Testing for Cognitive Bias in AI Applications

Peter Varhol, Gerie Owen

peter@petervarhol.com, gerie.owen@gerieowen.com

1. Abstract

We would like to think that AI-based machine learning systems always produce the right answer within their problem domain. However, in reality their performance is a direct result of the data used to train them. The answers in production are only as good as that training data.

But data collected by human means, such as surveys, observations, or estimates can have built-in human biases, such as the confirmation bias or the representative bias. Even seemingly objective measurements can be measuring the wrong things, or can be missing essential information about the problem domain.

The effects of biased data can be even more insidious. AI systems often function as black boxes, which means technologists are unaware of how an AI came to its conclusion. This can make it particularly hard to identify any inequality, bias, or discrimination feeding into a particular decision.

This paper explains how AI systems can suffer from the same biases as human experts, and how that could lead to biased results. It examines how testers, data scientists, and other stakeholders can develop test cases to recognize biases, both in data and the resulting system, and how to address those biases.

2. Biography

Peter Varhol is a software strategist and evangelist who closely observes the testing industry and uses his knowledge and experience to identify new technologies to identify trends and help companies respond to those trends. He speaks frequently at conferences, local meetups, and on webinars on software development, testing, machine learning, and DevOps topics. He also writes for popular technology publications such as Information Week and InfoWorld.

Peter has been a tenure-track professor in computer science and mathematics, technology journalist and editor, software developer and tester, and software product manager. He has MS degrees in computer science, applied mathematics, and psychology, along with doctoral work in data science.

Gerie Owen is a Testing Strategist and Evangelist. She is a Certified Scrum Master, Conference Presenter and Author on technology and testing topics. She enjoys mentoring new QA Leads and brings a cohesive team approach to testing. Gerie is the author of many articles on technology including Agile and DevOps topics. Gerie chooses her presentation topics based on her experiences in technology, what she has learned from them and what she would like to do to improve them.

3. Introduction

The uses for artificial intelligence (AI) and machine learning systems have exploded into the mainstream over the last couple of years. These types of systems are unique in that they "learn" based on data that is available about the problem domain. For example, by taking thousands or even hundreds of thousands of readings of the price of a stock in the stock market, we might be able to predict future movements of that stock with some accuracy.

These AI, or machine learning, systems are dependent upon data to enable them to learn. Machine learning systems consist of layered sets of mathematical algorithms that attempt to model a particular problem domain. It's not really learning as we understand it; it is simply using data to model a problem domain, and using that model to make decisions within that domain.

The most common type of machine learning architecture is the neural network. At a high level, neural networks model the neurons of the human brain. They include an input layer of neurons, an output layer, and one or more hidden layers of neurons. Each neuron, or node, contains a mathematical algorithm that transforms the data sent to it in some way. The network produces one or more outputs that are then compared to the known "correct" answer.

These systems are trained by applying data representing the problem and its solution. The algorithms manipulate that data at each stage of the process, until an output is obtained. After each training run, the software goes back and adjusts the algorithms and tries again. At first, the answers produced likely bear little resemblance to the correct ones, but after a number of training runs, the algorithm solutions should converge to the known correct solutions.

If an acceptable solution is not forthcoming from a particular neural network after a number of training runs, then it is time to go back and redesign the network. Perhaps more hidden layers can provide better accuracy, or different data are needed. But it is common for teams to go back and rethink their original assumptions regarding the data or the design.

That is the primary reason why issues are difficult to uncover and address. You won't get the exact right answer; instead, the goal is to get answers that are "good enough". What does good enough mean? It really all depends on the goals of the application. For an e-commerce recommendation engine, an advantage is usually worthwhile. For self-driving cars, it has to be pretty exact.

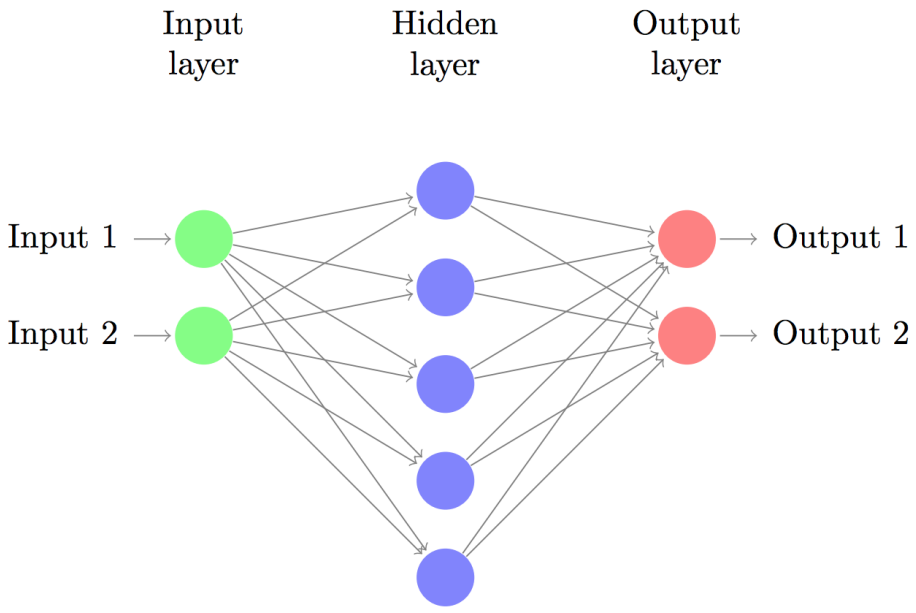

Figure 1. The basic neural network architecture, with an input layer for data, one or more hidden layers, and an output layer with the results.

1.1. About That Learning

AI and machine learning systems don't learn in the way that we normally think about learning. In humans, learning is a cognitive process that incorporates physiological changes to the brain and a change in behavior. Often as we learn new things, we integrate those into our existing knowledge, and are able to synthesize new concepts and applications.

While researchers are attempting to create machines with more general intelligence (which will introduce an entirely new and more complex series of problems), today's AI systems are almost entirely domain-specific, using algorithms and training techniques designed to operate in a particular problem domain.

Simply, what is occurring here in many cases is relatively straightforward curve-fitting, according to longtime researcher Judea Pearl. We can introduce advanced mathematical techniques that take into account things like prior probabilities or N-dimensional discriminant analysis, but the fundamental idea is that we are doing an advanced and largely black-box form of regression analysis. It's a black box in that we can't see inside these groups of algorithms to determine why a particular result has occurred.

2. What Causes Bias?

How can an intelligent system like this be biased? How can it produce results that don't appropriately reflect the problem domain? If it's trained with data that doesn't accurately represent the problem domain, it could produce results that aren't the best or most accurate responses. That can be difficult to determine, especially if the relationship between the training data and the domain isn't well-defined.

Further, it's very difficult to identify biases because many different types of tests are needed to determine whether or not bias exists in the first place. Even if we are looking for possible bias, we may not be looking at the domain and solution in the right way (Wachter-Boettcher, 2017).

So the training data is the key to a successful machine learning application. Data that accurately reflects the problem domain in all aspects will minimize bias. Data that is in some way skewed against the problem domain will likely cause biased results.

For example, in 2016 Amazon wrote an intelligent bot to search the Web and find potential candidates for jobs (Quartz, 2018). The training data was resumes from applicants that Amazon had received for the last ten years. For IT positions, the overwhelming number of applicants were men. As a result, the bot "learned" that men were better candidates for IT jobs. That bias was so pervasive in the algorithms that Amazon decided not to use the bot for IT positions.

This kind of bias is difficult to test for, and very difficult to identify, because the neural network itself is a black box, making it difficult for anyone in the design, development, and testing process to peer under the hood in order to understand where individual results are derived from. Yet as machine learning applications move more and more into the mainstream, unbiased results are critical to making the right business decisions. So identifying and correcting bias in these systems becomes a very high priority.

3. Bias Isn't a Bug

It's important to note that bias is not a bug, even though it may sound like one. A bug is an identifiable and measurable error in process or result of the software execution, and can usually be fixed with a code change. A bias, instead, is a bending of most or all results in a particular direction. In other words, it is incorrect, usually in a particular way. More importantly, bias cannot be fixed by a code change. The problem is deeper than that.

How does bias manifest itself? Because the AI system learns from data, how we choose our data affects the results that we obtain. If we choose data that doesn't

represent the entire problem domain objectively, then we have a strong potential for bias. So testers need to be able to identify if the data is biased, how it is biased, and how it might be corrected.

A lot of data used to train AI systems today is at least partially subjective. For example, opinions on physical attractiveness are used to train these systems that judge beauty contests. Those opinions may differ among reasonable people, but those whose opinions are included in the training data are those that matter. Also, parole officers' opinions are used as data to predict criminal recidivism. Both data sets are not generally used in isolation, but in combination with other data. Still, it is likely that any bias in those opinions will influence the results.

But even seemingly objective data can be biased. The data may simply not reflect actual use of the system once it reaches production, even if it is objectively measured.

For example, an electronic wind sensor, which uses the cooling of filaments to determine wind speed and direction, uses training data collected in a wind tunnel. The data are scientifically accurate, but were collected under the same humidity and temperature conditions. That could well produce biased results when deployed into the wild, which has often substantially different values in both factors. These authors created just such a design (Figure 2), in the 1990s, and found bias when the system was deployed into the wild (Varhol, 1993).

Figure 2. An electronic wind sensor that used wind tunnel data to train algorithms to estimate wind speed and direction (courtesy Armtek Industries).

4. Sources of Bias

There are three primary different sources of bias in AI systems. First is **data selection**; we have simply selected training data that represents only a part of the problem domain, or over represents part of the problem domain. This was Amazon's problem with its applicant application; it simply overrepresented male job candidates.

Data selection is probably the most common source of bias in AI systems, but it can be difficult to identify because the AI results seem to reflect real human decisions rather than an accurate view of the problem space.

A second source of bias is known as **latent bias**. This is where concepts become incorrectly correlated. Correlations are relationships between variables that can be used in prediction. To at least some extent, correlation is at the heart of many machine learning technologies, because we are using algorithms to relate known data to the results we want to use.

However, correlation doesn't mean causation. There may be a correlation between human intelligence and monetary income level, for example, but that doesn't mean that higher intelligence causes higher income. Instead, higher intelligence may result in people being more educated, which may in fact be the root cause of higher income. Without a good deal of further analysis, causation is difficult or sometimes even impossible to achieve.

The third source of bias is through **interactions**. If it is an AI that people interact with, those interactions can be misunderstood by the AI because it is being used in a way that was not intended. Microsoft Tay is a case in point. Tay was a Twitter bot that Microsoft launched to interact with others and dynamically learn from those interactions. Twitterdom quickly came to the realization that Tay could be trained for any behavior, without any conventional boundaries. As a result, nefarious tweets within the space of hours rapidly turned Tay into a racist and bigoted creation. As this was learned behavior, Microsoft had to withdraw it from use until it could start over again.

5. Testing for Bias

Testing for bias in AI systems is a significant challenge, because bias is difficult to identify in a limited period of testing time. Bias is, after all, a tendency, not a clear-cut right or wrong. It's not like a particular test case succeeds or fails. Therefore, it requires a different approach than traditional applications.

The sources of bias listed above also provide a guidepost on how to identify and test for that particular bias. Testers need to carefully examine the data used for training, and ideally have similar data for use in testing. Saying that data does not reflect the problem domain, or underrepresents or overrepresents the problem domain, has to be

demonstrated. Testers also need to look for unintended correlations, and ways that the AI can be incorrectly trained by interaction with people. Data that is seemingly innocuous may well have a significant influence on the results.

Testing for bias is substantially different than testing for defects in a traditional application. It starts with requirements. Rather than stating requirements in terms of feature capability or user interface controls, AI requirements have to be very specific in terms of both results and required margin of error. Depending on the problem domain, the quantitative requirements will be very different.

The electronic wind sensor developed by these authors is a case in point. Like any wind sensor, even a mechanical one, it was not going to precisely determine the exact wind speed and direction. In the wild, there is always an uncertainty factor. The question then became how much uncertainty was acceptable. In this case, we determined that for 95 percent of the readings, within 2 degrees of direction and within 2.5 knots of wind speed was acceptable, following a normal curve. With our testing data, we were able to achieve this goal, although subsequent tests in the wild were mixed because of confounding factors, such as dust.

Actual test planning and testing for bias can be accomplished through an analysis similar to that used for testing safety critical systems such as medical devices or aviation systems. With these systems, we look at potential risks, and attempt to determine the likelihood of those risks. For example, a cardiac monitoring device may have exposed leads, with the potential for electric shock from those leads. Safety critical testing involves determining the risk, the hazard inherent from that risk, and the likelihood that the risk will happen in reality. From that analysis you derive your test plan and test cases.

Likewise, testing for bias involves looking at any potential for a systemic incorrect response, the hazard inherent in that incorrect response, the likelihood that the hazard will manifest itself, and the harm it may do as a result.

For example, machine learning systems are being actively developed and used for health care applications, especially diagnosis and treatment regimens. Diagnosis is both difficult and uncertain; one of the authors in the recent past had been diagnosed with a terminal condition by multiple doctors that turned out not to be the case (obviously). The information was incomplete, and these doctors relied on their experience to fill in the blanks (Quartz, 2017).

Now imagine an AI system coming to such a conclusion. Was the AI making a truly objective diagnosis based on data, or did that data introduce bias based on a lack of causation or an overreliance on certain diagnostic outcomes? The only way to know is to go back and attempt to find flaws in the data methodology.

Testing for bias requires many more test cases than traditional functional testing. Testing individual requirements in traditional testing for correct or incorrect results usually only requires a handful of test cases; with AI system, each requirement may have hundreds or even thousands. That’s because bias will be very difficult to uncover

with just a few tests. The entire range of responses has to have representative test cases, including both mainstream and edge cases.

Further, testers must exercise discipline in running these cases. A methodology that runs test cases in a logical sequence will usually make it easier to identify trends. Exploratory testing, while valuable in many circumstances, doesn't exercise the discipline needed to identify bias.

6. Beginning the Testing Process

The best place to start such test cases is with the causes of bias mentioned above – data selection, latent bias, and interaction bias. Starting with data selection, testers quantify the problem domain, then map out how the planned data fully encompasses that domain. In the Amazon applicant case, the company could have looked at the number of resumes for each gender they had in the applicant pool and determined a method for equalizing them.

With latent bias, testers have to ask the question of why a particular data collection accurately reflects the causation behind a particular outcome. We want causation in order to ensure validity rather than a simple random relationship. Past performance does not always predict future results, as investment companies keep telling us.

Last, interaction bias determines if the data and architecture used can withstand use in the wild, especially in interactions with actual users. That will be the most difficult to determine, and requires looking into future use of the application. Will interactions with users influence future results? For static systems, those that don't learn after deployment, this is not a problem, but for those that do continue to learn, which are often referred to as adaptive or dynamic systems, it could be an issue. Adaptive systems require ongoing testing after deployment to ensure that the original intent of the application isn't subverted by ongoing use and changes.

How would this type of testing occur? The best approach is to maintain a set of test cases that you use initially to determine bias and correctness of results, then execute those same test cases at regular intervals throughout the life of the application. Ideally, those results should be approximately the same, with some accuracy improvements due to adaptive training.

However, if results start to diverge from correctness, testers have to consider the possibility of an interaction bias or other data-based bias. That could come from unexpected interactions, an over-reliance on a particular data set in adaptive training, or even that the problem domain has gradually changed over time. While a problem can be identified in a fairly straightforward fashion, the cause of that problem is much more difficult to determine. Ultimately, testers looking for a source of bias in adaptive

systems may have to make educated guesses based on their domain knowledge and deep understanding of the underlying data.

If there is bias in an AI system, addressing it is a difficult problem. As mentioned above, in most cases a code change won't fix the problem. In general, it will require starting over again with an untrained neural network, and retraining it with different data or an entirely different network. Although it won't require starting the development process all over again, it can be a time-consuming process. As mentioned earlier, unlike a bug, a bias encompasses virtually all of the code base, rather than a small section that can be easily changed (Barkan, 2018).

Teams and users that are concerned about bias will have to take into account the possibility that multiple trainings, and even algorithm changes, will be needed to get it right. It is best that the potential for bias is determined prior to the final selection of training data sets, so that any issues can be addressed before serious development begins.

7. Beyond AI Systems

It's important to note that all software is biased in some way, because it is designed and written by humans, who are biased in general. The biases may be trivial and unimportant, or they may have serious consequences. It doesn't necessarily have to be an AI application.

For example, a woman wasn't able to access the women's locker room of her gym, as her programmed RFID card wouldn't work. While the gym acknowledged that there was an error, it had no idea what was wrong or how to address it.

The software development team knew what was wrong. Among the registration information for the gym was the salutation. For expediency, the team had hard-coded the salutation "Doctor" as male, allowing access only to the men's locker room. This and similar biases can conceivably be fixed with a code change, although identifying the cause of the bias can be challenging (The Atlantic, 2017).

Biases such as this occur in many applications. Many of these types of biases are just as difficult to find, although they are usually easier to fix. Developers can code solutions that address the example above, as long as the problem can be found. Because developers can also introduce bias, design and development are the most appropriate places to observe and address it (King, 2010).

In the above example, such a decision may not have been designed or even documented; it may have just been an unconscious decision that an individual developer did under pressure to finish code. But it was bias, and it had implications to users.

8. Bias is Not Clear-Cut

There is some controversy in testing and fixing bias. If we want any bias to be uncovered and removed, then testing is an important aspect of the development process. But there may be circumstances where we want to reflect human bias in our AI systems. Bias isn't always bad, although we need to be aware that it is present. There may be problem domains where we believe the human expert is the best decision-maker, such as in medical diagnosis. Doctors may jump to a correct conclusion based on their experience and the bias that comes from that experience, and we may want the AI system to react in a similar way.

In problem domains such as this, we may want the AI system to reflect the expert biases. But are we getting the best diagnosis, or simply the same human diagnosis? And more importantly, which of these outcomes do we want in practice?

There is also ongoing research on "explainable AI", which is an attempt to offer an explanation of why a particular result occurred (Dosilovic, Brcic, and Hlupic, 2018). This research typically uses alternative AI techniques, such as decision trees, that may not provide the power of neural networks or genetic algorithms in solving domain-specific problems.

One approach that may be possible is to incorporate code that looks at what each algorithm does at each layer, and explains how a particular piece of data was interpreted by that algorithm. While this approach may not provide a human rationale for a decision, it could provide insight into what occurred at each step of the process (Müller and Bostrom, 2016).

These types of questions will continue to haunt bias in AI systems, and there is no easy answer. But until we can identify, quantify, and potentially remove bias, we won't understand what alternatives are available.

9. References

Müller, Vincent C. and Bostrom, Nick (2016), 'Future progress in artificial intelligence: A survey of expert opinion', in Vincent C. Müller (ed.), Fundamental Issues of Artificial Intelligence (Synthese Library; Berlin: Springer), 553-571.

Dosilovic, Filip; Brcic, Mario; Hlupic, Nikica (2018-05-25). "Explainable Artificial Intelligence: A Survey" (PDF). *MIPRO 2018 - 41st International Convention Proceedings*. MIPRO 2018. Opatija, Croatia. pp. 210–215.

M King, Tariq. (2010). A self-testing approach for autonomic software. ProQuest ETD Collection for FIU.

[Bar 18] Personal correspondence with Joseph Barkan, director of IBM Research Labs, 2018.

Wachter-Boettcher, Sara, 2017. Technically Wrong: Sexist Apps, Biased Algorithms, and Other Threats of Toxic Tech. New York: W.W. Norton.

Quartz, 2017. “When a Robot AI Doctor Misdiagnoses You, Who’s to Blame?” Posted May 23, 2017, https://qz.com/989137/when-a-robot-ai-doctor-misdiagnoses-you-whos-to-blame/ (accessed July 12, 2019).

Quartz, 2017. “AI hacks are trying to turn code into intelligence like alchemists tried turning lead into gold.” Posted September 28, 2017. https://qz.com/1089555/artificial-intelligence-programmers-are-trying-to-turn-code-into-intelligence-like-alchemists-tried-turning-lead-into-gold/ (accessed July 12, 2019).

The Atlantic, 2017. “The Coming Software Apocalypse.” Posted September 28, 2017. https://www.theatlantic.com/technology/archive/2017/09/saving-the-world-from-code/540393/ (accessed July 12, 2019).

Quartz, 2018. “Amazon’s sexist hiring algorithm could still be better than a human.” Posted November 7, 2018. https://qz.com/work/1454396/amazons-sexist-hiring-algorithm-could-still-be-better-than-a-human/ (accessed July 12, 2019).

Varhol, Peter D. “Neural Networks for Predicting Behavior.” Dr. Dobbs Journal, February 1993. http://collaboration.cmc.ec.gc.ca/science/rpn/biblio/ddj/Website/articles/DDJ/1993/9302/9302h/9302h.htm