Algorithms and the Auditor

Recent advances in machine learning (ML) and artificial intelligence (AI) have raised new awareness and concerns of algorithms, including their uses and misuses and their potential and limitations. It is important for practitioners, specifically auditors, to understand what algorithms are and why they matter, that smart algorithms are not new, and the decisive role humans have in algorithm design and metrics. It is the auditor's job to ask questions using the correct tools, interpret results and remember that errors are possible even with the most advanced algorithms.

Algorithms Are and Have Always Been Everywhere

The term "algorithm" is typically associated with mathematicians and computer scientists and is, therefore, often considered too specialized and advanced for the layman to understand. However, in essence, an algorithm is a recipe—one way of solving a specific problem that anyone could understand, for instance, when babies solve their needs for nourishment, pain management or attention by crying. An algorithm can be as simple as "If hungry, then cry." Similarly, algorithms are used for solving every type of activity from cooking to driving to troubleshooting or to diagnosing a medical condition. Whether it is thought of it this way or not, every time humans solve a problem, no matter how simple or complex it appears to be, they are using an algorithm. This algorithm may have been invented by others and passed along or developed out of need.

An algorithm does not have to be complex. All algorithms are not equal—there are better and worse ways of solving a problem, and there may be trade-offs for each variation. For example, programming the solution of a second-order algebraic equation using the discriminant formula is not a good way to solve the problem with a computer.[1] Similarly, if a program's workspace fits in memory, it is typically significantly more efficient to avoid the use of a database altogether. This is not practical or possible if a large amount of data needs to be used. The point is that what may be fast and easy with one algorithm may be prohibitively slow or even impossible with another.

The feasibility of potentially game-changing technological advances is critically dependent on finding an efficient algorithm that makes computations fast. An example is homomorphic encryption, which enables the manipulation of encrypted data without the need to convert them to cleartext first.[2] However, this greatly complicates the operations that must be performed.

The same concept applies to audits. Like any activity, audits involve tasks such as checking "as is" vs. the "as should be" situations or finding correlations, which may be broken into smaller subtasks. Checking the "as is" vs. the "as should be" situations imply an algorithm to:

- Obtain the "as is" and "as should be" versions
- Perform whatever operations are needed to enable a comparison

Spiros Alexiou, Ph.D., CISA, CSX-F, CIA
Has been an IT auditor for the last 13 years. He has more than 25 years of experience in IT systems, has written a number of sophisticated computer programs and algorithms, and has taught university courses on algorithms. He can be reached at spiralexiou@gmail.com.

- Perform the comparison
- Assess the results and their significance

A complete algorithm involves a detailed prescription of all of the general tasks and how to perform each of the subtasks. All the complexity and sophistication should not obscure the fact that an algorithm is just a recipe, or simply one way to tackle a problem, even if it is sophisticated or intelligent. Highly sophisticated and intelligent algorithms that can recognize and change their way of solving problems have existed for many years. For instance, adaptive solvers of differential equations can adjust their time step or method of attack using explicit or implicit methods depending on the problem.

> "A COMPLETE ALGORITHM INVOLVES A DETAILED PRESCRIPTION OF ALL OF THE GENERAL TASKS AND HOW TO PERFORM EACH OF THE SUBTASKS."

Algorithm Design and Metrics: Humans Are Important

A computer and the algorithms it employs to solve problems are merely following programmed prescriptions. This is also true for intelligent algorithms that can learn. Even highly advanced programs able to solve problems for which intelligence is needed are following prescriptions, probably with an elaborate feedback mechanism. In fact, the more intelligent the algorithm, the more specialized it typically is. However, both the input data and the relevant features the algorithm considers are determined by the user and programmer, and this must be taken into account when interpreting the output.

For instance, if a computer is given information on existing cars, their characteristics and their history and is asked to find correlations between characteristics and accidents, the characteristics chosen to be processed by the algorithm are crucial. If maximum speed, brake quality and other factors are not included in the characteristics processed, then the results may include spurious correlations. For example, the results may find a correlation between car color and traffic accident propensity where one does not actually affect the other. Similarly, if an auditor analyzes thefts in a branch or across branches but fails to record the monetary values of the items involved, then the analysis is likely to miss very important information and conclusions may be flawed.

Domain expertise is vitally important in designing algorithms. Someone tasked with designing a solution should at least be able to formulate the problem and outline the key characteristics of its solution, no matter how complicated the details may be. A number of algorithms, such as decision trees, can easily overfit and find spurious correlations without some type of guidance. This guidance can range from defining acceptable tolerances to including or excluding pieces of information, such as car color. This raises the issue of possible bias (i.e., ignoring or weighing potentially less important factors that do not fit the auditor's experience, presumptions or prejudices). In the previous example, why should the auditor consider the monetary value of items stolen and not their color? It is possible that the thief may choose to steal items of a certain color. The fact that a domain expert would choose the monetary value as a relevant characteristic but not the color of the stolen items or the manufacturer is an example of bias, as relevant characteristics are the projections of the auditor's experience, reasoning and beliefs. This is unavoidable and is also the case in many industries. For example, when studying cause and effect, only a limited number of possible causes are considered. Even the most convincing scientific experiments need an underlying theory to test (i.e., effect X depends on parameters A, B and C). In another example, when letting an object drop to test gravity, most people theorize about the force, speed and acceleration. That is, there must first be a hypothesis regarding the force and not whether the day is sunny or cloudy, for example, that determines the fall of the object. No experiment can handle an infinite number of possible causes. Similarly, in devising an elaborate algorithm to check for correlations and, ultimately, solve a problem, there can be a wide range of factors (e.g., monetary value, manufacturer, color). No algorithm can handle an infinite number of variables, which, in principle, may be related in a causal way to the end effect.

Particularly in ML, algorithms must be trained, and the training is what determines the algorithm's performance. This training can be compared to teaching a child. If a child has been trained in one area, for instance farming, the child will typically perform well in tasks related to farming and not as well in unfamiliar subjects. If there is bias in training (i.e., unimportant factors feature heavily and important factors are not featured all), especially with relatively small training sets, those trained algorithms will have a built-in bias. For example, a child who has never been exposed to a dangerous animal has only the notion of friendly animals. Similarly, if an algorithm is shown a number of hospitals and their costs for medical supplies and both the majority of the hospitals and hospitals with excessive costs are children's hospitals, then the algorithm may conclude that there may be a connection between children and excessive supplies. In other words, if an algorithm is to learn (from data), there is no such thing as bias-free learning. The best expectation is a pluralistic algorithm that allows multiple data sources.

> "IF AN ALGORITHM IS TO LEARN (FROM DATA), THERE IS NO SUCH THING AS BIAS-FREE LEARNING."

The bias in algorithms has no relation to the much more biased versions of history of different countries as discovered in past or present conflicts. There is a very good reason for this—algorithm designers are typically interested in covering as large a range of cases as possible, so that even if a rare case is encountered, the algorithm handles it well. When people talk about algorithmic bias, they typically mean bias in the data fed to the algorithm, using the incorrect algorithm to solve a problem or using an algorithm on a data set where it performs poorly.

Because a certain amount of bias in data is unavoidable, it is important to check for it. Just as scientists try to push theories to their limits to test them and then revise if necessary, human assumptions should also be tested, especially if a positive result has been produced. For example, in the car color case, a researcher can discover if different age groups with the same car color have similar accident rates. If not, then the car color is not the deciding factor. Similarly, in the hospital example, one might check the average cost for all children's hospitals and the average cost for general hospitals.

Another important issue is the necessary data. Algorithms operate on data. To prove or disprove a hypothesis such as a cost-benefit analysis or determine whether there is fraud in reporting, an algorithm needs data—not just any data, but the data that are necessary for answering the specific problem. Algorithms do not create this data—it is the job of humans to make sure that the algorithm has all the necessary information and data to solve the problem. Poor planning may lead to using whatever data are available to answer a problem for which the available data are inadequate or irrelevant. Even if the exact problem has not yet surfaced, planning to have the kind of data needed to answer classes of problems is essential.

Humans judge the relative successes or failures of algorithms. In many cases, one algorithm is not necessarily better than another, but it may be more robust or more suitable for a particular situation. A well-known example in numerical analysis is the use of implicit solvers (only) for stiff problems (i.e., improve the stability and, therefore, ensure correctness of the solution by solving a system of equations with a much larger time step).

For instance, matching algorithms can return false positives or false negatives. Typically, this is a product of fine-tuning the algorithm's parameters, such as acceptable error tolerances. The definition of acceptable depends on the application. If the cost of investigating false positives outweighs the benefit, then it might make sense to accept more false negatives if that means having significantly fewer false positives.

In functions such as fraud detection, using a watchdog algorithm to identify fraud cases, it is not desirable to be alerted for every encounter. The algorithm is instead set to tolerate some fraud (if the cost of investigating outweighs the potential loss) and concentrate on the more important cases. The same algorithm that performs well in this situation may be unsuitable in a situation with many positives. In a situation with a handful of true cases per day, an algorithm that effectively introduces as many false negatives is still acceptable since the cases to be investigated are manageable. On the

Enjoying this article?

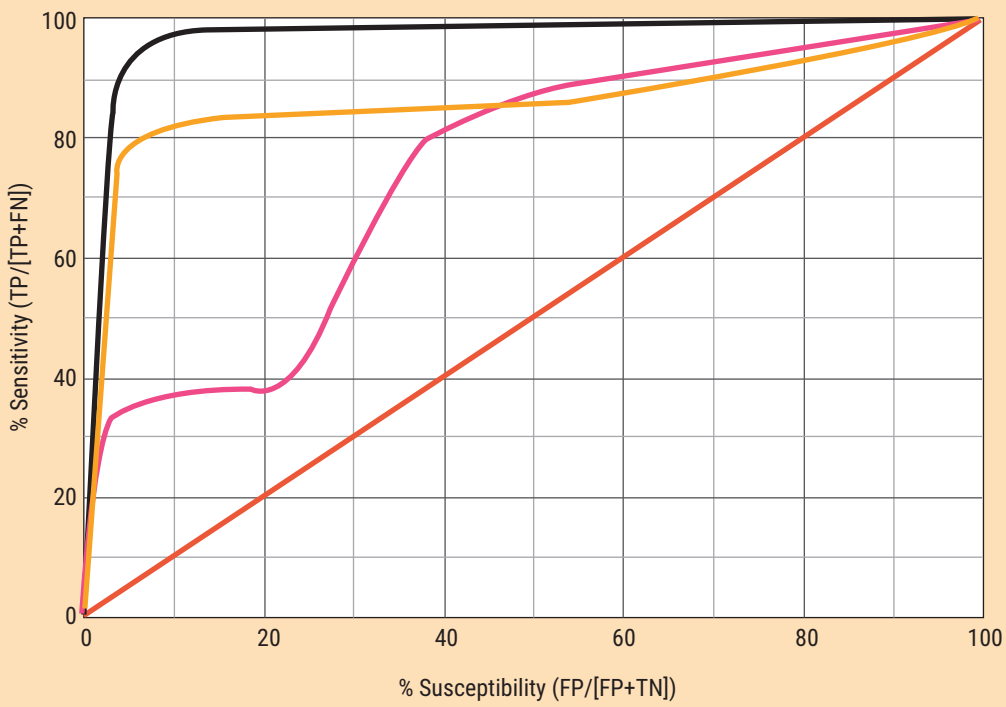
- Read *Auditing Artificial Intelligence. www.isaca.org/auditing-AI*
- Learn more about, discuss and collaborate on audit and assurance in ISACA's Online Forums. *https://engage.isaca.org/onlineforums*

other hand, hundreds of daily cases introducing as many or more false negatives may mean having to double staff to investigate the false positives.

Figure 1 illustrates the concept by comparing susceptibility and sensitivity in four different algorithms. Susceptibility is the tendency to raise a false alarm, defined as the percent ratio of false positives (FPs), or cases wrongly identified by the algorithm as exceptions or interesting, to the sum of FPs and true negatives (TNs), where cases are correctly identified as not being exceptions or interesting. Sensitivity is the percent ratio of true positives (TPs), or cases correctly identified as interesting or exceptions, to the sum of TPs and false negatives (FNs), where cases are erroneously labeled as not interesting or not exceptions. The ideal plot on this graph is the upper left corner, which indicates 100 percent sensitivity (no FNs) and zero percent susceptibility (no FPs). However, in practical application, this is not always possible, and some compromise must be made.

Auditors do not usually have much input in algorithm design unless they have designed it themselves. However, their input is valuable because when an algorithm is designed or evaluated, the data and processing for each of its potential uses must be considered. For example, a sophisticated system may measure customer satisfaction and find that all faulty equipment was replaced within one day, but this may not be the whole story. The problem may be caused by an auxiliary unit, such as a power source, that is incompatible with upgraded equipment. Failure to identify the root cause might mean that the enterprise is replacing perfectly operational and expensive equipment instead of the much cheaper power source. Algorithms can mimic human thinking but only if they are designed to do so.

Figure 1—Plotting Sensitivity vs. Susceptibility for Different Algorithms



Interpretation of the Results Is a Human Task

Algorithms can return incorrect results, either because of flaws in the algorithm itself (logic) or implementation (coding) errors.

Algorithmic (logic) errors typically result from a condition that arises during the execution that was not considered in the algorithmic design. For example, interpolation schemes (AI algorithms such as neural networks may be thought of as an interpolation scheme[3]) typically work well in the range for which data were provided (training data in the jargon of ML); however, they can easily go haywire if data outside of that range are encountered. Some logic errors, such as division by zero, will cause program failure—especially if testing is not thorough enough to consider them—but some will not. Complex algorithms make testing even more difficult and can lead to more errors, which are often exploited to hack systems.

Coding errors refer to cases where the algorithm is correct, but a coding error such as a simple typo has occurred. Logic or coding errors may be triggered by an event that is rare enough to have gone undetected during testing. This is why the response by the algorithm should ideally be accompanied by the exact raw data that it is based on and an explanation. For some algorithms, no explanation is needed. For example, a sorting algorithm's explanation is the data sorted as requested. Some algorithms can be programmed to provide the reasoning for their conclusion; however, for others, such as neural networks, it can be difficult to explain the results in terms that are easily understood by the average person.

However, assuming that neither logic nor coding errors have crept in, results must ultimately be interpreted by humans. The computer crunches numbers and returns an answer to the mathematical translation of the problem that was asked. How this relates to the actual problem may not be straightforward and depends on how well the solution to the mathematical problem answers the actual problem. For example, when checking reported shop income vs. the sum of the prices of the goods sold, it is possible to answer conclusively if there is a discrepancy. However, in comparing key performance indicators (KPIs), it is important to be careful about drawing conclusions that are not supported by the KPIs. As indicators, KPIs do not conclusively prove that there is or is not an issue. Nor do they always give insight on what is actually happening. Designing the right algorithm to prove something requires not just mathematical skills, but also a thorough understanding of the actual problem and what it will take to validate it or not. There is no shortage of readily accessible data, but if the proper steps are not taken to ensure that all necessary data are available, then key pieces of information will not be stored anywhere.

> "AUDITORS MUST MAKE SURE THAT TESTING IS DESIGNED TO ANSWER THE QUESTION OF WHETHER A TOOL BEING USED IS ABLE TO ANSWER THE QUESTION THAT THE AUDITOR IS ASKING."

Auditors must make sure that testing is designed to answer the question of whether a tool being used is able to answer the question that the auditor is asking. Enterprises typically have many tools of varying sophistication, but these are normally designed to answer operational questions, which may not be the questions auditors need to answer. For example, alarms indicating a potential problem, such as unusually heavy traffic, possible equipment malfunctions, low battery or adverse environmental conditions such as temperature or humidity, may be raised, but these may not be kept once resolved. To the auditor, trends or patterns in such alarms may be interesting, but in everyday operations there may not be time to solve anything but the immediate problem. When enterprises buy or build a new system, auditors should make sure it includes export capabilities. Using a new system may interfere with normal operations; therefore, it is ideal for auditors to have the option of obtaining the system's raw data and using the appropriate algorithm to answer the audit questions.

Understanding statistical significance is also important. For instance, a recommendation engine, a program that makes suggestions based on user data and preferences of similar users, makes an implicit assumption that it believes the user to be similar to some group. Even if this similarity is correct, a small sample in some such groups can easily result in poor statistics. In the car example, if the algorithm was trained with a set where few cars had accidents and those that did were predominantly of a certain color, an algorithm could associate a propensity to accidents for this color of car. In many cases, users only care about a definite answer from the algorithm and, as a result, algorithm designers may hide or fail to produce information that adds error bars (i.e., the results are believed to be accurate to 10 percent) or reservations to an answer if most users will discard such information anyway.

Support for the actual conclusions cannot be arbitrary, but rather must be based on information. Failure to understand the results, with all their assumptions and caveats, can be extremely problematic—especially if the need for proof is substituted by computer output. A well-known example is the use of a facial recognition algorithm on blurred pictures that wrongly identified suspects.[4] In these cases, an algorithm performed worse when the picture contrast was not as sharp, and the error was exacerbated by the police blindly acting on those suggestions. The best use of the algorithm in these cases would have been to take the suggestion of possible suspects and start an investigation, instead of directly accusing people. The same applies to audit. It is one thing to use a correlation to find indicators of possible fraud, but it is completely different to accuse people of fraud based on some correlations discovered by an algorithm. Regardless of what data were used, indicators are not proof, and justifying beliefs based on an algorithm is extremely dangerous.

Algorithmic bias has generated complaints.[5] In one case, protests forced the UK government to abandon an algorithm that was used to determine university admittance based on a student's intraschool ranking and historical school performance. Protesters argued that poorer students were discriminated against, and they had a valid point. The problem was that the algorithm was not designed to solve the actual problem. Ranking individual students made no more sense than assuming that bad teams cannot become good or that the best player on a bad team is worse than the best player on a good team. The protesters took aim at the algorithm; however, the error actually came from the person who decided to use the wrong algorithm.

The term algorithmic bias means that smart algorithms may use statistical data to discriminate against some people. One example is an enterprise's recommendation engine for evaluating applicants, which was reported to discriminate against women.[6]

> "WHETHER THEY THINK OF IT IN THESE TERMS OR NOT, AUDITORS ARE NO STRANGERS TO DEVISING ALGORITHMS."

It appears that the algorithm was using past successful and unsuccessful resumes to pick out desirable features in a candidate, and gender or gender-correlated information was one such piece of information, not unlike the children's hospital example. Even ignoring issues of statistical significance in a field where there may be too many applicants of one gender and too few of another, the key point is that if the exact, desirable qualities in a candidate are not considered, no smart algorithm can determine those qualities for an employer. Refusing to take responsibility for a decision and delegating it to a program or algorithm is an admission by the employment recruiter that the job of the recruiter could be replaced by a machine.

As theoretical physicist Eugene Wigner remarked, "It is nice to know that the computer understands the problem. But I would like to understand it, too."[7] Ultimately, everything is measured by success or failure. If the algorithms result in recommendations that are better and cheaper than human performance, then there will be strong pressure to

substitute human decisions with machine decisions. On the other hand, algorithms have no legal capacity and cannot be sued if their results are detrimental to anyone. This is a common debate and it will likely result in a compromise that algorithms must explain their decisions, and there have already been steps to this effect.[8, 9]

> "HUMAN NOTIONS SUCH AS FAIRNESS MUST BE PRECISELY DEFINED AND BUILT INTO THE ALGORITHM SINCE THEY ARE NOT SOMETHING THAT THE ALGORITHM CAN LEARN BY ITSELF."

Effects on the Auditor

Algorithms are relevant to auditors in two ways. First, auditors typically have a task (audit), which requires them to investigate and answer a problem such as whether or not a particular control or system of controls is working. To solve the problem, they need to devise an algorithm, or audit plan, that also describes how they will tackle the problem. Whether the audit plan is written or not and whether they use third-party software or write the software themselves, the auditors are using an algorithm. Whether they think of it in these terms or not, auditors are no strangers to devising algorithms. They should at least understand, in nontechnical terms, the questions that must be answered to reach a conclusion. If they cannot manage the technical aspects, they can seek technical help, but knowing what must be compared is the auditor's job.

The use of algorithms, particularly ML algorithms, to make decisions exposes enterprises to risk. Although management will ultimately make the decision on the risk it is willing to accept, it is the auditor's job to objectively inform management of the risk. Common areas of risk include:

- Errors in algorithms or their implementation and cases that may come up in practice that the algorithm has not considered. This may be diagnosed either via an analysis of the algorithm or through testing. Programs are often proprietary, and documentation, if available at all, is typically not detailed enough to understand exactly what the algorithm is doing. Even if it is, it may be complex and hard to understand for a nonexpert. User acceptance testing (UAT) is typically designed by users and focuses on demonstrating that the usual cases work. Rare cases are usually not considered, but these are most interesting to the auditor. The focus for users and auditors is different yet complementary; users focus on doing their everyday work well and efficiently, while the auditor is interested in what can go wrong.
- Correctness and adequacy of data fed to algorithms, particularly ML algorithms, to train them. Such data should be able to cover both usual and unusual cases. Auditors are concerned about poor training in some rare cases, resulting in the algorithms producing incorrect results.
- The tendency to trust the machine answer is strong but only justified if the correctness has been exhaustively tested and the machine actually answers the appropriate questions.

Conclusion

Auditors are the experts on algorithms used to solve audit problems, even if the technical aspects of the algorithm design need to be delegated to specialists. The ownership and responsibility for answering audit questions rests with auditors.

Early influence by auditors when systems are built or procured will anticipate potential needs with regard to the data and the ability of the algorithm to handle the data range of interest to the auditor and to answer the questions of interest in the audit.

When using the results of a software tool, auditors should ensure that the results provide a reasonable answer to the actual question asked by the audit and understand the assumptions and caveats that are a part of the algorithm design.

Auditors should also be aware of potential biases in the logic or data, understand how they may affect the results and understand what may be done to mitigate those biases. It is important to control the

behavior of the algorithm. Human notions such as fairness must be precisely defined and built into the algorithm since they are not something that the algorithm can learn by itself.

Endnotes

1 Teukolsky, S; W. Vetterling; B. Flannery; *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, United Kingdom, 2007

2 Armknecht, F.; C. Boyd; C. Carr; K. Gjøsteen; A. Jaschke; C. A. Reuter; M. Strand; *A Guide to Fully Homomorphic Encryption*, 2015, *https://eprint.iacr.org/2015/1192.pdf*

3 Alexiou, S.; "Advanced Data Analytics for IT Auditors," *ISACA® Journal*, vol. 6, 2016, *https://www.isaca.org/archives*

4 Hill, K.; "Wrongfully Accused by an Algorithm," *The New York Times*, 24 June 2020, *https://www.nytimes.com/2020/06/24/technology/facial-recognition-arrest.html*

5 Porter, J.; "UK Ditches Exam Results Generated By Biased Algorithm After Student Protests," *The Verge*, 17 August 2020, *https://www.theverge.com/2020/8/17/21372045/uk-a-level-results-algorithm-biased-coronavirus-covid-19-pandemic-university-applications*

6 Dastin, J.; "Amazon Scraps Secret AI Recruiting Tool That Showed Bias Against Women," Reuters, 10 October 2018, *https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G*

7 "Can Biological Phenomena Be Understood By Humans?" *Nature*, vol. 403, iss. 345, 27 January 2000

8 Smith, A.; "Using Artificial Intelligence and Algorithms," US Federal Trade Commission, 8 April 2020, *https://www.ftc.gov/news-events/blogs/business-blog/2020/04/using-artificial-intelligence-algorithms*

9 Sartor, G.; F. Lagioia; *The Impact of the General Data Protection Regulation (GDPR) on Artificial Intelligence*, European Parliamentary Research Service (EPRS), Belgium, June 2020, *https://www.europarl.europa.eu/RegData/etudes/STUD/2020/641530/EPRS_STU(2020)641530_EN.pdf*