

# Real-Time Digital Signal Processing Final Project

Sagar Patel, CID: 00842688, and Pranav Malhotra, CID: 00823617  
 (Dated: 6th March 2017)

## I. TRAIN DECISION FOREST

### A. Bootstrap Aggregating

In Bootstrap Aggregating (Bagging), the multiple data subsets are generated by uniformly choosing data points from the training set with replacement. This introduces some randomness into the training data used to grow each tree in the random decision forest. The in-built matlab function `randsample` is used to generate each dataset.

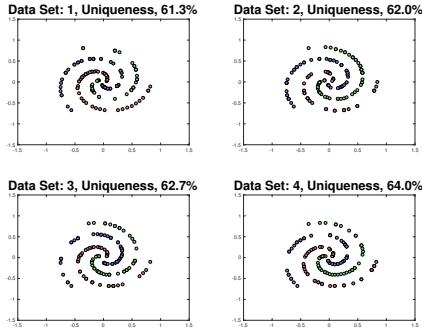


FIG. 1. Overview of spectral subtraction process

The resulting data subsets sampled from the Toy Spiral data are shown in Figure 1. Each dataset has 150 points however each spiral is not complete; some part of the spiral are missing. This is due to the fact that samples are drawn from the dataset with replacement. The amount of uniqueness in each dataset is presented in Figure 1 as well. Ideally, as the size of our data set increases, the uniqueness percentage should approach 63.3%. The finite size of our dataset results in uniqueness values spread around the ideal. The table below shows the probability distributions of each class within the dataset. The roughly equal distributions corroborate the fact that sampling was performed uniformly.

Dataset Index	Class Probabilities
1	[0.3733, 0.2733, 0.3533]
2	[0.2800, 0.3667, 0.3533]
3	[0.3333, 0.3200, 0.3467]
4	[0.3467, 0.3133, 0.3400]

### B. Split Function

The first step in the training of the decision forest is to develop the weak-learner functions that will be used to split the data at each node of the tree. Multiple weak-learner functions were implemented, namely axis-aligned and two-pixel. Linear, quadratic and cubic split functions were also implemented. Note that for the quadratic and cubic split functions have the form expressed in (1) and (2) respectively.

$$h(\mathbf{v}, \theta) = [a_1x_1^2 + a_2x_2^2 + a_3x_1x_2 + a_4x_1 + a_5x_2 > \tau] \quad (1)$$

$$\begin{aligned} h(\mathbf{v}, \theta) = & [a_1x_1^3 + a_2x_2^3 + a_3x_1^2x_2 + a_4x_1x_2^2 + a_5x_1^2 \\ & + a_6x_2^2 + a_7x_1x_2 + a_8x_1 + a_9x_2 > \tau] \end{aligned} \quad (2)$$

The graphs below show the results obtained for all the weak-leaners tested, except for the two-pixel test. The two-pixel test did not perform well when tested with the 2-dimensional toy spiral data. Notice that the graphs obtained show the results when `param.splitNum` was set at 3. The results are indicative of the fact that a stronger class of learner functions does not necessarily translate into a greater information gain when the number of split functions tested is small.

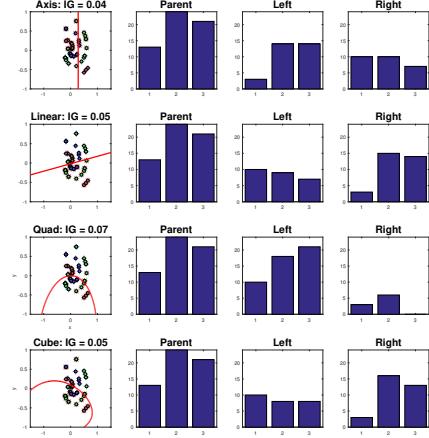


FIG. 2. Overview of spectral subtraction process

The graph in Figure 3 shows the effects that the strength of the learning class has when tested on nodes with a small number of data points. Observe the split function obtained for the quadratic and cubic learners. **There is clear overfitting of the data.** This is completely in line with the results obtained from VapnikChervonenkis theory. Quadratic and Cubic functions have a greater VapnikChervonenkis (VC) dimension and trained on a small number of points, these learners will overfit data not generalize well. The random forest algorithm provides a mechanism, the committee machine, to combat over-fitting in each individual tree. As such, in the next section, it should be noted that if strong non-linear learners are used, the number of trees should also be increased to achieve the same performance as a weak learner.

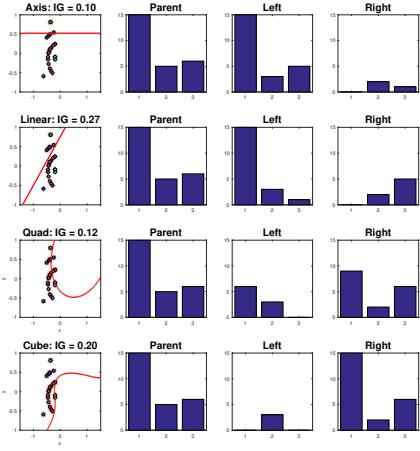


FIG. 3. Overview of spectral subtraction process

By simply analyzing Figure 2 and 3, we clearly see the advantages of using a stronger learning class. For this, we have to set `param.splitNum` to 20. Figure 4 makes it clear that stronger learning classes are able to split the data more efficiently and obtain a higher information gain. The discriminating power of the learning classes is not obvious when the number of split functions tested is small however increasing `param.splitNum` clearly highlights the each classes discriminating power.

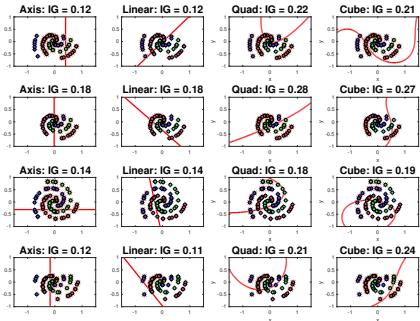


FIG. 4. Overview of spectral subtraction process

Using a stronger learning class such as the quadratic or cubic non-linear functions has its benefits. They do lead to an increased information gain at each node. However, the increasing the strength of the learning class should also be accompanied with a small increase in the number of split functions tested. Stronger class have more degrees of freedom and thus search for a linear separator in a higher dimensional space. As such, keeping the number of split functions that we test constant is not ideal.

### C. Growing Tree

Lastly, the leaf nodes have been visualized in Figure 6. The leaf nodes have been generated using the axis-aligned weak learner with `param.splitNum` set to 3. Most leaves contain only one class of data points; the recursive splitting of the data points down the tree has a distilling effect. There are some leaves with 2 or more classes of data. This can be attributed to the difficulty in splitting the region near the middle of the Toy Spiral where the data points are tightly clustered.

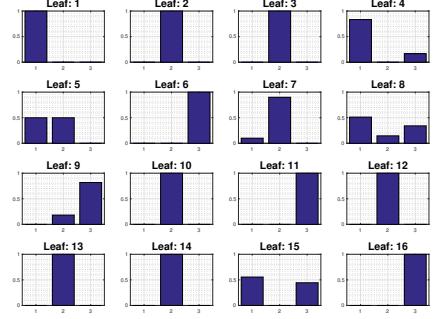


FIG. 5. Overview of spectral subtraction process

The tree was built using two spotting criteria. Firstly, a node is considered a leaf if there are less than 5 training points reach the node. The reason for this is because, training on such a limited number of points is not going to lead to good results. If only 5 points arrive at a node, they will be concentrated within a small region. Training based on these points will lead to over-fitting. The second stopping criteria is the maximum tree depth. This is simple to implement and gives us direct control on the maximum number of leaves. Using the tree-depth to control the number of leaves will come in handy when growing an random forest code-book.

## II. EVALUATING DECISION FOREST

The 4 test points are evaluated on a forest containing 3 trees; the results obtained are graphed below. The histograms show the probability distribution functions at each leaf for each point. The averaged values have also been plotted. Note that test point 1 (-0.5, -0.7), has been misclassified; ideally, the point should be blue. The reasons for this misclassification is multi-fold. One of the main factors is that the random forest currently only have 3 trees and thus the averaging effect of the committee machine is limited.

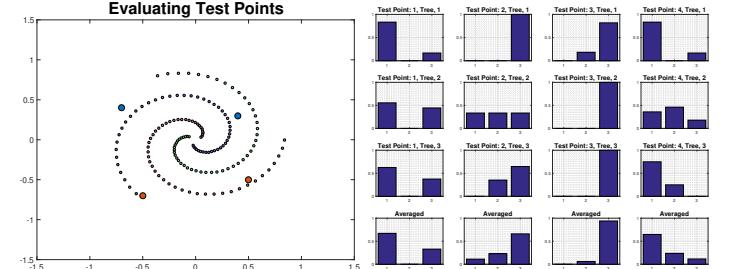


FIG. 6. Overview of spectral subtraction process

### A. Varying Number of Trees

To directly combat the misclassification observed above, the first parameter that is varied is the number of trees in the forest. The graph below shows that increasing the number of trees has a considerable effect on the classification process, when the absolute number of trees is small. Increasing the number of trees from 100 to 200 has negligible effect. Notice the straight axis-aligned class boundaries obtained in the region of the grid for which extrapolation is required. Training data is contained within the unit square whereas the region for which the classifier is tested extends beyond the unit square. The classifier is expected to extrapolate the

class boundaries. Since the axis-aligned weak learner is used, the extrapolation leads to straight axis-aligned lines in the extrapolated region.

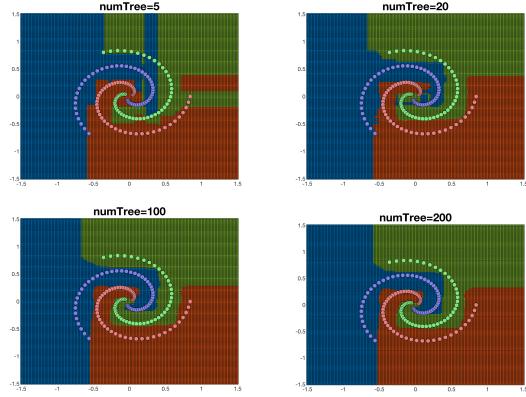


FIG. 7. Overview of spectral subtraction process

It is important to note that increasing the number of trees has computation costs associated with it. Although this was not evident when dealing with the toy spiral dataset, the computational penalty that one has to pay for increasing the number of trees becomes abundantly clear when dealing with the Caltech 101 dataset.

## B. Varying Number of Split Functions

Next, we vary the number of split functions, `param.numTree` is fixed at 10. By increasing the number of split functions that we try, we are decreasing the randomness of inherent within each tree. If all trees are assumed to be completely uncorrelated, then  $E_{com} = E_{av}/T$ ; where  $E_{com}$  is the error incurred by the random forest and  $E_{av}$  is the average error of each tree. If the trees are correlated, then  $E_{com} \leq E_{av}/T$ . Note that, even with correlated trees, it is possible that the committee machine produces good results with  $E_{av}$  is low. A good balance point has to be found, such that to achieve a low  $E_{av}$ , some correlation is allowed. Observe below that with small number of split numbers, artifacts are observed. Each tree is a very bad learner and thus, even though  $E_{com} \approx E_{av}/T$ ,  $E_{com}$  is still rather high. Increasing the number of split functions to 7, the random forest is clearly trying its best to map the spiral nature of the training data using axis-aligned weak learners. Setting `param.splitNum` to 15, not only is each tree highly over-fitted, the committee machine cannot average away the over-fitting artifacts because the trees are highly correlated.

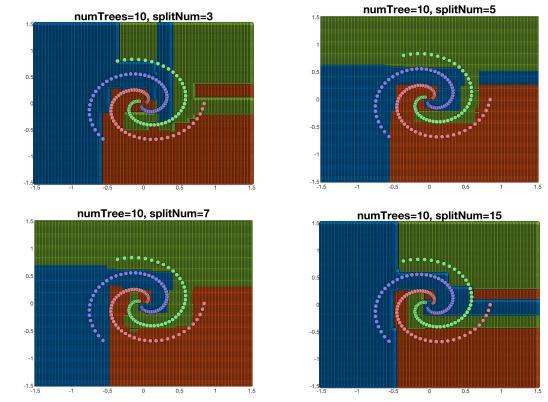


FIG. 8. Overview of spectral subtraction process

## C. Varying Tree Depth

The third parameter that can be varied is the depth of the tree. It can be seen that more accurate results are produced when tree depth is increased. With tree depth set at 3 or 5, there are very obvious artifacts that appear; a thin red line when depth is set at 3, and a massive blue region when the depth is set at 5. These are clearly anomalies that have come about because the classes have not been sufficiently refined at the leaf nodes. Increasing the depth to 7, the obvious anomalies have now disappeared but small artifacts still remain at the center of the graph. These can be further refined if the tree depth is increased to 9.

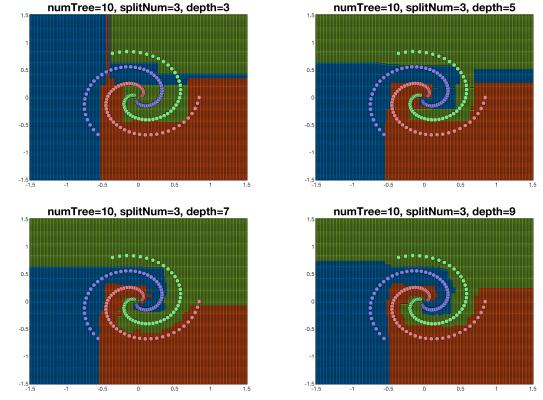


FIG. 9. Overview of spectral subtraction process

Tree depth should not be increased arbitrarily for two reasons. As we descend down the tree, training points become refined and get clustered together. If we arbitrarily keep increasing the depth, we will be attempting to separate points which lie very close to one another. The weak-learner will have an effect over the entire range of values for which the algorithm is used. Observe what happens when we train on very few number of points in Figure 3 .

Secondly, increasing tree depth will increase computational complexity exponentially. It is much wiser to average out anomalies using a greater number of trees because while trees can be trained and tested in parallel, each tree must be grown sequentially.

#### D. Testing with Different Weak Learners

Note that all the discussion above has all been made with respect to the axis-aligned weak learner. This is important to note that ideal values of splitNum and the tree depth vary with the weak learner used. The graphs below show the effects of using different weak-learner classes. Notice that the class boundaries in the extrapolated region follow the general shape of the learner class used. For reference, an random forest trained using the axis-aligned weak learner with all 3 parameters turned has also been graphed.

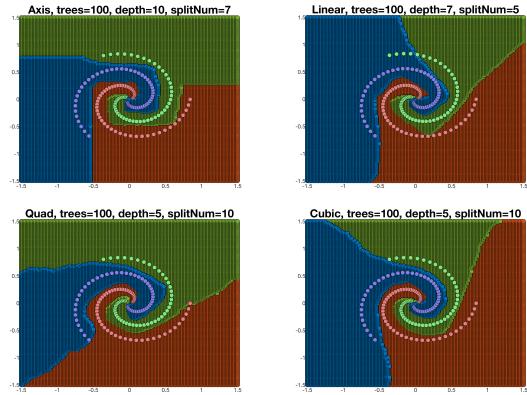


FIG. 10. Overview of spectral subtraction process

### III. RF CLASSIFIER

The RF is trained using the a vocabulary generated using the K-means algorithm and subsequently performance is tested. The confusion matrix below shows that perfect object classification is obtained for the Trilobite and the Windsor Chair, whereas algorithm is the worst at identifying watches.

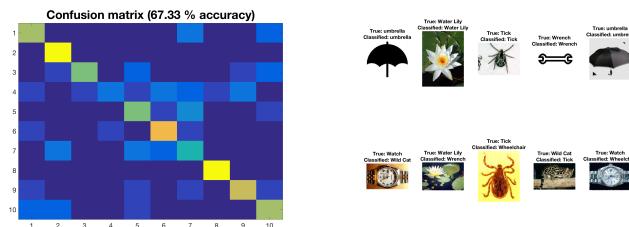


FIG. 11. Overview of spectral subtraction process

#### Vocabulary Size

Using the K-means algorithm to form a Bag-of-Words (BoW) representation, we have direct control over vocabulary size. The BoW can be interpreted as a dimensionality reduction technique that shrinks an image into a histogram indicating the frequency of occurrence of codewords, which in this case are the cluster centers. Using a small vocabulary, the dimension of the space in which the Random Forest Classifier acts is small. The classes will be harder to separate in a low dimensional space. As such, we observe that increasing the number of bins, thereby increasing the vocabulary size, we obtain a greater classification accuracy.

Notice however the performance increase by increasing the vocabulary size from 512 to 1024 is marginal, when there are 50 or 150 trees in our forest. Performance drops when an extremely large number of bins are used but the number of trees is kept low. With 1024 codewords in our dictionary, the  $E_{av}$  is high. The high error that each tree incurs can either be averaged away using the committee machine, or by increasing depth and the number of split functions tried, we can lower  $E_{av}$ .

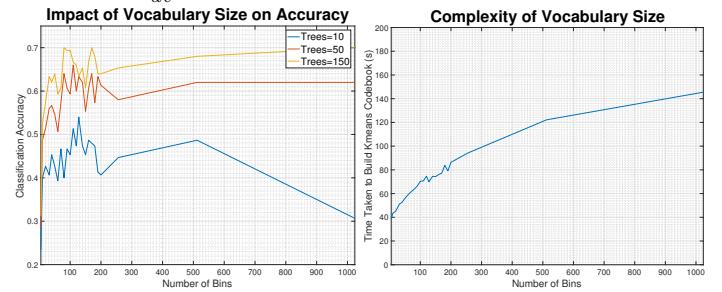


FIG. 12. Overview of spectral subtraction process

#### Number of Trees

The graphs below corroborate the discussion above about the effects that the number of trees has on performance of the Random Forest algorithm. Regardless of the vocabulary size, increasing the number of trees yields an increase in performance. However, setting `numTrees` to be greater than 180 does not lead to any significant improvements in the recognition accuracy. Also, we observe that increasing the number of trees causes computational complexity to grow linear. Note that, no parallel processing has been done as of yet; the algorithm is susceptible to parallelism.

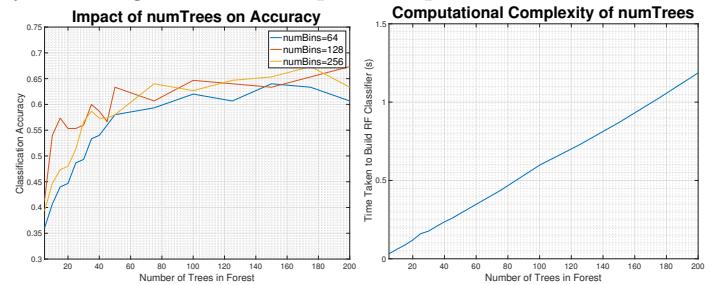


FIG. 13. Overview of spectral subtraction process

#### Number of Split Functions

Again, the graphs below corroborate the discussion above about the effects of increasing the number of split functions. We observed visually that setting `numSplits` to be small lead to artifacts appearing the Toy Spiral data. These artifacts are analogous to the poor recognition accuracy obtained in the figure below. We also observed that setting `numSplits` to be large, overfitting started to occur. Again, we observe this numerically as the performance starts to decrease after a certain point. Although the general patterns are similar, notice how different the scales are. For performance to drop, we need to set the number of split functions tested at each node to be 150. The reason for this is that instead of separating points in a 2-dimensional space as in the case of the Toy

Spiral data, we are now separating points in much higher dimensions. In fact, the graphs below are obtained using a vocabulary consisting of 256 codewords. As such, to correlate the trees in such a large dimension, we need to try a lot of split functions.

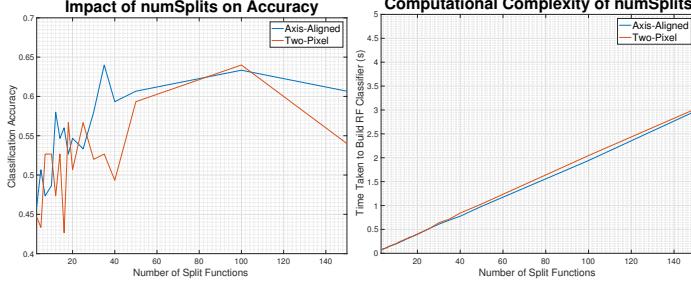


FIG. 14. Overview of spectral subtraction process

### Depth of Tree

Lastly, the impact of tree depth is studied. The most important thing to notice from the graphs is how poorly the algorithm scales in terms of complexity. Note that the graph shows the average time taken to grow each tree in the forest. This is expected since the number of nodes in our tree grows exponentially with tree depth. For relative good performance, we would need at least 150 trees in our forest. If each tree took 10 seconds to grow, simply training the forest would take approximately 25 minutes. As mentioned above, increasing tree depth is not that wise and other parameters should be tuned to increase performance. Anyway, performance does not always increase with an increase in tree depth. Notice that for the axis-aligned split function, the peak performance occurs at a depth of 8.

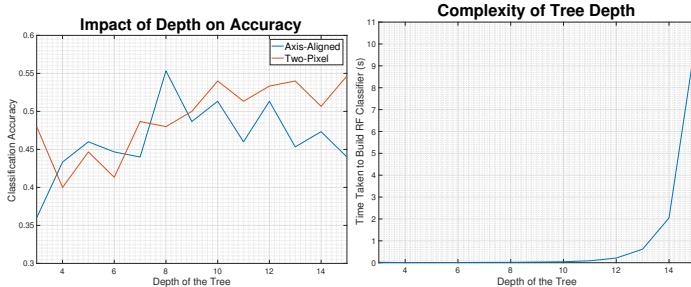


FIG. 15. Overview of spectral subtraction process