Black Tree AutoML

Humanity's Fastest Deep Learning Software

Black Tree AutoML

Black Tree is the fastest machine learning software known to mankind.

It is also fully autonomous, and does not require a data scientist to run.

Anyone can use it.

Black Tree
vs.
Traditional Machine Learning

Black Tree is among the most accurate machine learning software on the market:

Dataset	Black Tree Accuracy
UCI Iris	95.65 %
UCI Wine	96.70 %
UCI Ionosphere	94.11 %

For context, traditional machine learning methods achieved accuracies that were generally in the range of 90.00 % to 94.00 %, when applied to 121 UCI datasets.*

*SOURCE: "Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?"

How did you do it?

How did you do it?

Answer: Parallel Computing, Compression, Information Theory

Serial vs. Parallel Computation

For example:

Calculate, A = 1+2+4+5+7+9

Serial Computation

```
Add each term in order

     step 1:1+2
     step 2: +4
     step 3: +5
     step 4: +7
     step 5: +9
     __________
     result: 28
```

Parallel Computation

```
Add each pair simultaneously

step 1:          1+2; 4+5; 7+9
step 2:                    3+9
step 3:                    +16
------------------------------
result:                     28
```

So in serial, 5 steps are required, whereas in parallel, only 3 steps are required. By making maximum use of **parallel computing**, Black Tree minimizes the number of calculations necessary to complete tasks.

Theoretical Basis

Consider the picture below. Colors are similar over small distances, except at boundaries. As a consequence, given a point in the image, if you want to predict the color of that point, you will as a general matter be correct if you use the region near that point.

Nearest Neighbor

This observation implies the notion of **local consistency**. Specifically, a dataset of Euclidean vectors is locally consistent if, for every vector, there is some radius delta, such that all other vectors within a sphere of radius delta, with an origin of that vector, have the same classifier.

Said in simple terms, if I give you a row of the dataset, there's some sphere around that row within which all other rows have the same classifier.

I proved in Analyzing Dataset Consistency, that if and only if a dataset is locally consistent, the Nearest Neighbor algorithm will produce literally perfect accuracy.

Nearest Neighbor

Given a vector **x**, the **nearest neighbor** of **x** over a dataset is the vector **y**, such that the norm of the difference between **x** and **y** is minimum over the dataset. In geometric terms, **y** is the point closest to **x** in the set of vectors.

As you can already tell, if a dataset is locally consistent, then **x** and **y** will have the same classifiers, provided that **y** is within the applicable sphere around **x**.

Delta Classification

If however, **y** is not within the sphere around **x** (i.e., it’s distance from **x** is greater than delta), then we can’t be certain that **y** has the same classifier as **x**.

As a result, my Delta Classification algorithm accepts a prediction only if the distance from the prediction is less than or equal to the applicable value of delta.

Delta Classification

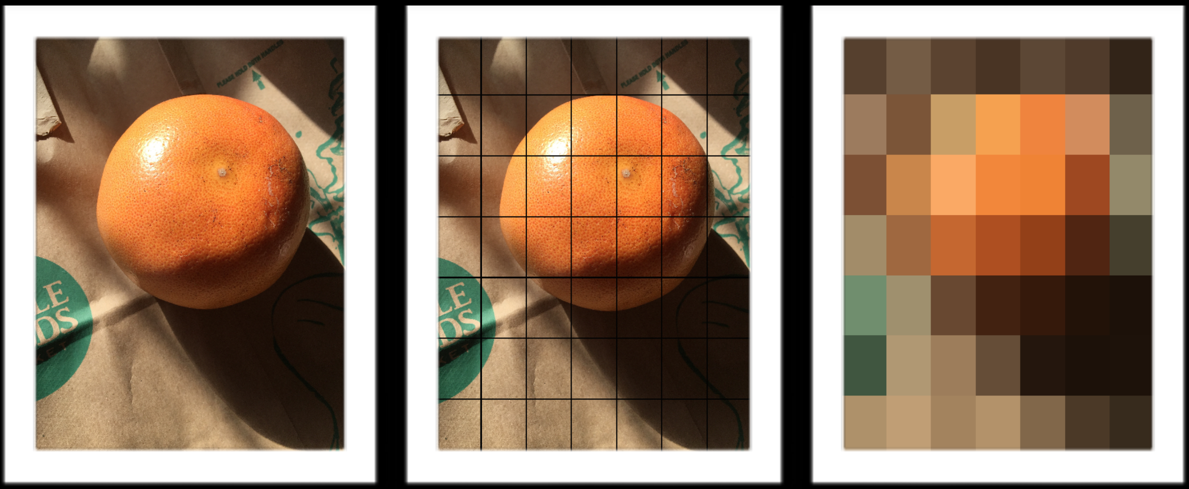
In parallel, Nearest Neighbor can be applied to an entire dataset all at once, in linear time. In serial, it still has a polynomial runtime.

Delta Classification requires only one additional step of testing whether the distance is less than or equal to the applicable delta, which in parallel can be accomplished in one step over the entire dataset. Therefore, even in serial, Delta Classification has a polynomial runtime.

As you can tell, Black Tree's accuracies are just as good as Neural Networks, but the runtime is incomparable, producing categorically superior algorithms.

Image Classification

A simple iPhone photo can contain millions of pixels. For example, the image on the left contains 7,990,272 pixels. This means comparing two images could easily require billions of calculations, which adds up when you're comparing thousands of images, which is common in Machine Learning. In contrast, the image on the right contains just 49 cells after being compressed by Black Tree, which is just 0.000613% of the data in the original image.



Sizing a full color image for compression takes about 10 seconds, whereas actually compressing images after that initial analysis takes about 0.007 seconds per image. The sizing analysis looks for boundaries, essentially forcing the actual object into a set of rectangular structures, which requires more runtime. The actual compression is done in parallel, where the average color of each resultant region is calculated independently, and so it is much faster.

Image Classification

Obviously, this only works for single object classification, but I have other algorithms not included in Black Tree that can quickly partition an image into objects defined by their boundaries. You can then take these objects and feed them to a single object classification algorithm. That said, the commercial target for Black Tree's Image Classification algorithm is the medical imaging business. Specifically, it allows for mass diagnosis on inexpensive consumer devices. Other algorithms such as YOLO are perfectly fine for identifying objects in a scene, which is not the commercial purpose of Black Tree's imaging algorithms.

Because the imaging algorithms are so fast, there are of course other algorithms that can quickly classify video.

Are there datasets that are not locally consistent?

Yes, but they are in my experience rare. Just looking at the physical world, you can see that data is in fact typically locally consistent.

My opinion is that this follows from the fact that many real world systems are described by continuous functions, that we then quantize on the basis of similarity. In this view, a boundary is where one continuous function ceases and another one takes over, changing the quantized classifier.

However, boundaries are real, creating quantized systems:

1. Electron orbitals;
2. Bond defaults;
3. Elections.

As such, I've developed another set of algorithms based upon a totally different set of theoretical results that can solve datasets that are not locally consistent, and should be correct even near boundaries.

These algorithms are included only in the Massive version of Black Tree, and they can process 500,000 vectors in about 10 minutes.

Are there datasets that are not locally consistent?

There are two papers in scope:

1. Sorting, Information, and Recursion, in which I proved that a list of real numbers is sorted if and only if the distances between adjacent terms is minimized.

This theorem implies that sorting a list of vectors gives you the nearest neighbors of those vectors. This also implies that sorting generates clusters. Because sorting can be implemented quickly in Octave, these sort-based algorithms are orders of magnitude faster than the nearest neighbor-based algorithms.

2. Information, Knowledge, and Uncertainty, in which I present empirical evidence for a measure of Confidence.

This allows you to filter predictions as a function of Confidence, which in turn increases accuracy.

Though these algorithms also work on locally consistent datasets, they can be applied to any dataset that has at least some statistical consistency to it, in that the correct classifier is the most dense in a region around each vector.

Uncertainty and Confidence

In my paper, Information, Knowledge, and Uncertainty, I showed that we can express information, knowledge, and uncertainty in a single equation that follows from the tautology that all things are either in a given set, or not in that set.

Specifically, everything that is knowable about a system is either known to some observer, or unknown to that observer.

This implies the following:

$$I = K + U.$$

Uncertainty and Confidence

Consider a set of N boxes, one of which contains a pebble. Assume we want to guess the location of the pebble, which is otherwise unknown.

The location of the pebble requires exactly log(N) bits to specify, and so that is the maximum amount of information we can know about the system. Ex ante our Knowledge is assumed to be zero, and so we have that,

$$I = U = \log(N).$$

Uncertainty and Confidence

Now assume we are told the pebble is not in the first box, and the source of this information is perfectly reliable.

It follows that we are now considering a system with N-1 states, since the pebble is not in the first box, leaving only the remaining N-1 boxes. As such, our Uncertainty is now given by log(N-1).

Therefore,

$$K = I - U = \log(N) - \log(N-1).$$

Note the limit of this expression is zero, as N approaches infinity.

Uncertainty and Confidence

Claude E. Shannon proved in his seminal paper, A Mathematical Theory of Communication, that his equation for the entropy of a probability distribution, is also a measure of average uncertainty. The previous example is a special case that is easier to understand.

The more general equation for the Knowledge associated with N observations of a system with M possible states is therefore given by,

$$K = N \log(M) - NH,$$

Where H is the entropy of the distribution of states of the system.

Uncertainty and Confidence

Applying this, we find that large consistent observations impart more Knowledge, than small inconsistent observations, which is of course correct.

Just imagine observing a system a large number of times, and the resultant measurements are all consistent. In that case, you have a high degree of Knowledge. Compare that to a small number of observations, that are all inconsistent. In that case, you have a low degree of Knowledge.

Uncertainty and Confidence

The algorithms use a measure of Confidence given by the minimum of the modal probability of a cluster, and the normalized value of Knowledge. So for example, consider two cluster predictions that contains the following classifiers:

$$A = [1\ 1\ 1\ 1\ 1\ 5\ 4];$$
$$B = [3\ 3\ 3\ 4].$$

The modal class of A is 1, and has a probability of 5/7 = 0.714. The modal class of B is 3, and has a probability of 3/4 = 0.75. Calculating Knowledge for A, the length of the vector is 7, and there are 3 classes, implying N = 7, M = 3, and H = 1.149. Therefore,

$$K = 7 \text{ x } \log(3) - 7 \text{ x } 1.149 = 3.05 \text{ bits.}$$

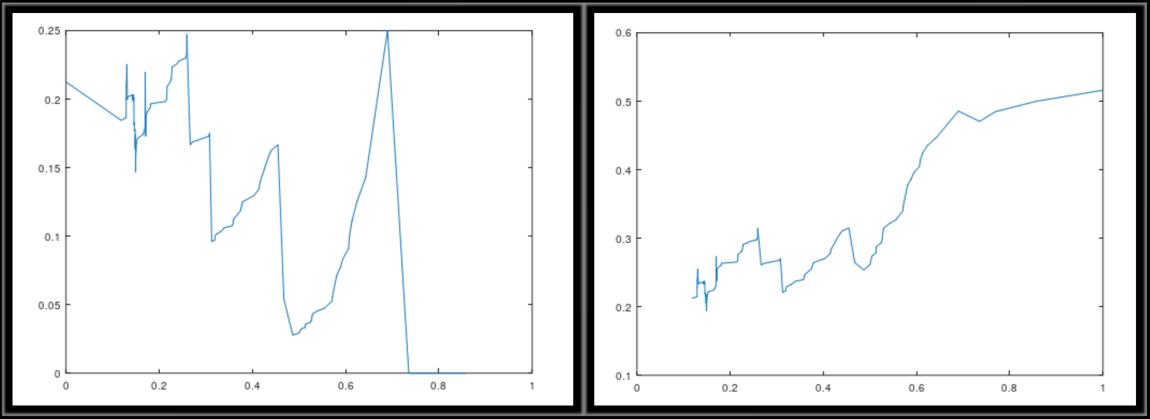
The same calculation for B yields,

$$K = 4 \text{ x } \log(2) - 4 \text{ x } 0.8112 = 0.76 \text{ bits.}$$

If these are the only two observations, then A has a Confidence of min(0.714, 3.05/3.05) = 0.714, whereas B has a Confidence of min(0.75, 0.76/3.05) = min(0.75, 0.25) = 0.25. Note that we divide by the larger of the two values of K to normalize both measures to [0,1]. Therefore, our Confidence in prediction A is greater than our Confidence in prediction B.

Uncertainty and Confidence

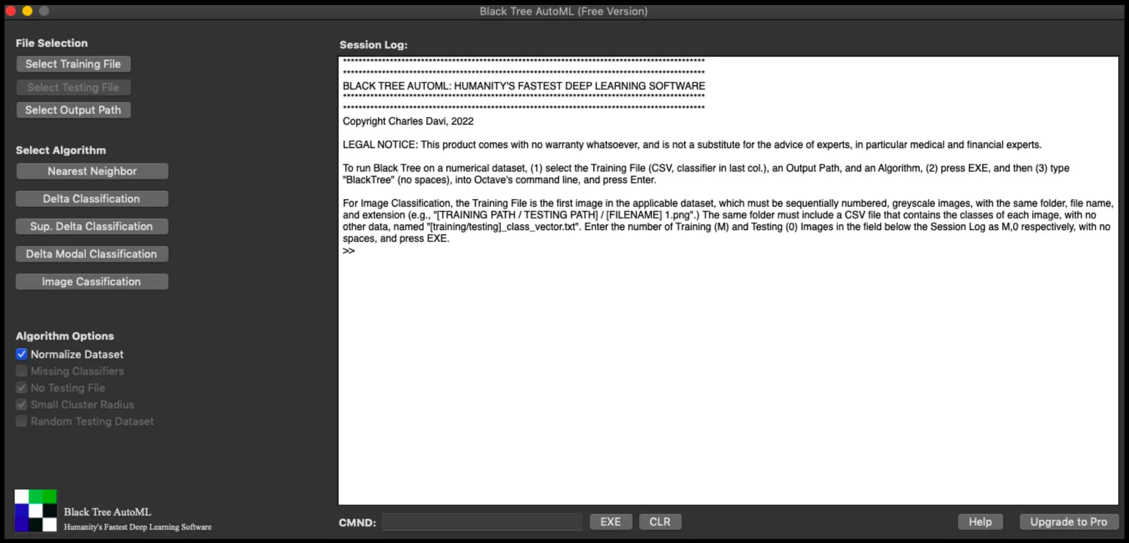
Intuition suggests that the higher modal probability prediction B is “better”, because the modal probability is higher. This is empirically false, in that if you filter predictions using the modal probability alone, you see no correlation between modal probability and accuracy.



Instead, if you filter on the basis of Confidence, you find a plain correlation between Confidence and accuracy. As a consequence, prediction A is “better”, in the sense that if we fix a higher threshold for Confidence, prediction A will survive, whereas prediction B will not. Doing so will also increase accuracy as an empirical matter.

Black Tree AutoML

Because these algorithms are so universal and fast, I decided to build a GUI in Swift that would allow basically anyone with decent computer skills to do Machine Learning. The economic value of Black Tree is therefore obvious:



[✓] Normalize Dataset
[ ] Missing Classifiers
[✓] No Testing File
[✓] Small Cluster Radius
[ ] Random Testing Dataset

```
**************************************************************************************
**************************************************************************************
BLACK TREE AUTOML: HUMANITY'S FASTEST DEEP LEARNING SOFTWARE
**************************************************************************************
**************************************************************************************
Copyright Charles Davi, 2022

LEGAL NOTICE: This product comes with no warranty whatsoever, and is not a substitute for the advice of experts, in particular medical and financial experts.

To run Black Tree on a numerical dataset, (1) select the Training File (CSV, classifier in last col.), an Output Path, and an Algorithm, (2) press EXE, and then (3) type
"BlackTree" (no spaces), into Octave's command line, and press Enter.

For Image Classification, the Training File is the first image in the applicable dataset, which must be sequentially numbered, greyscale images, with the same folder, file name,
and extension (e.g., "[TRAINING PATH / TESTING PATH] / [FILENAME] 1.png".) The same folder must include a CSV file that contains the classes of each image, with no
other data, named "[training/testing]_class_vector.txt". Enter the number of Training (M) and Testing (0) Images in the field below the Session Log as M,0 respectively, with no
spaces, and press EXE.
>>
```

An administrator can run Machine Learning with high accuracy. This allows you to both reduce headcount, and experiment with Machine Learning for a low cost.

Black Tree is in my honest opinion the lightbulb of A.I.

Questions?

Do these concepts work for Non-Euclidean Data?

Yes, but you have to be careful in applying the ideas analogously.

The largest case study I've worked on that does not involve Euclidean data is genetics.

Specifically, I analyzed a dataset of 403 complete human mtDNA genomes, taken from the National Institute of Health Database.

The tasks included predicting ethnicity, and analyzing connections among populations.

The accuracy is very good, and the runtimes are comparable, even though the dimension is very large (i.e., approximately 17,000 columns).

This work is summarized in my paper, A New Model of Computational Genomics.

Genetic Data

DNA is assembled in pairs of **bases**.

Bases are always one of the following molecules:

Base	Label	Pair
Adenine	A	T
Cytosine	C	G
Guanine	G	C
Thymine	T	A

Because each of the bases always pairs with the same other base, we only need to read one strand of a given DNA molecule.

Genetic Data

As a consequence, mathematically, a **genome** is a vector of labels over the set {A,C,G,T}.

We can compare two such vectors by testing whether or not the bases at a given index are equal or not.

For example, given genomes A = (A,C,G) and B = (A,G,G), we simply compare corresponding entries, and count the number of matching bases, which we call the **match count**. In this case, the match count is 2, since the first and last bases are matching bases.

Note this is not however a measure of distance, and is instead a measure of similarity.

We can nonetheless define the **nearest neighbor of a genome** rigorously, as the other genome that has the largest of number of bases in common with a given genome, in the dataset in question.

Genetic Data

As an example, below is a table showing the predictions generated using the nearest neighbor of a genome, over a dataset of 10 samples of the Rota Virus, also from the NIH. The classifiers are the 5 host species for the virus. Accuracy is 70%, and chance implies 20%.

Row No.	Actual Classifier	Predicted Classifier	Match Count
1.	Human	Human	3,301
2.	Bovine	Bos Indicus	2,807
3.	Human	Human	2,987
4.	Human	Human	3,301
5.	Chicken	Human	1,089
6.	Bos Indicus	Bovine	2,807
7.	Human	Human	2,987
8.	Simian	Simian	3,301
9.	Simian	Simian	3,301
10.	Human	Human	2,824

Unlike in typical applications of Machine Learning, where the goal is to cause the predicted classifiers to match the actual classifiers as closely as possible, as we'll see, in the case of genomics, superficially incorrect classifiers are at times indicative of connections between genomes. For example, in this case, there is only one Chicken, Bos Indicus, and Bovine sample, forcing the algorithm to find the genome most similar to each. However, the predictions provided are perfectly sensible, as the Bovine and Bos Indicus samples mutually map to each other, and both are types of cattle. In contrast, the Chicken sample has no obvious match, and the algorithm is simply reporting that its closest match is one of the Human samples, though the match count is the lowest.

Genetic Data

Nearest Neighbor performs significantly better than chance (3.00%), without filtering predictions as a function of confidence, returning an initial accuracy of 39.00% over the full human mtDNA Dataset.

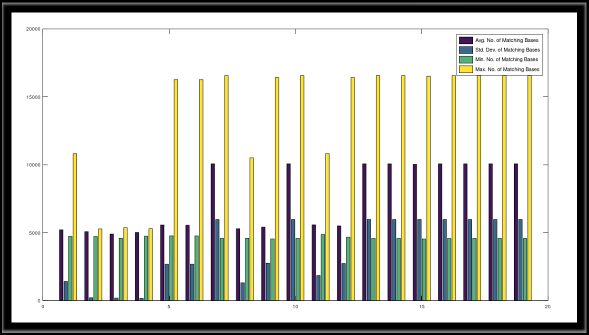
However, a simple mapping from genomes to Euclidean space improves accuracy significantly, producing an initial accuracy of 79.00%, which can then be improved using confidence and other methods. This method has a runtime of just 1.43 seconds, which is even faster than running Nearest Neighbor on the entire mtDNA Dataset, because we begin by compressing the dataset.

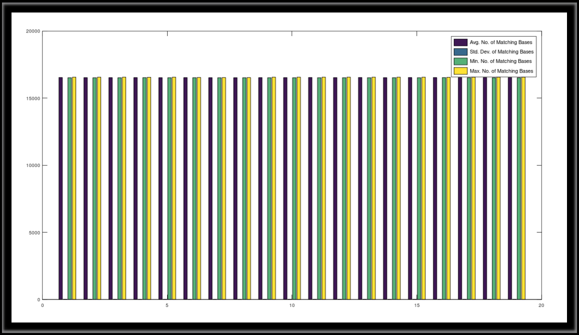
Specifically, when counting matching bases between a given genome, and the rest of its population, the average A, standard deviation S, minimum m, and maximum M of the number of matching bases, viewed as a vector P = (A, S, m, M), forms a unique profile for each population.

Intuitively, each genome in a population has a roughly similar relationship to all the other genomes in the population, producing a signature profile pattern in the form of the charts in the next slide.

Genetic Data

The figure on the left for the Italian population plainly includes outlier genomes, whereas the figure on the right for the Iberian Roma population does not, implying that the Italians are more diverse on their maternal line than the Iberian Roma. Note the x-axis gives the genome indexes within the populations, both of which contain 19 complete mtDNA genomes.





We do this for every genome in a given population, individually, and this will construct a dataset of vectors in the form of P, one for each genome in the population (i.e., a number of rows equal to the number of genomes, each row in the form of P, together forming a matrix with four columns). Note however, we are comparing genomes to other genomes in the same population (e.g., German mtDNA compared to German mtDNA). We do this for every population, separately, constructing unique matrices for each population (i.e., one matrix for German, Italian, etc.). Then combine all of the matrices into a single matrix dataset, and treat the known classifiers as unknown, and try to predict the classifier of a given profile (in this case ethnicity, e.g., German). To do so, simply run Euclidean Nearest Neighbor, mapping Pi to Pj, for which the Euclidean norm of the difference, ||Pi − Pj|| is minimum, treating the classifier of Pj as the predicted classifier of Pi.