Face Detection Using Ensemble Methods – Random Forest and Mixture of Experts

Wasim Khan
Dept. of Electrical & Computer Eng.
Boston University
wasimk95@bu.edu

Lingshan Yang
Dept. of Electrical & Computer Eng.
Boston University
yls@bu.edu

Abstract

Face detection is one kind of biometric detection technique based on facial features. It has many practical applications such as in security systems, electronic passports or identity cards, and tracking fugitives. We plan to explore, understand and implement face detection system using Random Forest and Mixture of Experts, two implementations of ensemble methods. In addition, we will implement K-Nearest Neighbor as a baseline and then compare the performance based on several metrics.

1. Introduction

Face detection is a problem that involves extracting features from an image or a video, and identifying human faces. Research of face detection began in 1960s and quickly improved due to the development of computer techniques and optical imaging techniques in the 1980s. There are several challenges in face detection. First are the internal changes of faces such as different skin colors, face shapes, facial expressions and facial shelters (like hair, beard, glasses and masks). Second is the external changes such as different filming angles, light conditions and imaging methods. [1]

We will be implementing ensemble methods and evaluating their performance face detection. An ensemble method itself is a supervised learning algorithm. It combines several learning algorithms to gain a better predictive performance. Although there are appropriate learning algorithms for a particular problem, ensemble methods are designed to find the best solution when facing a new problem. In other words, the ensemble methods have more flexibility to fit the training data. Due to the diversity among the algorithms, it can address over-fitting problems in some cases. [2]

2. Literature Review

2.1. Random Forest

Random forest is one kind of classification method to combine many tree classifiers. There are two characteristics of random forests, one is randomly selecting training data and the other is randomly selecting features. Using ensemble of trees and letting these trees vote for the most popular class can significantly increase the accuracy of the tree classifiers. Random vectors are created for each tree independently and each tree grows according to the values of the random vectors. Each node of a tree will also randomly select features. Compared to the Adaboost, random forests' error rate is more robust regarding the noise. [3]

2.2. Mixture of Experts

Typical ensemble methods train all of the base learners using the same algorithm, and then combine results to make a prediction. Instead, Mixture of Experts trains each weak learner to specialize in a specific part of the dataset. A gating function is used to assign probabilities to each expert, which determines the likelihood of picking that expert given the specific input, and is modeled by a softmax function [4]. Since the gating network outputs higher probabilities when the input belongs to the region that the learner specializes in, the parameters for each weak learner will only be affected by data that the weak classifier is an expert in. And just like other ensemble methods, the predictions of each weak classifier can be combined, although weighted by the gating network manager, to predict the final outcome.

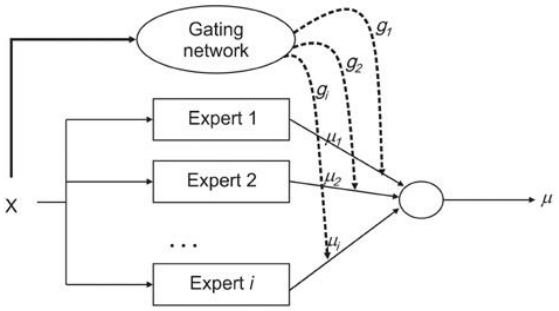


Figure 2.2.1 Mixture of Experts Model.

2.3. Baseline - K-Nearest Neighbor Classifier

One of the most basic and popular classifiers, K-Nearest Neighbor (K-NN) has good performance in many cases. The essential meaning of K-NN is to put the training data into a vector space, then count the k nearest points' classes of one sample in the vector space and assign the most frequent class to that sample. In the case of face detection, there are only two classes, one positive class that has faces and one negative class that does not. It assumes that all the data with faces will cluster together. [5] A test sample inside the cluster will be classified as a positive sample while any test samples outside the cluster will be treated as a negative one.

3. Algorithms and Solution Approaches

3.1 Algorithm 1: Random Forest

Random Selection of Data

First, take some samples from the training set to form a subset, which has the same number of samples as the training set. Samples in different subsets can be repeated and the samples in the same subset can be repeated. Second, use the subsets to construct decision trees. When testing, we classify the test samples using all the decision trees that we constructed in the previous step. The predicting labels are given by counting the votes of different decision trees.

Random Selection of Features

Similar to the random selection of data, construction process of each decision tree does not use all the features, but randomly selects certain features to form a smaller feature set and then selects the best features from it. In this way, the decision trees are different from each other which results in the diversity of the whole ensemble system. According to the principle of ensemble method, the diversity can improve the system's classification performance.

How to Build a Decision tree

When constructing the decision trees, we should compute information gain to sort all the remaining selected features. The feature with the most information gain will be selected to construct the node (start from root node). In the next level of the tree, repeat the previous steps. There are three algorithms to split the decision tree.

(i) ID3: using the information gain as the splitting rule - select the feature with the biggest information gain.

Entropy:

$$H(P) = -\sum_{i=1}^{k} p_i \log_2 p_i \quad (1)$$

Conditional Entropy:

$$H(P,X) = -\sum_{i=1}^{m} P_{x_i} \sum_{j=1}^{k} p_j \log_2 p_j \quad (2)$$

Information Gain:

$$Gain(P,X) = H(P) - H(P,X) \quad (3)$$

(ii) C4.5: using the information gain rate as the splitting rule. It can prevent selecting the features with most classes in ID3.

Splitting information:

$$Split(P,X) = -\sum_{i=1}^{m} \frac{|D_j|}{|D|} \times log_2 \left(\frac{|D_j|}{|D|}\right) \quad (4)$$

Information gain rate:

$$Split(P,X) = \frac{Gain(P,X)}{Split(P,X)} \quad (5)$$

(iii) CART (Classification and Regression Tree): Using the gini index as the splitting rule.

$$gini(T) = 1 - \Sigma p_j^2 = 1 - \Sigma \left(\frac{n_j}{S}\right)^2 \quad (6)$$

$$gini_{split}(T) = \frac{S_1}{S_1+S_2} gini(T_1) + \frac{S_2}{S_1+S_2} gini(T_2) \quad (7)$$

The feature with the smallest gini index is the most decisive and powerful feature to split the feature space.

How to Build a Random Forest

For each tree inside the forest, the building steps should be as following:

a) Random and put-back selection of samples to form a random training set (bootstrap sampling). The size of the new training set should be same as the original one. If so, nearly 2/3 of the training samples will be selected for each loop.

b) If the feature dimension M is too large, randomly select m (m<<M) features to form a new training feature list

c) Use the new sample list and feature list to train a decision tree.

3.2 Algorithm 2: Mixture of Experts

Mixture of experts is a discriminative mixture model, and so it can be solved using the expectation maximization (EM) algorithm. By introducing a hidden variable Z, and taking the expectation of the log likelihood, the problem can be solved in two separate optimization steps. The likelihood can be written as the following equation

$$p(y|x,\theta) = \prod_{i=1}^{N} \sum_k p(z_k|x_i,\theta)\, p(y|x_i,z_k,\theta) \quad (8)$$

Taking the expectation of the log of the likelihood allows us to divide the likelihood into two separate models since the log can be distributed to both. In addition, they are weighted by some terms determined by the weights from the previous iteration, which are used to calculate weights for the current iteration. The E step consists of finding these weights. The following equation is the expected log likelihood and shows this simplification

Expected log Likelihood:

$$\sum_i \sum_k p(z_i = k|x_i,\theta) \log[p(z_k|x_i,\theta) p(y_i|x_i,z_k,\theta)] \quad (9)$$

In this case, $p(z_i = k|x_i,\theta)$ is found in the E step. In the M step we maximize, the above equation. The models we chose for the face detection application were logistic regression for both the gating function and the experts. This is because it is a classification problem and using a

softmax seems like a logical decision to make. The following equations represent the models

Gating Function: $p(z_k|x_i,\theta) = \frac{e^{v_k^T x_i}}{\sum_{l=1}^{K} e^{v_l^T x_i}}$ (10)

Experts: $p(y_i|x_i,z_k,\theta) = \frac{e^{(W_k^T x_i)y_i}}{\sum_{l=1}^{K} e^{(W_l^T x_i)y_i}}$ (11)

$r_{ik} = \frac{p(z_k|x_i,\theta)_{old} p(y_i|x_i,z_k,\theta)_{old}}{\sum_{l=1}^{K} p(z_l|x_i,\theta)_{old} p(y_i|x_i,z_l,\theta)_{old}}$ (12)

where $r_{ik} \triangleq p(z_i = k|x_i,\theta)$.

The two equations for the gating function and experts can be optimized through gradient descent at each iteration of the EM algorithm.

The general flow of the implementation is to randomly set the weights for the gating function and experts. Then, calculate

4. Implementation

4.1 Dataset

For the face detection, we selected the Pascal Object Class (VOC) 2012 as the dataset. There are more than 17,000 samples of varying resolution with the ground truth labels.

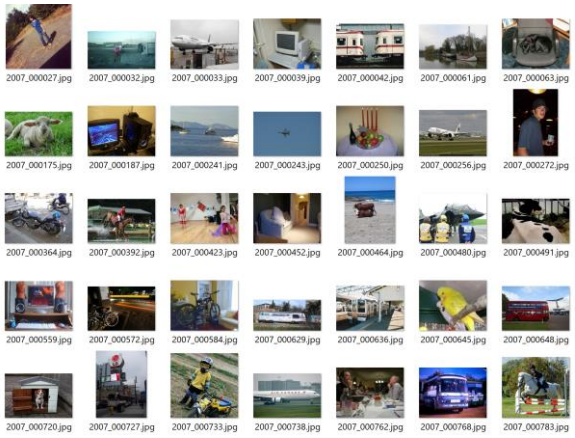


Figure 4.1 Pascal VOC 2012 dataset

4.2 Implementation of Random Forest

First, we transformed the original rgb pictures in the dataset to greyscale pictures. In addition, we shrunk all pictures to a standard size of 300*300 pixels. For these rescaled greyscale pictures, we used the edge detection (see an edge detection example in the appendix (i)) to detect whether there is an edge in every pixel and generate a 90,000-dimension feature vector for every sample in the dataset. Because this matrix was very sparse and has a very large feature dimension, we also use the Matlab built-in Principal Components Analysis (PCA) to make dimension reduction, which reduces the dimension from 90,000 to 2500.

Then we follow the steps as mentioned earlier to build a random forest. At first, we directly used the 90,000-dim original data. We tried different parameters like different feature list size, different stop criterion when building trees and different forest size. For all these results, the cross validation CCR are below 58% that are much lower than our expectation. However, for the 2,500-dim data, the CCR has a great improvement. Using the PCA, we succeeded in solving the sparse problem of the data and applied the random forest algorithm to a lower dimension data.

The time complexity of random forest should be O(nmf) while the n = size of forest, m = number of nodes for each decision tree and f = size of feature list (f>>m).

4.3 Implementation of Mixture of Experts

The implementation consists of first creating the algorithm and testing it on synthetic data, and then running it on the face detection dataset and evaluating the performance using experiments. The first part of both of these steps was defining the parameters and then executing the algorithm to predict the labels. Through trial and error, we chose parameters that took into account both runtime and correctness of results so that we may run it in a timely fashion and get as accurate results as possible with the current implementation.

Gradient Descent

The parameters that did best, which we determined through trial and error were a fixed step size of 10^-5 for the difference the log likelihood and the gradient and a max iteration size of 600 for the synthetic data and 100 for the Face detection algorithm. The gradient descent algorithm was used for both the experts and the gating function. The only difference was the number of parameters, or weights, for the experts and gating function. At each iteration, we took the gradient of the negative log likelihood and subtracted it from the negative log likelihood. This was repeated a max iteration number of times.

Expectation Maximization

For the EM algorithm, we chose 600 max iterations for the synthetic data and 100 for the face detection data. We evaluated several combinations and concluded it worked best for the synthetic dataset since the number of samples were minimal. In each iteration of the EM algorithm, we optimized weights using gradient descent and used these weights to keep a running track of r_{ik} to calculate weights for the following iteration. For debugging purposes, we kept track of the expected log likelihood to make sure it was indeed monotonically increasing.

Regression

This was the initial step towards implementing a working mixture of experts implementation. The difference between regression and classification was that

for the maximum likelihood estimate, we used a Gaussian model instead of logistic regression to model the experts. This resulted in a higher runtime since the optimization for the Gaussian model could be done in one step. This is almost the same as MMSE, except that it was a weighted sum of least squares. Furthermore, the decisions made after estimating the parameters were soft decisions, and a weighted sum of the decisions from the individual experts was taken with the weights determined by the gating function.

Classification

The classification implementation was used for both synthetic data and the face detection application. The difference between this and regression was that the model for the experts was logistic regression, and a hard decision was made based on the posterior that was calculated for the mixture model. The classification model was much slower due to calculating logistic regression optimization using gradient descent for k experts.

5. Experimental Results

5.1 Results of Random Forest

For the 90,000-dim data, we get the best cross validation test CCR = 57.75% when the number of nodes m = 150 as shown in figure 5.1.1.

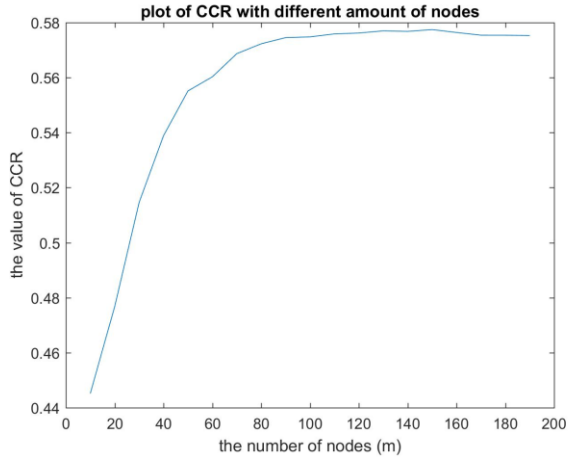


Figure 5.1.1 the test CCR of random forest for the 90,000-dim data as a function of the number of nodes

For the 90,000-dim data, we get the best cross validation test CCR = 57.86% when the number of trees >= 45 as shown in figure 5.1.2.

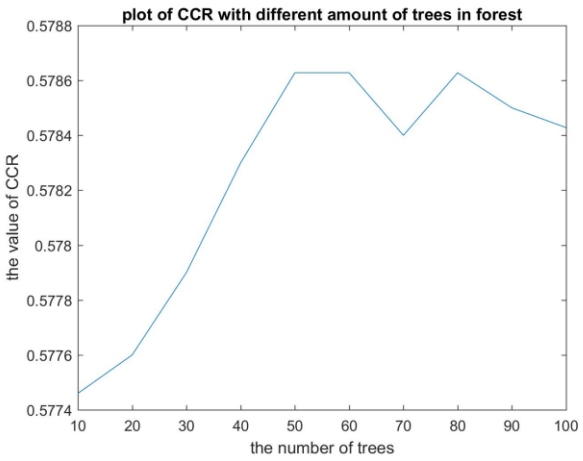


Figure 5.1.2 the test CCR of random forest for the 90,000-dim data as a function of the forest size

For the 2,500-dim data, we get the best cross validation test CCR = 82.40% when the number of trees >= 25 as shown in figure 5.1.3.

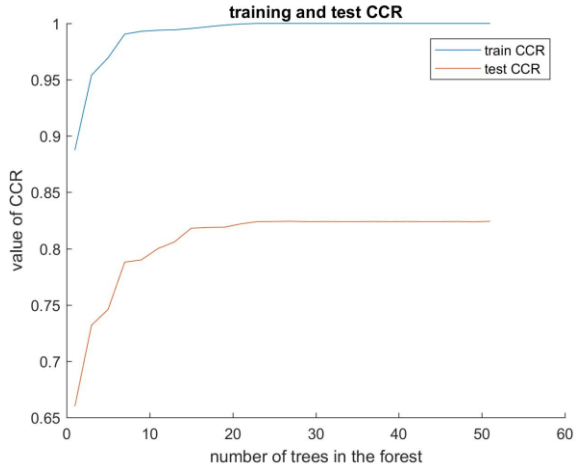


Figure 5.1.3 the training and test CCR of random forest for the 2,500-dim data as a function of the forest size

5.2 Results of Mixture of Experts

Regression – Synthetic

We generated 5000 one dimensional samples that were clearly separated in space. This allowed us to show the correctness of the algorithm without introducing complexity. As can be seen from the figure, the data is modeled fairly accurately for 5 experts with 5 clearly separated regions. The prediction is the weighted sum of all of the experts, as a result, the borders between regions are not well defined.

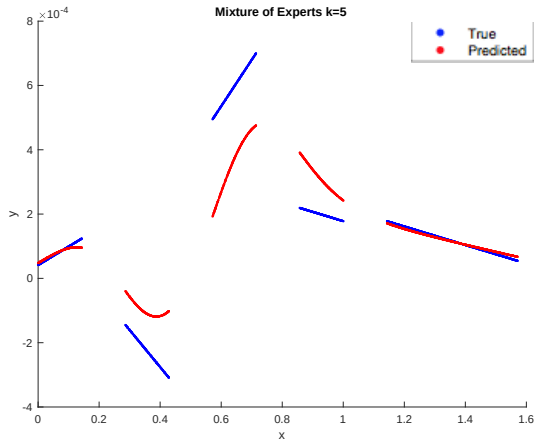


Figure 5.4.1 True and Predicted labels for MOE Regression n=5000 k=5

Classification – Synthetic

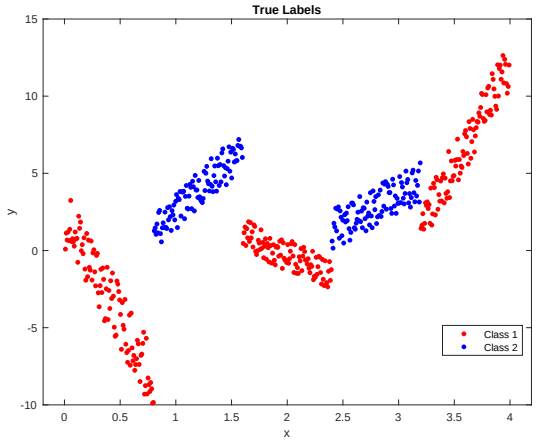


Figure 5.4.1 Synthetic dataset for classification experiments

Using the toy dataset in figure 5.4.1 with 500 samples, MOE predicted results with an increasing accuracy as the number of experts increased.

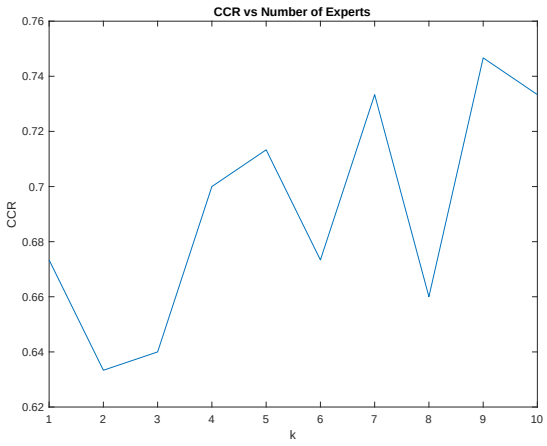


Figure 5.4.2 CCR for MOE classification n=500

Face Detection

The CCR for the face detection algorithm using 10% of the samples, n=1700, and number of experts equal to 2^k from k=5-11 resulted in a consistent CCR of 50% as shown in figure 5.4.3.

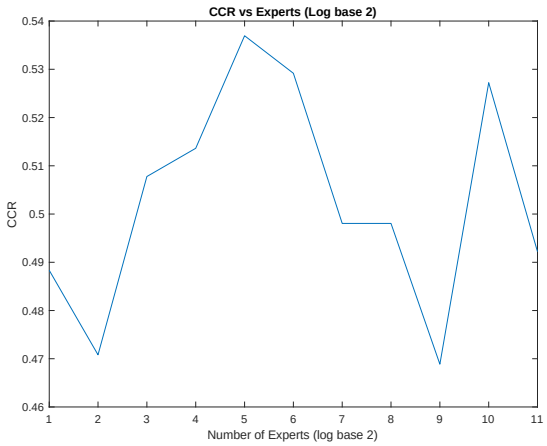


Figure 5.4.3 CCR for face detection

5.3 Results of Baseline K-NN

The highest CCR is 55.97% when the parameter k is 5133 as shown in the figure. Since there are 17125 samples in total and 9583 of them are positive samples, the K-NN seems to converge when k = half of the number of positive samples.

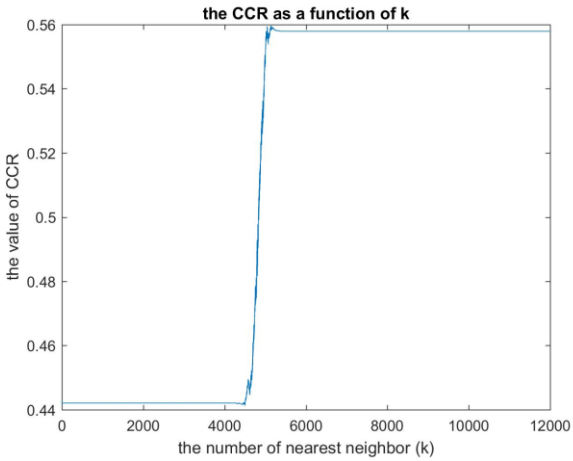


Figure 5.3.1 the CCR of K-NN classifier as a function of k

5.4 Results Comparison

Random forest classification performed significantly better than Mixture of Experts and K-NN when predicting the class labels. This is because the random forest has the attribute to fit the data with high feature dimension very well. In addition, Random Forests has a much lower runtime than MOE and K-NN.

6 Conclusion

6.1 Random Forest

- For each decision tree inside the random forest, the number of nodes of it should be some appropriate value. This is because the classification will not be precise enough if the nodes are too few and it will arise overfitting problems on training set if the nodes of tree are too much.
- As the number of trees increasing, the performance of the random forest tends to converge. This is because if the size of the forest is big enough, those randomly selected features, which are redundant and less informative, will counteract each other and make some offset in the final prediction and those which are most decisive and powerful will be strengthened and play significant roles in the final result.

6.2 Mixture of Experts

The performance of mixture of experts was highly dependent on the separation of data in space. This was influenced by two key points.

a) The number of samples in the dataset

b) Number of experts

In general, mixture of experts does well when the data is separated. This means that each expert, which focuses on a specific part of the dataset, has a large number of samples to model that sub region. In addition, the data in that sub region should be linear in the case of regression or a single class in the case of classification. As the number of experts increases, the CCR increases, up to a specific point determined by the number of samples in the training set. Any increase in experts after that point produces problems of over-fitting. In general, how well each expert can model the sub region will determine the accuracy of the model.

On the other hand, one also has to take into account the complexity of the algorithm. As, the number of experts increases, so does the runtime. In addition, since we are using two gradient descent algorithms in each EM iteration, the number of samples and experts is limited. This produces a big problem for large datasets with several classes.

Face Detection

Mixture of experts does not perform well for face detection using Pascal VOC dataset. This is due to the fact that the number of samples in the dataset is small. In addition, the size of each feature, which is the dimension size, is larger than the number of samples. Although the application is binary classification, the lack of samples and large runtime with such high dimensions limits the testing of Mixture of experts and produces results as bad as random guessing.

7 Description of Individual Effort

Lingshan Yang:

- Literature of survey of random forest
- Implementation of dimension reduction (PCA)
- Implementation of code of random forest
- Theoretical analysis of the experimental results of random forest
- Describe random forest algorithm in the final report

Wasim Khan:

- Literature of survey of mixture of expert
- Implementation of feature extraction through edge detection
- Theoretical analysis of the experimental results of mixture of experts
- Describe mixture of experts algorithm in the final report

References

[1] http://facedetection.com/

[2] R. Polikar. Ensemble based systems in Decision Making. IEEE Circuits and Systems Magazine. 6(3):21-45.2006.

[3] Leo Breiman. Random Forests. Machine Learning. 45(1):5-32, 2001.

[4] Zhihua Zhou. Ensemble Methods Foundations and Algorithms, Taylor & Francis Group, 2012.

[5] Y. Liao and V. R. Vemuri. Use of K-Nearest Neighbor Classifier for Instruction Detection. Computers & Security, 21(5):439-448, 2002.

[6] S.E. Yuksel, J. N. Wilson. and P. D. Gader. Twenty Years of Mixture of Experts. 2012.

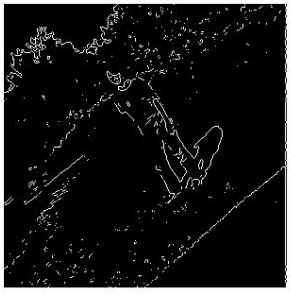
[7] https://github.com/karpathy/Random-Forest-Matlab (only for 2D data generation)

Appendix

1. An example of edge detection

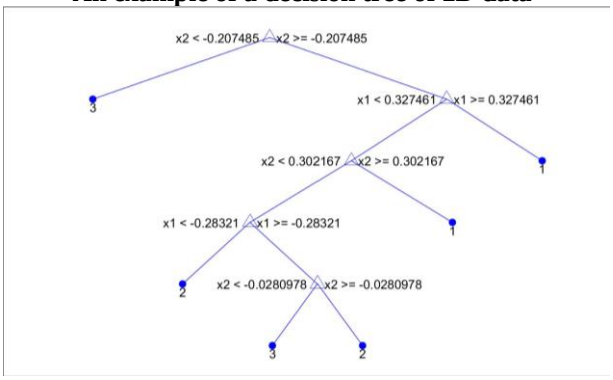
Original rgb picture

Rescaled greyscale pictures

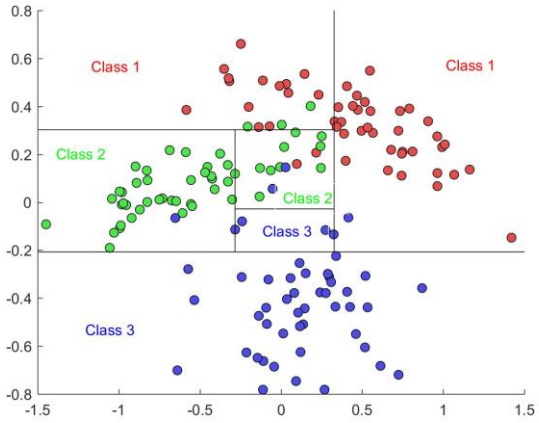


Rescaled edge pictures

2. An example of a decision tree of 2D data



Plot of a decision tree



The decision regions in 2D

3. Dataset

PASCAL VOC 2012 can be download at this link: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

4. Source code

You can see our source code at Github: https://github.com/wasimk1995/ec503_final_project