Vectorized Deep Learning

Charles Davi

November 18, 2020

Abstract

Many consumer devices can be used to perform parallel computations, and in a series of approximately five-hundred research notes,[1] I introduced a new and comprehensive model of artificial intelligence rooted in information theory and parallel computing that allows for classification and prediction in worst-case polynomial time. This results in run-times that are simply incomparable to any other approach to A.I. of which I'm aware, with classifications at times taking seconds over datasets comprised of tens of millions of vectors, even when run on consumer devices. Below is a summary of the results of this model as applied to UCI and MNIST datasets, as well as several novel datasets rooted in thermodynamics. All of the code necessary to follow along is linked to below.

[1] All of the research notes and applicable code are publicly available on my ResearchGate Homepage.

1 Vectorized Deep Learning

To follow along, simply download and install Octave, which is free, and download my A.I. library, which includes all of the code referenced in this article. Note that my A.I. library may be NOT be used for commercial purposes.[2]

1.1 Data Classification

UCI Ionosphere Dataset

This algorithm effectively iterates through increasing levels of discernment, until it finds the level that generates the greatest change in what would be the perceived structure of the dataset. This instance of the algorithm is fully vectorized, with only a hypothetical number of iterations.

- **Size**: 351×34.
- **Task**: Unsupervised Clustering.

- **Average Accuracy**: 99.972%.
- **Runtime**: 0.14465 seconds.[3]

The accuracy reported above is the average accuracy across all clusters. For a given cluster, the accuracy is calculated by counting the number of errors in the cluster, and dividing by the number of rows in the dataset, since the clusters are not mutually exclusive. This ratio is then subtracted from 1. The average number of elements per cluster is 6.9259, the minimum number of elements is 1, and the maximum is 46. The minimum accuracy is 98.006%, and the maximum accuracy is 1.

UCI Wine Dataset

This algorithm is the same as the one used for the Ionosphere Dataset above, with an additional step that first normalizes the dataset.

[2]I retain all rights (copyright and otherwise) to all of the materials, algorithms, and all other works published in this article. The algorithms in my A.I. library are NOT to be used for commercial purposes without my express, prior, written consent. For the avoidance of doubt, you may NOT modify, or redistribute any of this material, in particular the algorithms, without my express, prior, written consent.

[3]Run on an iMac 3.2 GHz Intel Core i5.

- **Size**: 178×13.
- **Task**: Normalization; Unsupervised Clustering.

- **Average Accuracy**: 95.881%.
- **Runtime**: Normalization, 0.925937 seconds; Clustering, 0.0356669 seconds.[4]

The accuracy reported above is the average accuracy across all clusters. For a given cluster, the accuracy is calculated by counting the number of errors in the cluster, and dividing by the number of rows in the dataset, since the clusters are not mutually exclusive. This ratio is then subtracted from 1. The average number of elements per cluster is 2.3708, the minimum number of elements is 1, and the maximum is 9. The minimum accuracy is 98.315%, and the maximum accuracy is 1.

MNIST Numerical Dataset

This algorithm is a fully vectorized implementation of the nearest neighbor method.

- **Size**: $2,500 \times 121$.[5]
- **Task**: Unsupervised Classification Prediction.

- **Accuracy**: 93.60%.
- **Runtime**: 52.4577 seconds.[6]

The accuracy reported above is the accuracy over all predictions, calculated by counting the total number of prediction errors, and dividing by the number of rows in the dataset. This ratio is then subtracted from 1.

[4]Run on an iMac 3.2 GHz Intel Core i5.

[5]$2,500$ images from the dataset are first loaded into memory, and then processed, prior to clustering, generating a $2,500 \times 121$ matrix. The runtime listed below is the runtime for only the clustering algorithm itself. For an explanation of the entire process, see, “A New Model of Artificial Intelligence: Application to Data”.

[6]Run on an iMac 3.2 GHz Intel Core i5.

1.2 Massive Datasets

The algorithms used in this section were developed to allow for deep learning techniques to be applied to thermodynamic systems, making use of both fully and partially vectorized algorithms. They are, however, generalized algorithms that likely have applications in other areas of study.

Two-State Gas Dataset

This dataset consists of two classes of collections of vectors:

(a) one representing the particles of a gas in a compressed volume, and

(b) another representing the particles of the gas in an expanded volume.

These two classes are intended to represent the two possible macrostates of the gas, compressed or expanded. Each class consists of 50 configurations, for a total of 100 configurations, intended to represent the microstates of the gas. Each configuration consists of $15,000$ vectors. The classification task is to cluster the 100 microstate configurations in a manner that is consistent with the two hidden macrostate classifiers, compressed or expanded.

The algorithm applied to this dataset also iterates through increasing levels of discernment, like the algorithms used in Section 1.1 above. However, this algorithm makes use of a vectorized operator that can quickly compare two large collections of vectors, as single operands, in turn allowing for the efficient comparison of microstates of complex systems.

- **Size**: $1,500,000 \times 3$.
- **Task**: Identify the macrostates of a gas.

- **Accuracy**: 100%.
- **Runtime**: 15.6721 seconds.[7]

The accuracy is calculated by counting the number of classification errors, and dividing by the number of rows in the dataset. This ratio is then subtracted from 1.

Expanding Gas Dataset

[7] Run on an iMac 3.2 GHz Intel Core i5.

This dataset consists of two classes of sequences:

(a) one representing the particles of a gas expanding at a slow rate, and

(b) another representing the particles of the gas expanding at a fast rate.

Each sequence of expansion consists of 15 observations, and there are 600 sequences. Each observation consists of $10,000$ vectors, representing the particles of the gas. The classification task is to cluster the 600 sequences in a manner that is consistent with the two hidden classifiers, slow or fast.

The algorithm applied to this dataset first compresses the dataset, by sorting and then embedding the dataset on the real number line. The sorting algorithm again makes use of a vectorized operator that can quickly compare two large collections of vectors. Then, a clustering algorithm similar to the one used in Section 1.1 above is applied to the embedded dataset. The bulk of the work done by the algorithm is sorting the dataset, ultimately allowing the dataset to be compressed from a $90,000,000 \times 3$ matrix, into a 600×15 matrix.

- **Size**: $90,000,000 \times 3$.
- **Task**: Identify the different rates at which a gas expands.

- **Accuracy**: 100%.
- **Runtime**: Sorting, 21.242 minutes; Embedding, 49.8727 seconds; Clustering, 0.0923202 seconds.[8]

The accuracy is calculated by counting the number of classification errors, and dividing by the number of rows in the dataset. This ratio is then subtracted from 1.

Statistical Spheres Dataset

This dataset consists of some fixed number of K spheres in Euclidean 3-space, each of which consists of some number of points, producing shapes that are not solid, but nonetheless visually distinct. The classification task is to cluster the points in a manner that is consistent with the K hidden classifiers, representing the K distinct objects in the space.

The algorithm applied to this dataset also iterates through increasing levels of discernment, like the algorithms used in Section 1.1 above. However, this algorithm makes use of a different technique that can quickly cluster large collections of low-dimensional vectors.

[8]Run on an iMac 3.2 GHz Intel Core i5.

- **Size**: $1,048,724 \times 3$.
- **Task**: Identify and cluster objects in Euclidean 3-Space.

- **Accuracy**: 100%.
- **Runtime**: 114.036 seconds.[9]

The accuracy is calculated by counting the number of classification errors, and dividing by the number of rows in the dataset. This ratio is then subtracted from 1.

1.3 Other Algorithms

As noted above, the balance of my work is available on my ResearchGate Homepage, which includes A.I. algorithms for images, videos, and object tracking, as well as others specific to physics.

[9]Run on an iMac 3.2 GHz Intel Core i5.