## Introduction / Statement of Contribution

This is the lab for block2 in the Machine Learning. In this lab, contains the following tasks:1. ENSEMBLE METHODS. 2.MIXTURE MODELS. 3. Theory.
Han Xia and Liuxi Mei was responsible for the code writing of assignment 1 and assignment 2 respectively, and Xiaochen Liu was mainly responsible for the answer of assignment 3. For each assignment, everyone participated in the problem analysis to ensure that everyone understood the final result.

## Assignment 1:ENSEMBLE METHODS

**#Build test data set**

First of all, we fixed the data of the test set to avoid the result error caused by the difference of the test set in the subsequent calculation process.

```{r test dataset}
set.seed(1234)
x1 <- runif(1000)
x2 <- runif(1000)
testdata <- cbind(x1, x2)
colnames(testdata) <- c("x1", "x2")
y1 <- as.numeric(x1 < x2)
testlabels <- as.factor(y1)
y2 <- as.numeric(x1 < 0.5)
testlabels2 <- as.factor(y2)
y3 <- as.numeric((x1<0.5 & x2<0.5) | (x1>0.5 & x2>0.5))
testlabels3 <- as.factor(y3)
```

**#Build the train dataset for 1000 times with the size of 100**

We created 1000 training data sets of size 100 and fixed them, learn a random forest from each data set.

```{r train dataset}
set.seed(123)
train_data_list <- lapply(1:1000, function(i) {
  x3 <- runif(100)
  x4 <- runif(100)
  trdata <- cbind(x3, x4)
  colnames(trdata) <- c("x1", "x2")
  list(trdata = trdata)
})
```

**# Compute the misclassification error**

First, the list is initialized to store data. Labels are created for the training set, depending on the

condition. Then, we computed the misclassification error in the same test dataset of size 1000 in condition 1. Report results for when the random forest has 1, 10 and 100 trees.

```{r include=FALSE}
error_rate_1<- list(
  number1 = rep(0,1000),
  number2 = rep(0,1000),
  number3 = rep(0,1000)
)
mean_error_1<- c()
var_error_1<- c()
for (i in 1:1000) {
  trdata <- train_data_list[[i]]$trdata
  y <- as.numeric(trdata[, 1] < trdata[, 2])
  trlabels <- as.factor(y)

  #Build the models
  rf_model1 <- randomForest(trdata, trlabels, ntree = 1, nodesize = 25, keep.forest = TRUE)
  rf_model2 <- randomForest(trdata, trlabels, ntree = 10, nodesize = 25, keep.forest = TRUE)
  rf_model3 <- randomForest(trdata, trlabels, ntree = 100, nodesize = 25, keep.forest = TRUE)

  #predictions and error rates
  predictions1<- predict(rf_model1,testdata)
  error_rate_1$number1[i] <- mean(predictions1 != testlabels)
  predictions2<- predict(rf_model2,testdata)
  error_rate_1$number2[i] <- mean(predictions2 != testlabels)
  predictions3<- predict(rf_model3,testdata)
  error_rate_1$number3[i] <- mean(predictions3 != testlabels)
}
```

**#Compute the mean and variance of error rates**

Evaluate the model's overall predictive performance and sensitivity to input data.

```{r include=FALSE}
mean_error_1[1]<- mean(error_rate_1$number1)
mean_error_1[2]<- mean(error_rate_1$number2)
mean_error_1[3]<- mean(error_rate_1$number3)

var_error_1[1] <- var(error_rate_1$number1)
var_error_1[2] <- var(error_rate_1$number2)
var_error_1[3] <- var(error_rate_1$number3)
```

Then, we can change the condition,and the calculation is the same as before.

**#Summerize the results**

Repeat for the conditions (x1\<0.5) and the conditions ((x1\<0.5 & x2\<0.5) \| (x1\>0.5 & x2\>0.5)) for 1,10,100 trees, the results are summarized in the table below.

Mean_error:

| | ntree = 1 | ntree = 10 | ntree = 100 |
|---|---|---|---|
| Condition 1 | 0.212314 | 0.134672 | 0.110031 |
| Condition 2 | 0.094541 | 0.015954 | 0.006759 |
| Condition 3 | 0.244643 | 0.119890 | 0.076860 |

Var_error:

| | ntree = 1 | ntree = 10 | ntree = 100 |
|---|---|---|---|
| Condition 1 | 0.0034664178 | 0.0009542587 | 0.0008674575 |
| Condition 2 | 1.895466e-02 | 7.167787e-04 | 6.893385e-05 |
| Condition 3 | 0.013417381 | 0.002800795 | 0.001326289 |

**#### Questions:**

**#### What happens with the mean error rate when the number of trees in the random forest grows? Why?**

As the number of trees in the random forest increases, the mean error rate decreases. This happens because: a random forest combines multiple decision trees. As more trees are added, the predictions are averaged, reducing the variance of the model. With more trees, the random forest effectively samples a larger variety of subsets from the training data. This increases the likelihood that the model captures the true underlying patterns.

**#### The third data set represents a slightly more complicated classification problem than the first one. Still, you should get better performance for it when using sufficient trees in the random forest. Explain why you get better performance.**

Despite being more complex, the random forest performs better on the third data set with sufficient trees because: the third data set is more complex and may contain multiple non-linear boundaries or classes that are more difficult to distinguish. In this case, a single tree may not be enough to capture all the complex patterns, and increasing the number of trees can provide more decision rules, thereby improving the performance of the model. More trees can improve the generalization ability of the model in an integrated way.

## Assignment 2:MIXTURE MODELS

In this assignment,we implemented the EM algorithm for Bernoulli mixture model.And for different values of M, compared their results. The E and M steps are calculated as follows:

**Learn** the GMM

**Data:** Unlabeled training data $\mathcal{T} = \{\mathbf{x}_i\}_{i=1}^n$, number of clusters $M$.
**Result:** Gaussian mixture model

1 Initialize $\widehat{\boldsymbol{\theta}} = \{\widehat{\pi}_m, \widehat{\boldsymbol{\mu}}_m, \widehat{\boldsymbol{\Sigma}}_m\}_{m=1}^M$
2 **repeat**
3 For each $\mathbf{x}_i$ in $\{\mathbf{x}_i\}_{i=1}^n$, compute the prediction $p(y \mid \mathbf{x}_i, \widehat{\boldsymbol{\theta}})$ according to (10.5) using the current parameter estimates $\widehat{\boldsymbol{\theta}}$.
4 Update the parameter estimates $\widehat{\boldsymbol{\theta}} \leftarrow \{\widehat{\pi}_m, \widehat{\boldsymbol{\mu}}_m, \widehat{\boldsymbol{\Sigma}}_m\}_{m=1}^M$ according to (10.16)
5 **until** *convergence*

**Predict** as QDA, Method 10.1

$$p(y = m \mid \mathbf{x}_\star) = \frac{\widehat{\pi}_m \mathcal{N}\left(\mathbf{x}_\star \mid \widehat{\boldsymbol{\mu}}_m, \widehat{\boldsymbol{\Sigma}}_m\right)}{\sum_{j=1}^M \widehat{\pi}_j \mathcal{N}\left(\mathbf{x}_\star \mid \widehat{\boldsymbol{\mu}}_j, \widehat{\boldsymbol{\Sigma}}_j\right)}. \tag{10.5}$$

$$\widehat{\pi}_m = \frac{1}{n} \sum_{i=1}^n w_i(m), \tag{10.16a}$$

$$\widehat{\boldsymbol{\mu}}_m = \frac{1}{\sum_{i=1}^n w_i(m)} \sum_{i=1}^n w_i(m) \mathbf{x}_i, \tag{10.16b}$$

$$\widehat{\boldsymbol{\Sigma}}_m = \frac{1}{\sum_{i=1}^n w_i(m)} \sum_{i=1}^n w_i(m)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_m)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_m)^\mathsf{T}. \tag{10.16c}$$

where $w_i(m) = p(y_i = m \mid \mathbf{x}_i, \widehat{\boldsymbol{\theta}})$

**Method 10.3:** Unsupervised learning of the GMM

- The algorithm above is also known as EM algorithm: Step 3 is called Expectation, and step 4 is called Maximization.
- It converges to a **local maximum** of the likelihood function of the training data.

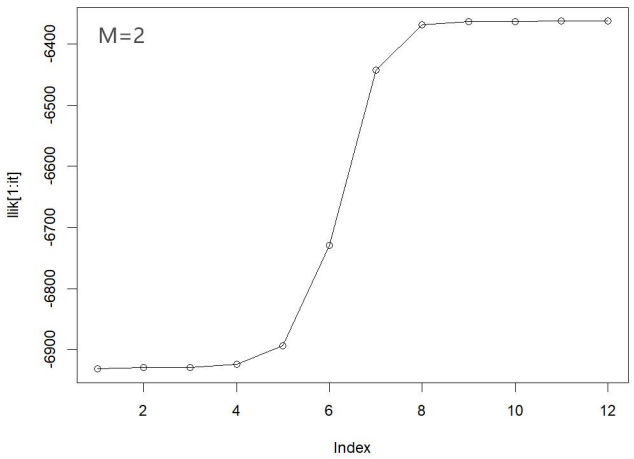
The function N () is calculated from the Bernoulli distribution

And the formula for calculating the log-likelihood value is as follows:

$$\log L(\Theta \mid X) = \sum_{i=1}^{N} \log \left( \sum_{k=1}^{M} \pi_k \cdot p(x_i \mid \mu_k, \Sigma_k) \right)$$

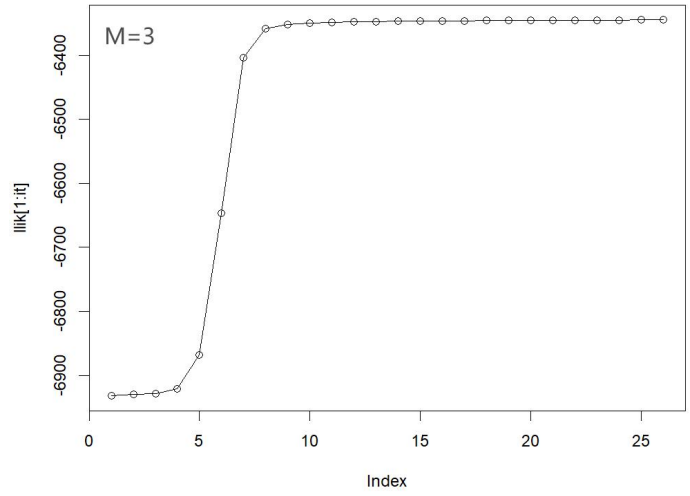
We can get results for different M values. The results are summarized as follows.

For M=2:



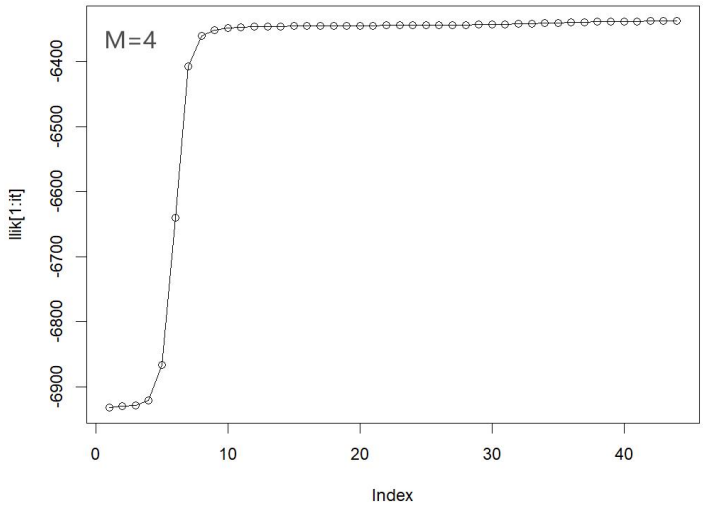

```
> pi
[1] 0.497125 0.502875
> mu
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]      [,10]
[1,] 0.4775488 0.4113939 0.5892308 0.3472420 0.6583712 0.2686589 0.7089490 0.2118629 0.7957549 0.08905747
[2,] 0.5062860 0.5597531 0.4177551 0.6728856 0.3354854 0.7247188 0.2616231 0.8007511 0.1678555 0.90027808
```

For M=3:




```
> pi
[1] 0.3416794 0.2690298 0.3892909
> mu
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]      [,10]
[1,] 0.4727544 0.3869396 0.6302224 0.3156325 0.6875038 0.2030173 0.7832090 0.1435650 0.8827796 0.03422816
[2,] 0.4939501 0.4757687 0.4584644 0.4711358 0.5413928 0.4976325 0.4569664 0.4869015 0.4909904 0.37087402
[3,] 0.5075441 0.5800156 0.4221148 0.7100227 0.2965478 0.7571593 0.2400675 0.8424441 0.1188864 0.99033611
```

For M=4:




```
> pi
[1] 0.1547196 0.1418652 0.3514089 0.3520062
> mu
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]       [,9]      [,10]
[1,] 0.4426228 0.5955990 0.5973038 0.4611075 0.4148259 0.3224465 0.4616759 0.5068223 0.57827821 0.52366273
[2,] 0.5347882 0.3763616 0.3116137 0.5256451 0.6254569 0.6980795 0.4139865 0.5327794 0.34159869 0.41722943
[3,] 0.5103748 0.5869840 0.4219499 0.7218615 0.2825337 0.7763136 0.2299954 0.8591562 0.09774851 0.99998228
[4,] 0.4781150 0.3812010 0.6195949 0.3165236 0.6926095 0.2166850 0.7756026 0.1479707 0.87418437 0.01530099
```

When M is equal to 2, the coefficient (pi) approaches 0.5, meaning that the two clusters contribute nearly equally to the data. Although there are significant differences in the conditional distribution (mu) on each dimension,there is a big deviation from the true value, resulting in underfitting. When M is equal to 3, the convergence rate is slightly slower than when M is equal to 2, but the mu is more dispersed and closer to the true value, which can accurately capture the true structure of the data without being overly complicated. When M is equal to 4, the log-likelihood converges at the 44th iteration, and the convergence speed is the slowest, which

indicates that the complexity of the model is significantly increased, and the weight of some clusters (such as pi1 and pi2) is very small, which indicates that the data may be unnecessarily split and overfitting may occur.