NET-ID: JPM190001

ML OVERVIEW

from the perspective of Jason P. McElhenney

To me, Machine Learning has always been some sort of oxymoron superposition: after all, what truly separates a human from a machine if not the ability to learn from experience? With this question always simmering at the back of my mind when researching the topic of machine learning, my interest was piqued continuously.

As I use it today, Machine Learning is synonymous with the use of neural networks of some kind coupled with a set of human-labeled training data and a training process which allows the network to be tuned to recognize underlying patterns within the training set, and is thereby able to reliably recognize these patterns within organic unlabeled input data. The term given to the practice of a piece of software being fed training data in order to minimize a cost function which evaluates and modifies initially arbitrary weights and the bias of each layer within the network could then, perhaps, be the actual process of a “machine learning.”

Of course, any computer science student certainly–or really any adult with an internet connection–has at least heard of this touted revolutionary technological advancement. It’s mainstream adoption by industry leaders such as Google for the verification of human users and rejection of automated ones with their game-changing Captcha tool has led to the concept of training a neural model on your data for purposes of complex recognition, classification, generation, bounding, or all of the above being a staple in any proficient developer’s toolbox.These tools have underpinned the advancements we see today as the bleeding edge of what the digital age can do, facilitating among other use cases the vast improvement in self-driving technology, natural language processing, and hyper-targeted online advertising campaigns. With the number of neural models used in technology that could at the extreme potentially save or lose a human life, it is obviously crucial that we go to great lengths to validate our model results on completely new data so as to evaluate its accuracy. The world changes all around us, and therefore so too must these new algorithms continue to update their neurons in response to new classified data in order to keep up with the current use of the technology.

Many people classify these trained neural networks as artificial intelligence, or simply AI for brevity; however, the term actually encompasses a wide range of ideas, and will vary drastically in definition from one individual to the next. Personally, I dislike the use of AI as a shorthand for neural networks as it considerably reduces the scope of the term to cover only one subset of what AI could be accepted to mean. To me, a much better phrase to describe these trained models would be Machine Learning, or ML, which to me gives a much more intuitive idea to an uninformed listener as to the nature of the underlying technologies at play in a system. This is also encouraged by the fact that I tend to define Artificial Intelligence as having a requirement of at least basic sentience, which is just as impossible to quantify as it is to manufacture with a neural network. I would also personally require AI to include a sense of self or personal existence.

Returning to two examples from the beginning of this document, let us start with the revolutionary self-driving technology available even in personal vehicles. This sort of pseudo-intelligent, self-steering behavior would simply be infeasible to accomplish without the use of trained neural network models. A slew of data are provided as input to the steering, braking, and gas pedal intensity/direction evaluation, including but not limited to multiple LiDAR sensor levels for depth assessment, as well as the pixels and intensity levels within an image of the road and surroundings in front of the car. These data must be evaluated accurately within the span of a second, outputting potentially minute or very extreme instructions as to the car's control system, which is simply an impossible analysis problem for simple algorithms to assess and report back accurately, especially within the necessary time constraints.

We can examine a similar problem within my second previous example, the study of Natural Language Processing, or NLP. While this field certainly existed before the popularization of trained neural type models, the introduction of the training process to the field resulted in an unprecedented increase in accuracy, evaluation speed, and breadth of classifiable inputs. While previous methods make attempts at recognizing patterns as features within characters (or syllables) of a language, these algorithms were ultimately unreliable due to the variance between hand-writing idiosyncrasies, fonts, or accents and pronunciations. Furthermore, these human designed classifiers were limited by the features which could be identified manually by the algorithm's designer. Machine Learning models are not subject to this limitation, as weights and the bias of the input and all internal layers can become as finely tuned to externally unknown features as the size of the human-labeled training dataset can provide to the training (cost-function minimization) algorithm.

Within Machine Learning we see two fundamental parameters become the most prominent when evaluating methods of training, as well as for general description of any network itself: the feature, or predictor, and the observation, or label. If we consider the classic example of the MNIST hand-drawn numeral dataset classifier model, we can evaluate a feature as an indivisible unit containing a distinct pattern: with this example, perhaps this is the two unequal length lines that make up a four, or the curvature relationship between the two concave circle segments making up a three. Any of these features, and indeed any number of them, can then be identified with an observation. An observation is simply the association of a known set of output layer weight values to a particular feature or group of features. Frequently the quantity of features is used as an input layer size, and the quantity of observations as an output size. While this approach is commonly practiced, it is not always accurate or meaningful to all potential problems, with in my opinion the largest flaw in blindly following this guideline being that frequently the observations, or output layer, is conceived in structure before it is effectively determined whether or not this will accurately convey all the potential features within a given input data set. Along with this common practice, we also come upon the inherent distinction between quantitative and qualitative data when choosing a neural network structure. Classification is the action of placing an input data set into one of any number of entirely mutually exclusive classes–which are themselves divided based on qualitative differences observed in the data. A common example is color, where a network might be trained to sort images of (normalized in size) sports balls based on their main color. The qualitative differences between each ball in turn determines its class distinction. Quantitative then aims to provide a continuous output, with no implicit requirement that mutual exclusion be enforced or even acknowledged.

I have personally always been entirely fascinated by the concept of neural networks. My first proper research paper reads were adversarial generative neural network analysis reports by the University of Texas, and I have gone as far as to participate in the rite-of-passage that is creating a simple convolutional neural network to classify the MNIST handwritten digit dataset[1]. Overall, though I haven't actually utilized any trained models of my own in production projects, (as my work experience mainly consists of web development for small businesses and my own hobby programs thrown together in Java) I am eagerly awaiting an opportunity to do so… as long as the model would serve a both realistic and useful purpose[2].

[1] This was developed *almost* from scratch, however I decided I didn't want to spend an entire afternoon writing linear algebra functions and used one of the popular libraries for this task.

[2] I'm certainly not looking forward to labeling the training / testing data sets though. Talk about grunt work.