A4-Learning-Image Orientation

# Random Forest

## Design Decisions and Tunable Parameters

Random forest is a collection of decision trees. We have taken different number of decision trees and tried out different parameters for creating the decision trees.

High level design decisions of the algorithm: We are trying to find a decision tree which correctly classifies the images according to their orientations. We are calling the build\_tree function recursively to generate the subtrees under the tree, depending on the value of one particular dimension in the function. This dimension/predicate is chosen by finding out the minimum entropy among the remaining predicates. After we find the predicate, we split the dataset into 2 subsets. Even though we have a continuous variable like pixel value, we are making it binary with respect to a threshold. So, if a pixel is lesser than the threshold, we send the data point to the left subtree of the decision tree, if it is greater, it goes to the right subtree.

We tested by tuning certain parameters:

1. Purity: This is the purity of the dataset. We define the purity as the percentage/fraction of data of the current dataset that is required for this dataset to be a leaf node in the tree. We noticed that a lot of times we are stuck with no predicates in the function, and we are forced to make the highest orientation count as the leaf node, so we came up with the concept of purity. Purity is a value between 0 to 1. For example, for purity value=0.9 if in the current dataset, the fraction of labels of a certain orientation, say 90 degrees is more than 0.9, then the function will make it a leaf node of the decision tree and return the same.
2. Number of trees: This is the number of trees in the forest. We tested on different values for this
3. Number of features/predicates in each tree: The number of predicates in each decision tree. We tested on different values for this.
4. Threshold: We are taking the threshold value as 128 and forming a binary tree.

## Data Structure of Trees

We are using a dictionary to keep the trees, inside each dictionary there is a list of 2 subtrees, generated recursively.

Predicate[i]: {

[Predicate[j]: {….}, Predicate[k]: {…}]

}

Each dictionary will contain another subtree. When it reaches to the leaf nodes, it will store as below:

{Label: [None, None]}

## Algorithm Build Trees

Build-Tree(Subset S, Attributes P)

Count number of each label in S

If any label occurs more than purity threshold (e.g. 90%), return Label

If P has only 1 attribute, split S based on P’s threshold value, and return { Build-Tree (S-left, []), Build-Tree (S-right, [])}// This returns a dictionary with labels as the leaf nodes

IF P is empty, Return the maximum count label.

Else, if none of the above conditions are true,

C=Find the minimum entropy predicate from P.

Split S based on C’s threshold value.

Return { Build-Tree (S-left, P-{C}), Build-Tree (S-right, P-{C})}// This returns a dictionary with the predicate C as key, and a list of 2 subtrees as value.

## Testing Results:

We tested the accuracies with different values of the parameters, below are some of the results.

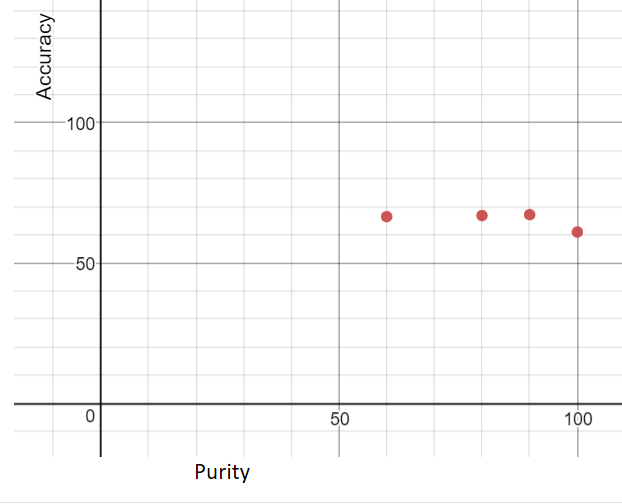


Figure 1. Fixing Number of Trees=100, Features=14, Threshold=128. Plotted Accuracy v/s Purity.

Values:

For Purity 60%: Accuracy=66.5%

For Purity 80%: Accuracy=66.9%

For Purity 90%: Accuracy=67.2%

For Purity 100%: Accuracy=61%

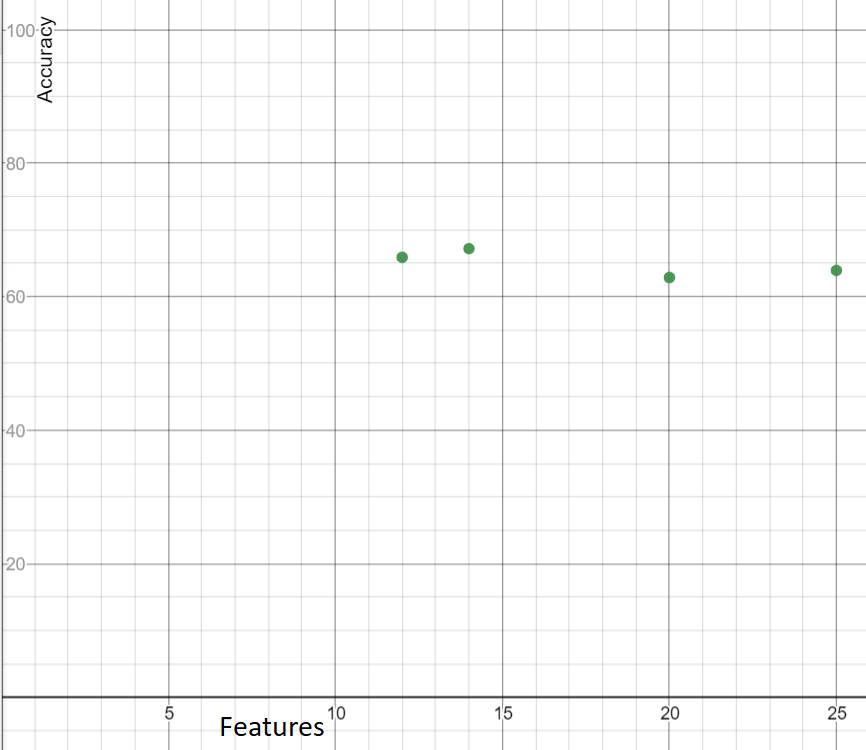


Figure 2. Fixing Number of Trees=100, Threshold=128, Purity=90%, Plotted Accuracy v/s Features:

Features: 20, Accuracy=62.88%

Features: 25, Accuracy=63.94%

Features: 14, Accuracy=67.2%

Features: 12, Accuracy=65.9%

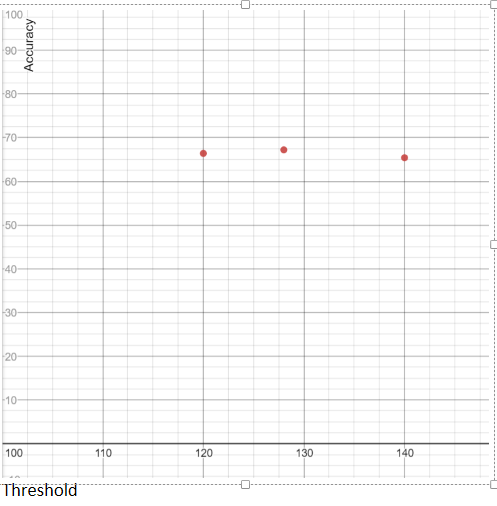


Figure 3: Fixing Number of Trees=100, Purity:90%,Features: 14, Plotted Accuracy v/s Threshold:

Threshold: 120, Accuracy=66.4%

Threshold: 128, Accuracy=67.2%

Threshold: 140, Accuracy=65.4%

Considering All these, the best accuracy is given by:

Threshold=128

Purity=0.9

Features=14

We observed that accuracy slightly increases with the number of trees, we are fixing that as 100 here, after that it takes a long time to train, and accuracy doesn’t increase that much.