

Introduction to Machine Learning

Lab 2 Block 2

Rasmus Holm

2016-12-07

Contents

Assignment 1a	2
Assignment 1b	3
Assignment 2a	4
1	4
2	4
3	4
Assignment 2b	5
Assignment 3a	9
1	9
2	9
Assignment 4a	10
Appendix	12
Code for Assignment 1b	12
Code for Assignment 2a	14
Code for Assignment 2b	15
Code for Assignment 3a	19
Code for Assignment 4a	19

Assignment 1a

Assumptions:

$$\begin{aligned} \mathbb{E} [\epsilon^b(x)] &= 0, \\ \forall_{i,j}, i \neq j : \mathbb{E} [\epsilon^i(x)\epsilon^j(x)] &= 0 \end{aligned}$$

Prove:

$$\mathbb{E}_X [(f_{bag}(x) - h(x))^2] = \frac{1}{B} \left[\frac{1}{B} \mathbb{E}_X [(f^b(x) - h(x))^2] \right]$$

We know:

$$f^b(x) = h(x) + \epsilon^b(x)$$

Proof:

$$\begin{aligned} \mathbb{E}_X [(f_{bag}(x) - h(x))^2] &= \\ \mathbb{E}_X \left[\left(\frac{1}{B} \sum_b f^b(x) - h(x) \right)^2 \right] &= \\ \mathbb{E}_X \left[\left(\frac{1}{B} \sum_b \epsilon^b(x) \right)^2 \right] &= \\ \frac{1}{B^2} \mathbb{E}_X [(\epsilon^1(x))^2 + 2\epsilon^1(x)\epsilon^2(x) + \dots + 2\epsilon^{b-1}(x)\epsilon^b(x) + (\epsilon^b(x))^2] &= \\ \frac{1}{B^2} (\mathbb{E}_X [(\epsilon^1(x))^2] + 2\mathbb{E}_X [\epsilon^1(x)\epsilon^2(x)] + \dots + 2\mathbb{E}_X [\epsilon^{b-1}(x)\epsilon^b(x)] + \mathbb{E}_X [(\epsilon^b(x))^2]) &= \\ \frac{1}{B^2} \sum_b \mathbb{E}_X [(\epsilon^b(x))^2] &= \\ \frac{1}{B^2} \sum_b \mathbb{E}_X [(f^b(x) - h(x))^2] &= \\ \frac{1}{B} \left[\frac{1}{B} \mathbb{E}_X [(f^b(x) - h(x))^2] \right] \end{aligned}$$

Assignment 1b

Assignment 2a

1

```
#> [1] 37.10301
```

2

```
#> [1] 40.19377
```

3

Assignment 2b

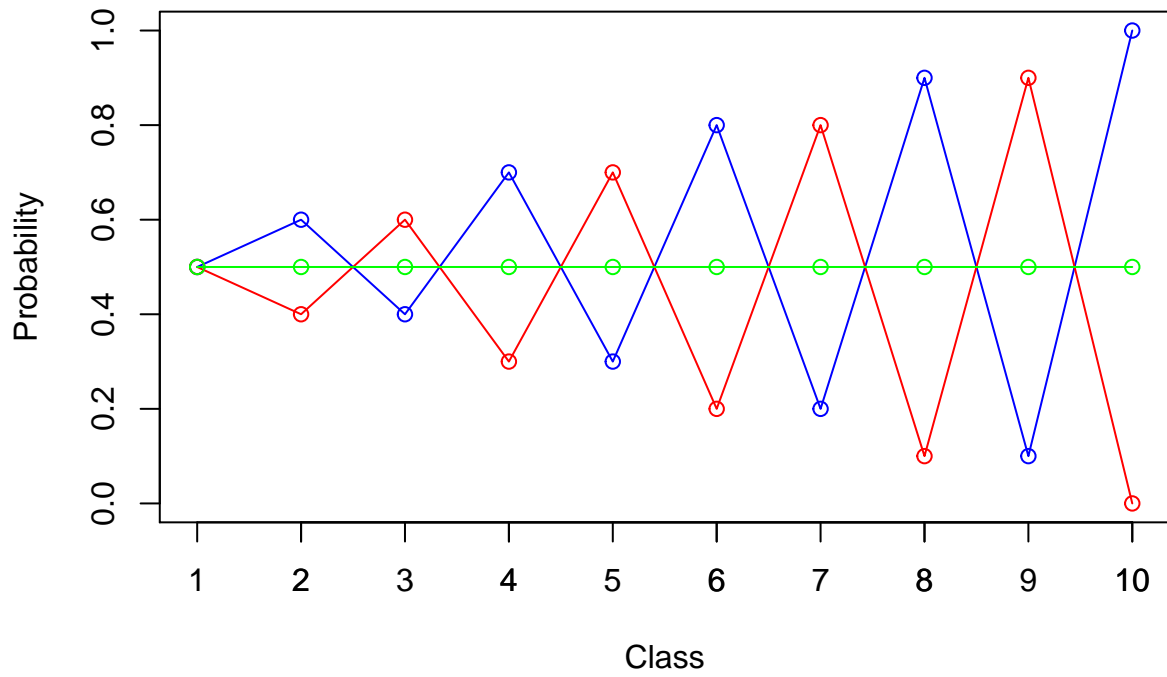


Figure 1: The true probabilities of the multinomial distributions.

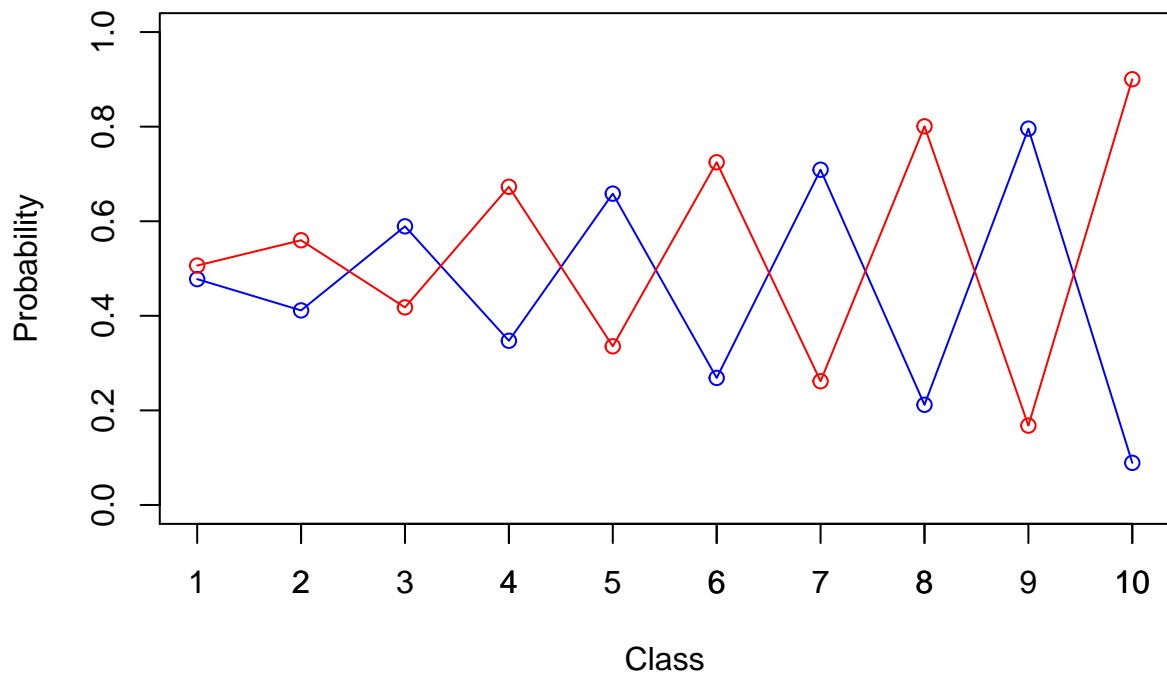


Figure 2: The estimated probabilities of the multinomial distributions.

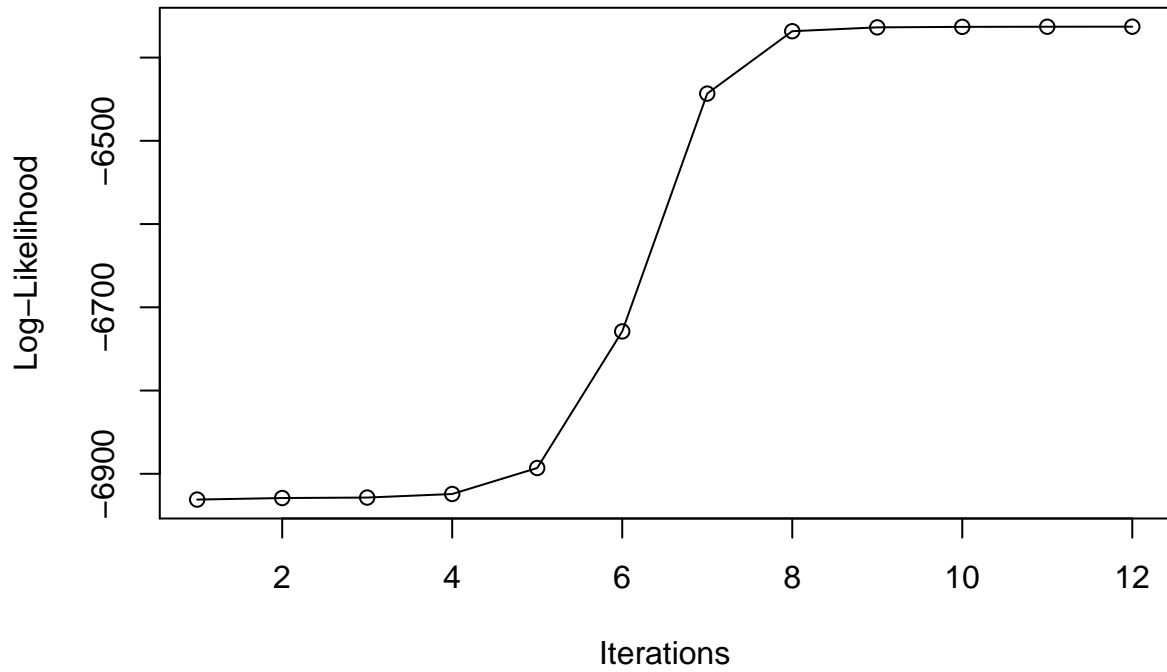


Figure 3: The log-likelihood versus the number of iterations.

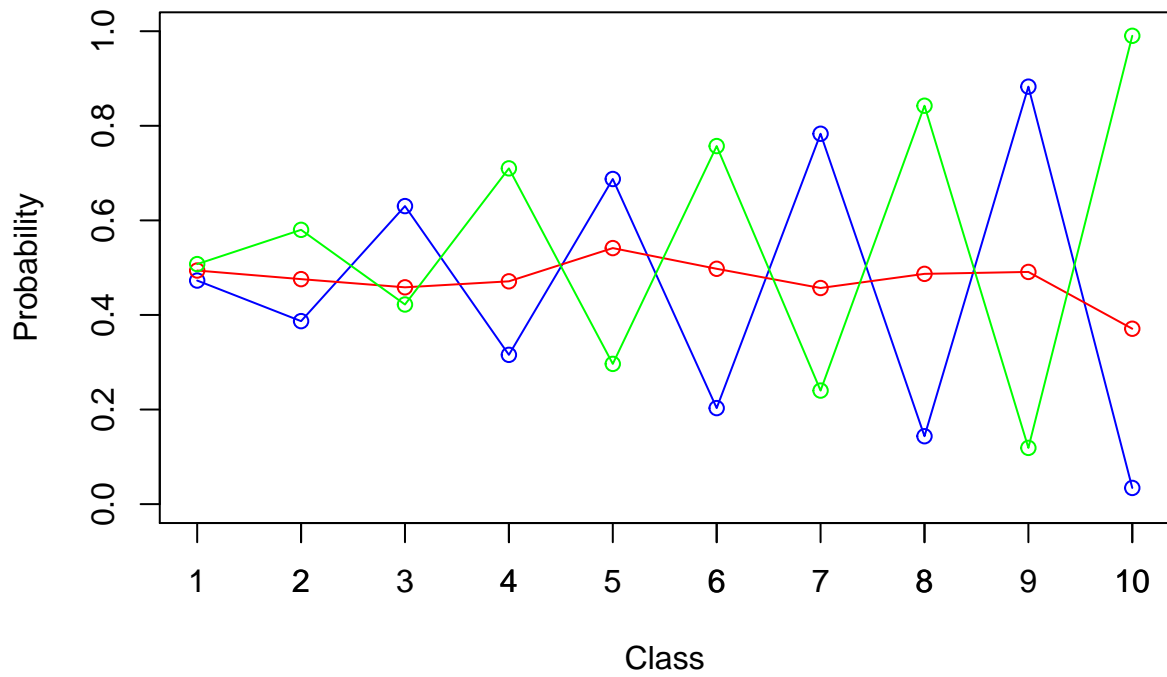


Figure 4: The estimated probabilities of the multinomial distributions.

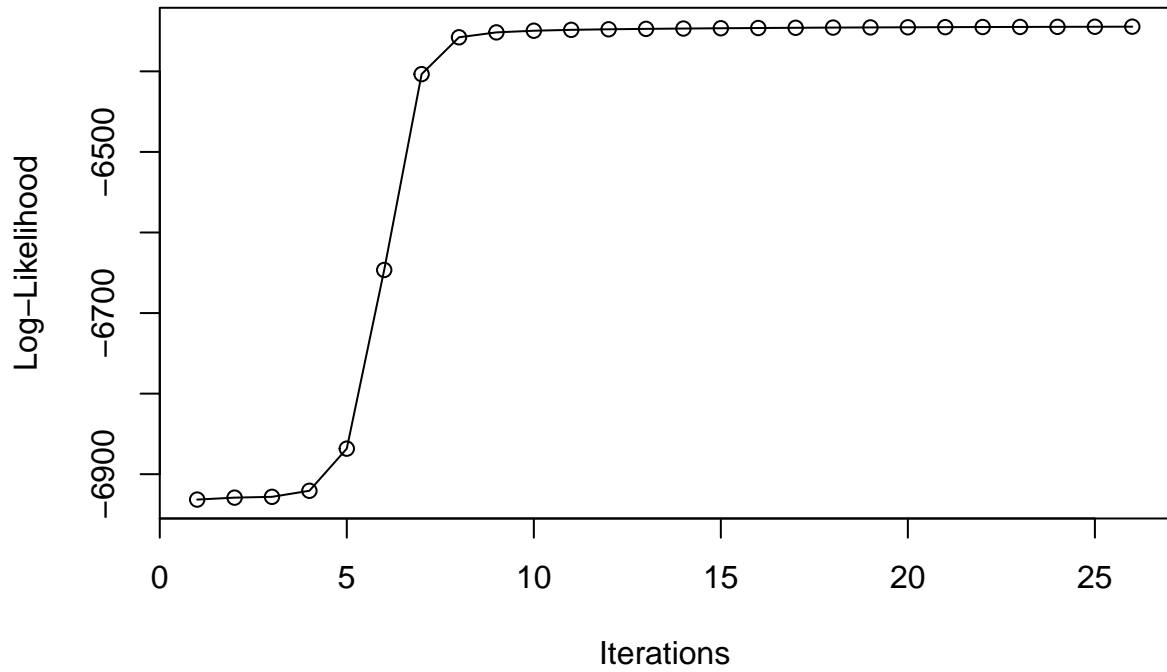


Figure 5: The log-likelihood versus the number of iterations.

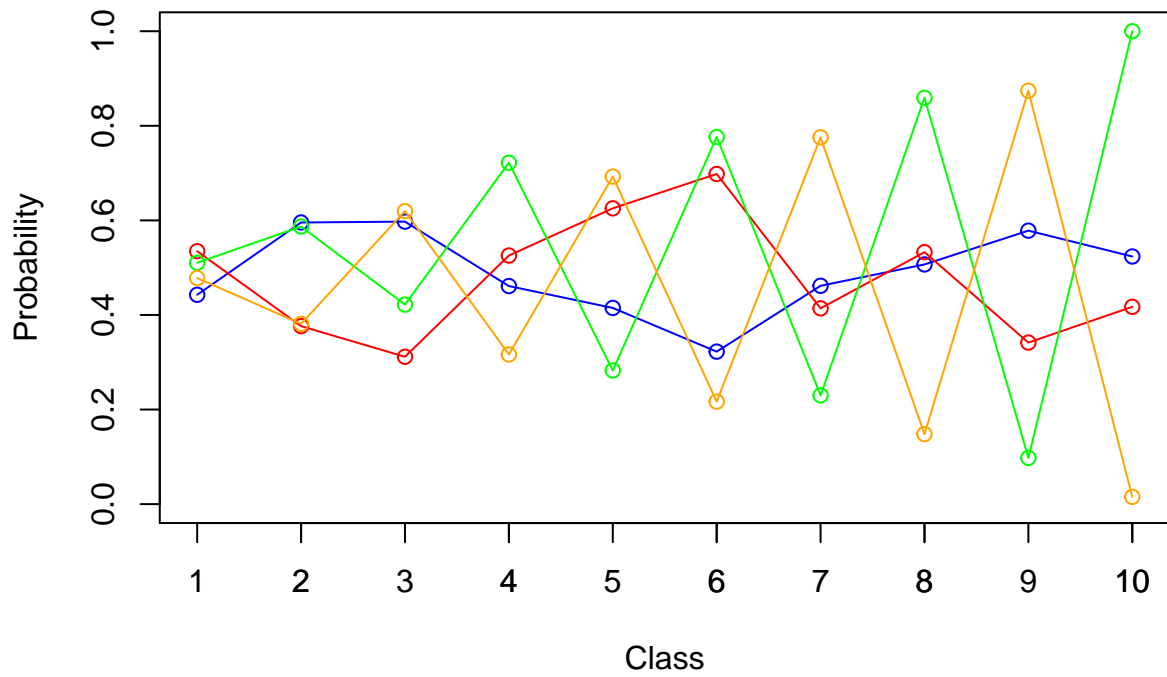


Figure 6: The estimated probabilities of the multinomial distributions.

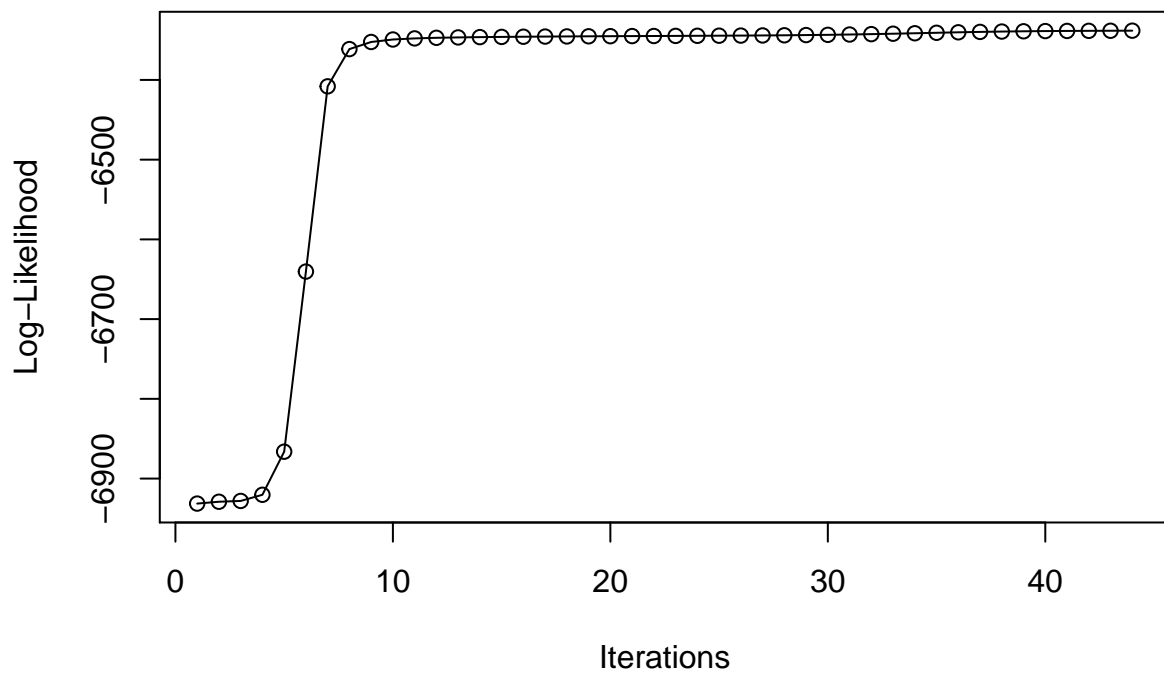
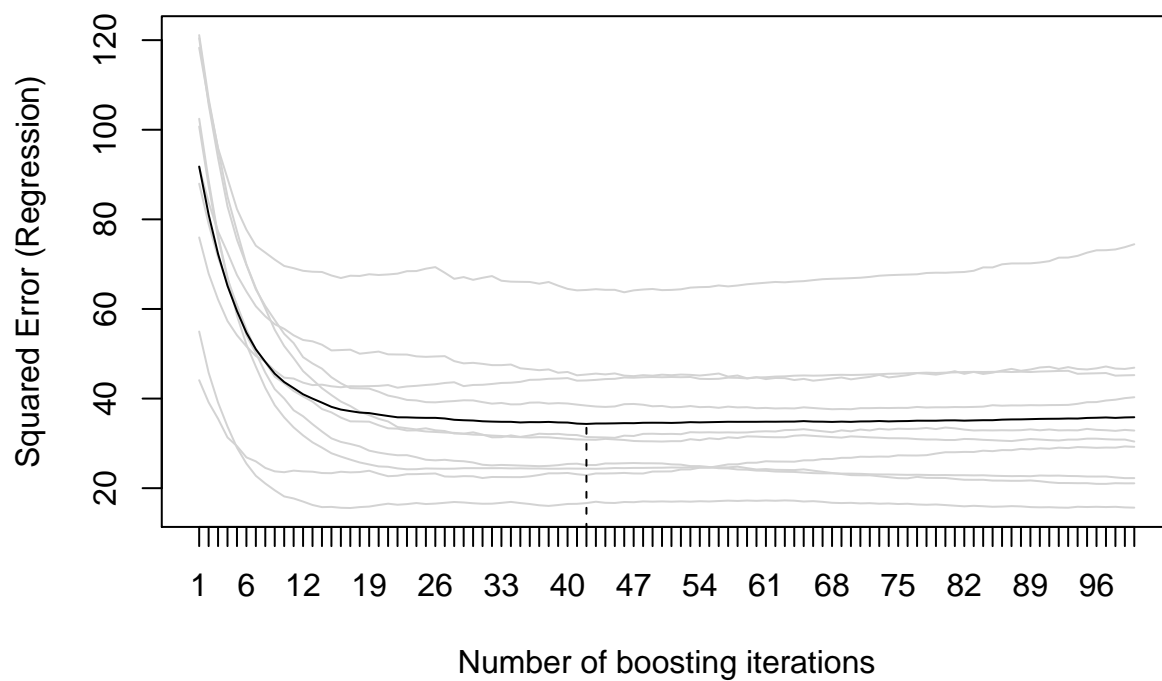


Figure 7: The log-likelihood versus the number of iterations.

Assignment 3a

1

10-fold kfold

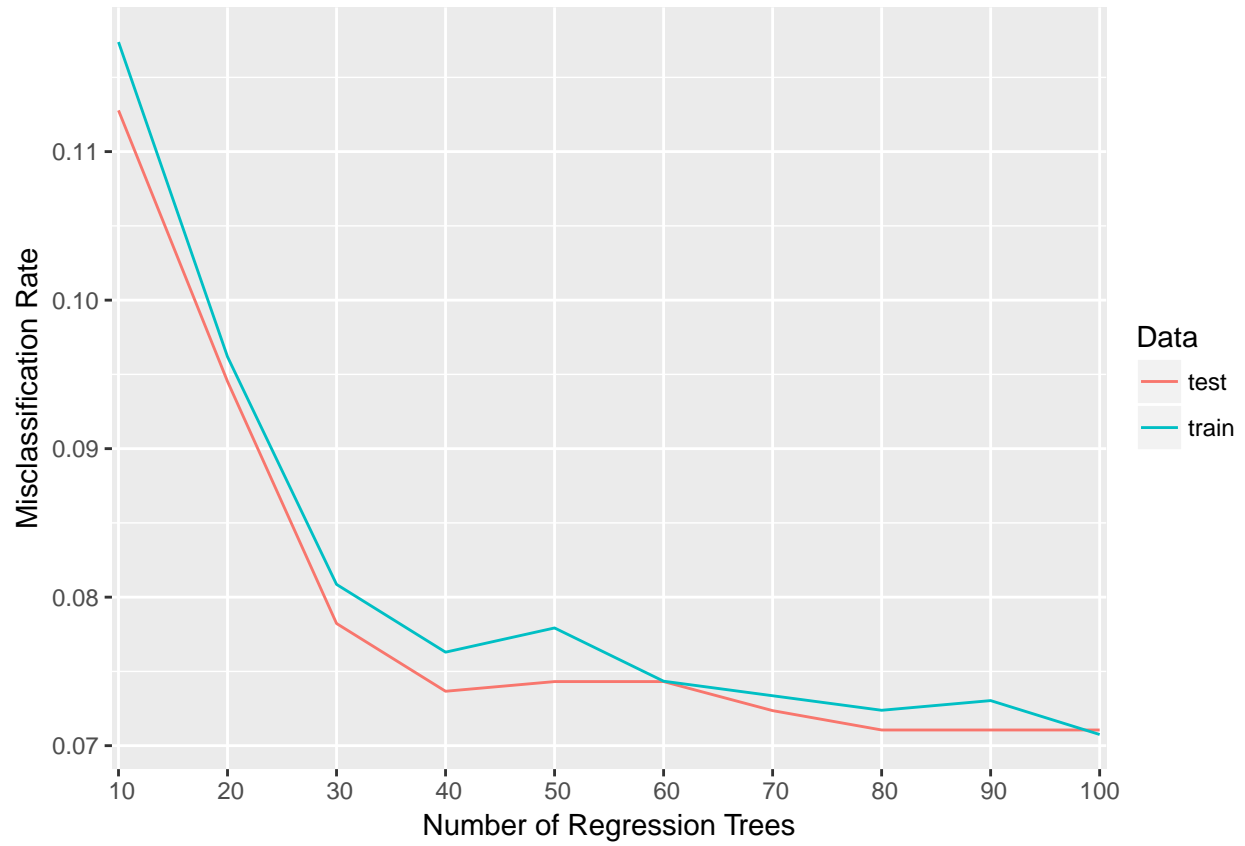


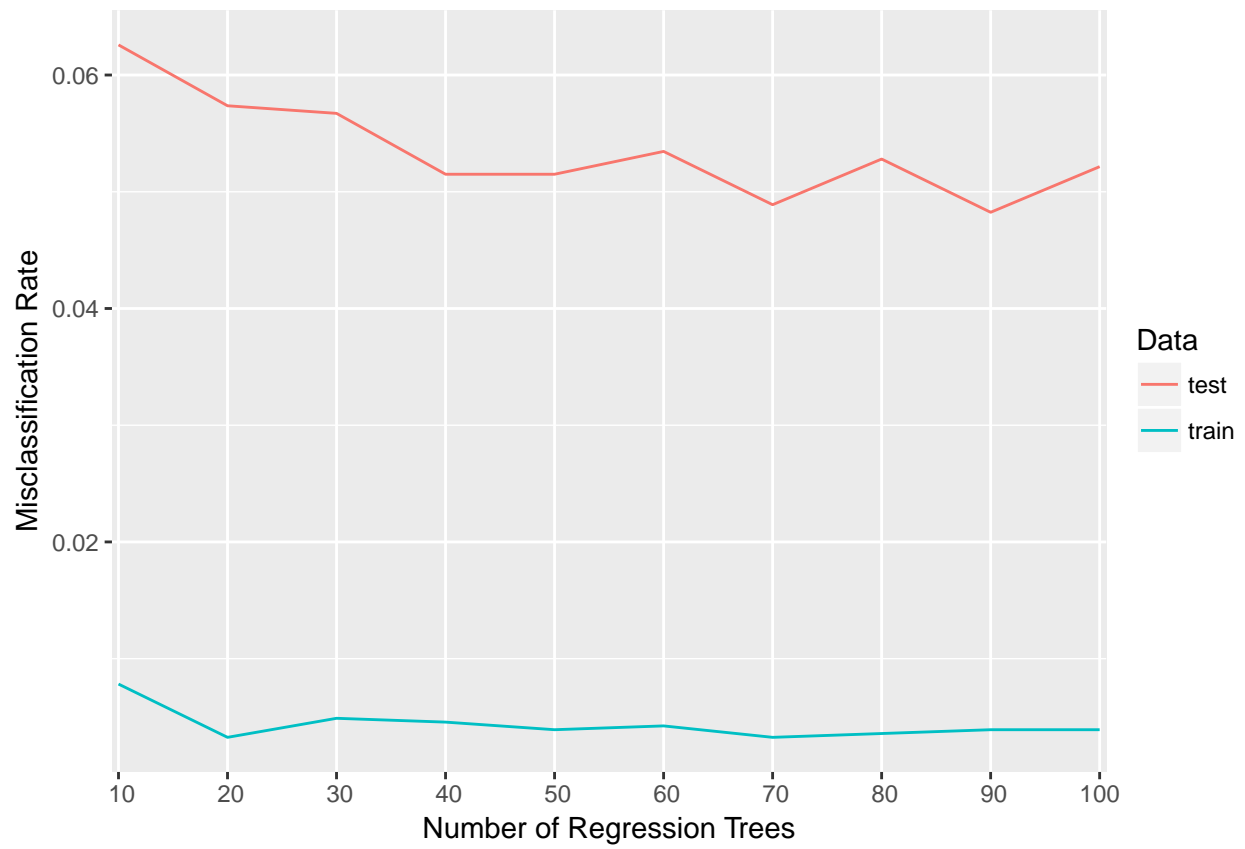
2

```
#> [1] 988.8233  
#> [1] 1433.941
```

Assignment 4a

```
#> [1] 0.11277705 0.09452412 0.07822686 0.07366362 0.07431551 0.07431551
#> [7] 0.07235984 0.07105606 0.07105606 0.07105606
#> [1] 0.11737855 0.09618520 0.08086078 0.07629605 0.07792631 0.07433975
#> [7] 0.07336159 0.07238344 0.07303554 0.07075318
```





Appendix

Code for Assignment 1b

```
x_given_mu <- function(x, mu) {
  x_mu <- matrix(1, nrow=nrow(x), ncol=nrow(mu))

  for (n in 1:N) {
    for (k in 1:K) {
      for (i in 1:D) {
        prob <- mu[k, i]^x[n, i] * (1 - mu[k, i])^(1 - x[n, i])
        x_mu[n, k] <- x_mu[n, k] * prob
      }
    }
  }

  x_mu
}

expectation.step <- function(x, x_given_mu, pi) {
  z <- matrix(nrow=nrow(x), ncol=length(pi))

  for (n in 1:N) {
    denominator <- sum(pi * x_given_mu[n,])

    for (k in 1:K) {
      nominator <- pi[k] * x_given_mu[n, k]

      z[n, k] <- nominator / denominator
    }
  }

  z
}

loglikelihood <- function(x, x_given_mu, pi) {
  llik <- 0
  for (n in 1:N) {
    inner_summation <- 0
    for (k in 1:K) {
      inner_summation <- inner_summation + pi[k] * x_given_mu[n, k]
    }
    llik <- llik + log(inner_summation)
  }

  llik
}

maximization.step <- function(x, z) {
  pi <- vector(length=ncol(z))
  mu <- matrix(nrow=ncol(z), ncol=ncol(x))
```

```

for (k in 1:K) {
  pi[k] <- sum(z[, k]) / nrow(x)
}

for (k in 1:K) {
  denominator <- sum(z[, k])
  for (i in 1:D) {
    nominator <- sum(x[, i] * z[, k])
    mu[k, i] <- nominator / denominator
  }
}

list(pi=pi, mu=mu)
}

EM <- function(N, D, K, max_it, min_change, true_pi, true_mu) {

  ## Producing the training data
  x <- matrix(nrow=N, ncol=D)

  for(n in 1:N) {
    k <- sample(1:K, 1, prob=true_pi)
    for(d in 1:D) {
      x[n, d] <- rbinom(1, 1, true_mu[k, d])
    }
  }

  z <- matrix(nrow=N, ncol=K) # fractional component assignments
  pi <- vector(length=K) # mixing coefficients
  mu <- matrix(nrow=K, ncol=D) # conditional distributions
  llik <- vector(length=max_it) # log likelihood of the EM iterations

  ## Random initialization of the paramters
  pi <- runif(K, 0.49, 0.51)
  pi <- pi / sum(pi)
  for(k in 1:K) {
    mu[k,] <- runif(D, 0.49, 0.51)
  }

  for(it in 1:max_it) {
    x_mu <- x_given_mu(x, mu)

    ## E-step: Computation of the fractional component assignments
    z <- expectation.step(x, x_mu, pi)

    ## Log likelihood computation.
    llik[it] <- loglikelihood(x, x_mu, pi)

    ## Stop if the log likelihood has not changed significantly
    if (it > 1 && abs(llik[it] - llik[it-1]) < min_change) break

    ## M-step: ML parameter estimation from the data and fractional component assignments
    result <- maximization.step(x, z)
  }
}

```

```

    pi <- result$pi
    mu <- result$mu
  }

  list(pi=pi, mu=mu, llik=llik, it=it)
}

max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations

N <- 1000 # number of training points
D <- 10 # number of dimensions
K <- 3 # number of guessed components

## true mixing coefficients
true_pi <- vector(length=3)
true_pi <- c(1/3, 1/3, 1/3)

## true conditional distributions
true_mu <- matrix(nrow=3, ncol=D)
true_mu[1,] <- c(0.5, 0.6, 0.4, 0.7, 0.3, 0.8, 0.2, 0.9, 0.1, 1)
true_mu[2,] <- c(0.5, 0.4, 0.6, 0.3, 0.7, 0.2, 0.8, 0.1, 0.9, 0)
true_mu[3,] <- c(0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1),
     xlab="Class", ylab="Probability")
axis(side=1, at=c(1:D))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")

```

Code for Assignment 2a

```

library(tree)

data <- read.csv2("../data/bodyfatregression.csv")
names(data) <- c("Waist", "Weight", "Bodyfat")
set.seed(1234567890)
train_idx <- sample(nrow(data), floor(nrow(data) * (2 / 3)))
train <- data[train_idx,]
test <- data[-train_idx,]

set.seed(1234567890)

tree_count <- 100
test_errors <- rep(0, tree_count)

for (i in 1:tree_count) {
  newdata <- train[sample(nrow(train), replace=TRUE),]
  fit <- tree(Bodyfat ~ ., data=newdata, split="deviance")

  test_error <- mean((predict(fit, test) - test$Bodyfat)^2)
  test_errors[i] <- test_error
}

```

```

mean(test_errors)
tree_count <- 100
fold_count <- 3
test_errors <- matrix(0, nrow=tree_count, ncol=fold_count)

set.seed(1234567890)

folds <- suppressWarnings(split(1:nrow(data), f=1:fold_count))

for (j in 1:fold_count) {
  train <- data[-folds[[j]],]
  test <- data[folds[[j]],]

  for (i in 1:tree_count) {
    newdata <- train[sample(nrow(train), replace=TRUE),]
    fit <- tree(Bodyfat ~ ., data=newdata, split="deviance")

    test_error <- mean((predict(fit, test) - test$Bodyfat)^2)
    test_errors[i, j] <- test_error
  }
}

mean(test_errors)
bagging.regtrees <- function(formula, data, newdata, b) {
  predictions <- matrix(0, nrow=nrow(newdata), ncol=b)
  trees <- list()

  for (i in 1:b) {
    bootstrap_sample <- data[sample(nrow(data), replace=TRUE),]
    fit <- tree(formula, data=bootstrap_sample, split="deviance")
    trees[[i]] <- fit
    predictions[, i] <- predict(fit, newdata)
  }

  list(trees=trees, predictions=rowMeans(predictions))
}

cv.regtrees <- function(formula, data, newdata, b, k) {
}

```

Code for Assignment 2b

```

x_given_mu <- function(x, mu) {
  x_mu <- matrix(1, nrow=nrow(x), ncol=nrow(mu))

  for (n in 1:N) {
    for (k in 1:K) {
      for (i in 1:D) {
        prob <- mu[k, i]^x[n, i] * (1 - mu[k, i])^(1 - x[n, i])
        x_mu[n, k] <- x_mu[n, k] * prob
      }
    }
  }
}

```

```

    }
  }

  x_mu
}

expectation.step <- function(x, x_given_mu, pi) {
  z <- matrix(nrow=nrow(x), ncol=length(pi))

  for (n in 1:N) {
    denominator <- sum(pi * x_given_mu[n,])

    for (k in 1:K) {
      nominator <- pi[k] * x_given_mu[n, k]

      z[n, k] <- nominator / denominator
    }
  }

  z
}

loglikelihood <- function(x, x_given_mu, pi) {
  llik <- 0
  for (n in 1:N) {
    inner_summation <- 0
    for (k in 1:K) {
      inner_summation <- inner_summation + pi[k] * x_given_mu[n, k]
    }
    llik <- llik + log(inner_summation)
  }

  llik
}

maximization.step <- function(x, z) {
  pi <- vector(length=ncol(z))
  mu <- matrix(nrow=ncol(z), ncol=ncol(x))

  for (k in 1:K) {
    pi[k] <- sum(z[, k]) / nrow(x)
  }

  for (k in 1:K) {
    denominator <- sum(z[, k])
    for (i in 1:D) {
      nominator <- sum(x[, i] * z[, k])
      mu[k, i] <- nominator / denominator
    }
  }

  list(pi=pi, mu=mu)
}

```



```

}

EM <- function(N, D, K, max_it, min_change, true_pi, true_mu) {

  ## Producing the training data
  x <- matrix(nrow=N, ncol=D)

  for(n in 1:N) {
    k <- sample(1:K, 1, prob=true_pi)
    for(d in 1:D) {
      x[n, d] <- rbinom(1, 1, true_mu[k, d])
    }
  }

  z <- matrix(nrow=N, ncol=K) # fractional component assignments
  pi <- vector(length=K) # mixing coefficients
  mu <- matrix(nrow=K, ncol=D) # conditional distributions
  llik <- vector(length=max_it) # log likelihood of the EM iterations

  ## Random initialization of the paramters
  pi <- runif(K, 0.49, 0.51)
  pi <- pi / sum(pi)
  for(k in 1:K) {
    mu[k,] <- runif(D, 0.49, 0.51)
  }

  for(it in 1:max_it) {
    x_mu <- x_given_mu(x, mu)

    ## E-step: Computation of the fractional component assignments
    z <- expectation.step(x, x_mu, pi)

    ## Log likelihood computation.
    llik[it] <- loglikelihood(x, x_mu, pi)

    ## Stop if the log likelihood has not changed significantly
    if (it > 1 && abs(llik[it] - llik[it-1]) < min_change) break

    ## M-step: ML parameter estimation from the data and fractional component assignments
    result <- maximization.step(x, z)
    pi <- result$pi
    mu <- result$mu
  }

  list(pi=pi, mu=mu, llik=llik, it=it)
}

max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations

N <- 1000 # number of training points
D <- 10 # number of dimensions
K <- 3 # number of guessed components

```

```

## true mixing coefficients
true_pi <- vector(length=3)
true_pi <- c(1/3, 1/3, 1/3)

## true conditional distributions
true_mu <- matrix(nrow=3, ncol=D)
true_mu[1,] <- c(0.5, 0.6, 0.4, 0.7, 0.3, 0.8, 0.2, 0.9, 0.1, 1)
true_mu[2,] <- c(0.5, 0.4, 0.6, 0.3, 0.7, 0.2, 0.8, 0.1, 0.9, 0)
true_mu[3,] <- c(0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1),
      xlab="Class", ylab="Probability")
axis(side=1, at=c(1:D))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")
set.seed(1234567890)

K <- 2
result <- EM(N, D, K, max_it, min_change, true_pi, true_mu)
mu <- result$mu
pi <- result$pi
llik <- result$llik
it <- result$it
plot(mu[1,], type="o", col="blue", ylim=c(0,1),
      xlab="Class", ylab="Probability")
axis(side=1, at=c(1:D))
points(mu[2,], type="o", col="red")
plot(llik[1:it], type="o", xlab="Iterations",
      ylab="Log-Likelihood")
set.seed(1234567890)

K <- 3
result <- EM(N, D, K, max_it, min_change, true_pi, true_mu)
mu <- result$mu
pi <- result$pi
llik <- result$llik
it <- result$it
plot(mu[1,], type="o", col="blue", ylim=c(0,1),
      xlab="Class", ylab="Probability")
axis(side=1, at=c(1:D))
points(mu[2,], type="o", col="red")
points(mu[3,], type="o", col="green")
plot(llik[1:it], type="o", xlab="Iterations",
      ylab="Log-Likelihood")
set.seed(1234567890)

K <- 4
result <- EM(N, D, K, max_it, min_change, true_pi, true_mu)
mu <- result$mu
pi <- result$pi
llik <- result$llik
it <- result$it
plot(mu[1,], type="o", col="blue", ylim=c(0,1),
      xlab="Class", ylab="Probability")

```

```

axis(side=1, at=c(1:D))
points(mu[2,], type="o", col="red")
points(mu[3,], type="o", col="green")
points(mu[4,], type="o", col="orange")
plot(llik[1:it], type="o", xlab="Iterations",
      ylab="Log-Likelihood")

```

Code for Assignment 3a

```

library(mboost)

data <- read.csv2("../data/bodyfatregression.csv")

fit <- blackboost(Bodyfat_percent ~ Waist_cm + Weight_kg, data=data)

cvf <- cv(model.weights(fit), type="kfold")
cvm <- cvrisk(fit, folds=cvf, grid=1:100)
plot(cvm)
set.seed(1234567890)
train_idx <- sample(nrow(data), floor(nrow(data) * (2 / 3)))
train <- data[train_idx,]
test <- data[-train_idx,]

fit <- blackboost(Bodyfat_percent ~ Waist_cm + Weight_kg, data=train,
                  control=boost_control(mstop=mstop(cvm)))
test_error <- sum((predict(fit, test) - test$Bodyfat_percent)^2)
train_error <- sum((predict(fit, train) - train$Bodyfat_percent)^2)

test_error
train_error

```

Code for Assignment 4a

```

library(mboost)
library(randomForest)
library(ggplot2)
library(reshape2)

data <- read.csv2("../data/spambase.csv")
data$Spam <- as.factor(data$Spam)

set.seed(1234567890)
train_idx <- sample(nrow(data), floor(nrow(data) * (2 / 3)))
train <- data[train_idx,]
test <- data[-train_idx,]
tree_counts <- seq(10, 100, by=10)
test_errors <- rep(0, length(tree_counts))
train_errors <- rep(0, length(tree_counts))

for (i in 1:length(tree_counts)) {

```

```

fit <- blackboost(Spam ~ ., data=train, family=AdaExp(),
                  control=boost_control(mstop=tree_counts[i]))
test_error <- 1 - (sum(predict(fit, test, type="class") == test$Spam) / nrow(test))
train_error <- 1 - (sum(predict(fit, train, type="class") == train$Spam) / nrow(train))
test_errors[i] <- test_error
train_errors[i] <- train_error
}

test_errors
train_errors
plot_data <- data.frame(Trees=tree_counts, test=test_errors, train=train_errors)
plot_data <- melt(plot_data, id="Trees", value.name="Error", variable.name="Data")

ggplot(plot_data) +
  xlab("Number of Regression Trees") +
  ylab("Misclassification Rate") +
  geom_line(aes(x=Trees, y=Error, color=Data)) +
  scale_x_discrete(limits=tree_counts)
test_errors <- rep(0, length(tree_counts))
train_errors <- rep(0, length(tree_counts))

for (i in 1:length(tree_counts)) {
  fit <- randomForest(Spam ~ ., data=train, ntree=tree_counts[i])
  test_error <- 1 - (sum(predict(fit, test, type="class") == test$Spam) / nrow(test))
  train_error <- 1 - (sum(predict(fit, train, type="class") == train$Spam) / nrow(train))
  test_errors[i] <- test_error
  train_errors[i] <- train_error
}

plot_data <- data.frame(Trees=tree_counts, test=test_errors, train=train_errors)
plot_data <- melt(plot_data, id="Trees", value.name="Error", variable.name="Data")

ggplot(plot_data) +
  xlab("Number of Regression Trees") +
  ylab("Misclassification Rate") +
  geom_line(aes(x=Trees, y=Error, color=Data)) +
  scale_x_discrete(limits=tree_counts)

```