Jin Kweon - 3032235207 - Lab9

Jin Kweon

10/28/2017

https://onlinecourses.science.psu.edu/stat857/node/82

Q. What do you mean by "LDA classifier can be viewed as an estimated version of the Bayes classifier"???

Q. How do I check sigma_hat with lda function???

Q. Does the covariance/variance function automatically caclulate sigma hat for us? ===> No it's not....

```
iris <- iris

my_lda <- function(X, y){
  combined <- as.data.frame(cbind(y = y, x = X))
  splited <- split(combined, as.numeric(combined$y))
  n <- length(y)
  k <- length(splited)
  p <- ncol(X)

  #our prior probability is NOT uniform, as we have different number of observations for for each 3 gro

  pihat <- sapply(splited, nrow) / length(y) #lapply not work since y is categorical...

  #It should be 3 by 4 matrix.
  muhat <- matrix(0, 3, 4)
  for(i in 1:3){
    nk <- sapply(splited, nrow)[[i]]
    for(j in 1:4){
      muhat[i,j] <- apply(splited[[i]][,-1], 2, sum)[j] / nk
    }
  }

  names(pihat) <- levels(iris[1:140, 5])

  sigmahat <- matrix(0, p, p)
  #This algorithm efficiency sucks,,, but it works ...
  for(i in 1:k){
    for(j in 1:nrow(splited[[i]])){
      sigmahat <- sigmahat +
        tcrossprod(as.matrix(as.numeric(unname(splited[[i]][j, -1] - muhat[i,]))))
    }
  }
  sigmahat <- sigmahat / (n - k)



  lists <- list(pi_hat = pihat, mu_hat = muhat, sigma_hat = sigmahat)

  return(lists)
}
```

```r
ldafunc <- my_lda(iris[1:140, 1:4], iris[1:140, 5])
ldafunc
```

```
## $pi_hat
##     setosa versicolor  virginica
##  0.3571429  0.3571429  0.2857143
##
## $mu_hat
##        [,1]  [,2]   [,3]  [,4]
## [1,] 5.0060 3.428 1.4620 0.246
## [2,] 5.9360 2.770 4.2600 1.326
## [3,] 6.6225 2.960 5.6075 1.990
##
## $sigma_hat
##            [,1]       [,2]       [,3]       [,4]
## [1,] 0.27294270 0.09738394 0.17311423 0.03823650
## [2,] 0.09738394 0.11884526 0.05682628 0.03123066
## [3,] 0.17311423 0.05682628 0.18806971 0.04520000
## [4,] 0.03823650 0.03123066 0.04520000 0.03909781
```

```r
ldar <- lda(Species ~.,data = iris[1:140, ])
summary(ldar)
```

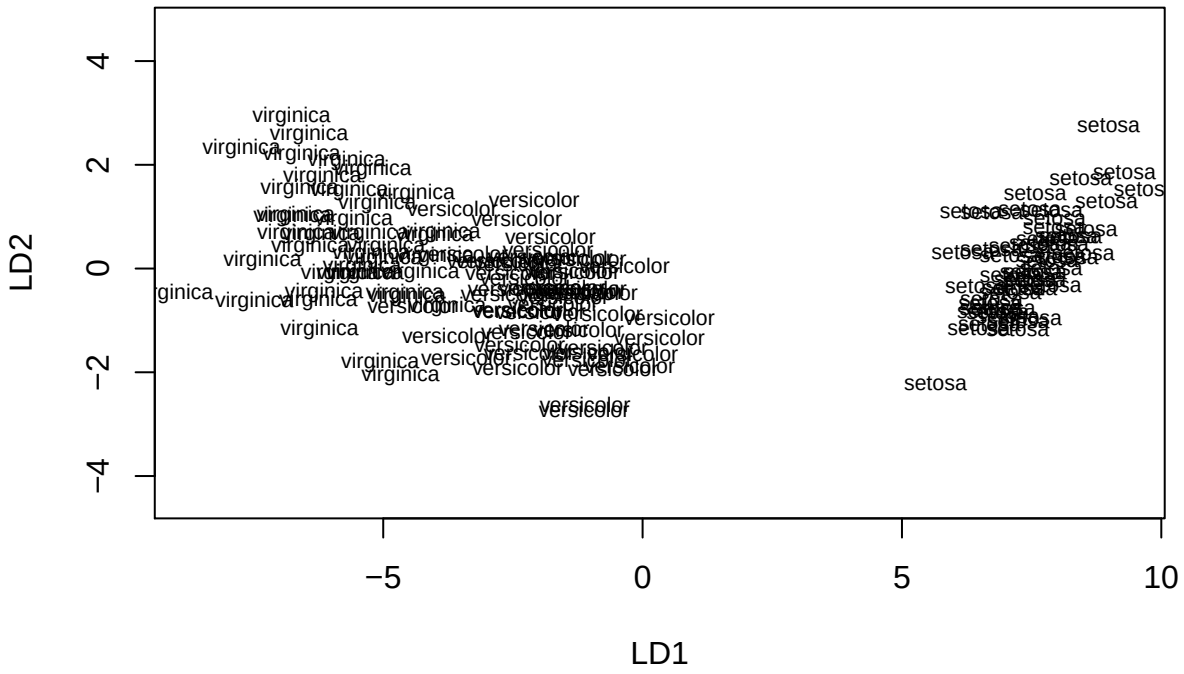
```
##         Length Class  Mode
## prior    3     -none- numeric
## counts   3     -none- numeric
## means   12     -none- numeric
## scaling  8     -none- numeric
## lev      3     -none- character
## svd      2     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```r
ldar
```

```
## Call:
## lda(Species ~ ., data = iris[1:140, ])
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3571429  0.3571429  0.2857143
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.0060       3.428       1.4620       0.246
## versicolor       5.9360       2.770       4.2600       1.326
## virginica        6.6225       2.960       5.6075       1.990
##
## Coefficients of linear discriminants:
##                     LD1         LD2
## Sepal.Length  0.8312266 -0.01087252
## Sepal.Width   1.4548314  2.32273722
## Petal.Length -2.2081308 -0.71432112
## Petal.Width  -2.7036332  2.53641482
```

```
##
## Proportion of trace:
##    LD1    LD2
## 0.9925 0.0075
```

```
plot(ldar)
```



https://www.quora.com/Mathematical-Modeling-How-are-posterior-probabilities-calculated-in-linear-discriminant-analysis (Clover told me...)

Q. How to use discriminant score....??? Why do we even need to caclulate posterior probability, if we can just calculate discriminat score and find the maximum of this....

```
predict_my_lda <- function(fit, newdata){
  m <- nrow(newdata)
  k <- length(fit$pi_hat) #group number

  pos <- matrix(0, m, k)

  #Use dmvnorm to generate the multivariate normal density!!! (we calculated mean and sigma hats before

  #each observation has four columns, and we are trying to impose three different groups' densities usi
  density <- dmvnorm(newdata, fit$mu_hat[1, ], fit$sigma_hat)
  for(i in 2:k){
    density <- rbind(density, dmvnorm(newdata, fit$mu_hat[i, ], fit$sigma_hat))
  }
```

```
  #get p(x) - denominator
  denom <- rep(0, m)

  for(i in 1:k){
    denom <- denom + fit$pi_hat[i] * density[i,]
  }

  for(i in 1:k){
    pos[, i] <- (fit$pi_hat[i] * density[i,]) / denom
  }

  colnames(pos) <- names(fit$pi_hat)
  class <- c()
  for(i in 1: m){
    class[i] <- colnames(pos)[which.max(pos[i, ])]
  }

  lists <- list(posterior = pos, class = class)
  return(lists)
}

predict_my_lda(ldafunc, iris[141:150, -5])
```

```
## $posterior
##             setosa   versicolor virginica
##  [1,] 1.822023e-43 2.360129e-06 0.9999976
##  [2,] 1.204284e-34 8.851349e-04 0.9991149
##  [3,] 1.002964e-36 1.618792e-03 0.9983812
##  [4,] 2.289667e-44 1.633764e-06 0.9999984
##  [5,] 1.027581e-44 5.095900e-07 0.9999995
##  [6,] 1.184605e-37 1.553062e-04 0.9998447
##  [7,] 1.098815e-34 9.868582e-03 0.9901314
##  [8,] 7.724661e-34 4.664455e-03 0.9953355
##  [9,] 2.353301e-39 2.112746e-05 0.9999789
## [10,] 2.848375e-32 2.112626e-02 0.9788737
##
## $class
##  [1] "virginica" "virginica" "virginica" "virginica" "virginica"
##  [6] "virginica" "virginica" "virginica" "virginica" "virginica"
```

```
predict(ldar, iris[141:150, -5])
```

```
## $class
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: setosa versicolor virginica
##
## $posterior
##           setosa   versicolor virginica
## 141 1.822023e-43 2.360129e-06 0.9999976
## 142 1.204284e-34 8.851349e-04 0.9991149
## 143 1.002964e-36 1.618792e-03 0.9983812
## 144 2.289667e-44 1.633764e-06 0.9999984
## 145 1.027581e-44 5.095900e-07 0.9999995
```

```
## 146 1.184605e-37 1.553062e-04 0.9998447
## 147 1.098815e-34 9.868582e-03 0.9901314
## 148 7.724661e-34 4.664455e-03 0.9953355
## 149 2.353301e-39 2.112746e-05 0.9999789
## 150 2.848375e-32 2.112626e-02 0.9788737
##
## $x
##           LD1         LD2
## 141 -6.941596  1.91004258
## 142 -5.400922  2.01138715
## 143 -5.815751  0.07968611
## 144 -7.105066  1.67329123
## 145 -7.141806  2.55679939
## 146 -5.933464  1.70985582
## 147 -5.470291 -0.31886548
## 148 -5.288619  0.95110588
## 149 -6.208771  2.50152274
## 150 -5.025815  0.52177854
```

```
actual <- iris[141:150, 5]
actual
```

```
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: setosa versicolor virginica
```

```
ldar
```

```
## Call:
## lda(Species ~ ., data = iris[1:140, ])
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3571429  0.3571429  0.2857143
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.0060       3.428       1.4620       0.246
## versicolor       5.9360       2.770       4.2600       1.326
## virginica        6.6225       2.960       5.6075       1.990
##
## Coefficients of linear discriminants:
##                     LD1         LD2
## Sepal.Length  0.8312266 -0.01087252
## Sepal.Width   1.4548314  2.32273722
## Petal.Length -2.2081308 -0.71432112
## Petal.Width  -2.7036332  2.53641482
##
## Proportion of trace:
##    LD1    LD2
## 0.9925 0.0075
```

```r
my_qda <- function(X, y){
  combined <- as.data.frame(cbind(y = y, x = X))
  splited <- split(combined, as.numeric(combined$y))
  n <- length(y)
  k <- length(splited)
  p <- ncol(X)

  #our prior probability is NOT uniform, as we have different number of observations for for each 3 gro

  pihat <- sapply(splited, nrow) / length(y) #lapply not work since y is categorical...

  #It should be 3 by 4 matrix.
  muhat <- matrix(0, 3, 4)
  for(i in 1:3){
    nk <- sapply(splited, nrow)[[i]]
    for(j in 1:4){
      muhat[i,j] <- apply(splited[[i]][,-1], 2, sum)[j] / nk
    }
  }

  names(pihat) <- levels(iris[1:140, 5])

  sigmahat <- array(0, dim = c(p, p, k))

  #QDA takes covrariance matrix!!!
  for(i in 1:k){
    sigmahat[,,i] <- sigmahat[,,i] + cov(splited[[i]][,-1])
  }

  lists <- list(pi_hat = pihat, mu_hat = muhat, sigma_hat = sigmahat)

  return(lists)
}

qdafunc <- my_qda(iris[1:140, 1:4], iris[1:140, 5])
qdafunc
```

```
## $pi_hat
##     setosa versicolor  virginica
##  0.3571429  0.3571429  0.2857143
##
## $mu_hat
##        [,1]  [,2]   [,3]  [,4]
## [1,] 5.0060 3.428 1.4620 0.246
## [2,] 5.9360 2.770 4.2600 1.326
## [3,] 6.6225 2.960 5.6075 1.990
##
## $sigma_hat
## , , 1
##
##            [,1]        [,2]        [,3]        [,4]
## [1,] 0.12424898 0.099216327 0.016355102 0.010330612
```

```
## [2,] 0.09921633 0.143689796 0.011697959 0.009297959
## [3,] 0.01635510 0.011697959 0.030159184 0.006069388
## [4,] 0.01033061 0.009297959 0.006069388 0.011106122
##
## , , 2
##
##            [,1]       [,2]       [,3]       [,4]
## [1,] 0.26643265 0.08518367 0.18289796 0.05577959
## [2,] 0.08518367 0.09846939 0.08265306 0.04120408
## [3,] 0.18289796 0.08265306 0.22081633 0.07310204
## [4,] 0.05577959 0.04120408 0.07310204 0.03910612
##
## , , 3
##
##            [,1]       [,2]       [,3]       [,4]
## [1,] 0.46794231 0.11041026 0.35777564 0.05125641
## [2,] 0.11041026 0.11323077 0.08107692 0.04625641
## [3,] 0.35777564 0.08107692 0.34532692 0.05930769
## [4,] 0.05125641 0.04625641 0.05930769 0.07425641
```

```
qdar <- qda(Species ~.,data = iris[1:140, ])
summary(qdar)
```

```
##         Length Class  Mode
## prior    3     -none- numeric
## counts   3     -none- numeric
## means   12     -none- numeric
## scaling 48     -none- numeric
## ldet     3     -none- numeric
## lev      3     -none- character
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
qdar
```

```
## Call:
## qda(Species ~ ., data = iris[1:140, ])
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3571429  0.3571429  0.2857143
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.0060       3.428       1.4620       0.246
## versicolor       5.9360       2.770       4.2600       1.326
## virginica        6.6225       2.960       5.6075       1.990
```

```r
predict_my_qda <- function(fit, newdata){
  m <- nrow(newdata)
  k <- length(fit$pi_hat) #group number

  pos <- matrix(0, m, k)

  #Use dmvnorm to generate the multivariate normal density!!! (we calculated mean and sigma hats before

  #each observation has four columns, and we are trying to impose three different groups' densities usi

  #sigmahat is array...
  density <- dmvnorm(newdata, fit$mu_hat[1, ], fit$sigma_hat[,, 1])
  for(i in 2:k){
    density <- rbind(density, dmvnorm(newdata, fit$mu_hat[i, ], fit$sigma_hat[,, i]))
  }


  #get p(x) - denominator
  denom <- rep(0, m)

  for(i in 1:k){
    denom <- denom + fit$pi_hat[i] * density[i,]
  }

  #posterior!!
  for(i in 1:k){
    pos[, i] <- (fit$pi_hat[i] * density[i,]) / denom
  }

  colnames(pos) <- names(fit$pi_hat) #Change posterior colnames


  #Get the column name where the posterior is the biggest
  class <- c()
  for(i in 1: m){
    class[i] <- colnames(pos)[which.max(pos[i, ])]
  }

  lists <- list(posterior = pos, class = class)
  return(lists)
}

predict_my_qda(qdafunc, iris[141:150, -5])
```

```
## $posterior
##              setosa   versicolor virginica
##  [1,] 1.593400e-174 2.124111e-09 1.0000000
##  [2,] 1.657172e-144 4.562809e-08 1.0000000
##  [3,] 7.217888e-126 5.351414e-04 0.9994649
##  [4,] 9.559272e-184 1.278474e-06 0.9999987
##  [5,] 9.198115e-184 3.512176e-10 1.0000000
##  [6,] 5.455780e-150 1.315944e-08 1.0000000
##  [7,] 3.404338e-124 3.143837e-04 0.9996856
```

```
##  [8,] 1.323189e-133 1.767812e-03 0.9982322
##  [9,] 2.679955e-155 1.731190e-06 0.9999983
## [10,] 8.559298e-119 7.284787e-02 0.9271521
##
## $class
##  [1] "virginica" "virginica" "virginica" "virginica" "virginica"
##  [6] "virginica" "virginica" "virginica" "virginica" "virginica"
```

```
predict(qdar, iris[141:150, -5])
```

```
## $class
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: setosa versicolor virginica
##
## $posterior
##            setosa   versicolor virginica
## 141 1.593400e-174 2.124111e-09 1.0000000
## 142 1.657172e-144 4.562809e-08 1.0000000
## 143 7.217888e-126 5.351414e-04 0.9994649
## 144 9.559272e-184 1.278474e-06 0.9999987
## 145 9.198115e-184 3.512176e-10 1.0000000
## 146 5.455780e-150 1.315944e-08 1.0000000
## 147 3.404338e-124 3.143837e-04 0.9996856
## 148 1.323189e-133 1.767812e-03 0.9982322
## 149 2.679955e-155 1.731190e-06 0.9999983
## 150 8.559298e-119 7.284787e-02 0.9271521
```

```
actual <- iris[141:150, 5]
actual
```

```
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: setosa versicolor virginica
```

```
qdar
```

```
## Call:
## qda(Species ~ ., data = iris[1:140, ])
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3571429  0.3571429  0.2857143
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.0060       3.428       1.4620       0.246
## versicolor       5.9360       2.770       4.2600       1.326
## virginica        6.6225       2.960       5.6075       1.990
```

Q. Why do they not predict 100% perfect???

```
set.seed(100)
k <- length(levels(iris$Species))
confusion <- matrix(0, k, k)

train_idx <- sample(nrow(iris), 90)
train_set <- iris[train_idx, ]
test_set <- iris[-train_idx, ]

#LDA
lda1 <- my_lda(train_set[,1:4], train_set[,5])
predictlda1 <- predict_my_lda(lda1, test_set[,1:4])
#compare our predicted classes with actual classes.
table(predictlda1$class, test_set[,5])
```

```
##
##              setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
```

```
confusionMatrix(predictlda1$class, test_set[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
##
## Overall Statistics
##
##                Accuracy : 0.9833
##                  95% CI : (0.9106, 0.9996)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9747
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                    1.0            1.0000           0.9474
## Specificity                    1.0            0.9767           1.0000
## Pos Pred Value                 1.0            0.9444           1.0000
## Neg Pred Value                 1.0            1.0000           0.9762
## Prevalence                     0.4            0.2833           0.3167
## Detection Rate                 0.4            0.2833           0.3000
## Detection Prevalence           0.4            0.3000           0.3000
## Balanced Accuracy              1.0            0.9884           0.9737
```

```
#LDA with r-embedded function
lda2 <- lda(train_set[,1:4], train_set[,5])
```

```r
predictlda2 <- predict(lda2, test_set[,1:4])
#compare our predicted classes with actual classes.
table(predictlda2$class, test_set[,5])
```

```
##
##              setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
```

```r
confusionMatrix(predictlda2$class, test_set[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
##
## Overall Statistics
##
##                Accuracy : 0.9833
##                  95% CI : (0.9106, 0.9996)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9747
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                    1.0            1.0000           0.9474
## Specificity                    1.0            0.9767           1.0000
## Pos Pred Value                 1.0            0.9444           1.0000
## Neg Pred Value                 1.0            1.0000           0.9762
## Prevalence                     0.4            0.2833           0.3167
## Detection Rate                 0.4            0.2833           0.3000
## Detection Prevalence           0.4            0.3000           0.3000
## Balanced Accuracy              1.0            0.9884           0.9737
```

```r
#QDA
qda1 <- my_qda(train_set[,1:4], train_set[,5])
predictqda1 <- predict_my_qda(qda1, test_set[,1:4])

table(predictqda1$class, test_set[,5])
```

```
##
##              setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
```

```r
confusionMatrix(predictqda1$class, test_set[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
##
## Overall Statistics
##
##                Accuracy : 0.9833
##                  95% CI : (0.9106, 0.9996)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9747
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                    1.0            1.0000           0.9474
## Specificity                    1.0            0.9767           1.0000
## Pos Pred Value                 1.0            0.9444           1.0000
## Neg Pred Value                 1.0            1.0000           0.9762
## Prevalence                     0.4            0.2833           0.3167
## Detection Rate                 0.4            0.2833           0.3000
## Detection Prevalence           0.4            0.3000           0.3000
## Balanced Accuracy              1.0            0.9884           0.9737
```

Q. Why do we take out the first column particularly....??? Why does the first column should be the baseline category???

```r
find_multinom_coef <- function(X, y){
  y_category <- levels(y)
  K <- length(y_category)
  #X=iris[1:140, 1:4]
  p <- ncol(X)
  X <- cbind(1, X)
  n <- nrow(X)
  X <- as.matrix(X)
  ydummy <- dummy(y) #indicator for each entry
  Y <- ydummy[,-1]

  likelihood <- function(beta){
```

```
    beta <- matrix(beta, (p+1), (K-1)) #optim function sends vectors, so I need to change to matrix....

    totalsum <- 0

    for(i in 1:n){
      leftsum <- 0
      for(k in 1:K-1){
        leftsum <- sum(leftsum, Y[i, k] * (X[i, ] %*% beta[, k]))
      }

      rightsum <- 0
      rightexpsum <- 0

      for(k in 1:K-1){
        rightexpsum <- sum(rightexpsum,(exp(X[i, ] %*% beta[, k])))
      }
      rightsum <- log(1 + rightexpsum)

      totalsum <- totalsum + leftsum - rightsum
    }

    return(-totalsum)
  }


  #check -> if I plug in zero, I should get -n*logK.  (remeber i need to input vector only for optim!!!
  # l <- likelihood(rep(0, (p+1) * (K-1)))


  mlebeta <- optim(par = matrix(0, (p+1), (K-1)) , fn = likelihood, method = "BFGS")$par

  return(mlebeta)
}


mult <- find_multinom_coef(X=iris[1:140, 1:4], y=iris$Species[1:140])
mult
```

```
##            [,1]       [,2]
## [1,] 17.7254637 -24.631223
## [2,] -6.7005422  -9.107771
## [3,] -6.2433338 -12.869906
## [4,] 13.7900526  23.118285
## [5,] -0.5066336  17.596108
```

```
#Optim only minimizes -> so minimizes -f() will be maximization


#Check answer
iris_multi <- multinom(Species ~ ., data=iris[1:140, ])
```

```
## # weights:  18 (10 variable)
## initial  value 153.805720
## iter  10 value 24.082349
## iter  20 value 6.036653
```

```
## iter  30 value 5.937954
## iter  40 value 5.930515
## iter  50 value 5.926939
## iter  60 value 5.925467
## final  value 5.923988
## converged
```

```
t(coef(iris_multi))
```

```
##              versicolor  virginica
## (Intercept)  17.7252583 -24.630925
## Sepal.Length -6.7006986  -9.107935
## Sepal.Width  -6.2434619 -12.870044
## Petal.Length 13.7902839  23.118434
## Petal.Width  -0.5060067  17.596721
```