that have already been coded. Estimated ratings for existing conditions can be calculated, and those with extremely low values might represent codings that should be reconsidered to ensure there was not perhaps a mistake during data entry.

Accurately documenting a patient's clinical status will be increasingly important as more health care providers enter into alternative payment arrangements. Provider organizations face a growing scrutiny on the quality and cost of care. As a result, advanced analytics must find their way into daily workflows. Collaborative filtering systems provide a unique perspective toward coding improvements that produce useful suggestions of uncoded conditions. ■



Shea Parkes, FSA, MAAA, is an actuary at Milliman in Indianapolis. He can be reached at shea.parkes@milliman.com.



Ben Copeland is a data scientist and actuarial student at Milliman in Indianapolis. He can be reached at ben.copeland@milliman.com

Getting Started with Deep Learning and TensorFlow

By Jeff Heaton

Deep learning is a rapidly evolving machine learning technology. The world's largest technology companies are investing heavily in deep learning. They are sharing this investment with the world by open sourcing their deep learning technologies. Currently the tech titans of the world have open-sourced the following deep learning frameworks:

- Amazon – Deep Scalable Sparse Tensor Network Engine (DSSTNE)¹
- Baidu – PArallel Distributed Deep Learning (PADDLE)²
- Google – TensorFlow³
- Microsoft – Computational Network Toolkit (CNTK)⁴

All of these frameworks have their complete source code available on GitHub, which is a web platform that allows everyone from individual programmers to Fortune 500 companies to share source code and collaborate. As of the late 2016 writing of this article, DSSTNE and PADDLE both only work with the Linux operating system. TensorFlow works both with Macintosh and Linux. Not too surprisingly, Microsoft's CNTK is the only one of the four to support Microsoft Windows. The platforms supported by these frameworks will increase in the future. Work is already underway for Windows support in TensorFlow.

GOOGLE'S TENSORFLOW

Since its recent introduction in 2015, TensorFlow has taken the world of deep learning by storm. Though typically associated with deep learning, TensorFlow is actually a mathematics package specifically designed to leverage machine learning across CPU, GPU, and grid computing. Many machine learning models can be adapted to TensorFlow. It works best with neural network-like models, such as deep belief neural networks, generalized linear regression (GLM), support vector machines (SVM), and Long Short Term Memory (LSTM). While it might be possible to adapt TensorFlow to tree-based models, such as Random Forests or Gradient Boosting Machines (GBMs), these are not a focus for current versions of TensorFlow.

Python is the most widely supported language for TensorFlow. TensorFlow itself is implemented in C++, so it is also possible to directly access TensorFlow from a less widely known C++ based application programming interface (API). Typically, models are

defined as TensorFlow compute graphs that define the order that calculations must occur in order to calculate a model. For example, a GLM would be defined as dot products feeding into a link function. The graph ensures that the link function is not calculated until the precursor dot products have been calculated. Similarly, neural networks are layers of dot product calculations and activation functions. The graph defines the exact order of these calculations. Python code is used to define this compute graph. However, Python would be too slow to calculate and fit these models in any acceptable timeframe. Rather, TensorFlow transforms these compute graphs into highly efficient C++ and Graphical Processing Unit (GPU) code. Deep learning is very well adapted to GPUs, and TensorFlow contains extensive support for GPUs.

INSTALLING TENSORFLOW

Because performance is paramount in deep learning, every reasonable optimization has been employed in its design. Many of these optimizations are platform specific. Currently, TensorFlow officially supports the Macintosh OSX and Linux operating systems. Windows is not currently supported. Google suggests using a virtual machine or Docker (a software containerization platform) if you must make use of the Windows operating system. At some point, TensorFlow might support windows natively. However, that time has not yet arrived. Google provides installation instructions for TensorFlow for Mac, Linux, and Windows (using an emulator).⁵

It is also possible to use TensorFlow entirely from the cloud in a web browser. This frees you from the complexities of installing binary Python packages. Jupyter notebooks provide a convenient web-hosted environment to program Python. IBM provides a free Jupyter notebook that can be used directly from the web. The Data Scientist Workbench⁶ is a free and open Jupyter notebook that can be used to run the examples provided in this article.

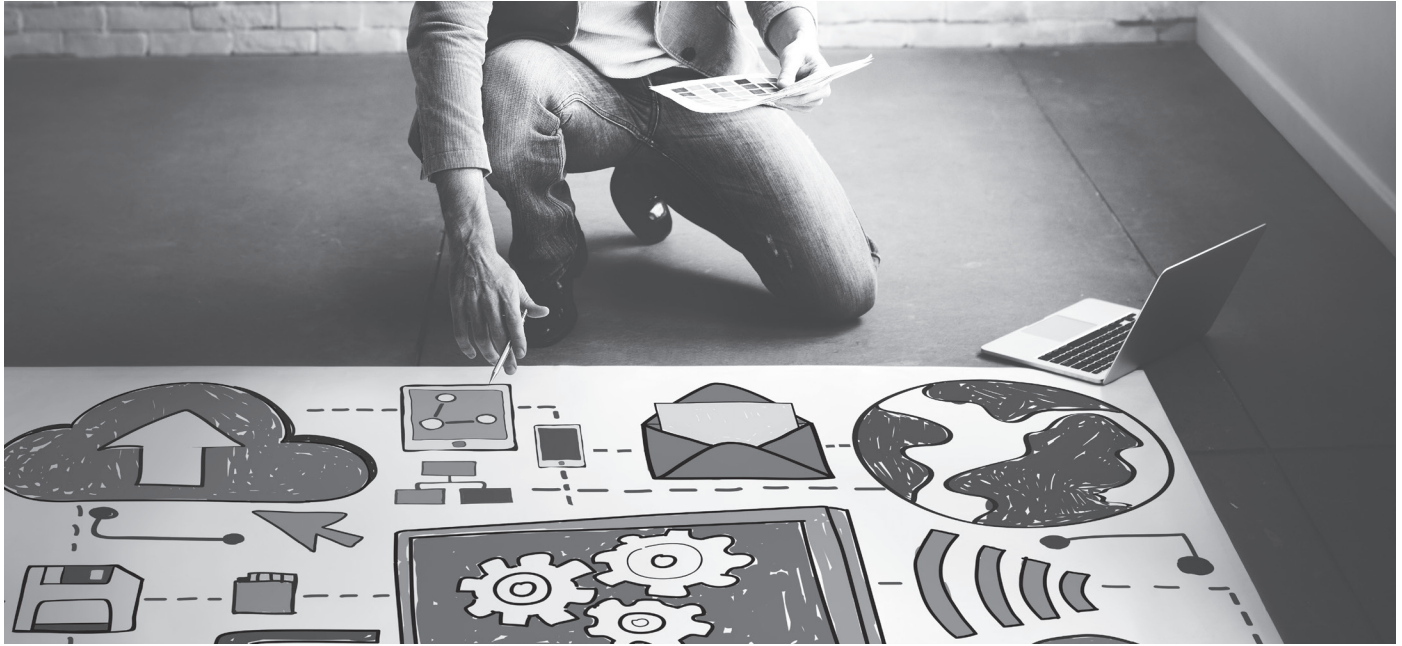
USING TENSORFLOW

To make use of TensorFlow you must import it into Python. The following two lines of code import TensorFlow and report what version of it you are using.

```
import tensorflow as tf

print("Tensor Flow Version: {}".format(tf.__version__))
```

The above code should report that you are using TensorFlow 0.8 or higher. The examples provided with this article were all created with 0.8 of TensorFlow. These examples are stored at GitHub and will likely be updated for future versions of TensorFlow.⁷ The reference link refers to a deep learning class at Washington University in St. Louis that is taught by the author of this article.



ENCODING CATEGORICAL DATA FOR A DEEP NEURAL NETWORK

Neural networks function similarly to traditional classification and regression models. An input vector (x) of predictors is provided to the neural network and a result (y) is returned from the network. The data provided to the neural network's input vector must be encoded to numeric form. If a categorical value is a predictor, meaning it is part of the information given to the neural network to make a prediction, then it should be encoded into dummy variables. The following Python function can be used to encode a dummy variable:

```
def encode_text_dummy(df,name):
    dummies = pd.get_dummies(df[name])
    for x in dummies.columns:
        dummy_name = "{}-{}".format(name,x)
        df[dummy_name] = dummies[x]
    df.drop(name, axis=1, inplace=True)
```

For example, to encode a dummy variable named "state," in the dataframe "df," use the following call:

```
encode_text_dummy( df, "state")
```

This will remove the column "state" from your dataframe and replace it with 50 dummy variables that represent each of the 50 U.S. states (assuming all 50 were present in your dataset). Dummy variables replace a categorical variable with a number of 1/0 (true/false) columns that represent the categorical value.

For each row, there would be 50 such fields, all of which would be zero, except for the state that the row corresponds to.

If we were predicting the state, and it were on the y side of the model, then we must encode this categorical value to an index. The following code accomplishes this:

```
def encode_text_index(df,name):
    le = preprocessing.LabelEncoder()
    df[name] = le.fit_transform(df[name])
    return le.classes_
```

The following code would encode the state to an index:

```
encode_text_index(df, "state")
```

Encoding to an index removes textual state abbreviations and assigns an index to each. Rather than getting 50 dummy variables, you have a single column variable. It is important that you not encode predictors as indexes. This will introduce bias. For example, two states might have indexes that are very close to each other. The distance between state indexes would convey undesired bias information to the network. This limitation does not exist for the output (y) values as TensorFlow simply treats each output as a separate independent category.

ENCODING CONTINUOUS DATA FOR A DEEP NEURAL NETWORK

Neural networks prefer their input columns to be centered near zero. They do not need to be normally distributed, but the zero



centering has been shown to help with neural network accuracy. Statistical z-scores are a great way to accomplish this. The following function will normalize a column to z-scores:

```
def encode_numeric_
zscore(df, name, mean=None, sd=None):
    if mean is None:
        mean = df[name].mean()

    if sd is None:
        sd = df[name].std()

    df[name] = (df[name]-mean)/sd
```

This function allows the mean and standard deviation to either be passed in or calculated. If you are training the initial dataset you should not provide mean and standard deviation, as you have enough data to calculate them. However, later you might have only a few values to generate predictions on, so it is helpful to provide the mean and standard deviations from the original training set. To convert the column “income” to z-scores use the following call:

```
encode_numeric_zscore(df, "income")
```

Later, if you wanted to normalize just a few rows, and you already knew the mean and standard deviation were 50,000 and 15,000, you would call:

```
encode_numeric_zscore(df, "income", 50000,
15000)
```

Once all of the input columns have been normalized correctly, you are ready to train a neural network.

TRAINING A DEEP NEURAL NETWORK

A TensorFlow network is trained using two sets of data named *x* and *y*. The dataframe (*df*) must be separated into these predictors (*x*) and the expected output (*y*). The following function can be used to do this:

```
def to_xy(df, target):
    result = []
    for x in df.columns:
        if x != target:
            result.append(x)
    return df.as_matrix(result), df[target]
```

If you wanted to predict (*y*) the income column for the dataframe (*df*) you would use the following call to separate into *x* and *y*:

```
x, y = to_xy(df, "income")
```

Now that the dataframe is separated, the neural network can be created and then trained.

To create and train a neural network for classification, use the following code:

```
classifier = skflow.TensorFlowDNNClassifier(  
    hidden_units=[30, 20, 10],  
    n_classes=3, steps=200)  
classifier.fit(x, y)
```

The above neural network would have hidden layers with 30, 20 and 10 hidden neurons. The number of hidden neurons can affect the accuracy of the neural network. Usually you will start with a larger number of hidden neurons (30) and add layers working down to a smaller number (10). This network would be able to classify three classes, and 200 steps would be used to train it.

To create and train a neural network for regression, use the following code:

```
regressor = skflow.TensorFlowDNNRegressor(  
    hidden_units=[10, 20, 10],  
    steps=200)  
regressor.fit(x, y)
```

Fitting the neural network may take a while, depending on how many steps you have specified. The more steps, the more accurate the neural network will become.

EVALUATING A DEEP NEURAL NETWORK

There are a variety of ways to evaluate a neural network. Two of the most simple are root mean square error (RMSE) for regression and accuracy for classification. The following calculates the RMSE score:

```
score = \  
    np.sqrt(metrics.mean_squared_  
    error(regressor.predict(x), y))  
print("Final score (RMSE): {}".  
    format(score))
```

The RMSE error simply measures the magnitude of the average difference between the expected outcome and the actual outcome. Lower RMSE scores are better.

Accuracy is measured similarly:

```
score = metrics.accuracy_score(y,  
    classifier.predict(x))  
print("Final score: {}".format(score))
```

Accuracy is simply the percent of data items that were classified correctly. Higher accuracy scores are better.

OTHER APPLICATIONS OF DEEP LEARNING

Deep neural networks can accomplish the same type of classification and regression tasks that other models like support vector machines, GLMs and decision trees are used for. While deep neural networks might sometimes provide better results than other model types, there are two important areas where deep neural networks really shine: computer vision and time series prediction.

Computer vision might seem like a technology more suited to a Google self-driving car than an insurance company. However, there are cases where computer vision can be very useful to an insurance company. Two recent Kaggle competitions highlighted these areas. The first was the Kaggle Diabetic Retinopathy Detection.⁸ This challenge used predictive models to look at retinopathy images and predict if an individual had diabetes. Additionally, State Farm ran a Kaggle competition to analyze images and detect distracted drivers.⁹ Both of these computer vision applications could help insurers to determine risk.

Time series is another area where neural networks are particularly adept. This is because neural networks can be recurrent. By allowing connections backwards through the neural network they are able to learn to predict patterns in a series of inputs, not just patterns within individual input. A neural network could have two inputs to read the systolic and diastolic blood pressures. However, a traditional model would always output the same for a given reading. A recurrent neural network could detect a pattern in a series of readings. Two of the most current types of recurrent neural networks are the LSTM and gated recurrent unit (GRU) networks. Time series and neural networks will be the topic of a future article for this newsletter. ■



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

ENDNOTES

- 1 <https://github.com/amznlabs/amazon-dsstne>
- 2 <https://github.com/baidu/Paddle>
- 3 <https://github.com/tensorflow/tensorflow>
- 4 <https://github.com/Microsoft/CNTK>
- 5 https://www.tensorflow.org/versions/r0.10/get_started/os_setup.html
- 6 <https://datascientistworkbench.com/>
- 7 https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class2_tensor_flow.ipynb
- 8 <https://www.kaggle.com/c/diabetic-retinopathy-detection>
- 9 <https://www.kaggle.com/c/state-farm-distracted-driver-detection>

Guide to Deep Learning

By Syed Danish Ali

Up until the recent past, the artificial intelligence (AI) portion of data science was looked upon cautiously due to its history of booms and flops.¹ In the latest stream of events, major improvements have taken place in this field and now deep learning, the new leading front for AI, presents a promising prospect for overcoming problems of big data. Deep learning is a method of machine learning that undertakes calculations in a layered fashion starting from high level abstractions (vision, language and other AI related tasks) to more and more specific features². Deep learning algorithms essentially attempt to model high-level abstractions of the data using architectures composed of multiple non-linear transformations. The machine is able to progressively learn as it digests more and more data, and its ability to transform abstract concepts into concrete realities has opened up a diverse plethora of areas where it can be utilized. Deep learning has various architectures such as deep neural networks, deep belief networks, Deep Boltzmann machines and so on that are able to handle and decode complex structures that have multiple non-linear features.³

Deep learning offers us considerable insight into the relatively unknown unstructured data which is 80 percent of the data that we generate as per IBM.⁴ While traditional data analysis before 2005 focused on just the tip of the iceberg, the big data revolution sprang up and now deep learning offers us a better glimpse into the unconscious segment of data that we know exists, but is constrained in realizing its true potential. Deep learning helps us in both exploring the data and identifying connections in descriptive analytics for ratemaking, but these connections also help us in price forecasting what the result will likely be, given the particular combination as the machine learns from the data.

Deep learning has inputs, hidden layers where they are transformed by the weights/biases and output which is achieved through choice of activation function from various functions available (Softmax, sigmoid, hyperbolic tangent, rectified linear, maxout and so on). The weights/biases are learned by feeding training data to the particular deep learning architecture.

A de-mystified foundation of deep learning is that deep learning is mostly a way of using backpropagation with gradient descent and a larger number of hidden neural network layers, which is



certainly not new. However, revival of deep learning was possible after 2010 and onwards due to drastically more computational power from GPUs, bigger datasets, and some key algorithm tweaks—mainly dropout and AdaGrad to increase accuracy rates. Moreover, the unique feature of deep learning is that it allows individual parts of the model to be trained independently of the other parts.⁵

Deep learning models can recognize human faces with more than 97 percent accuracy, as well as recognize arbitrary images and even moving videos. Deep learning systems now can process real-time video, interpret it, and provide a natural language description. It is becoming increasingly established that deep learning can perform exceptionally well on problems involving perceptual data like speech recognition image classification and text analytics.⁶

In a single formula, this is the formula for neural networks (for hyperbolic tangent activation function).⁷

$$p(x) = \beta_0 + \sum_{i=1}^{n_h} \beta_i \tanh \left(\alpha_{i,0} + \sum_{j=1}^n \alpha_{i,j} x_j \right).$$

So that essentially, $p(x)$ = linear + non- linear.

Aside from exposures, the other side of ratemaking of general insurance is losses and loss trends. By building deep learning models we can analyze images to estimate repair costs. Also deep learning techniques can be applied to automatically categorize the severity of damage to vehicles involved in accidents. This will more quickly update us with more accurate severity data for modeling pure premiums.⁸

Deep learning is becoming the method of choice for its exceptional accuracy and capturing capacity for unstructured data. This is also emphasized ahead in the section machine learning-unstructured data mining and text analytics.⁹

One issue, however, with deep learning is trying to find the hyper-parameters that are optimal. The possible space for consideration is very large and it is difficult and computationally intensive to understand each hyper parameter in depth. One potential solution that the author of this report identifies is the possible use of a genetic algorithm to find optimal hyper parameters. Genetic algorithms are already used on GLMs on R “glmulti” packages to select optimum GLM equation as per a given criteria usually Akaike Information Criterion or Bayesian Information Criterion.

Moreover, another algorithm has been used to optimize both structure and weights of a neural network. ES HyperNEAT is Evolving Substrate Hyperbolic Neuroevolution Of Augmenting Topologies developed by Ken Stanley. It uses a genetic algorithm to optimize both the structure and weights of a neural network. Following from this, maybe the ES HyperNEAT framework can be extended to deep learning so that a genetic algorithm can optimize both the structure and weights of the neural networks in deep learning as well.¹⁰

Another problem is over-fitting. Machine unlearning can be used to solve this. We will try to explain machine unlearning in one sentence. Machine unlearning puts a new layer of a small number of summations between the training data and the learning algorithm so that the dependency between these two is eliminated. Now the learning algorithms depend only on the summations instead of the individual data from which over-fitting can arise more easily. No retraining or remodeling is required.¹¹

Finally, there are huge numbers of variants of deep architectures as it's a fast developing field and so it helps to mention other leading algorithms. This list is intended to be comprehensive

but not exhaustive since so many algorithms are being developed.^{12, 13}

1. Deep High-order Neural Network with Structured Output (HNNSO)
2. Deep convex networks
3. Spectral networks
4. noBackTrack algorithm to solve the online training of RNN (recurrent neural networks) problem
5. Neural reasoner
6. Recurrent Neural Networks
7. Long short term memory
8. Hidden Markov Models
9. Deep belief networks
10. Convolutional deep networks
11. LAMSTAR
12. a) Large memory storage and retrieval neural networks
13. b) Increasingly being used in medical and financial applications
14. Deep Q-network agent
15. a) Used by Google DeepMIND
16. b) Based on reinforcement learning, which is a major branch of psychology, aside from evolution ■



Syed Danish Ali is a senior consultant at SIR consultants, a leading actuarial consultancy in the Middle East and South Asia. He can be reached at sd.ali90@gmail.com.

ENDNOTES

- 1 Jack Clark (Feb. 3, 2015); Bloomberg Business, “I’ll be back: The Return of Artificial Intelligence.”
- 2 Will Knight (May 31, 2015); MIT Technology Review, “Deep Learning catches on in new industries, from fashion to finance.”
- 3 Yoshua Bengio, University of Montreal, “Learning deep architectures for AI.”
- 4 IBM Website Smarter Planet, “Improve decision making with content analytics.”
- 5 Sean Lorenz (June 2016), Domino Datalab, “Deep learning with h2o.ai.”
- 6 PwC, March 2016, Top Issues, “AI in Insurance: Hype or reality?”
- 7 Dugas, et. al.; “Statistical Learning Algorithms Applied to Automobile Insurance Ratemaking.”
- 8 PwC, March 2016, Top Issues, “AI in Insurance: Hype or reality?”
- 9 Ibid
- 10 Risi, S. and Stanley, K. University of Central Florida, The ES-HyperNEAT Users Page
- 11 Cao and Yang, 2015. IEEE symposium on security and privacy, pgs. 463-480. “Towards Making Systems Forget with Machine Unlearning.”
- 12 Mayo M, Larochelle H, (Oct 2015) KD Nuggets.com. “Top 5 arXiv Deep Learning Papers, Explained.”
- 13 Mayo M, Larochelle H, (Jan 2016) KD Nuggets.com. “5 More arXiv Deep Learning Papers, Explained.”

Introduction to Using Graphical Processing Units for Variable Annuity Guarantee Modeling

By Bryon Robidoux

Recently, I was asked to give a webcast on using Graphical Processing Unit (GPU) for predictive modeling. This paper will be an introduction to the GPU and will be a precursor for the webcast. A GPU is nothing more than the graphics card in your computer which creates the images on your monitor. The laptop I am working on now has four cores in its Central Processing Unit (CPU) where a GPU will have thousands of cores. Each core allows calculations to be done in parallel. It became apparent to scientists that a computer's GPU was a cheap way to get massive parallelization on a desktop computer. NVIDIA, a company that manufactures GPU cards for computers, introduced CUDA (originally, an acronym for Compute Unified Device Architecture) extensions to the C programming language and CUDA cores to encourage scientists to use GPUs for scientific computing. Originally scientists had to transform their calculations to fool the GPU, but now most NVIDIA graphics cards are CUDA compliant. This manor of using a GPU is quickly going from cutting-edge to mainstream in many different scien-

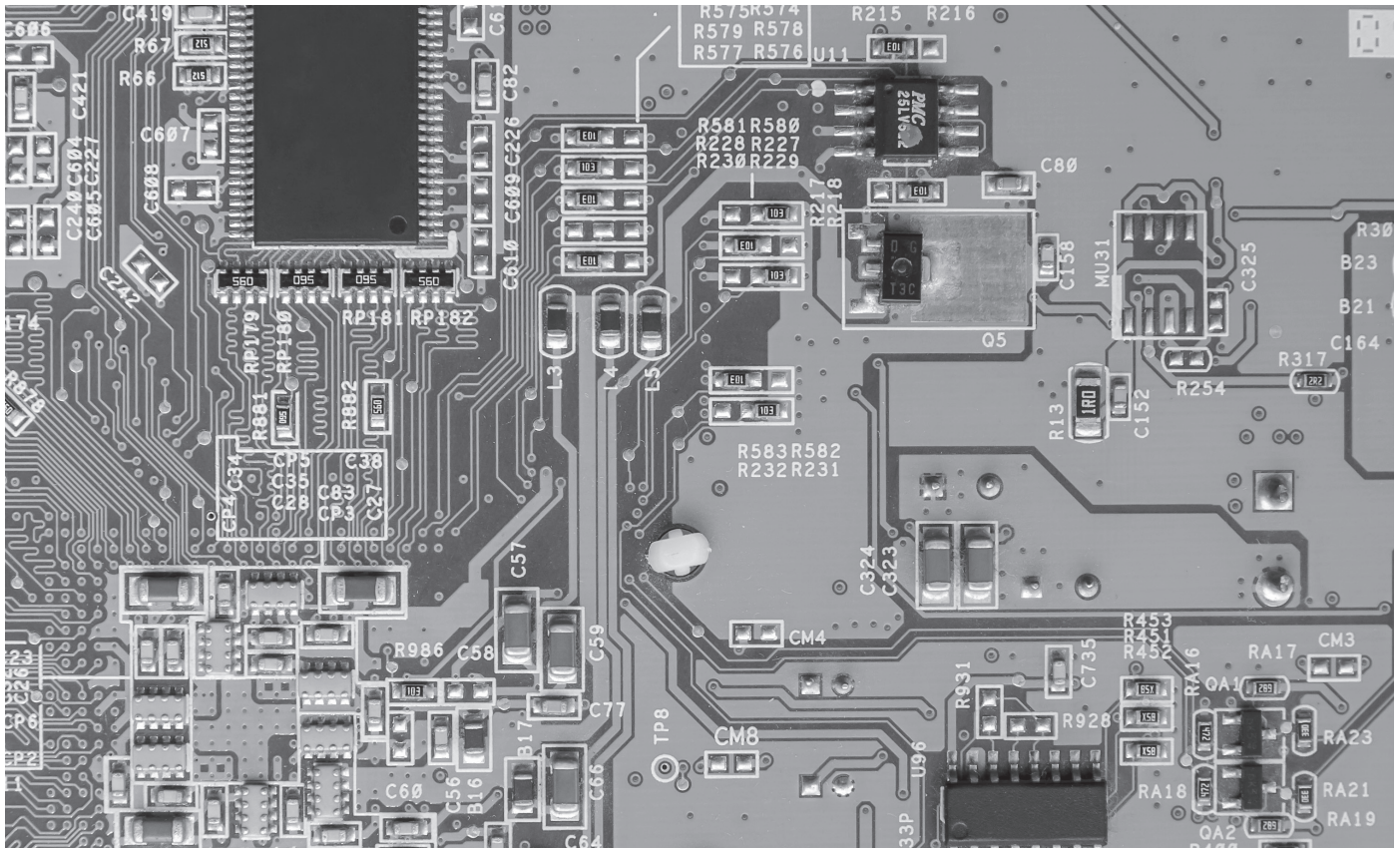
tific professions, but I don't think this is true yet for the actuarial profession.

As I have been approached about working on various GPU projects, the person touting the project usually describes it as fairly straightforward. They state that all that is required is to take the existing model, change a few lines of code, and abracadabra the new calculations will run just as fast as a compute cluster. This could not be further from the truth! This mentality may get something to work, but it will be nowhere near the speed advertised or possible. It may actually be slower than no GPUs at all! Building a variable annuity guarantee model is way different than building a predictive model, but at the same time, you will potentially run into similar types of bottlenecks and issues. Even though 99 percent of the time you will be using a library such as Theano, which is a high-level python library, or Thrust, which is a C++ library, to do the predictive modeling on GPUs, understanding the finer points of GPU architecture will help you understand why a particular model is running slower than anticipated or why it may not be able to be ported to a GPU. The first part of this article will give a high level overview of the GPU architecture. The second part of the article will describe different aspects of modeling a variable annuity guarantee. The third part of the article will try to combine the constraints of the GPU architecture with the common features of the variable annuity guarantee model to create solutions to likely bottlenecks in the variable annuity guarantee model.

In GPU literature, the CPU is called the host, whereas the GPU is called the device. Other than the host calling functions to execute on the device, the host and device run separately from each other. The host and device have their own memory also. To use the device, the data must be migrated from the host memory into the device's global memory. This is one of the first barriers to using a GPU. It can take longer to transfer data to and from

FEATURE*	TESLA K80	FEATURE	KEPLER GK210
GPU Chip(s)	2x Kepler GK210	Compute Capability	3.7
Peak Single Precision (base clocks)	5.60 TFLOPS (both GPUs combined)	Threads per Warp	32
Peak Double Precision (base clocks)	1.87 TFLOPS (both GPUs combined)	Max Warps per SM	64
Peak Single Precision (GPU Boost)	8.73 TFLOPS (both GPUs combined)	Max Threads per SM	2048
Peak Double Precision (GPU Boost)	2.91 TFLOPS (both GPUs combined)	Max Thread Blocks per SM	16
Onboard GDDR5 Memory1	24GB (12GB per GPU)	32-bit Registers per SM	128K
Memory Bandwidth1	480 GB/s (240 GB/s per GPU)	Max Registers per Thread Block	64K
Achievable PCI-E transfer bandwidth	12 GB/s	Max Registers per Thread	255
# of SMX Units	15	Max Threads per Thread Block	1024
# of CUDA Cores	2880	Shared Memory Configurations	16KB + 112KB L1 Cache
Memory Clock	3004 MHz	Max Shared Memory per Thread Block	48KB
GPU Base Clock	745 MHz		

*table from <https://www.microway.com/knowledge-center-articles/in-depth-comparison-of-nvidia-tesla-kepler-gpu-accelerators/>



the host and device than it does to run the actual calculation. In these types of situations, it is best not to use the device. The device contains one or more kernels. Kernels are major functions used to run code on the device. It consists of many blocks that work independently of each other. Each block consists of a user defined number of threads. The threads are what actually execute the code.

The key to understanding the bottlenecks when developing for GPUs is to understand how the threads get scheduled and the memory resources available to the threads. There are multiple layers of schedulers. The top level is GigaThread global scheduler which controls the scheduling of the kernels and the streaming multiprocessors (SM). Within the global scheduler, the SMs control the scheduling of the blocks. Each SM schedules its blocks independently of the other SMs. This is why if synchronization is required among threads, it must happen within the block. The SM's basic unit of scheduling threads is the warp, which is a block of 32 threads. The compute capacity is defined by NVIDIA as the maximum number of resident threads per SM.¹ The larger the compute capacity the better the GPU is suited for scientific calculations.

For this article I am going to use the Kepler K80 GPU for the example. This is worthy of doing actuarial modeling. Don't be

fooled into thinking that your gaming graphics card is good for actuarial modeling. Even though it probably contains CUDA cores, the compute capacity is not sufficient. It would be worthy for proof of concept, but not production requirements. The following table shows the specifications for the K80.

A GPU is a Single Instruction Multiple Data (SIMD) device. This means that all the threads within a warp must process the same instruction in order to run in parallel and only the data can be different. Warp divergence occurs if all the threads in the warp are not running the same instruction. This means that an "if statement" can have a huge impact on speed because this is a natural place for instructions to diverge. In a worst case scenario, each thread within the warp will have to be run serially because each thread has to run a separate instruction, in the case of the K80, causing a potential 32X slowdown to occur.¹ The next large hurdle is memory resources.

The following table gives the specifications on different types of memory within the GPU. The important information to get from the table is where the memory is located and its bandwidth. For this article, the local memory will be restricted to just L1-Cache. (A level 1 cache (L1 cache) is a memory cache that is directly built into the microprocessor, which is used for storing the microprocessor's recently accessed information, thus it is

MEMORY	LOCATION	CACHED	ACCESS	SCOPE	BANDWIDTH GB/S*	ON-CHIP/OFF-CHIP
Register	On-Chip	No	Read/write	One Thread	10,847	45
Local	On-Chip	Yes	Read/write	One Thread	2,169	9
Shared	On-Chip	N/A	Read/write	All threads in the block	2,169	9
Global	Off-Chip (unless cached)	Yes	Read/write	All threads + host	240	1
Constant	Off-Chip (unless cached)	Yes	Read	All threads + host	240	1

* This table is from 1. It only provides the information for the Fermi C2050 on page 101. The values for shared, local and register are derived by multiplying the ratio of the memory bandwidth between the K80 and Fermi C2050 which is 240/177 to keep the relative speeds the same between the GPU models.

also called the primary cache.²) The on-chip memory is inside the SM, whereas off-chip memory sits outside of the SM. It is easy to see that on-chip memory is faster than off-chip memory. The problem is that the faster the memory, the less of it there is available.

From the K80 specification table, the global memory size is 24GB whereas the register memory size is 128KB per SM. Register memory is a very valuable and limited resource. The only memory fast enough to keep the GPU running at full capacity is the register memory. The global memory is nowhere near fast enough. For example, a very simple C++ code snippet would be

```
for(i=0;i<N;++i) c[i]=a[i]+b[i];
```

where a, b and c are all in global memory and single precision 4-byte variables. This simple little program requires two mem-

It is obvious that a major factor of making the device run faster is developing a good strategy for moving data from the global memory to the register memory.

ory reads from a and b and one memory write to c. In order to keep the 5.6 trillion floating point operations (TFlop) busy on the K80, there would need to be 3 instructions * 4 bytes * 5.6 Tflops = 67.2 Terabytes/second (TB/s) of memory bandwidth, but there is currently only 240 GB/s available.¹ It is obvious that a major factor of making the device run faster is developing a good strategy for moving data from the global memory to the register memory. It is very easy to create a situation called register spillover which occurs when a thread block requires more registers than are available. In early generations of the device, the register spillover went into global memory, but for later generations the spillover first goes to the L1-Cache and if that is

exhausted it spills over into global memory. The L1-Cache is also a limited resource and needs to be managed carefully, but it does reduce some of the penalty of register spillover. The problem with the slow memory is that the warp will not schedule threads to run until all its resources are available. This means that very few warps can operate in parallel because the required data is in a traffic jam. The key strategy in GPU programming is to maximize data reuse, so you avoid unnecessary trips to fetch data from global memory. Now that we know some of the major constraints of the GPU it is time to move onto the modeling of the variable annuity.

This article will stay focused on the Guaranteed Minimum Withdrawal Benefit for Life (GMWBL) rider. The best way to approach it is to break down the rider into its fundamental modeling components so we can best try to map them to the GPU. From the top down, there is obviously the rider plan, policy, withdrawal cohorts for the policy, the time steps and lastly the order of transactions within the time step. The withdrawal cohort is just specifying the likely time someone will exercise their benefit along with the probability of them doing it at that time. From all the models I have worked on, they follow Mary Hardy's suggestion to make the time of withdrawal deterministic.³ At the very least, withdrawal cohorts are dimensioned by issue age, with four to 10 per issue age, but I have also seen them dimensioned by qualified and non-qualified status. Qualified status means the policy was purchased with proceeds from a before-tax account such as a 401k. Qualified policies will have significantly different behavior from non-qualified policies. Depending on the time of withdrawal, the policyholder can be rewarded through a credit or penalized through a loss of benefit. The order of transactions is my way of generalizing Mary Hardy's³ characterization of her two transaction types, which were before fees and after fees. In practice, there are usually three or four transaction types. They are usually label beginning of period (BOP), middle of period (MOP), and end of period (EOP). BOP is when the market mechanics, such as fund returns, and withdrawal behavior are calculated. MOP is when fees are applied and any ratchets or rollups occur, if applicable. EOP is when the decrements are applied such as mortality and dynamic lapse. Now that the fundamental components of modeling have been established, it is time to combine the GPU and the

variable annuity guarantee model to see where potential calculation bottlenecks and problems can occur.

From an actuarial perspective, the most intuitive way to model a GMWBL rider would be to put each withdrawal cohort of a policy on its own thread and to sort the policy file by rider plan and policy number. (Modeling at any lower level doesn't make sense because by definition they are serially dependent.) A few simple examples will show that this will easily lead to a massive register spillover and warp divergence if done haphazardly. Currently, I am working on a GMWBL model that requires 423 bytes of inputs to model a single withdrawal cohort of a policy. The inputs are a mixture of Booleans, single precision floating-point numbers, and integers. If each withdrawal cohort of a policy is allocated to a thread this implies 2048 threads * 423 bytes = 866KB of register memory is needed to calculate the block. There is only 128KB of registers available for the block so the capacity of the registers has been exceeded by 6X. Even with the ability to spill over into the L1-cache, the L1-cache is 112KB so this is still insufficient to handle all the data. If I were to port my current model to a GPU, as is, there is little I could do to avoid register spillover and not have huge speed reduction. The purposed strategy will also lead to warp divergence because the consequences and rewards to the policyholder depend on the timing of withdrawal. The consequences and rewards cause each withdrawal cohort to have a different behavior, which leads each withdrawal cohort down a different logical path.

In order to address the issues above, the minimum requirement is to perform some preprocessing. In order to get the data size requirement down, there needs to be a strategy for data reuse. Some of this can be accomplished by strategically sorting the policy file and creating a sort key by rider plan, then by issue age, then by assumed withdrawal time, then by qualified status, and then by any other fields that cause material changes to calculation behavior. This forces the policies with the most homogenous information and behavior together. Within the code, I would force all blocks to be homogenous by requiring only policies with the same sort key to be in a block. Enforcing homogeneity through the sort order and code should help to reduce thread divergence. It should also reduce register spillover and promote data reuse because roughly a quarter to a half of the 423 bytes of the data for the GMWBL policy are to describe variations within the rider features which are not policy specific. The common rider features can be migrated to shared memory and shared among threads. The shared memory has the speed as L1-cache, which is much better than global memory. The idea of grouping homogenous policies together to speed up calculations should not be unfamiliar to modeling and valuation actuaries because this is very similar exercise to get good cell compression on policy files.

One last topic that should be mentioned is the creation of the market dynamics from the economic scenario generator (ESG). The gold standard of random number generators (RNG) for use in ESG is the Mersenne Twister, because of its enormous periodicity. The Mersenne Twister is a serial generator because, after the seed is applied, each number generated depends on the previous number generated. It may be tempting to think that each thread should receive its own seed, but this would likely not preserve the statistical properties of the random number generator such as mean and standard deviation. In order to work properly, it is highly recommended that cuRand® be used to generate random numbers. The RNGs have been specifically designed, such as the MTGP Mersenne Twister, so that each thread can generate its own set of random numbers and still preserve the proper statistical properties. At this point, the only issue is with testing. It is very likely that individual policies would be checked with Excel which implies the testing will use the original Mersenne Twister. The original Mersenne Twister and the MTGP Mersenne Twister will not produce the same set of random numbers. They are only guaranteed to have the same statistical properties. As a part of testing, it will be required to isolate the random numbers from the device so calculations will match.

In conclusion, modeling variable annuities on GPUs can be a fun and challenging problem. It is not a simple migration to rework a model built for a compute grid to work on a GPU. This article by no means addresses all the issues of modeling VA riders with a GPU, but it should give you a good flavor of the types of issues that can occur. Even though building a variable annuity model on a GPU is much different than building a predictive model with a library such as Theano or Thrust, it should give you a good appreciation for some of the challenges of creating those libraries, demonstrate some underlying reasons on why the model may be calculating slower than expected, and possible ideas on how to speed the model up. ■



Bryon Robidoux, FSA, is director and actuary, at AIG in Chesterfield, Mo. He can be reached at Bryon.Robidoux@aig.com

ENDNOTES

- 1 Rob Farber, CUDA Application Design and Development, Morgan Kaufman 2011
- 2 <https://www.techopedia.com/definition/8048/level-1-cache-l1-cache>
- 3 Mary Hardy, Investment Guarantees: Modeling and Risk Management for Equity-Linked Life Insurance.
- 4 <https://www.microway.com/knowledge-center-articles/in-depth-comparison-of-nvidia-tesla-kepler-gpu-accelerators/>



SOCIETY OF ACTUARIES

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

NONPROFIT
ORGANIZATION
U.S. POSTAGE
PAID
SAINT JOSEPH, MI
PERMIT NO. 263

