732A99/732A68 MACHINE LEARNING

LAB 1 BLOCK 2: ENSEMBLE METHODS AND MIXTURE MODELS

JOSE M. PEÑA
IDA, LINKÖPING UNIVERSITY, SWEDEN

INSTRUCTIONS

Each student must submit a report with his/her solutions to the lab. The report must be concise but complete. It should include (i) the code implemented or the calls made to existing functions, (ii) the results of the code or calls, and (iii) explanations for the code and results.

RESOURCES

The ensemble methods assignment is designed to be solved with the R packages `mboost` and `randomForest`. No R package is needed to solve the mixture models assignment. Note that there is no online learning assignment.

1. ENSEMBLE METHODS

The file `spambase.csv` contains information about the frequency of various words, characters, etc. for a total of 4601 e-mails. Furthermore, these e-mails have been classified as spams (spam = 1) or regular e-mails (spam = 0). You can find more information about these data at `https://archive.ics.uci.edu/ml/datasets/Spambase`

Your task is to evaluate the performance of Adaboost classification trees and random forests on the spam data. Specifically, provide a plot showing the error rates when the number of trees considered are $10, 20, \ldots, 100$. To estimate the error rates, use 2/3 of the data for training and 1/3 as hold-out test data.

To learn Adaboost classification trees, use the function `blackboost()` of the R package `mboost`. Specify the loss function corresponding to Adaboost with the parameter `family`. To learn random forests, use the function `randomForest` of the R package `randomForest`. To load the data, you may want to use the following code:

```
sp <- read.csv2("spambase.csv")
sp$Spam <- as.factor(sp$Spam)
```

2. MIXTURE MODELS

Your task is to implement the EM algorithm for mixtures of multivariate Benoulli distributions. Please use the template in the next page to solve the assignment. Then, use your implementation to show what happens when your mixture models has too few and too many components, i.e. set $K = 2, 3, 4$ and compare results. Please provide a short explanation as well.

```
set.seed(1234567890)

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

true_pi <- vector(length = 3) # true mixing coefficients
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
true_pi=c(1/3, 1/3, 1/3)
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))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")

# Producing the training data
for(n in 1:N) {
  k <- sample(1:3,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}

K=3 # number of guessed components
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)
}
pi
mu

for(it in 1:max_it) {
  plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  points(mu[2,], type="o", col="red")
  points(mu[3,], type="o", col="green")
  #points(mu[4,], type="o", col="yellow")
  Sys.sleep(0.5)

  # E-step: Computation of the fractional component assignments
  # Your code here

  #Log likelihood computation.
  # Your code here

  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  # Stop if the lok likelihood has not changed significantly
  # Your code here

  #M-step: ML parameter estimation from the data and fractional component assignments
  # Your code here
}
pi
mu
plot(llik[1:it], type="o")
```