Empirical Verification of the Error Bound of the Decision Tree Classifier Algorithm

Zhamila Kulchukova [1]

Abstract

The decision tree classifier is a type of learning algorithm that constructs a binary tree with simple Boolean operators at the nodes. In this project, the goal was to empirically verify the error bound of the given algorithm and test it on real-world data. As the result, the experimentally computed error was mostly in agreement with the theory.

1. Introduction

The decision tree classifier constructs a binary tree to predict the label y of the associated instance x. The nods of the tree perform simple Boolean operations that split the data based on one of the features of x, until the split data ends up at the leaf where it gets labeled.

The algorithm is very easy to understand and visualize, however, this comes at a certain cost - the tree is very prone to overfitting. This is because if the tree is let to grow to an arbitrary size, the depth of the tree can get arbitrarily large too. The VC dimension of the decision tree classifier corresponds to 2^d, so as d gets large, this will cause a problem. So to combat the overfitting, the MDL (minimum description length) principle is used which helps the algorithm to create a tree of optimal depth and minimal empirical loss.

In this project, the error bound of the decision tree classifier algorithm is empirically verified. Then, the algorithm is tested on some real-world data, and the results are compared with the theoretical error bound.

2. Methods

2.1. Theory

The algorithm has a very greedy approach to constructing the tree, which makes it hard to generalize the empirical risk. Therefore, several assumptions about the data must be made.

[1]Department of Physics, Nazarbayev University, Astana, Kazakhstan. Correspondence to: Zhamila Kulchukova <zhamila.kulchukova@nu.edu.kz>.

The main assumption is that $x = \{0,1\}^d$, which means that the elements of x are vectors of d features that are either 0 or 1. This allows the simplification of splitting rules into $\mathbb{1}_{[x_i=1]}$. According to S. Shalev-Shwartz and S. Ben-David (2014), this will lead to a construction of a decision tree with 2^d leaves and a depth of $d+1$. But as it was already mentioned, letting d be arbitrarily large has a very high chance of leading to overfitting, so some assumptions about the size of the tree must be made. The approach is to use MDL, which adds prior knowledge that smaller trees must be preferred over larger trees. Applying all these assumptions leads to the error bound in the following form:

$$L_{\mathcal{D}}(h) \le L_S(h) + \sqrt{\frac{(n+1)log_2(d+3) + log(2/\delta)}{2m}}, \quad (1)$$

2.2. Implementation

The true error is unknown, however, it can be approximated by the training error. The term ΔL was introduced to approximate the difference between the true error and the empirical risk:

$$\Delta L = L_{\mathcal{D}}(h) - L_S(h) = L_{train}(h) - L_{test}(h)$$

Since the decision tree classifier creates a binary tree, n depends on the depth of the tree according to a relation as $2^{depth} - 1$, which simplifies the expression down to 2 parameters d and m. So to empirically verify the error bound, the following must hold for all hypothesis classes h:

$$\Delta L \le \sqrt{\frac{2^{d+1}log_2(d+3) + log(2/\delta)}{2m}}$$

Aside from obeying the error bound, ΔL must also decay as m increases and grow as d increases. To check that, a set of experiments was done using the $\texttt{scikit} - \texttt{learn}$ package. It is worth noting that the package uses the CART (classification and regression trees) algorithm, which constructs a tree based on the largest information yield at each node for a chosen threshold. The link to the source code with the detailed computational algorithm can be found in the following repository:

`shorturl.at/lBEHT`

3. Results

3.1. Verification on randomly generated data

As described in 2.1, the data must have a form $x = \{0, 1\}^d, y = \{0, 1\}$. For x, a random list of size m containing arrays with d number of binary features was generated using `numpy` package. To ensure that the classification task is difficult enough so that it visibly approaches the error bound, the array of labels y was also generated completely randomly. It is expected that the experimentally computed ΔL (which is denoted as experimental error) will closely approach the error bound as m grows, and overall, both experimental error and error bound must decay and approach some constant value as $m \to \infty$. The following figure shows the graphical results of the error bound verification by varying the sample size m.

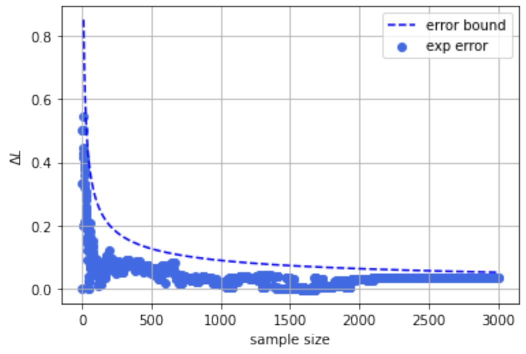

Figure 1. The relation between ΔL and randomly generated data sample of size m. The dashed line represents the error bound described by the theory.

As expected, ΔL follows the behavior as described by theory, and it visibly decays as $m \to \infty$, while not exceeding the error bound. It is also noteworthy that ΔL is quite high when m is arbitrarily small, which can be explained by underfitting. At the same time, for very large m, the error converges to some value close to zero. This means that the more sample points we have, the better the prediction will be. Many real-life data sets have a finite sample size, so it should be expected that there will be some uncertainty, even if the number of samples is large.

Now, the same procedure is repeated for a varying number of features. We should expect ΔL to grow as d gets larger. However, it was noticed that after $d = 10$, the error stops changing and converges to some fixed value, despite the fact that the error bound is growing rapidly, as presented in the figure below:

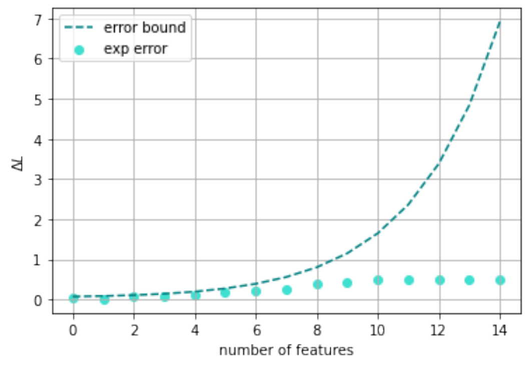

Figure 2. The relation between ΔL and randomly generated data with a number of features d. The dashed line represents the error bound described by the theory.

This can be explained. It was already mentioned in 2.1 that smaller trees are preferred over larger trees, as large trees are very prone to overfitting. The CART algorithm limits the tree depth to combat that. Since the tree depth is very closely correlated to the number of features (for small d specifically), at some point, the additional features will stop providing any useful information, and so the algorithm limits the depth of the tree. In this case, it can be seen that after d, the information gain is not affected, and so the error stops changing.

It was also very interesting to see how the error changes with the tree depths, as it could be more representative of how the error bound behaves. So in the following figure, the error bound is tested for varying tree depths:

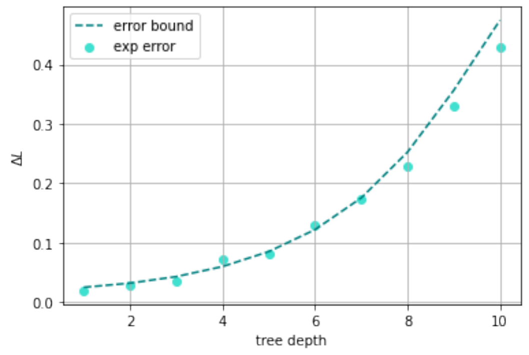

Figure 3. The relation between ΔL and a randomly generated data sample with tree depth d. The dashed line represents the error bound described by the theory.

As presented, the results are in agreement with the theory. The experimental error does not exceed the bound for the most part, and the error grows as tree depth increases.

3.2. Verification on real-world data

Now, it is required to test the error bound on some real-world data. For that, the `breastcancer` data set was used. However, the data set cannot be used directly to test the given error bound (1) since the feature vectors are not binary - they consist of real numbers. One of the main assumptions for the error bound was to have a feature set in form $\{0,1\}^d$, so to stay consistent with the assumption, the features of the data set were converted into binary. To do so, all of the feature values were compared to the average of the respective feature vector; if they were larger than the average, the feature was turned into 1, else - it was turned into 0. Of course, this is not ideal, and there will be a big loss of information which will ultimately affect the computed ΔL. But the main goal remains - that is to test if the ΔL behaves in agreement with the error bound.

After converting the data set to the binary, the same procedures were conducted. For varying m, the results are presented in the plot below:

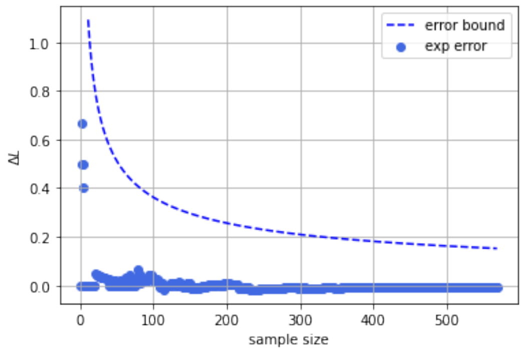

Figure 4. The relation between ΔL and the breast cancer data set with varying sample size. The dashed line represents the error bound described by the theory.

As expected, the error decays as m grows and converges to a number that is close to 0. The error, however, converges way faster than the error bound, which is expected since the bound describes the error for completely random data, whilst there is a clear relation between the features and the label in the breast cancer data set.

Now, for the varying tree depth $d+1$, the results are presented below:

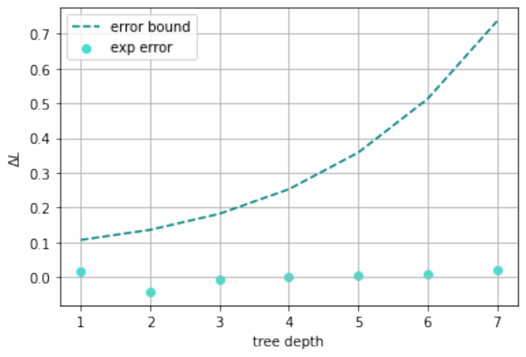

Figure 5. The relation between ΔL and the breast cancer data set with varying tree depth. The dashed line represents the error bound described by the theory.

Similar to the previous part with varying m, the error is way lower than the bound and it diverges quite slowly. This is also expected because the depth of the tree is highly dependent on the splitting rules that are defined by the information gain at the nodes. It is obvious that for real-world data the information gain will be more effective than for randomly generated data sets. In addition, the maximum depth that the algorithm could have done was 7, which is quite low, and still, an upward trend is barely visible. If the depth of the tree could have been manipulated to create bigger trees, the error might have grown even further.

Conclusion

Both randomly generated and real-world data sets are in agreement with the error bound. For the random data, the behavior of the error was very close to that of the error bound (as shown in Fig. 1), if not closely following (as in Fig. 3). For the real-world data, even though the error was much smaller, which is expected since the information gain was larger, the behavior followed that of the error bound, decaying quickly as m increased. Although the change in error was barely visible as d increased, it was still below the error bound. In addition, there was a pitfall - that the real-world data did not comply with the assumptions about the data distribution, so it was changed to fit the criteria. It would have been great if there was a data set with binary features that could showcase more accurate results. But aside from that, the expectations regarding the real-world data were fulfilled. Overall, the main objectives of the project were met.