Coursework on Randomised Decision Forest

Ashish Pandey
Dept. of Electrical and Electronic Engineering
Imperial College London
ashish.pandey17@imperial.ac.uk
CID-01383450; Login-ap8516

Ilias Chrysovergis
Dept. of Electrical and Electronic Engineering
Imperial College London
ilias.chrysovergis17@imperial.ac.uk
CID-01449042; Login-ic517

1. Train Decision Forest

1.1 Bootstrap Aggregating

We make use of bootstrap aggregation (Bagging), wherein we generate multiple datasets by uniform selection of data points from the provided training data **with replacement**. We make use of the inbuilt MATLAB function `randsample to` generate four datasets. A visual representation of each of the four generated datasets is given below. There are **150 data points** in each generated dataset. In each of the plot given above it can be observed that none of the spiral is complete. This happens because we draw samples from the training dataset with replacement. Average uniqueness is around 63% and as size of the dataset increases it will reach 63.3%. Probability distribution of three classes in the datasets is presented in Table I to corroborate the fact that sampling was indeed performed uniformly. It can be seen from the values of each class that the sampling is indeed uniform. All simulations were done on a system with Intel i7 Processor (2.9 GHz) and 16 GB of RAM.

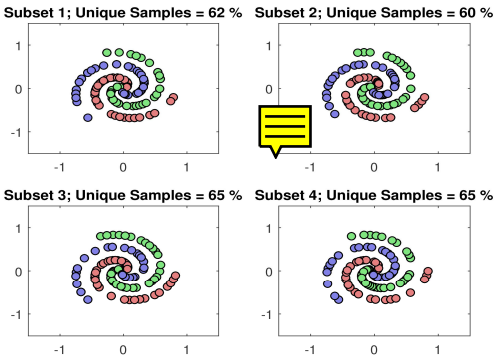

Figure 1: Dataset generation using Bagging.

Table I: Probability Distribution of Three Classes in Dataset.

Dataset	Class 1	Class 2	Class 3
1	0.3454	0.2839	0.3707
2	0.2915	0.3485	0.3600
3	0.3217	0.3612	0.3171
4	0.3509	0.3182	0.3309

1.2 Split Functions

We implemented five split functions: **axis-aligned, linear, conic, two-pixel and cubic**. The expression for conic and cubic weak-learners is given in Appendix-I to this report. To make use of these non-linear split functions we need to transform the data. Representative examples of the five split functions are presented in figure 2. Also shown alongside is the histogram representation for number of data samples per class at the parent node and left and right child nodes. For better analysis we again apply the split functions, but this time on few data points and observe the change. It is observed that when a strong learning class of split function is applied to few data points it results in clear over fitting of the data. For less data points, the generalization of cubic split function is poor as from Vapnik Chervonenkis (VC) theory **they have a larger VC dimension**. To combat effect of overfitting, random forest provides the solution in form of committee machine. To ensure that the performance obtained is consistent with use of strong non-linear split functions, we must also increase the number of decision trees in the forest.

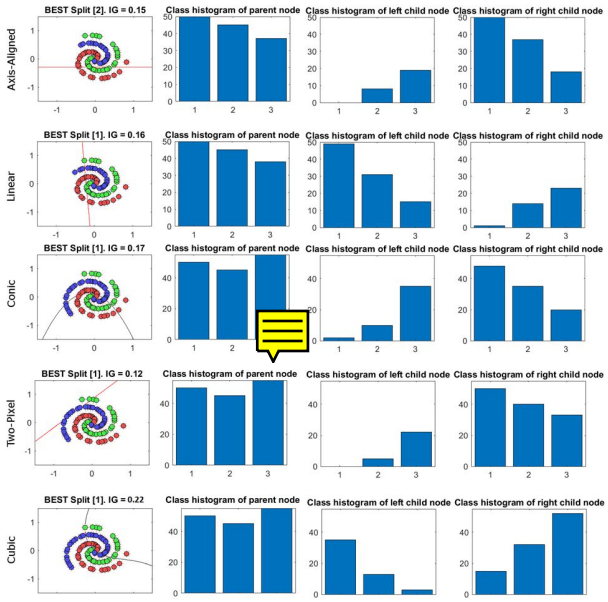

Figure 2: Five Different Weak-Learners trained on large number of Data Points for large value of `param.splitNum`.

We also observe the variation in Information gain with randomness parameter ρ, for each of the five split functions. We observe that when the number of split functions tested is small we are not able to clearly distinguish between efficiency of different weak-learner classes used to split the data and also their discriminating power. However, as we increase the number of split functions to be tested for each class by increasing the value of `param.splitNum`, we are able to clearly distinguish between performance of strong split-functions such as **cubic** from much weaker split function like **two-pixel**. Hence, it can be sufficiently concluded that while

using a strong learning class, we should increase the number of splitting trials for best results.

Figure 3: Variation in Information Gain after node splitting with Randomness for RF with 10 Decision Trees of Depth 5.

1.3 Growing Tree

Having split the root note in our prior discussion, we now recursively split the nodes to grow the tree. Once, tree is fully grown the leaf nodes contain the class distributions. In the figure given below, we visualize the leaf nodes of a decision tree that utilized axis-aligned weak-learners. The effect of recursively splitting data points in a node using a certain class of weak-learners is akin to **distilling**, wherein in the end we obtain pure data i.e. data pertaining to only one class in the leaf nodes. Stronger the class of split-function used, better distilling is obtained. It can be seen that in most leaves, we have data points that belong to just one class of data. Since the density of data points near the origin is very high, it becomes visibly more complex to separate them. This has led to obtaining four leaves which have data points from two classes. Another visualization of leaves for tree grown using a stronger class of weak-learners: **cubic** is shown as figure A3.1 in Appendix-II. In that figure all leaf nodes have data points from only one class. We make use of two stopping criterions, firstly, if **less than five training points** reach a node we call that node as leaf node and second criterion is by **fixing maximum depth of tree**. It is done by choosing appropriate value of `param.depth`. Our motivation for choosing less than five data point as stopping criterion stems from our previous discussion that training on very few data points leads to over-fitting.

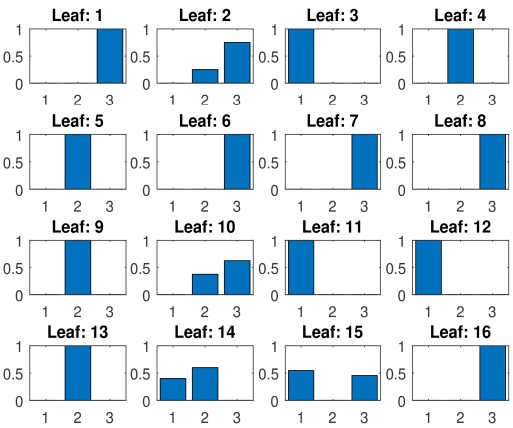

Figure 4: Visualization of leaf nodes of a decision tree grown using axis-aligned weak-learners.

2. Evaluating Decision Forest

We take 4 test points given in the coursework and evaluate them on **a RF with three trees, depth=5 and using linear weak-function.** The evaluation result of four novel data points and class distributions of the leaf nodes at which the data point arrives and the averaged class distributions are also visualized alongside.

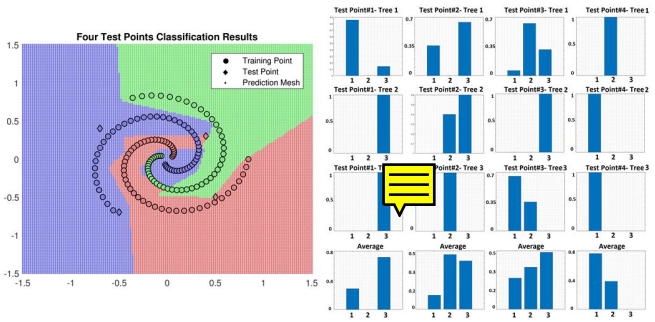

Figure 5: Visualization of evaluated novel data points and corresponding average and individual class distributions at leaf nodes.

We observe that one test point [0.4, 0.[illegible] has been misclassified. Its actual color should be blue. A major reason for this misclassification is that we have only three trees in our decision forest and this leads to limited averaging effect of committee machine. If we increase the number of trees to 5, we find that all points are correctly classified. For a start we know that the classification accuracy or generally stating the performance of Randomised Decision Forest (RF) depends on several parameters, essential ones being: **Depth of Decision Tree, Number of Decision Trees in Forest**, **Weak-Learner used** to split data, **Number of split functions tried.** We will examine the effect of these aforementioned parameters on performance of RF in sub-sections.

2.1 Varying number of Decision Trees in RF

We saw it above that having less number of decision trees in our RF limits the averaging effect of the committee machine. So, we first of all try to change the number of decision trees in our grown RF and then use it to classify the test points and observe the classification result. In figure 6, it can be clearly observed that the color encoded classification regions are dominated by blue color near the origin (where training points are tightly clustered). As we increase the number of trees we find that classification regions become more distinguished and even near the origin we now have classification regions for all three classes based on the spiral training data points. However, all this comes at a cost of increased processing time and it appears that there is not much change in performance when we increase the number of decision trees in RF from 50 to 80 and then 100.

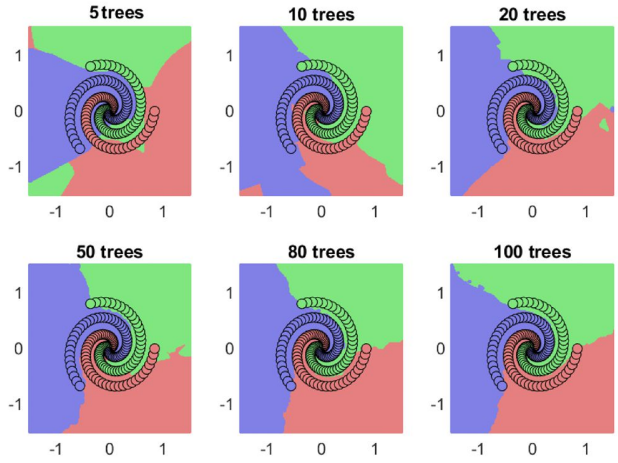

Figure 6: Effect of varying number of decision trees on classification result; Depth=5, `param.splitNum`=10, Conic weak-learner.

2.2 Varying depth of Decision Tree in RF

We now vary the maximum depth an individual decision tree can achieve in a RF. Theoretically, as we increase the depth value we should observe an improved distilling effect and as a result the leaf nodes start to become purer. We should take considerable care not to increase the depth arbitrarily because not only this will lead to increased processing time but also cause the training points to become refined and clustered together as we descend the tree. It will lead to separation of points which lie close to each other. A workaround to this problem is to increase number of trees and hence improve averaging effect. It is imperative to mention that individual trees can be trained and tested in parallel but for a single decision tree with large depth it has to be grown sequentially.

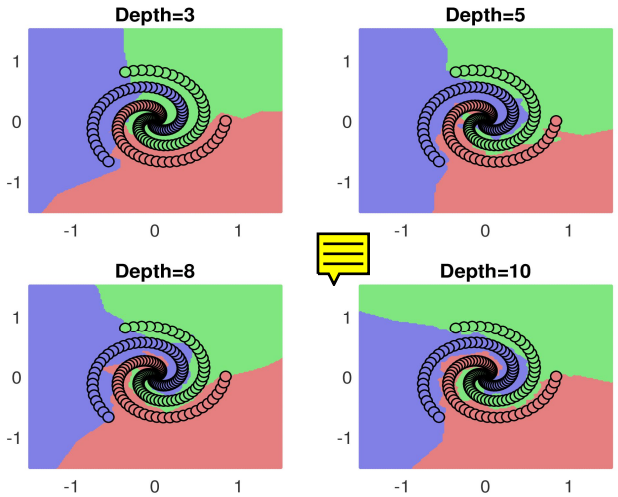

Figure 7: Effect of varying depth of decision tree on classification result; Trees=10, `param.splitNum`=10, Linear weak-learner.

2.3 Varying Number of Split Functions to be Tested

By choosing a large value of `param.splitNum` we are increase the number of split functions of a particular class, that are tried at each node for splitting. This reduces the inherent randomness within each tree. We do not observe improvement in performance on choosing a large value. This is also in complete conformance with the plot provided in figure 3.

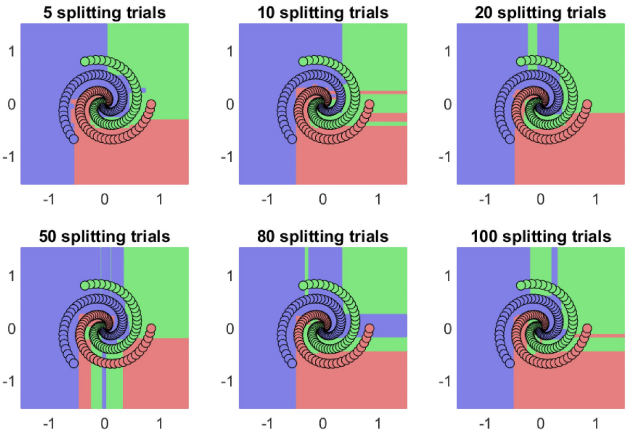

Figure 8: Effect of varying number of split-functions to be tested on classification result, Trees=10, Depth=5, Axis-aligned.

2.4 Use of different Weak-Learners in RF

The figure illustrated below shows the effect of using different weak-learners. Plot is not provided for cubic weak-learner. It is worthy to note that the class boundaries in the extrapolated region are very much in agreement with the class of weak-learner used. Performance of Two-Pixel Test is very poor compared to other weak-learners.

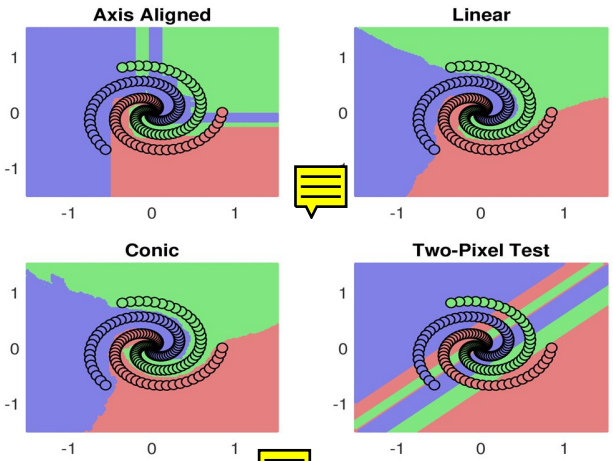

Figure 9: Effect of different weak-learners on classification result; Trees=50, `param.splitNum`=20, Depth=5.

3. Caltech Dataset for Image Categorisation

3.1 K-means Codebook

In order to construct a visual codebook (or vocabulary) for Image Categorization we apply the K-means algorithm, which is currently the most common algorithm to approach such problems. The descriptors used are called SIFT (Scale Invariant Feature Transform) descriptors. SIFT descriptors are robust to partial visibility, clutter and background noise since they are resistant to geometric and illumination variations as well as with scale and affine transformations.

We create SIFT descriptors for both training and testing sets of images and the feature dimension is 128. These descriptors will be used for the creation of K-Binned Bag-of-Words Histograms. To build the K-means codebook we randomly select 100k descriptors from the training dataset and the `vl_kmeans` algorithm clusters the descriptors into K groups. The vocabulary size K will be discussed in more detail later. The following table presents specific characteristics of the algorithm that were used in order to increase the performance and the time-efficiency of the algorithm.

Table II: Algorithm Specification.

Verbose	Yes
Distance	L2
Algorithm	ANN
Initialization	Plus-Plus

The verbose option has been used in order to see useful information about the clustering, while the Euclidean distance is regarded as the most common and appropriate one for k-means clustering. The Plus-Plus initialization method greedily picks K data points that are maximally different in order to increase the clustering performance, while the computational complexity achieved is $O(logK)$. Finally, the ANN (Approximated Nearest Neighbours) algorithm is used to accelerate the sample to center comparisons. This algorithm is suitable for large problems when time-efficiency is very important [1][2].

3.1 (a) Vocabulary Size

Using K-means we are in direct control over the vocabulary size, however this also presents an important question: What should be the ideal vocabulary size? One way is to select the final clustering by taking the one with the minimum classification error in image categorization. Since, we use an RF classifier for classification, where we have too many parameters to take into consideration, we decided to determine the vocabulary size by considering the time-efficiency and the total energy for different K. The optimal vocabulary size will need minimum time and provide the minimum energy [3]. In the following figure the time and energy for different codebook (vocabulary) sizes is presented:

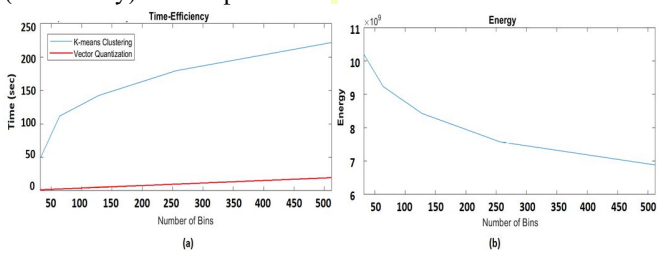

Figure 10: Impact of Vocabulary Size on the K-means Codebook (a) Time- Efficiency (b) Energy.

3.1 (b) Vector Quantisation Process

The Vector Quantisation process is used in order to extract bags of visual words (histograms) for each training and testing image, i.e. the features for training or testing the multi-class classifiers. Therefore, each visual word (descriptor) of a given -training or testing- image is compared with the codewords and through Nearest Neighbor matching (using Euclidean distance) it is assigned to the nearest codebook. Each histogram bin is a codeword and each bin counts the number of visual words assigned to its' corresponding codeword. Therefore, each image has dimension equal to the vocabulary size and thus the memory and computation needed for classification has drastically decreased.

3.1 (c) Example Images

The Bag-Of-Words Histograms of some example Training and Testing images is presented in the following figure:

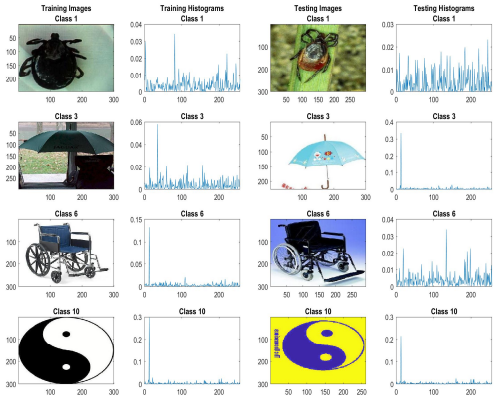

Figure 11: Bag-Of-Words Histograms (256 Bin) of Example Images.

3.2 RF Classifier

Randomized Decision Forests have multiple parameters which need to be tuned for optimal performance. These are the number of decision trees in RF, depth of trees, the degree of randomness and the type of weak-learner used. We use an axis-aligned weak-learner because it accelerates classification time.

3.2 (a) Recognition Accuracy / Confusion Matrix

The variation in Recognition Accuracy by varying different RF parameters is presented in the following figure.

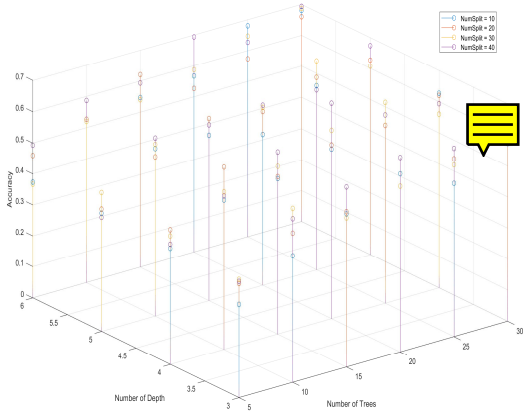

Figure 12: Accuracy vs Number of Trees, Number of Depth and Number of Split Functions using K-means Codebook

It is clear from the figure above that as the number of trees, depth and split functions are increasing, the recognition

accuracy increases too. The best classifier is found for 20 trees, depth equal to 4 and 40 split functions and the accuracy is equal to 69.33%. Confusion Matrix for the best classifier is given below. It is clear that RF classifies images from Class 2, 8 and 10 with best accuracy and images from Class 3 with worst accuracy amongst all classes. Sample success and failure images are given in figure 14. We can only provide a **heuristic explanation** as the possible reason for few misclassifications.

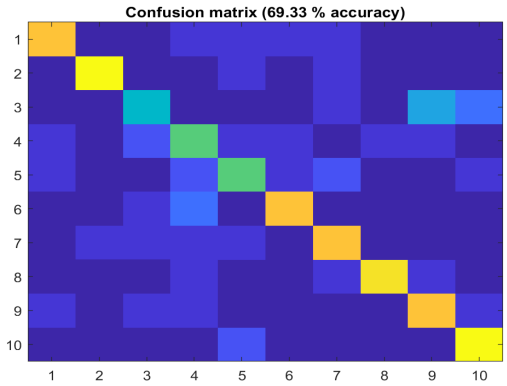

Figure 13: Confusion Matrix when `numTrees = 20, numDepth = 4 & splitNum = 40` (K-means Codebook).

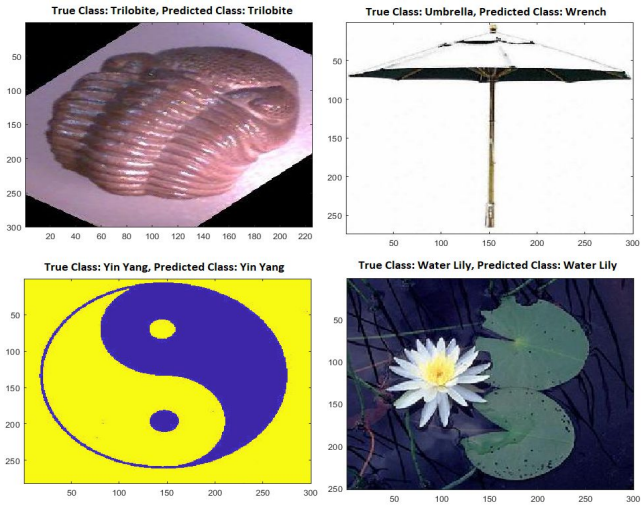

Figure 14: Sample success and failure cases from classification using K-means codebook and RF Classifier.

3.2 (b) Time-Efficiency

The Time-Efficiency of different RF Classifiers is presented in the following figure.

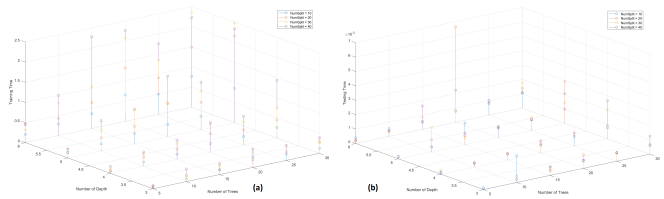

Figure 15: Comparison of Training and Testing Time for different RF Classifiers (K-means codebook).

From figure 15 it is obvious that training is much more computational expensive than testing. Testing time is almost 1000 times quicker that training time. This is one of the most important advantages of Random Forest Classifiers. Furthermore, as the number of trees, split functions and depth are increasing, computational complexity increases too. This is one of the most significant trade-offs in RF classifiers and in classification generally.

3.4 Impact of the Vocabulary Size

A low vocabulary size restricts us to operate in low dimensional space and class separation as such becomes difficult. However, having said this, we cannot increase the vocabulary size arbitrarily as it leads to excessive processing time. In the figure given below variation in recognition accuracy with vocabulary size is provided. It can be easily observed that accuracy decreases as number of bins increase beyond a certain optimal value. Also, the accuracy is mostly better for 20 decision trees in RF classifier compared to 10 trees in our RF. This occurs because a higher vocabulary size leads to large number of cluster centers and having very large number of such centers might lead to diminished discriminating power of codewords.

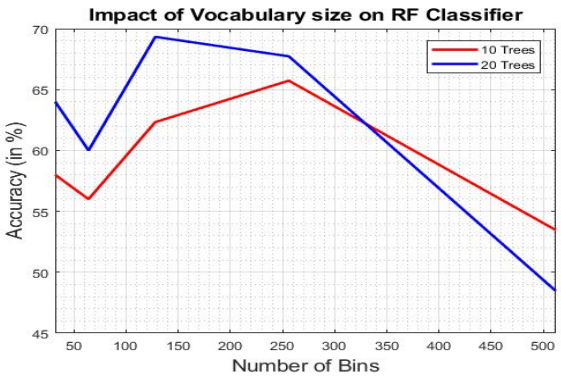

Figure 16: Impact of the Vocabulary Size on the RF Classifier.

3.3 RF Codebook

In this section we built a visual codebook by using a Randomized Decision Forests approach. Although the K-means codebook is currently the most common it has many issues, concerns and disadvantages that should be taken into consideration. Firstly, the vector quantization process is very slow, since it relies on nearest neighbor search, and computational expensive. Furthermore, the K-means method is generative (i.e. unsupervised) and therefore it is not able to achieve as good classification accuracy as a discriminative (i.e. supervised) method can. Finally, the K-means algorithm converges only to local optima since it depends significantly on initialization.

The K-means algorithm, however has an advantage that clusters are formed by considering all 128 dimensions of our descriptors. This leads to a higher discriminating power of codewords for our obtained codebook than we can ever obtain using a single decision tree. This occurs because we consider only small dimensions at each node in our tree. This problem is overcome by using an ensemble of decision trees.

3.3 (a) Recognition Accuracy / Confusion Matrix

In figure 17 the recognition accuracy for different RF classifiers using the RF codebook is presented. We did not manage to find better results than the ones using the K-means codebook.

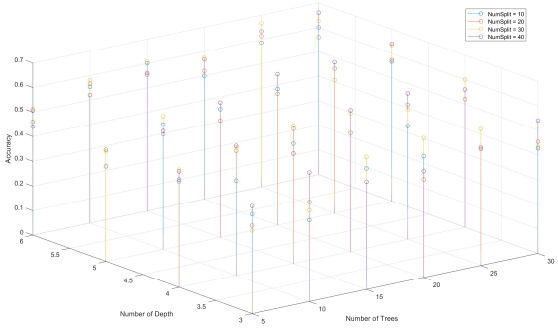

Figure 17: Variation in Recognition Accuracy vs Number of Trees, Number of Depth and Number of Split Functions using RF Codebook

The best classifier has recognition accuracy equal to 66.67% and its parameters are: `numTrees = 25, numDepth = 5 & splitNum = 30.` The confusion matrix for the best classifier can be found in figure A3.2 of Appendix-III along with sample success and failure cases in figure A3.3. Some possible reasons for a sub-optimal performance obtained using RF are provided below. Firstly, a better classification may not have been achieved by using the provided Caltech Dataset. Secondly, as opposed to tuning just one parameter in K-means algorithm, we have several parameters which need to be tuned in RF in order to obtain optimal performance for which cross-validation needs to be performed.

3.3 (b) Time-Efficiency

The time-efficiency diagrams for training and testing the RF classifiers using the RF codebook is provided in figure 18. Although the training time is almost the same the between the two different codebooks, the testing time has improved by a factor of 2.5 for some set of parameters. Although the dimensionality of the histograms has increased in that experiment (320 bins in the RF Codebook & 256 bins in the K-means Codebook), the time-efficiency is the same or better than previously.

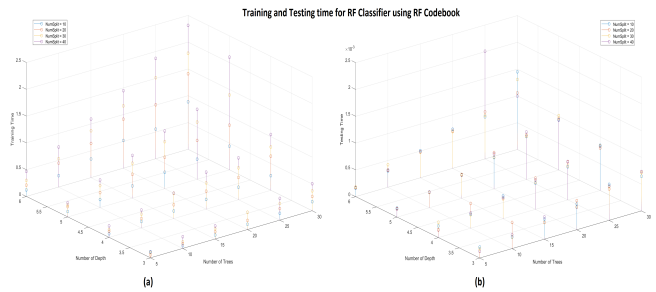

Figure 18: Comparison of Training and Testing time for different RF Classifiers (RF codebook).

3.3 (c) Impact of the Vocabulary Size

The randomness of the random forest restricts us from gaining essential insights of the impact of the vocabulary size on the classification accuracy using the RF Codebook. It is positive that as dimensionality of the histograms increases, the processing time also increases. It is very hard to confirm the same for increase in accuracy by having just one simulation result. By taking several realizations (in order of 100's) and averaging the obtained accuracy we can obtain a better estimate. However, this will demand days of computation with specification of our system.

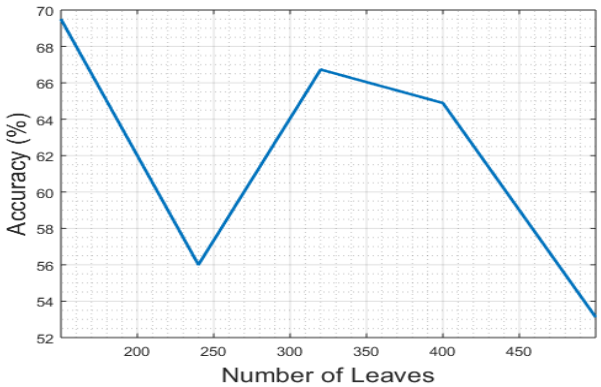

Figure 19: Impact of the Vocabulary Size on the RF Classifier.

3.3 (d) Vector Quantization Complexity

The Vector Quantization Complexity is almost two times worst when using `testTrees` function and nearly the same when using `testTrees_fast` function. Although the nearest-neighbor search has higher computational complexity than searching in binary trees in general, this attribute is not confirmed from our experiments. Several reasons which have already been discussed in previous subsections might have led to such an observation.

4. Conclusion

In our report we have successfully discussed and presented results as per the requirements laid out in the coursework. We learnt to train and test a fully grown random forest on Toy Dataset and then observe the change in classification performance by varying different parameters associated with a Random Forest. We learn to generate codebooks using two methods: K-Means algorithm and RF for Caltech 101 Image Dataset and compare their performance. A method to increase time-efficiency can be obtained by using PCA to reduce number of features before using RF for classification. This may also improve the accuracy of the classifier, since it can eliminate the noise which is inherent in given dataset.

References

[1] http://www.vlfeat.org/matlab/vl_kmeans.html
[2] http://www.vlfeat.org/overview/kmeans.html
[3] https://people.sc.fsu.edu/~jburkardt/m_src/kmeans/kmeans.html
[4] T.K Kim, Machine Learning for Computer Vision Lecture Notes, 2018, Imperial College London.

APPENDIX-I

Expression for cubic weak-learners used in the coursework is provided below:

Cubic: $h(\mathrm{v},\theta) = \left[a_1x_1^3 + a_2x_2^3 + a_3x_1^2x_2 + a_4x_1x_2^2 + a_5x_1^2 + a_6x_2^2 + a_7x_1x_2 + a_8x_1 + a_9x_2 > \tau\right]$ (1)

APPENDIX-II

The code used to build the K-means Codebook has been presented below.

```
[desc_tr, desc_te, imgIdx_tr, imgIdx_te]=
getData('Caltech');

disp('Building visual codebook...')
% Build visual vocabulary (codebook) for
'Bag-of-Words method'
desc_sel =
single(vl_colsubset(cat(2,desc_tr{:}),
10e4)); % Randomly select 100k SIFT
descriptors for clustering

%% K-means codebook
numBins = 256;
[centroids, ~] = vl_kmeans(desc_sel,
numBins, 'verbose', 'distance', 'l2',
'algorithm', 'ann','Initialization',
'plusplus');

disp('Encoding Images...')
% Vector Quantisation

% Training Data
data_train = zeros(10*15,numBins+1);
for i = 1:10 % number of classes
    for j = 1:15 % number of images
        [~,idx_train] =
min(vl_alldist(centroids,single(desc_tr{i
,j})));
        data_train(15*(i-1)+j,1:(end-1))
=
histc(idx_train,1:numBins)/length(idx_tra
in);
        data_train(15*(i-1)+j,end) = i;
    end
end

% Testing Data
data_test = zeros(10*15,numBins+1);
for i = 1:10 % number of classes
    for j = 1:15 % number of images
        [~,idx_test] =
min(vl_alldist(centroids,single(desc_te{i
,j})));
        data_test(15*(i-1)+j,1:(end-1)) =
histc(idx_test,1:numBins)/length(idx_test
);
        data_test(15*(i-1)+j,end) = i;
    end
end
```

APPENDIX-III

In this appendix we provide some additional figures which could not be incorporated in main body of our report due to page limit specified in our coursework.

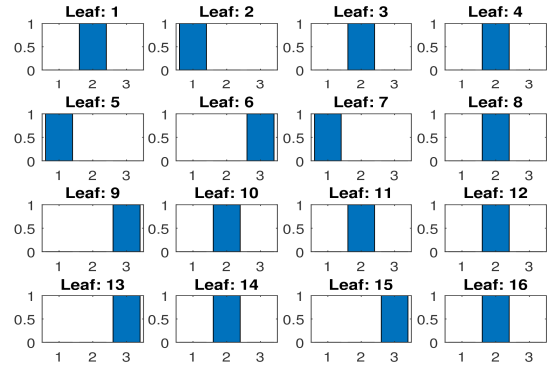

Figure A3.1: Visualization of leaf nodes of a decision tree grown using cubic class of weak-learners.

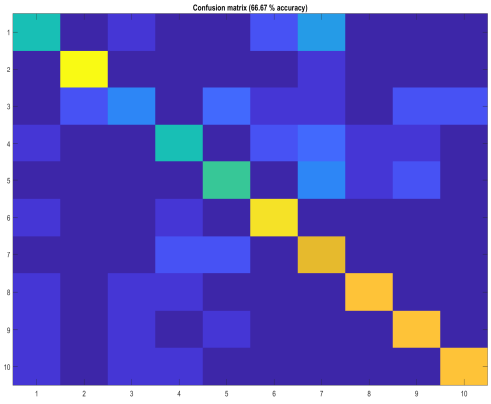

Figure A3.2: Confusion Matrix when `numTrees = 25, numDepth = 5 & splitNum = 30`. (RF Codebook).

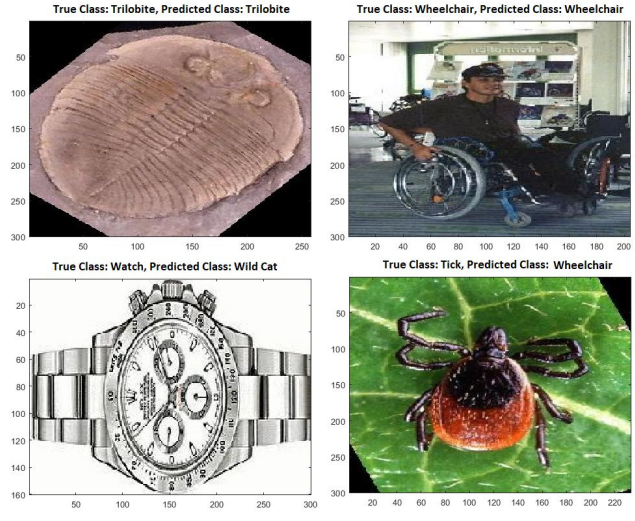

Figure A3.3: Sample Success and Failure classified objects (RF Codebook).