EE4-62 Machine Learning for Computer Vision Randomised Decision Forest Coursework

Biyuan WANG
CID 00821345
Imperial College London

Hiu Yan Cecilia KAN
CID 00828315

1. Training Decision Forest

We investigate the hyperparameters in decision forest training with a 2D spiral toy set of data.

To build the classifier with combined architecture, we generated 4 data subsets from the provided data by bagging, and visualise their distribution, shown in figure 1. Class 1, 2 and 3 are labeled red, green and blue respectively. We performed bagging with replacement to further introduce randomness, as decision trees are influenced by the data from which they stem. Each bagged subset contains 95 data points (63% of full set), but only at most 80% distinctive data, leading to more partial representation of the full data compared to bagging without replacement where all subset data would be distinct.

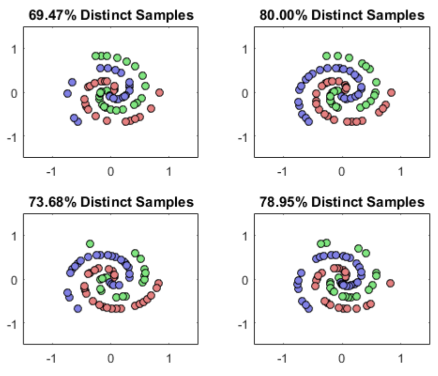


Figure 1. Data distribution in 4 subsets bagged with replacement, identified with percentage distinct samples

Using one data subset, we grow a tree with different split feature functions types and parameters. In figure 2, we show a best trial of an axis-aligned split feature function applied to the base node, with the class histograms of resultant nodes. We observe that the information gain may vary significantly for any single split, from 0.00 to 0.15 over 5 split trials (Appendix A), even if the split function changes in one dimension only, which highlights the importance to optimise tree growth through randomness control parameter ρ, which is the number of thresholds available for selection to yield the best information gain, and randomness can be expressed as a ratio of ρ and the number of all possible thresholds $|\tau|$. Compared with results for $\rho = 3$, used for this coursework unless otherwise specified, best information gain deteriorates drastically for maximum randomness at $\rho = 1$, while $\rho > 3$ does not improve the best information gain for splitting the base node significantly for this data set.

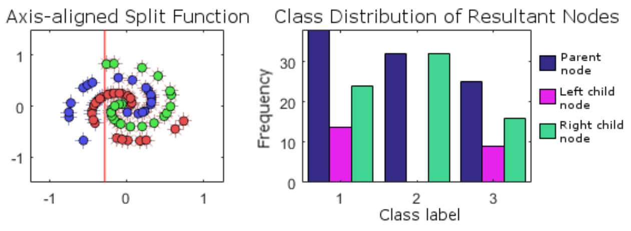


Figure 2. Axis-aligned split function applied to one data subset, and the class distributions in the resultant nodes. The information gain for this best split is 0.12.

We split the same node with a linear split feature function with the hyperplane equation $Ax_1 + Bx_2 + C = 0$, where x_1, x_2 are the coordinates in two random dimensions, A, B, C are randomly generated scalar coefficients (implementation method taken from Karpathy's library). The resultant nodes have class distribution shown in figure 3, and the information gain is 0.13.

Our stopping criteria used is maximum tree depth of 5, as well as minimum node size of 5 data points, to prevent overfitting. The latter criteria may be met as early as in trees of depth 3, therefore a crucial check. The ideal maximum tree depth has to be determined empirically. We look at sampled leaf nodes obtained at maximum tree depth of 5 and 10 (Appendix B for both split functions discussed). While we see a general decrease in the number of class the data in each leaf contains as the tree depth increase from 5 to 10, there is little indication of the tree's performance. On top of these criteria, a test for node with pure class, i.e. Shannon entropy of 0, should be used on top to prevent redundant splitting and improve computational efficiency for a larger tree depth.

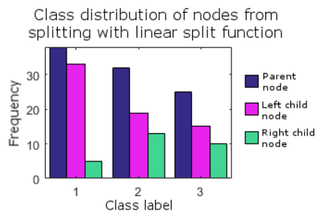


Figure 3. Class distributions in the resultant nodes split with linear split function. The information gain for this split is 0.04.

2. Evaluating Decision Forest on Test Data

For evaluation, four points (-0.5 -0.7), (-0.7, 0.4), (0.4, 0.3), (0.5 -0.5) are tested with random forest (RF) using axis-aligned split function. 10 trees are used for higher accuracy. Each data point is fed to each tree and the leaf node the point ends up is shown in Appendix C. Then majority vote by the 10 trees predicts the data label as in figure 4. We note that the RF is very confident about its labeling for the first point as class 1, even though by visual observation we see ambiguity as it may be classified as class 3. It has lower confidence for points 3 and 4 as there are training data of mixed class within a close region.

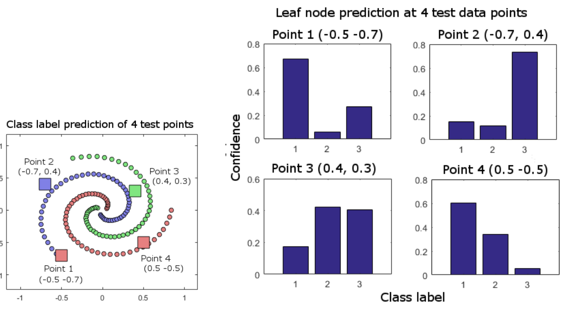


Figure 4. Predicted class labels of 4 test data points and the mode of their leaf node predictions

To have a better visualisation of how the RF extrapolate the data under different parameters, we use a 2D dense grid as testing data. The range of values tested for maximum tree depth and number of trees is limited by computational time. In this work nodes are split in series. As the number of nodes to split in growing the random forest is $2^d \times N$, where d is the maximum tree depth and N the number of trees, it is therefore expensive to increase the maximum tree depth, but less so to increase number of trees.

The visualisation of predicted class labels are shown in Appendix D and E. The plane of the data point is more clearly divided into 3 continuous regions as the parameters are pushed to settings that tend to produce better fit to the data (ie larger numbers of trees, larger maximum tree depth, higher $\frac{\rho}{|\tau|}$), and the boundaries fit the shape of the spiral test data better. The confidence of the RF on its classification also increases as demonstrated in Appendix F. It is difficult to distinguish from the labelling alone the effect of learning more features in the case of increased tree numbers, from increasing tree depth, which gives higher resolution, but for higher tree depth the confidence continues to increase for extrapolation as the class purity increases after each split, but for high number of trees, the confidence for labels from extrapolation remains, The influence of the split feature function type is obvious from the boundary of these regions away from the centre of the plane, where the appeared correlation of data points tends to differ for the two split function types, given similar tree depth or number of trees.

By visual inspection of these results, we labelled the 2D dense grid by RF with the lowest parameters that generates better fit to test data than a lower setting: 200 trees, maximum tree depth at 10, with 8 split trials for each node (Figure 5).

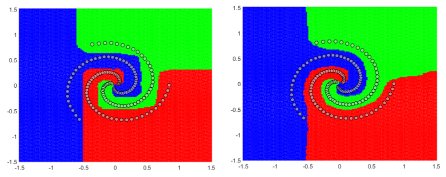


Figure 5. Predicted class labels of 2D dense grid test data with RF of 200 trees, maximum tree depth at 10, 8 split trials, and *Left:* axis-aligned split function; *Right:* linear split function

3. Image Categorisation with Caltech101 Dataset

For this section we assess the performance of random forest with data that is scaled up in size and dimension. RF has emerged as a popular algorithm for image categorisation. Using a subset of Caltech101 dataset, we compare RF against K-means for codebook generation, as well as between classification by RF applied to the resultant codebooks.

3.1. K-means Codebook

Among a pool of scale invariant feature transform (SIFT) descriptors from 15 images of each image class available, 100k are randomly selected. K-means is applied to these descriptors such that 256 cluster centres, each is one codeword, is obtained. Then SIFT descriptors obtained from one image are each assigned to a codeword which has the shortest Euclidean distance from it in feature space. The number assigned to each codeword is compiled into a bag-of-word histogram, the representation of that image after vector quantisation. Examples from the Caltech101 subset are shown in Appendix G.

We experiment with classifying images in their bag-of-word representation with RF of different parameters, and summarise the results in figure 6. Accuracy is measured as the averaged probability of correct prediction by RF across test images of all 10 classes. We find the performance of axis-aligned hyperplane splitting higher and stabler (accuracy variation within 40%-50% over 5 trials) than that of linear hyperplane (below 35% to above 45% for the same setting), due to large increase in $|\tau|$ and in turn randomness when using a more complex split function in high dimension. For axis-aligned splitting, the increase in ρ in range that does not improve best information gain in 2D toy data improves the accuracy for this case from below 35% to 50%, as the same ρ value increase produce a smaller but not insignificant increase in $\frac{\rho}{|\tau|}$ to allow irrelevant feature dimensions to be filtered out. For the forest size, accuracy increases more than 15 percentage points to 55% with increasing number of trees below 20. Above 20 the improvement over an increase to 100 trees is merely within 5 percentage points, and even less from 100 to 300 trees. The cost in added computation time to get more weak learners that learn different features of the data to improve accuracy is high above 20 trees. As for tree depth, overfitting begins to occur above maximum tree depth of 8, as accuracy decreases from peak of just below 50% to 45% at maximum tree depth 15. We deduce the optimal parameters for this task as 200 trees, maximum tree depth 8, 50 split trials, and achieve on average 74.7% accuracy. Figure 7 shows the test results confusion matrices. Categorisation accuracy for images from class 'Windsor chair', 'wrench', and 'Yin Yang' (class label 8, 9, and 10) is consistently high, even for non-optimisted parameters (see Appendix H) as many images from these three classes have plain backgrounds, main subject of similar shape and texture, as well as simple structures. While it is difficult to quantify the complexity of images for comparison, the classification accuracy may not be fully assessed without considering the intrinsic properties of the image data. Boosting algorithm should be considered to assign heavier weights to misclassified images during training, which may help the RF detect the correct features that distinguish image classes.

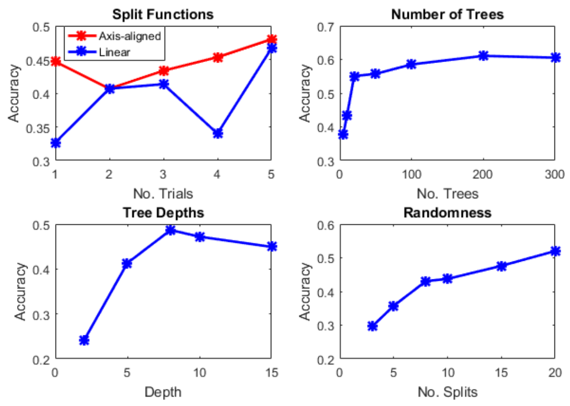


Figure 6. RF test image classification accuracy using bag-of-words from K-means codebook. Axis-aligned split function is used to test different parameters as it offers stabler performance. Parameters values for control are 10 trees, maximum tree depth of 5, 3 split trials

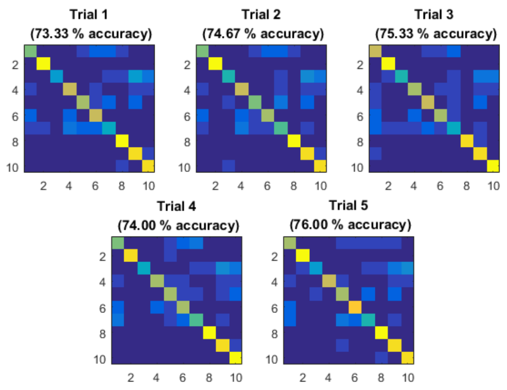


Figure 7. Confusion matrices of RF image classification using K-means codebook as a heat map, with optimal parameters of 50 trees, maximum tree depth 8, 50 split trials. On the diagonal, more yellow colour represents higher probability of accurate prediction, blue is vice versa.

3.2. Random Forest Codebook

The random forest codebook is built at the start using labelled SIFT descriptors. The decision trees firstly act to cluster instead of as a classifier, in the sense that each one is partitioning the feature space according to the labels of the bag of data, and their leaf nodes correspond to regions in feature space. The leaf nodes are the visual words, and are indexed and counted to form a bag-of-word histogram.

We apply the same RF again to construct the classifier, using instead the RF codebook with optimal parameters (tuning discussed in section 3.3), to the image data with different parameters. We observe similar trends in the effect of different parameters on accuracy to that in using K-means codebook from the results (Figure 8). The performance with using linear split feature function is more stable, but accuracy is still lower than using axis-aligned splitting. For tree depth, tree number, and ρ, the highest accuracy reached by tuning each parameter individually is 5 percentage points lower than that for the same settings when K-means codebook is used. Higher number of trees and maximum tree depth, 50 and 10 respectively, compared with 25 and 8, are also required to obtain that level of accuracy.

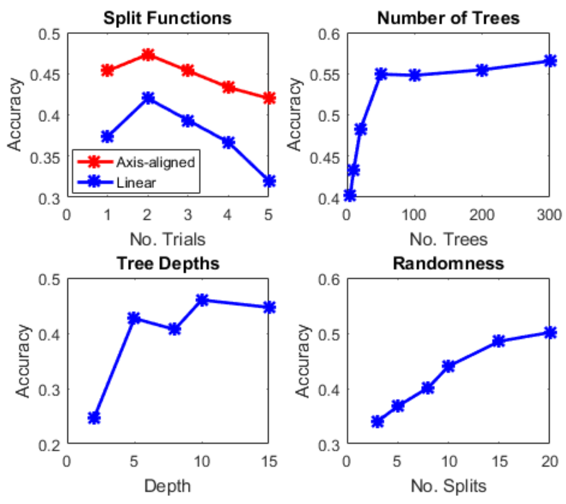


Figure 8. RF test image classification accuracy using bag-of-words from different RF codebooks. Classifier control parameters are: 200 trees, tree depth of 5, 3 split trials.

3.3. Codebooks Comparison

Apart from its intrinsic parameters, the performance of the RF image classifier is closely tied to the codebook used for the task.

One key parameter of concern to any codebook generation method is vocabulary size. Using a larger number of codewords leads to a more discriminative codebook, but the codebook may be less robust, as for instance the impact of descriptor misclassification increase. We explore the effect of changing vocabulary size for both K-means and RF codebooks and plot resultant classification accuracy in Figure 9. It is evident that increasing vocabulary size has a positive influence on accuracy for small number of codewords. From both graphs the vocabulary size above which overfitting sets in to decrease accuracy can also be identified, which is 64 and 128 codewords respectively for K-means and RF codebooks. For similar vocabulary size, classification using RF codebook offers higher accuracy ranging from 65% to 68%, in contrast to 40% to 46% for similar range of vocabulary size from around 64 to around 256 codewords. This may be due to randomness associated with the RF algorithm itself, without having to perform parameter regularisation. There is scope for introducing randomness into the method with K-mean with the choice of initial cluster centers, however at a prohibitively high computational time cost. With RF codebook, a larger vocabulary size can be tolerated before overfitting occurs, which may also explain the higher accuracy achievable, as the RF codebook can be more distinguishing without losing robustness.

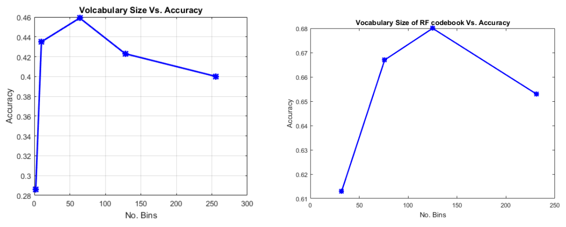


Figure 9. RF test image classification accuracy using different size of bag-of-words from *Left:* K-means codebooks (10 trees, depth of 5, and 3 split trials default settings are used); *Right:* RF codebooks (200 trees, depth of 8, and 50 split trials settings are used)

The RF used for codebook generation has its own parameters that requires adjustment. The forest size influences not only the vocabulary size, but the visual words themselves. Using the RF classifier parameters that produced the best results with the K-means codebook, we measure the accuracy dependency on the individual RF codebook generation parameters. Results shown in Figure 10 has higher accuracy values than results from testing on K-means codebook, so tuning the codebook RF is effective. Similar trends in accuracy dependency on maximum tree depth observed for the RF in codebook generation as in RF for classifier, with close to 70% accuracy achievable at a relatively low value of 5. Simply pushing the numbers up does not increase accuracy for number of trees or ρ for the codebook generation RF. Using 8 trees, equivalent to 125 codewords, yield the highest accuracy of 68%. Accuracy decreases to 65% for 15 trees (231 codewords). $\rho = 5$ gives the highest accuracy of 73%, and for larger values accuracy decreases, to 63% for $\rho = 20$. The codebook RF may require smaller values for these two parameters to prevent regions labelled by the leaf nodes becoming too specific, or small. For example in the case of many clustering trees where the dataset used for each tree would be thinned.

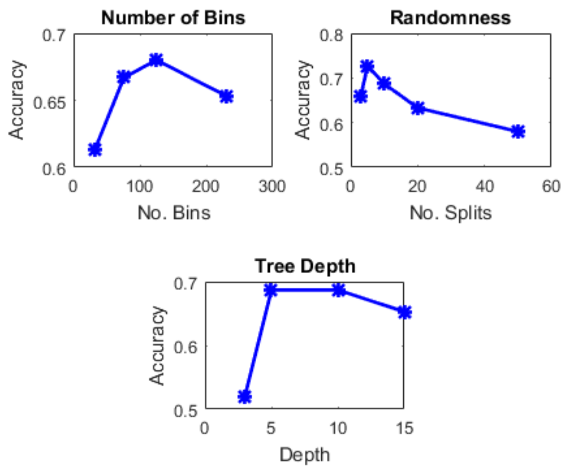


Figure 10. RF test image classification accuracy using bag-of-words from different RF codebooks. RF classifier settings are: 200 trees, tree depth of 8, 50 split trials

We also notice that the RF codebook generation optimisation for one RF classifier is not transferable to another, as we select the best values for individual parameters according to Figure 10 and Figure 8 to build a RF codebook, then train and test a RF classifier, the accuracy obtained is higher than that of optimising individual RF classifier parameters, but not higher than that using values optimised for K-means codebook (Figure 11). The averaged accuracy is calculated as 62.5%. From the confusion matrix, the deterioration in performance is across all image classes, so the lower accuracy is due to intrinsic properties of the RF. As the aim of this work is to investigate RF performance related to its parameters, not to produce optimise RF for a certain purpose, the parameter values are chosen to provide comparison for this cause. In literature, RF codebook has been shown to offer more accurate results than k-means codebooks, and have good resistance to background clutter, which is important as a significant proportion of descriptors detected from an image may not be from the subject of interest [1].

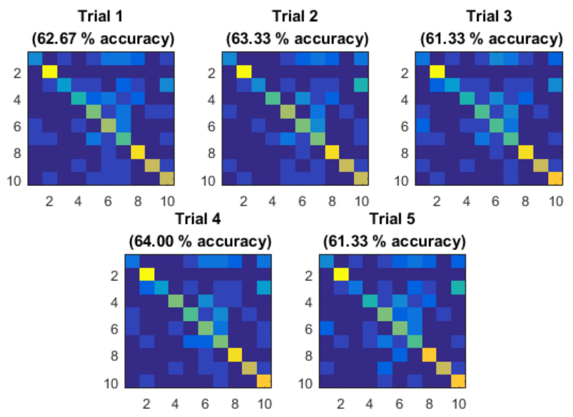


Figure 11. Confusion matrix for RF test image classification as heat map, using bag-of-words for the optimal RF codebook and optimal classifier settings of 200 trees, tree depth of 10, 50 split trials

Using RF for generating codebook is also much more time efficient than using K-means. RF performs little extra computation for data of higher dimensionality, whereas K-means has to compare the Euclidean distance of every descriptor with the cluster centres element-wise. Computational complexity for K-means is $O(DN'K)$, where D is descriptor dimensionality, N' is the number of scalar features from the image, K is the number of clusters (or leaf node for RF). Whereas RF has $O(\sqrt{D}N'logK)$. We measure the time taken to generate a codebook for both methods (Table 1 and Table 2) for changing vocabulary size, that is K. For K-means the increase in codebook building time roughly doubles for doubled vocabulary size, as the complexity is directly proportional to K. As for RF, the increase in codebook building time loosely follow increase in $\log K$ (Figure 12), as the actual complexity also depends on factors such as balance of the clustering tree. The codebook implemented also affects the efficiency of the RF classifier. For the optimised RF classifier settings, training time using K-means codebook is 12.7s, compared to 0.016s when using RF codebook.

Vocabulary Size	Building time / s
32 (2 trees)	0.76
79 (5 trees)	2.5
128 (8 trees)	2.9
237 (15 trees	5.4

Table 1. RF codebook generation time

Vocabulary Size	Building time / s
64	5.1
128	11.3
256	25

Table 2. K-means codebook generation time

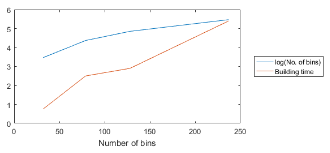


Figure 12. RF codebook generation time relation with theoretical complexity for changing vocabulary size

4. Conclusion

In this coursework, we explored the effect of different parameters of a random forest on its classification, and the application of random forest for image categorisation. While random forest offers fast training and testing and good accuracy, there is a drawback of the need to tune the parameters to fit the purpose of its use. The parameters have to be determined empirically for the best balance between different trade-offs.

Reference

[1] Frank Moosmann, Bill Triggs, and Frederic Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *Advances in Neural Information Processing Systems 19*, pages 985–992. MIT Press, Cambridge, MA, 2006.

Appendices

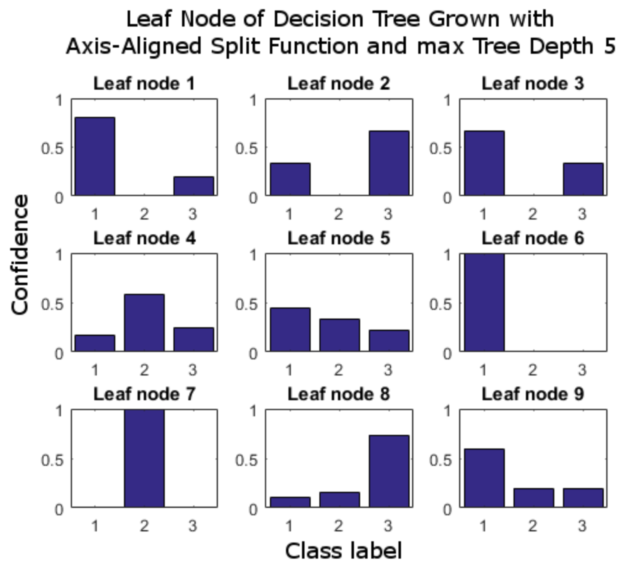
A. Information gain of axis-aligned split function applied to base node of one decision tree

No. of splits	3		5		8		10
Trial	1	2	1	2	1	2	1
IG	0.05	0.06	0.11	0.10	0.15	0.10	0.11
	0.11	0.01	0.01	0,11	0,15	0.11	0.15
	0.01	0.15	0.04	0.15	0.08	0.01	0.12
			0.15	0.10	0.05	0.11	0.02
			0.12	0.01	0.01	0.01	0.01
					0.06	0.07	0.02
					0.00	0.03	0.15
					0.15	0.08	0.07
							0.07
							0.07
Best	0.11	0.15	0.15	0.15	0.15	0.11	0.15

Table 3. Information gain of axis-aligned split function applied to base node of one decision tree for different number of split trials, and the best information gain for that split.

B. Selected leaf nodes of decision tree grown with different split feature function and tree depths

B.1. Leaf nodes from axis-aligned splitting



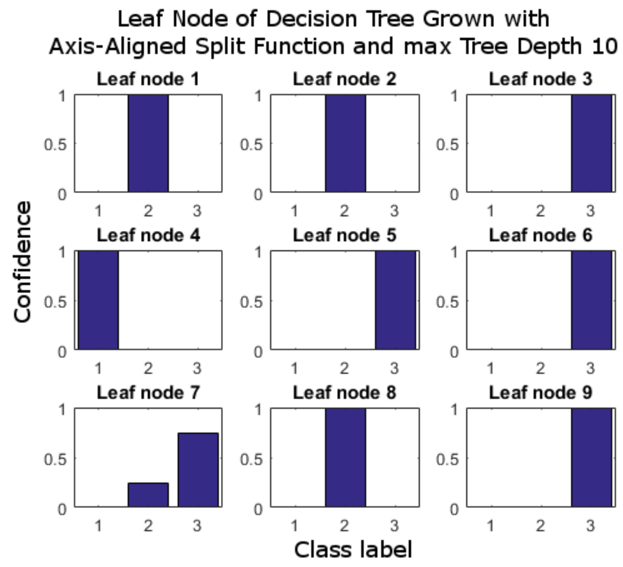
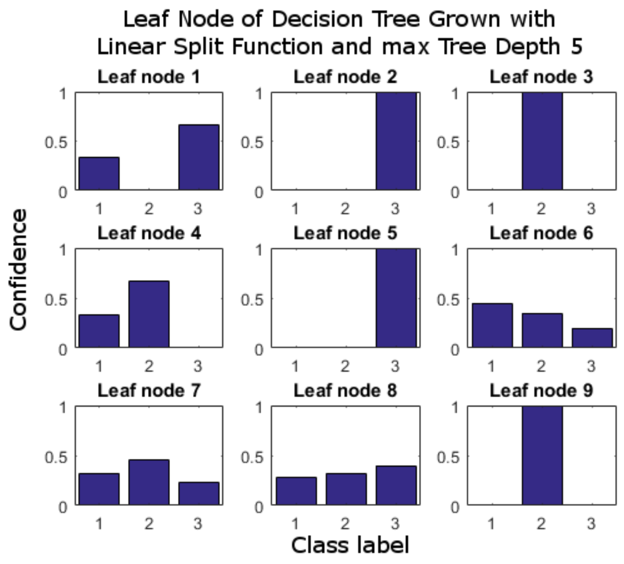


Figure 13. 9 leaf nodes of decision tree grown with axis-aligned split function, with a maximum tree depth of *Top:* 5 trees; *Bottom:* 10 trees.

B.2. Leaf nodes from linear splitting



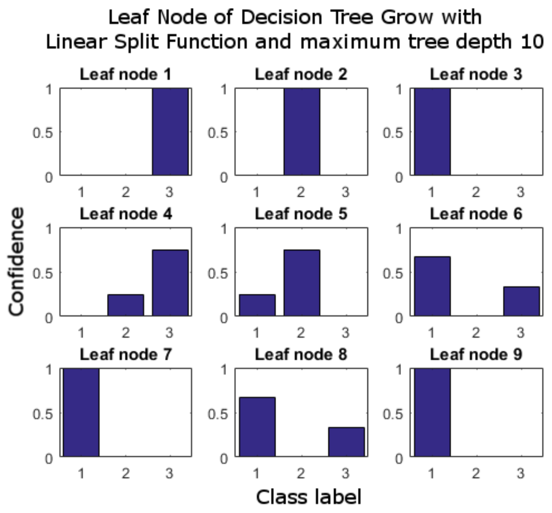


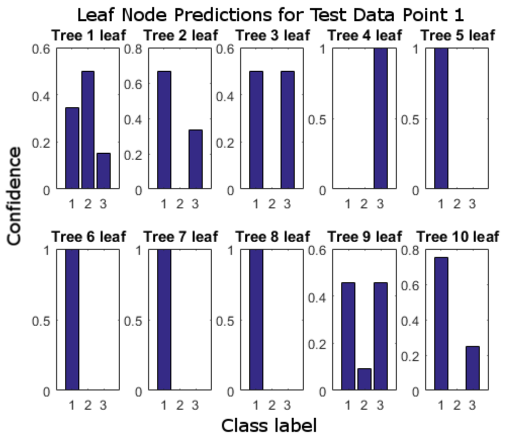
Figure 14. 9 leaf nodes of decision tree grown with linear split function, with a maximum tree depth of *Top:* 5 trees; *Bottom:* 10 trees.

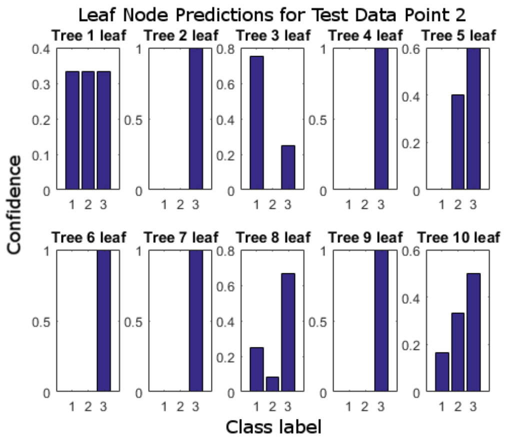
B.3. Count of data label classes in leaf nodes

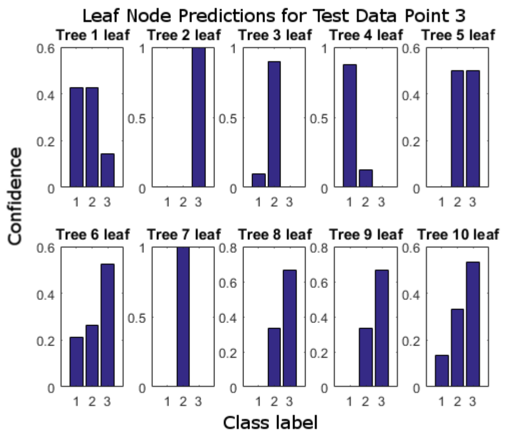
	Axis-aligned			Linear		
No. of class	1	2	3	1	2	3
Tree depth 5	2	3	4	4	3	2
Tree depth 10	8	1	0	5	4	0

Table 4. Count of leaf nodes with data from one, two, or all three classes for axis-aligned and linear split functions with maximum tree depths at 5 and 10.

C. Class distribution in leaf nodes predictors of 4 test data points







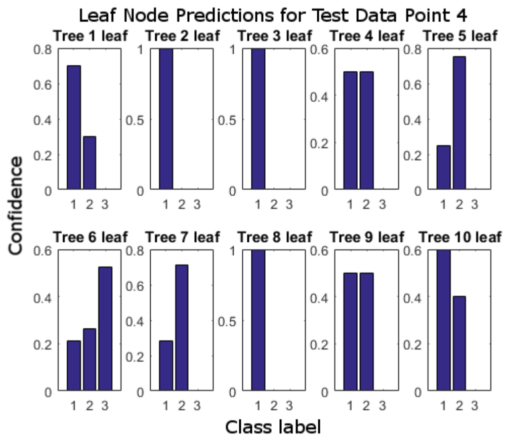


Figure 15. Class distribution in leaf nodes predictors of 4 test data points. Each data point is tested with a trained random forest of 10 trees, with a maximum tree depth of 5.

D. Class label prediction for 2D dense grid test data with axis-aligned splitting

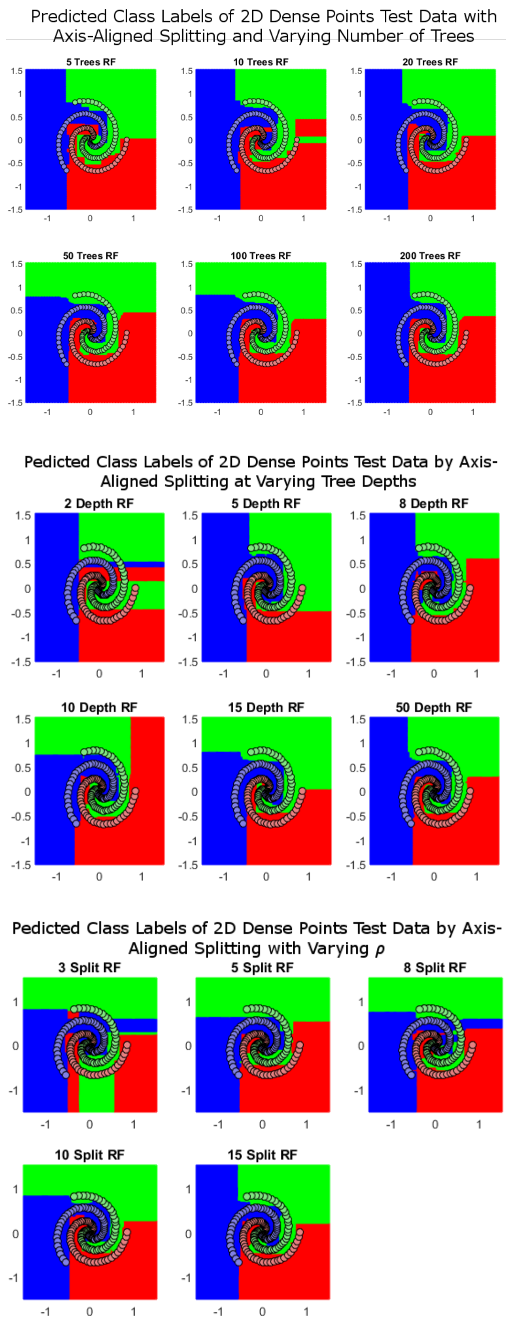


Figure 16. Predicted class labels of 2D dense grid test data with varying maximum tree depth, number of trees in RF, and randomness control parameter ρ.

E. Class label prediction for 2D dense grid test data with linear splitting

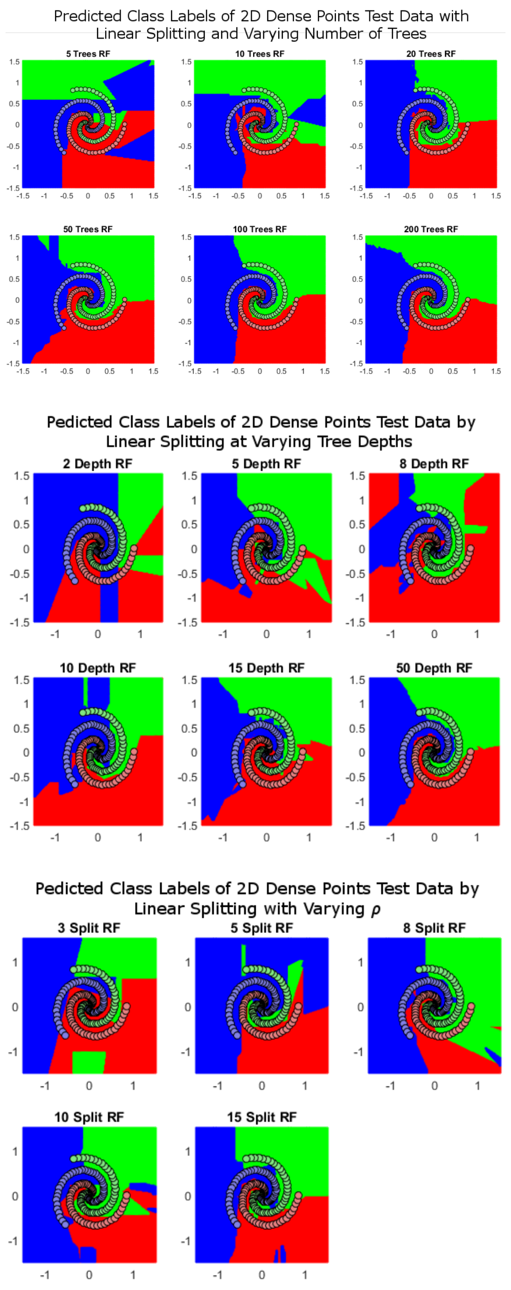


Figure 17. Predicted class labels of 2D dense grid test data with varying maximum tree depth, number of trees in RF, and randomness control parameter ρ.

F. Classification probability of RF with axis-aligned splitting

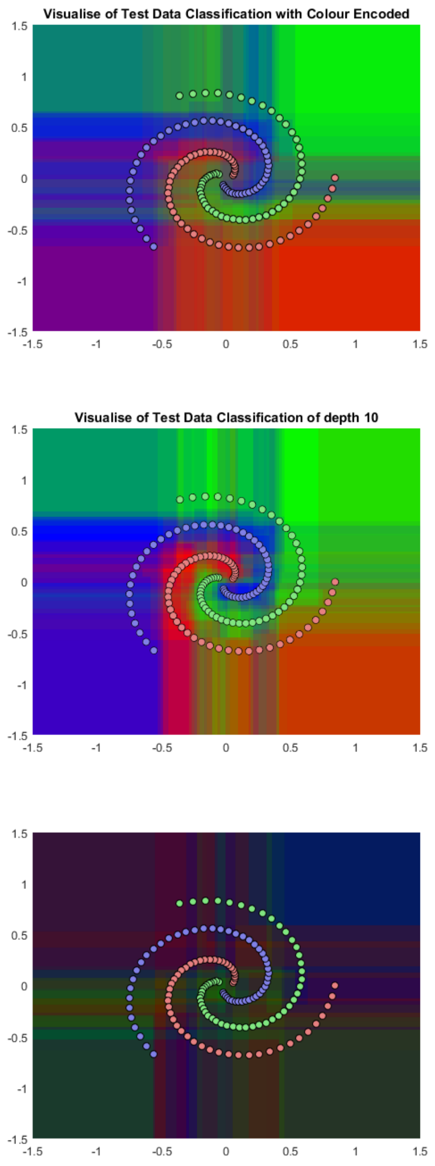


Figure 18. Classification probability of 2D dense grid test points with spiral toy data as training data by RF of *Top:* 10 trees, max depth 5; *Middle:* 5 trees, max depth 10; *Bottom:* 5 trees, max depth 5. The RF confidence is low with smaller number of trees or maximum depth, evident from the darker colour in bottom graph in contrast with other two results, which is due to a more similar probability for different class labels for each data point.

G. Bag-of-words histogram generated with K-means codebook

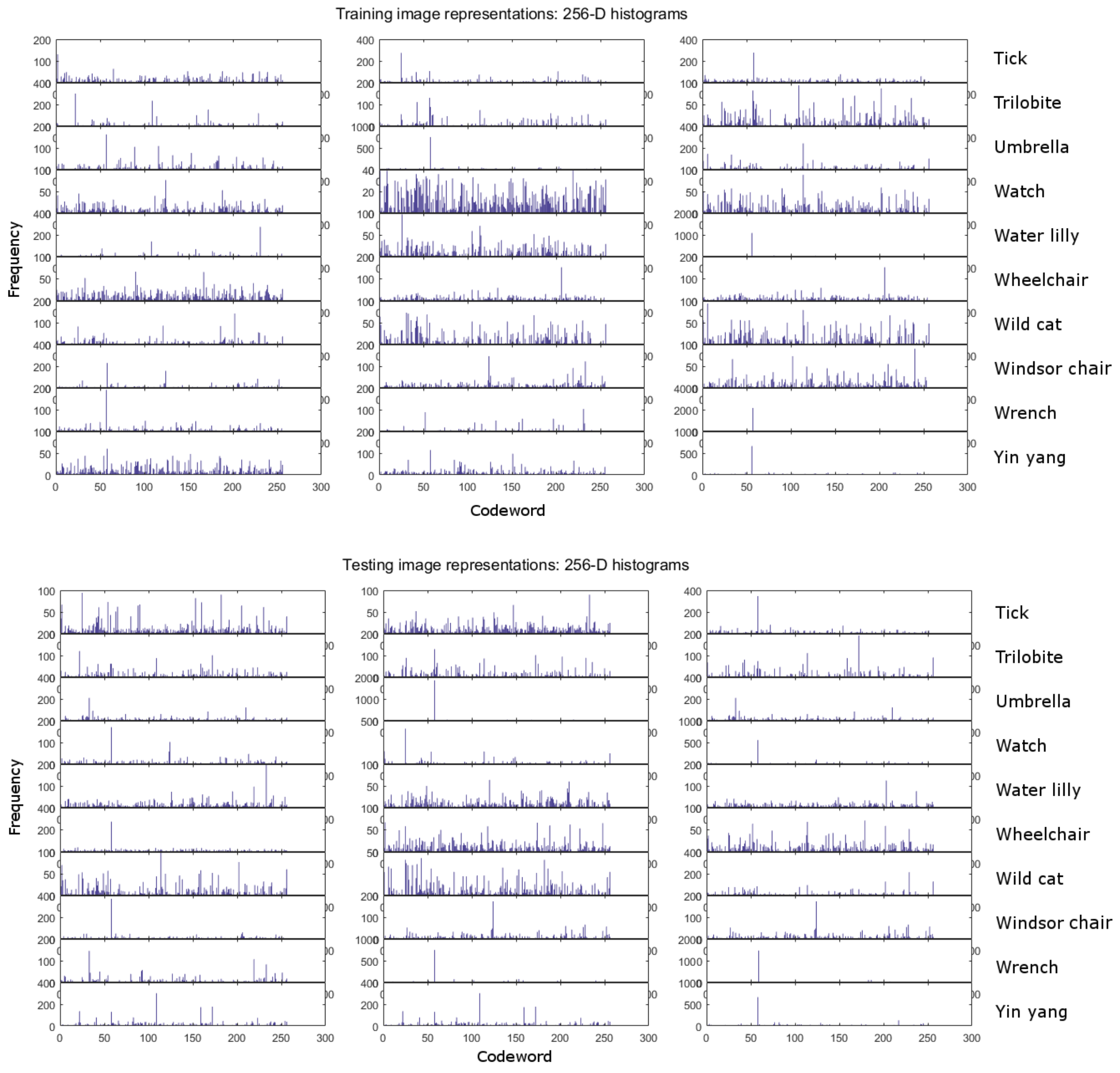


Figure 19. Bag-of-words histogram generated using K-means codebook. Each row shows histogram representations of 3 images from one class of images.

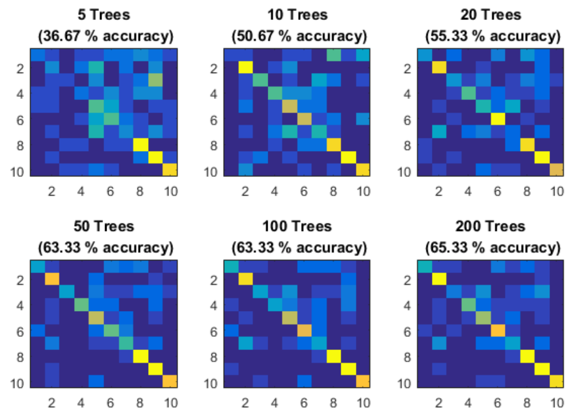


Figure 20. Confusion matrices for RF image classification using K-means codebook for different number of trees

H. Confusion matrices for RF image classification using K-means codebook for different number of trees

I. Comparison of different vocabulary size

Table 5. Comparison of different vocabulary size

Trial	**1**	**2**	**3**	**4**	**5**	**Average**
256 Bins	0.473	0.413	0.36	0.467	0.29	0.40
128 Bins	0.413	0.367	0.407	0.44	0.487	0.423
64 Bins	0.5	0.447	0.44	0.467	0.44	0.459
10 Bins	0.46	0.437	0.413	0.427	0.44	0.435
2 Bins	0.289	0.289	0.294	0.3	0.26	0.286

J. Code for K-means codebook generation

```matlab
% Build visual vocabulary (codebook)
        % K-means clustering (codebook) for 'Bag-of-Words method'
        % codebook: independent features (a representative of similar patches)
        desc_sel = single(vl_colsubset(cat(2,desc_tr{:}), 10e4)); % Randomly select 100k
            SIFT descriptors for clustering

        numBins = 64; % 64 feature
        tic;
        book = kmeans(desc_sel, numBins); % construct visual codebook (256 clusters)
        booktime=toc;
        % Vector Quantisation
        k = 1; figure('Units','normalized','Position',[.5 .1 .4 .9]);
        suptitle('Training image representations: 256-D histograms')
        [ num_class, num_img ] = size(desc_tr);
        data_train = zeros(num_class*num_img, numBins+1);
        for i = 1:num_class %  Number of classes in the training set
            for j = 1:num_img
                % calculate euclidean distance to assign the visual words
                % to the nearest cluster, counts the number of visual
                % words allocated in the cluster
                E_dist = pdist2(book' , desc_tr{i,j}');
                [ ~, hist_idx ]= min(E_dist); % find min in each col: 1xcol
                data_train(k,(1:end-1)) = hist(hist_idx, numBins);
                data_train(k,end) = i;  %
                k = k+1;
            end
            % visualise hist of training data bag of words
            for m=1:3
                if i==1
                    subaxis(length(classList),3,m,'SpacingVert',0,'MR',0.1);
                else
                    subaxis(length(classList),3,m+(i-1)*3,'SpacingVert',0,'MR',0.1);
                end
                if i==1
                    r = randi([1, 15],1);
                else
                    r = randi([(i-1)*15+1, i*15],1);
                end
                bar(data_train(r,1:end-1));
            end
        end
```

K. Code for vector quantisation using K-means codebook

```
% Quantisation
        k = 1; figure('Units','normalized','Position',[.5 .1 .4 .9]);
        suptitle('Testing image representations: 256-D histograms')
        [ num_class, num_img ] = size(desc_te);
       data_query = zeros(num_class*num_img, numBins+1);
        for i = 1:num_class %  Number of classes in the training set
            for j = 1:num_img
                E_dist = pdist2(book' , desc_te{i,j}');
                [ ~, hist_idx ]= min(E_dist);
                data_query(k,(1:end-1)) = hist(hist_idx, numBins);
                data_query(k,end) = i;
                k = k+1;
            end
            % visualise hist of testing data bag of words
            for m=1:3
                if i==1
                    subaxis(length(classList),3,m,'SpacingVert',0,'MR',0.1);
                else
                    subaxis(length(classList),3,m+(i-1)*3,'SpacingVert',0,'MR',0.1);
                end
                if i==1
                    r = randi([1, 15],1);
                else
                    r = randi([(i-1)*15+1, i*15],1);
                end
                bar(data_query(r,1:end-1));
            end
        end
```

L. Code for RF codebook generation

```
    disp('Building visual codebook...')
    % Build visual vocabulary (codebook)
    % K-means clustering (codebook) for 'Bag-of-Words method'
    % codebook: independent features (a representative of similar patches)
    %        desc_sel = single(vl_colsubset(cat(2,desc_tr{:}), 10e4)); % Randomly select
        100k SIFT descriptors for clustering
    desc_sel = [];
    for i = 1:size(desc_tr,1) % for all 10 classes
        desc_label = cat(2,desc_tr{i,:})'; % extract the training SIFT in current class
        idx = randsample(length(desc_label),10000); % randomly select 10k SIFT from this
            class
        desc_ = [desc_label(idx,:) i*ones(10000,1)]; % add class labels to the selected SIFT
        desc_sel = [desc_sel ; desc_]; % concatenate all selected classes together
    end
    tic
    param.num = 8;
    param.depth = 5;        % trees depth
    param.splitNum = 5;     % Number of split functions to try
    param.split = 'IG';
    desc_sel_tr = single(desc_sel);
    codebook = growTrees(desc_sel_tr,param);
    booktime=toc;
```

M. Code for vector quantisation using RF codebook

```
% Vector Quantisation -- get data_train

    data_train = zeros(num_img*num_class,length(codebook(1).prob)+1);
    for i = 1:num_class
        for k=1:num_img
            dense_leaves = testTrees_fast(desc_tr{i,k}',codebook);
            dense_leaves(dense_leaves==0)=1;
            for c=1:length(codebook(1).prob) % num of nodes in this tree
                data_train(15*(i-1)+k,c)= length(find(dense_leaves==c));
            end
            data_train(15*(i-1)+k,end) = i;
        end
    end
```