A HUMAN FACTORS ENGINEERING EDUCATION PERSPECTIVE ON DATA SCIENCE, MACHINE LEARNING AND AUTOMATION

Daniel Hannon[1], Esa Rantanen[2], Ben Sawyer[3], Raymond Ptucha[2], Ashley Hughes[4], Katherine Darveau[1,5], and John D. Lee[6]

[1]Tufts University, [2]Rochester Institute of Technology, [3]University of Central Florida, [4]University of Illinois, [5]GE Aviation, [6]University of Wisconsin-Madison

The explosion of data science (DS) in all areas of technology coupled with the rapid growth of machine learning (ML) techniques in the last decade create novel applications in automation. Many working with DS techniques rely on the concept of "black boxes" to explain how ML works, noting that algorithms find patterns in the data that humans might not. While the mathematics are still being developed, the implications for the application of ML, specifically to questions of automation, also are being studied, but still remain poorly understood. The decisions made by ML practitioners with respect to data selection, model training and testing, data visualization, and model applications remain relatively unconstrained and have the potential to yield unexpected results at the systems level. Unfortunately, human factors engineers concerned with automation often have limited training and awareness of DS and ML applications and are unable to provide the meaningful guidance that is needed to ensure the future safety of these newly emerging automated systems. Moreover, undergraduate and graduate programs in human factors engineering (HFE) have not kept pace with these developments and future HFEs may continue to find themselves unable to contribute meaningfully to the development of automated systems based on algorithms derived from ML. In this paper, human factors engineers and educators explore some of the challenges to our understanding of automation posed by specific ML techniques and contrast this with an outline of some of the historical work in HFE that has contributed to our understanding of safe and effective automation. Examples are provided from more conventional applications using both supervised and unsupervised learning techniques, that are explored with respect to implications for algorithm performance, use in system automation, and the potential for unintended results. Implications for human factors engineering education are discussed.

INTRODUCTION

Data science (DS) is one of the hottest topics in technology circles today, with tremendous opportunities throughout the tech world, the Silicon Valley, and on university and college campuses world-wide. Along with DS comes the promise of artificial intelligence (AI), machine learning (ML), and automation. DS is affecting almost all aspects of our lives, from smart appliances and consumer products to our interactions with banks and financial services, as well as throughout healthcare.

This explosion in DS has important implications for the field of human factors engineering (HFE), particularly with respect to human-machine automation. A database search of "Engineering Village" on "machine learning" and "human factors" yielded 2 returns in 1998, 19 returns in 2008, and 86 returns in 2018. Although a low number in comparison to 31,349 for all for "machine learning" alone in 2018, it is evident that ML is having a growing influence in the field of human factors.

There is much evidence of ML and automation in our daily lives as we gather information from various search engines and go shopping online. Almost all are familiar with recommender systems. Part of what makes these interactions so compelling is how natural and accurate they appear. Many people have experienced a shopping website show exactly what they are looking for without asking it, almost magically.

But, as history has shown us, automation designed without the human user in mind can lead to unintended consequences. Are we in such a state again? What should concern. us? Are HFEs prepared to offer effective human-centered solutions?

In this paper, we review basic mechanisms of how ML works. We explore implications for the way algorithms work and the implications of the decisions that are made in the design and creation of the models. Two ML examples are considered in an effort to reveal how ML practices could result in unintended consequences in systems designed based on these algorithms. Next, we take a step away from ML, and review some of the fundamental concepts we currently teach in HF curriculum about human-machine interaction and automation. Finally, the paper concludes with a consideration of the way in which HFE training programs should be considering the inclusion of DS and ML to better prepare researchers and practitioners to engage in the design of the automated systems of the future.

OVERVIEW OF MACHINE LEARNING

The field of DS has provided us with an almost overwhelming array of techniques for digesting and processing vast data stores. Among them, ML processes provide the opportunity to identify patterns in large datasets that are beyond the perception of researchers. Often, the algorithms require multiple iterations as they work to optimize the solution to a particular objective, giving these algorithms the illusion of having "learned" something. And indeed, ML practitioners, particularly those who use neural networks, often refer to the model as a "black box" of sorts since the patterns that are found do not necessarily align with the more conventional thinking that a researcher may bring to a problem (Shukla, 2018; Ch. 1). Combined with the difficulty of representing outcomes from multi-

dimensional data spaces, the field of ML at times is characterized as a "black art", practiced by a few who are in the know and to the exclusion of others who could benefit from the use of this rich set of research tools (see Gillies et al., 2016).

For the purposes of explication, two examples of ML techniques are provided in somewhat generic form that will serve to illustrate both the techniques as well as highlight the level of "art" that is involved in the decision making in the use of these tools. It will be these decisions that will later be explored as elements of concern in the design of future automated systems.

Recommender System Example

Perhaps one of the most ubiquitous of ML applications are recommender systems. These work by first having gathered enormous amounts of data on people who are doing the same activity (e.g., shopping), and noting the co-occurrence of their decisions. In the case of online shopping, this allows the owner of the data to know how often two items are purchased, and more importantly, by whom, when, etc., which allows them to target suggestions to individual consumers. As noted previously, the results can be quite appealing to a consumer who is provided with additional suggestions that meet their needs.

A recommender system may be based on a ML technique such as collaborative filtering (Coutinho, 2016), in which a dataset of consumer purchases is parsed based on instances unique to specific users and to specific categories of users (e.g., married, college educated males, over 50, living in New England). These data are often referred to as the "ground truth" in that they represent actual purchases by people in the specific category. The collaborative filtering process is then used to identify pairings of similar purchases between different people to identify how close or far apart people are with respect to their purchasing decisions. The algorithm learns these parings, allowing the algorithm to identify the likelihood of a product being of interest to a specific category of consumer based on the attributes of the product and then be able to predict the likelihood that a consumer will want other products.

The collaborative filtering process works based on an equation like

$$\frac{\sum_{i=1}^{n} A_1 \times B_i}{\sqrt{\sum_{i=1}^{n} (A_1)^2} \times \sqrt{\sum_{i=1}^{n} (B_1)^2}}$$

where A and B represent vectors of product ratings from two different people. In a typical application, the algorithm is used to determine distances between pairs of people, which is essentially an effort to find the minimal differences across all instances in the data set. The goal is to find the people with the smallest differences to make predictions from one to the other.

In some ML applications, after learning the relationships in the initial training data with respect to those items a person in a category has purchased and not purchased, a held-out test set is used to see if the model is now able to generalize and make predictions on unforeseen data. The performance on these unforeseen data samples reveals the success of the algorithm.

It is worth noting that the process of labeling the data in the training data set is generally referred to as "ground truth classification," and requires a human to decide what elements of the data constitute a particular label. In this case, it means deciding what constitutes a purchase and what does not.

Aside from needing a large set of data for the training and test data sets, the ML practitioner is faced with a number of decisions when setting up a recommender system. These include:

- Deciding how to label the training data. What will count as a purchase? If a person buys and then returns an item, or if a person buys a specific quantity and then reduces that quantity, should that change the label? How dependent is the algorithm on the specific way the data are labeled?
- Choosing representative training data. Are the data representative of the population who are being served? Are the data representative of the situations that will be encountered in the future?
- What is the influence of the optimization parameter(s) on the solution? Would more or less regularization have produced different outcomes?

  While a recommender system for purchases may not pose a threat of imminent harm, what if it were an "expert system" in healthcare, providing treatment or therapy suggestions based on patients with similar symptoms? What are the criteria for the training and test data sets? How should the learning be monitored and the results understood?
- How might people and organizations adapt to the output of the algorithm? Might this adaptation lead to unanticipated negative outcomes?

To be fair, many practitioners in the ML world are partnering with experts in various fields to better understand the implications of their models. The previous points are mentioned to point out that without the input of domain experts, ML practitioners often are blind to the implications of their decisions.

Neural Network Example

The recommender systems just described require the person initiating the learning algorithm establish the parameters of what is to be learned. In this case it is the features that were engineered or labeled in the training dataset. In general, ML techniques come in both supervised and unsupervised variants. Supervised datasets require some sort of human intervention to assign labels to each input sample in the dataset- for example in computer vision, this might be the assigning a of label to images such as "car", "truck", "cat", or "dog". These supervised methods then automatically learn relationships between data statistics and these labels. Unsupervised methods do not require this labeling step. In unsupervised ML, the algorithm both learns statistical features as well as relationships to desired outcomes. For example, unsupervised methods can learn lower dimensional representations of data, automatically cluster data, or even pre-learn weights for downstream supervised techniques.

One popular type of unsupervised learning is based on neural networks. Neural networks are loosely based on observations of how neurons in the mammalian cerebral cortex summate and propagate signals to fellow neurons. In a multi-layer perceptron neural network, matrices are used to represent

strengths of interactions between processing layers, and summation and propagation rules are used to transmit a signal (i.e., data) between layers. During learning, the difference between the network output and the desired outcome is used to adjust the weights. The network learning is considered complete when the set of weights have been optimized such that the error between the output and the desired output are at a minimum.

One of the more popular methods of working with neural networks is through the use of TensorFlow (Ganegedara, 2018), an open source machine learning framework, developed by Google, that simplifies the processes of setting up and running machine learning frameworks, including both unsupervised and supervised neural networks. Through TensorFlow and the Python or R programming languages it is possible to setup a number of tensors (i.e., input vectors and matrices) and to scale the number of layers in the network as needed for the complexity of the computation being performed. Relatively complex image processing problems requiring convolutional filtering and recurrent time-series data have become more tractable through the creation of TensorFlow.

Layers of neural networks are separated by non-linear activation functions which enable the network to learn more complex functions. Early activation functions of sigmoid and tanh have been largely replaced with the ReLU function and its variants. The ReLU activation function clips all negative values to zero, and passes all positive values. To reduce the number of learnable weights, fully connected matrices are often replaced with convolutions filters. To create an abstract hierarchy, data reduction layers called pooling layers reduce the data resolution, allowing the convolution filters to multiple representations of the input, similar to how the visual cortex produces different representations of visual stimuli.

A process called backpropagation is used to learn the weights in a neural network. Backpropagation computes the partial derivative of the current cost or error of the entire network, with respect to each weight in the network. Modern deep neural networks may have as many as 150M weights, meaning that for each training sample, 150M partial derivatives are computed. Further, these neural networks often require millions of training samples, each repeated dozens of times. The process is so computationally intensive, that even with hardware accelerators known as general processing units (GPUs), some large neural networks can take days or months to train.

While it is beyond the scope of this paper to provide a coherent example of a neural network model, we can still describe the steps in a basic use case as the following:

1. Clean data into usable elements, such as maximizing pixel contrast in an image or parsing speech into words.
2. Standardize the inputs, such as normalizing data values or making sure all speech utterances are the same length.
3. Convert data into a quantitative expression, such as assigning brightness values or quantifying the length of a word.
4. Learn relationship between inputs and labels. This is the backpropagation step. The data modified from steps 1 - 3 is processed through the model and the relationship to a desired output is learned. The desired output could be determining whether an image contains a face, or whether a speech utterance had a positive or negative sentiment.
5. Optimize the output based on a criterion value.

At this point, the reader may notice similarities between a neural network ML process and the recommender system described previously in that a large part of the process is involved with acquiring, cleaning and preparing the data for processing. Additionally, the output is dependent on the choice of criteria used to manage the learning process.

APPLICATION OF ML TO AUTOMATION

The foregoing descriptions and discussions of ML techniques serve to demonstrate two fundamental concerns. First, despite the advances in the quantitative approaches to ML, there are a number of steps requiring human decision making in the development of a ML-based model. Everything from the choice of the dataset, to the cleaning and labeling of the data in the training set, to parameters used to optimize the model require the skill of the ML practitioner. While the numerical methods are well-described, these decisions require thoughtful planning and comparison. Without careful cross-validation, the algoithms may appear to perform very well, but then perform very poorly when exposed to new data.

Second, the fidelity of the data and the labels to the real world that is being modeled is of central importance to the validity of the model. The availability of DS tools, such as TensorFlow and the ubiquity of processing language tools such as Python, put ML into many hands, potentially without the support of subject matter experts to guide the selection of data and labels. This leads ML practitioners, in some cases, to claim to be studying complex subjects, such as disease states, without requisite background knowledge within the medical field.

To further illustrate these concerns, there is a specific problem in the training and evaluation of ML models that is concerned with the number of parameters on the training set during learning. When a complex model is provided too many parameters it is possible to "over fit" to the data, similar to problems of having too many parameters in a linear regression model . That is, the model has learned every nuance of the training data set and has perfect or near perfect classification performance. Unfortunately, its performance on unforeseen data sets may be well under what it achieved with the training set. To prevent such problems, the unforeseen test data performance needs to be monitored throughout the training time along with the training set performance. Further, the test dataset statistical properties should match the training data, both of which should match what is expected when the trained model is released in product. For example, if all training and testing data was done with high resolution cameras, and the model is deployed using low resolution cell phone cameras, performance will degrade. Knowing when training is enough, knowing how to construct training and test sets, and understanding the statistical properties of each are key elements to the design of a successful ML model.

What HFEs Know About Automation

We turn now, briefly, to a summary what the field of HFE has already learned from decades of research on human-centered automation and from studying the failures when humans

are not integral in the design process. The syllabus of an Engineering Psychology course taught by one of the authors includes a segment on automation that is largely based on available texts (e.g., Wickens, Hollands, Banbury, & Parasuraman, 2015; Lee, Wickens, Liu, & Boyle, 2017, and earlier editions), and several articles and chapters (e.g., Fitts, 1951; Sheridan, & Verplank, 1978; Sarter, Woods, & Billings, 1997; Parasuraman & Riley, 1997; Parasuraman, Sheridan, & Wickens, 2000). However, as can be seen in the sampling of references common in HFE courses taught in universities, they dealt with what may be termed as "first-generation" of automation. Even then, technology has always (or as long as there has been technology) been running well ahead of HFE, and the history of HFE has been that of playing catch-up. A pertinent question is if there are "lessons learned" from past experience that could be applied to AI and ML, or is the history merely repeating itself?

HFE Contributions to the New Era of Automation

Given the history of contributions to the world of automation, particularly in human-machine systems, it is worth spending some time reconsidering what we know about human-machine system automation in light of the advancing push of ML and automation. First, with respect to when to automate, there clearly are differences between the human factors side and the ML side. As noted earlier, there is an explosion of applications of ML throughout the sciences and technology. Rather than attempting to automate when needed for safety or efficiency, it would appear that ML approaches are being taken because they are possible since data are available without the need to determine if there is a need for the solution for safety or efficiency.

The question of what is being modeled through ML for automation does appear to cover the concerns of the HFE. Dangerous and tedious tasks, such as those that occur in driving are actively being developed, as are approaches to problems that exceed human capabilities, such as finding patterns in high-dimensional data spaces. Yet even here, part of the challenge remains in how to keep the human user or operator engaged and knowledgeable about what the system is doing.

From reading many accounts of ML approaches, it is tempting to think that the human is being phased out. Yet, there is a far distance between the current recommender systems and a future system that automatically knows what we need and makes the purchases for us. Unfortunately, many ML projects are not considering a hybrid solution that uses a human operator as one of the processing elements, or as an interpreter of the output. There are exceptions where "usable machine learning" is being considered (Fiebrink & Gillies, 2018, Gillies et al., 2016). For example, before fully autonomous vehicles will be unleashed, there will be many iterations of human-assisted ML models. Cruise control systems have been augmented with ML-based automated braking and vehicle following. Driver assistance such as drowsiness detection, lane detection and following, blind spot detection, and high-beam light switching are all possible because of ML. Although full autonomy is still only possible in constrained scenarios, these ML methods are aiding the human driver, making driving safer.

We know from experience that there are unintended consequences from overreliance on automation. The primary problem with AI and ML is that they are not transparent. "Clumsy Automation" and "Automation Surprises" terms were coined in the 80s and the concepts are applicable to AI/ML, too. Human operators (mostly pilots in the 80s and 90s) ran into trouble when they could not understand what autopilots were doing, and those were more straightforward, deterministic systems. Today's algorithms are more capable, dangerous, and can cause more operator confusion (as recently illustrated with the Boeing 737 flight accidents). Automation based on AI/ML will be completely opaque and its actions inscrutable by humans.

Back in the 80s there were plenty of high-profile, tragic, aircraft crashes that were scrutinized and that brought up the problems with autopilots that were beyond pilots' training. If AI-based diagnostic systems misdiagnose one patient at a time at some (relatively low) rate, nobody will pay attention, or bother to collect such instances as data. There may be incident reporting systems in healthcare that could be accessed and analyzed, but that is just one domain of AI/ML applications.

There is no reason to believe that this will not continue to be true in the world of ML models that drive automation. In particular, as ML models are able more to mimic elements of human behavior and decision making, there may be an increased tendency to trust these models beyond a prudent level. Reconsider the recommender system that suggested the right additional products to solve someone's shopping needs. When the system starts acting like it has unique knowledge it becomes easier to trust, and harder to decide to counteract when needed.

In addition to the influence of AI/ML substituting human decision making in high risk industries is the impact of AI/ML on safety and operational data analysis. Traditional techniques rely on event data to understand causal factors and develop solutions to prevent recurrence. This approach often is limited by the volume of data and, therefore, the availability of applicable and effective models and tools. ML adds the dimension of non-event data, from which a more meaningful algorithm can be derived to detect patterns across a variety of features that may impact an event outcome. ML in the context of safety and operational data collection, processing, analysis, and automated action is an area for further exploration in high-risk industries.

Educating HFEs for Roles in Automation and ML

Although teaching about human-machine systems and automation is a fundamental element of a human factors engineering education, as practitioners and educators in this field, the authors find that the current curriculum has not kept pace with the advances in ML and DS in general. In fact, there are counterproductive forces stemming from the push toward understanding the user experience that lead away from understanding automation and toward a greater trust of automation. Yet, it is our belief that as with the automation failures of years past, HFEs will continue to be called upon to address human-system automation design of the current and future ML era.

Toward that end, the following ideas are offered as modifications to the education of human factors engineers in an effort to better prepare them for active participation in the design of the ML-driven automation projects of the future:

1. Human factors engineers need an understanding of DS. This may be taught within existing courses on behavioral

data analysis or analytical methods or may require a unique course. Concepts such as unsupervised and supervised learning and cross-validation may be naturally included in advanced statistics courses.

2. Working with large data sets requires familiarity with some type of programming tools. While a deep knowledge of C++ may not be needed, scripting languages such as Python, MATLAB, or R are needed to manipulate datasets, visualize data, and to make sense of ML outputs.
3. As effective ML practitioner, students must have an introduction to the array of ML models and their applications. DS problems will vary with input and outcome data, which will direct the ML approaches to consider. Decision trees to guide the choice of modeling techniques can be effective tools, followed by learning methods. Real-world examples can highlight the impact of ML on prediction compared to traditional techniques, and expose model limitations and ways to address them.
4. Many ML algorithms can be described in mathematical symbology, often times utilizing calculus and/or differential equations. Indeed, the concept of optimization utilizes these advanced mathematical concepts and without them requires extensive narrative to explain the process. However, it is the drive toward optimization that creates some of the unique difficulties and opportunities for contributions by the human factors engineer. Understanding advanced mathematics is not a factor behind the choice of a human factors career. Therefore, we need to efficiently and systematically describe optimization in terms understandable by a non-mathematical audience. This will expose opportunities for HFEs to make more contributions.
5. In general, the "usable machine learning" movement has espoused greater emphasis on how to frame ML questions, how to understand what is going on with processing iterations, and how data are being visualized. More research will lead to better teaching. This could include development of specific content in ML techniques with annotation and exploration of the implications of the optimization on human decision making. For example, work through multiple versions of a problem with different data label definitions to understand how the labeling affects the outcome.
6. Increase the challenge to students of HFE to become better consumers of the ML and automation literature and prepare them to participate effectively in ML projects. Make it an imperative that they know what questions to ask.
7. Initiate student-led research into turning user requirements into ML considerations. These results can better inform how the initial data set is created and how features or desired outcomes are labeled.
8. Initiate student-led research into designing a human/ ML system? Are there alternatives to a completely optimized approach? Designing a ML system for use by a non-DS practitioner is the perfect opportunity for an HFE student.

CONCLUSION

It is noted that existing courses within a university may be sufficient for meeting these goals. However, it might be more appropriate to develop the unique content for HFEs. The purpose of this paper is to initiate a broad and vivid discussion on new forms of automation afforded by ML and AI. This discussion should start with a review of the history with automation and HFE, and then approach the era of AI/ML within this cautionary framework. We wish to highlight the significant challenges ML/AI present to human-centered technology. We look forward to the furthering of the discussion over time to highlight how HF may or may not be participating.

A topic this large needs a multi-year approach. After the initial discussion this year, we hope to bring a large number of the best and brightest minds in our community to address both the apparent and potential problems with ML/AI-based automation. In subsequent years, we will bring experienced ML/AI practitioners/modelers to discuss their work in ML and automation and openly debate the interactions with human users. Other topics to be explored include the HF/E training concepts around ML and AI that we need to start providing to future HFEs.

REFERENCES

Couhtino, R. (2018). Ghost in the Machine: Learning Recommender Systems, downloaded from: https://www.outsystems.com/blog/recommender-systems.html, Mar 18, 2019.

Fiebrink, R. & Gillies, M. (2018). Introduction to the special issue on human-centered machine learning, *ACM Transactions on Interactive Intelligent Systems*, 8(2).

Fitts, P. M. (Ed.) (1951). *Human engineering for an effective air-navigation and traffic-control system*. Washington, DC: National Research Council.

Ganegedara, T. (2018). *Natural Language Processing with TensorFlow: Teach language to machines using Python's deep learning library*, Packt Publishing.

Gillies, M. et al. (2016) Human-centered machine learning. *Proc. Conference on Human Factors in Computing Systems*, v 07-12-May-2016, pp. 3558-3565

Lee, J. D., Wickens, C. D., Liu, Y., & Boyle, L. N. (2017). *Designing for people: An introduction to human factors engineering*. CreateSpace.

Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human Factors, 39*(2), 230-253.

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 30*(3), 286-297.

Sarter, N. B., Woods, D. D., & Billings, C. E. (1997). *Automation surprises*. In G. Salvendy (Ed.), Handbook of Human Factors and Ergonomics, 2nd ed., pp. 1926-1943.

Sheridan, T. B., & Verplank, W. L. (1978). *Human and computer control of undersea teleoperators.* Massachusetts Inst of Tech Cambridge Man-Machine Systems Lab.

Shukla, N (2018). *Machine Learning with TensorFlow* 1st Edition, Manning Publications

Wickens, C. D., Hollands, J. G., Banbury, S., & Parasuraman, R. (2015). *Engineering Psychology and Human Performance*. Psychology Press.