Autonomous Real-Time Deep Learning

Charles Davi

September 17, 2019

Abstract

In a previous paper [1],[1] I introduced a new model of artificial intelligence rooted in information theory that can solve deep learning problems quickly and accurately in polynomial time. In this paper, I'll present another set of algorithms that are so efficient, they allow for real-time deep learning on consumer devices. The obvious corollary of this paper is that existing consumer technology can, if properly exploited, drive artificial intelligence that is vastly more powerful than traditional machine learning and deep learning techniques.

1 Introduction

In [1], I introduced a model of artificial intelligence that has a worst-case runtime that is always a low-degree polynomial function of the number of items in the dataset.[2] Rather than make use of statistical approximation, or neural networks, the model makes use of information theory to achieve data compression, and vectorization to achieve operational efficiency. In this paper, I'm going to introduce a set of related algorithms that skip the data compression, and instead make maximal use of vectorization. The result is a set of algorithms that have a near constant run time, and show unambiguously that existing consumer technology can support autonomous deep learning in real-time.

To follow along, simply download Octave from the official GNU website here, and download the related algorithms, which are available on both my technical blog here, and my code-bin here.

[1]C. Davi, *A New Model of Artificial Intelligence*(2019).

[2]The measure of runtime used in [1] counts the number of built-in MATLAB functions called by an algorithm.

2 Data Classification

A **data classification** problem takes the items in a dataset, and assigns them to some predefined set of categories. For example, below we'll apply my classification algorithms to a dataset of handwritten numbers stored in image files, meaning that the algorithm will read the image file, and identify the digit it contains as a number from 0 to 9. Traditional machine learning and deep learning techniques solve classification problems by first having a data scientist select a **model**. The model could be a simple equation, or a complex neural network, but in either case, the model is then **trained** by running another algorithm that optimizes the performance of the model based upon a **training dataset**. Once the model generates sufficiently accurate predictions using the training dataset, it is then applied to the **testing dataset**. Generally, the **accuracy** of a model is measured using the percentage of correct classification predictions generated when the model is applied to the testing dataset.

The model of artificial intelligence I presented in [1] is autonomous, in that there is no need for a statistician or data scientist to select a mathematical model. Instead, the algorithms will, on their own, generate a model, based upon the dataset. This eliminates the need to spend time selecting the best model for the task at hand. The commercial motivation for my work is obvious: if you use my algorithms, then you don't need as many data scientists, which reduces costs. In fact, it's so simple, I recently turned this entire process into a point-and-click application that anyone can use.[3] However, it still takes some time for the algorithms to actually run, normalize the data, and build a model. This process is nonetheless radically more efficient than any other machine learning or deep learning algorithms that I'm aware of, and always terminates in low-degree polynomial time. This means that a machine learning task that could ordinarily take hours, could take just a few minutes. It also means that problems that typically require an industrial powered machine can be solved quickly and reliably on a cheap consumer device.

The model of artificial intelligence I'll present below can solve complex problems instantaneously, turning cheap consumer devices into truly intelligent machines.

2.1 Batch Classification

We'll begin by applying the classification algorithm to the Ionosphere Dataset,[4] courtesy of the UCI Machine Learning Repository. Running the classification algorithm once produced an accuracy of 97.143%, and took 0.0667388 seconds.[5]

[3] You can download the application Prometheus here.

[4] The Ionosphere Dataset is available for download here.

[5] All runtimes were generated on an iMac with a 3.2 GHz Intel Core i5 processor.

Running the classification algorithm 250 times on randomly selected subsets of the Ionosphere Dataset generated the accuracies shown in Figure 1 (left), and the runtimes shown in Figure 1 (right). The runtime is in this case the amount of time it takes to classify the entire testing dataset. The average accuracy was 86.400%, and the average runtime was 0.0030959 seconds.

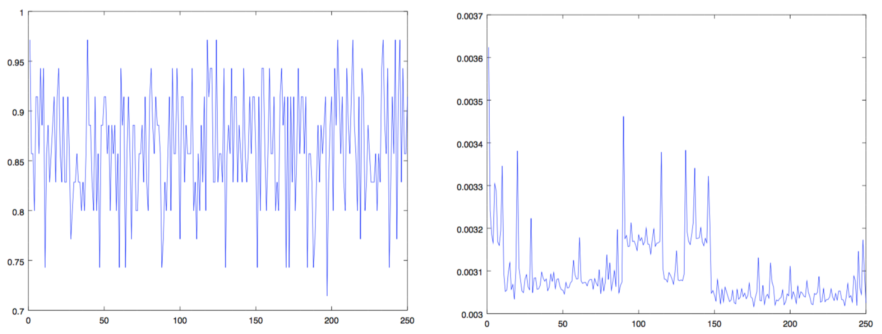

Figure 1: The accuracies of the classifications for each iteration (left). The runtime of each iteration (right).

For context, ordinarily, solving a machine learning problem of this type requires a human being to select a model, and then train that model on the dataset in question. This means that, at a minimum, a human being would have to be involved in solving the problem. In contrast, my real-time classification algorithm can autonomously, and instantaneously, without any prior training time or model selection, solve this problem, producing accurate results.

2.2 Real-Time Classification

We can simulate real-time learning by building up a dataset through "observations". That is, we treat the actual dataset like a buffer and make predictions based upon only those items that have been read from the buffer. This will cause the dataset to grow over time. As a result, the real-time classification algorithm assembles a single dataset of observations, which is then used to make predictions.

As an example, I've applied the real-time classification algorithm to the MNIST dataset, courtesy of the National Institute of Standards and Technology, though this particular version of it was uploaded by a third-party.[6] The

[6]You can download the dataset here. Note that you will have to adjust the directory and file names to match the instance of the download. Simply check the directory to which you've downloaded the dataset, take a look at the formatting of the individual files, which should be numbered in some consistent manner, and adjust the code I've provided accordingly.

individual unprocessed images are assumed to be available in memory, simulating reading from a buffer.

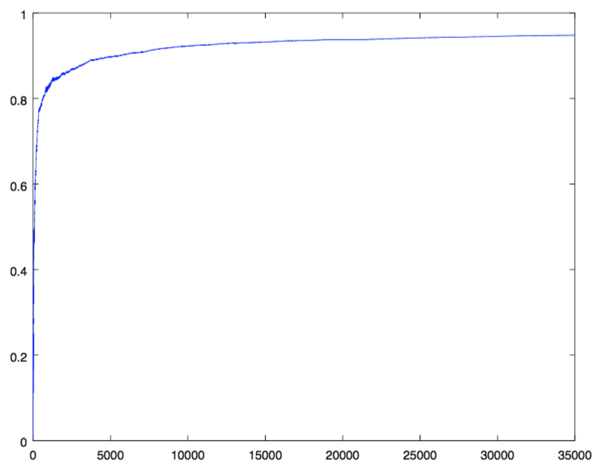

Figure 2: The accuracy of the classifications as a function of the number of observations.

The average runtime is 0.044219 seconds per prediction, which means the algorithm is reading and classifying 22.615 characters per second. The accuracy peaked at 94.794%, and Figure 2 shows the accuracy of the predictions as a function of the number of observations.[7] As you would expect, the predictions get better as a function of the number of observations.[8] Stated informally, the algorithm makes better predictions as it observes and learns more. Although this problem involves numerical character recognition, which is not as complex as distilling meaning from a collection of words, the bottom line is that this algorithm allows a consumer device to autonomously recognize characters at about the same speed that a human being reads.

2.3 Real-Time Image and Video Classification

The real-time classification algorithm can also be used to classify image and video files, at an average rate of approximately 3 frames per second. I've included an example that consists of 44 video files, where each video consists of 10 frames of HD images, which are roughly 700 KB per frame.[9] The individual

[7]I used $35,000$ images from the MNIST dataset. With a larger sample size, higher accuracy is likely possible.

[8]The algorithm includes optional code that automatically stops learning once the desired accuracy is achieved. If you'd like to disable this, set the "threshold" variable to any value greater than 1.

[9]The files for the video classification are available here.

unprocessed frames are assumed to be available in memory, simulating reading from a buffer.

The particular task solved by this algorithm is classifying the gestures made in each video: I raise either my left-hand or my right-hand in each video, and it is up to the algorithm to figure out which gesture I made. This algorithm requires no prior training, and learns on the fly as new frames become available. The accuracy is in this case 97.727%, in that the algorithm correctly classified 43 of the 44 videos. Accuracy for this problem typically varies from approximately 95% to 100%.[10]

Note that the algorithm doesn't know anything about left-hands or right-hands before it starts to run. Instead, the algorithm autonomously distinguishes between categories of gestures on its own, at least when the motions are sufficiently distinct, as they are in this case. The time-stamps printed to the command line represent the amount of time elapsed per video classification, not the amount of time elapsed per frame. To obtain the average time per frame, simply divide the time per video by 10.

3 The Need to Regulate AI

Though there is some novelty to these algorithms, unlike my previous model of AI, which is rooted in information theory, the code for the model above is incredibly simple. Though I'm not a patent lawyer, the basic idea is so banal, that I'm not sure it can even be patented - you simply look at the dataset, and take whatever prior observation is most similar to the current observation. All I've done in this implementation, is to do exactly that as efficiently as possible. As a result, this paper implies quite plainly that even cheap consumer devices can be easily transformed into powerful devices that are capable of possibly dangerous behavior. Algorithms of the type described above can probably be installed either as hidden hardware or software on consumer devices, given how little code they contain. Given the complexity of modern supply chains, and the fact that a significant amount of manufacturing and assembly takes place in states with which the U.S. has an adversarial relationship, this seems like a significant risk to national security.

Frankly, it shouldn't be possible for me to be conducting this kind of research without some kind of prior registration with the U.S. government. That said, I don't think we should limit the creativity of independent entrepreneurs and scientists at all, since the tech market is already over-consolidated, and anti-competitive. Instead, I think U.S. citizens doing research in A.I. should be registered with some agency of the U.S. Government, and have access to

[10]Even though the dataset is fixed, the algorithm has no information to use when making its first prediction, forcing it to guess, which creates some inherent randomness.

regulators that can advise them in the event that their research might pose a risk to the public.

Finally, my work also suggests that industrial scale computing is probably far more powerful than the public is aware of, and true machine intelligence might already be a reality. If this is the case, then it would imply that private firms have computers that could have military applications, with the ability to accurately predict markets, weather systems, and even the behavior of the human body, as well as engage in espionage through code-breaking, harassment, and spying that could be very difficult to detect. This type of power, like all other private power, needs to be disclosed, understood, regulated by the government, and explained to the public. We are probably decades behind in regulating private computing, and if true, this could pose a serious risk to the safety and well-being of the American people.