Clustering Accelerometer Data

#https://archive.ics.uci.edu/ml/datasets/Dataset+for+ADL+Recognition+with+Wrist-worn+Accelerometer

#Classification of the 14 activities in the accelerometer data. #Vector quantize technique is used to create feature set #Histogram of cluster centers #Random forest is used as a classifier.

```
set.seed(10)
w_path<-getwd()
train_data<-list()
test_data<-list()
dir_path<-paste(w_path,"/data1/",sep = "")
labels<-as.factor(list.files(path=dir_path))
int_dir<-list.files(path=dir_path)
f_num<-1
t_num<-1
for (f in 1:length(int_dir)) {
    rd_file <- list.files(paste(dir_path,int_dir[f],sep = ""))
    path <- paste(dir_path,int_dir[f],sep = "")
    full_path <- paste(path,"/",sep = "")
    size<-length(rd_file)*80/100
    if((length(rd_file) - size)<2) {
        size = size - 1
    }
    idx <- sample.int(length(rd_file),size,replace=FALSE)
    train_files <- rd_file[idx]
    test_files <- rd_file[-idx]
    dac<- read.table(paste(full_path,train_files[1],sep=""),
                      stringsAsFactors = FALSE,header=FALSE)
    dac$z<-f_num
    for (g in train_files[-1]){
        df <- read.table(paste(full_path,g,sep=""),
                        stringsAsFactors = FALSE,header=FALSE)
        f_num<<-f_num+1
        df$z<-f_num
        dac <- rbind(dac, df)
    }
    dac$y<- int_dir[f]
    train_data<-rbind(train_data,dac)
    train_data$y<-as.factor((train_data$y))

    tdac<- read.table(paste(full_path,test_files[1],sep=""),
                      stringsAsFactors = FALSE,header=FALSE)
    tdac$z<-t_num
    for (g in test_files[-1]){
      dt <- read.table(paste(full_path,g,sep=""),
                      stringsAsFactors = FALSE,header=FALSE)
      t_num<<-t_num+1
      dt$z<-t_num
      tdac <- rbind(tdac, dt)
    }
    tdac$y<- int_dir[f]
    test_data<-rbind(test_data,tdac)
```

```
    test_data$y<-as.factor((test_data$y))

    eighty_training<-train_data
    eighty_label<-train_data$y
    twenty_test <- test_data
    twenty_label <- test_data$y
}
```

```
set.seed(10)
ts<-length(eighty_training$V1)%%15
if (ts!= 0) {
  nitm<-(length(eighty_training$V1) - ts)
} else {
  nitm<- length(eighty_training$V1)
}
nr<-nitm/5
ntrain <- matrix(data=NA,nrow=nr,ncol=15)
k <-1
l<-1
for (n in 1:nitm) {
  for (j in 1:3) {
    if (k>nr) {
      break
    }
    ntrain[k,l] <- eighty_training[n,j]
    if (l == 15) {
      k<<-k+1
      l<<-1
    } else {
    l<<-l + 1
    }
  }
}
kmlist<-kmeans(ntrain, 100, nstart = 10, iter.max = 1000 ,algorithm="MacQueen")
```

```
library(pracma)
vfunc<-function(dacc) {
vts<-length(dacc$V1)%%15
if (vts!= 0) {
  vitm<-(length(dacc$V1) - vts)
} else {
  vitm<- length(dacc$V1)
}
vr<-vitm/5
nvector <- matrix(data=NA,nrow=vr,ncol=15)
k <-1
l<-1
for (n in 1:vitm) {
  for (j in 1:3) {
    if (k>vr) {
      break
    }
    nvector[k,l] <- dacc[n,j]
    if (l == 15) {
```

```
      k<-k+1
      l<-1
    } else {
    l<-l + 1
    }
  }
}
nvector
}

ll<-1
index <- data.frame(matrix(data=0,nrow=651,ncol=101))
cfunc<-function(ll,ii){
  index[ll,ii]<<-index[ll,ii]+1
}
tfunc<-function(ll,lbl){
  index[ll,101]<<-lbl
}
un<-length(unique(eighty_training$z))
for (f in 1:un) {
      dacc<-subset(eighty_training[,1:3], eighty_training[,4] == as.character(f))
      tl<-subset(eighty_training[,5], eighty_training[,4] == as.character(f))
      fvector<-vfunc(dacc)
      nrw<-nrow(fvector)
      for (feature in 1:nrw) {
        dist<-distmat(kmlist$centers,fvector[feature,])
        ii<-which.min(dist)

        if(which.min(dist)!=0) {
          cfunc(ll,ii)
          tfunc(ll,tl[1])
        }
      }
      ll<<-ll+1
}
```

```
library(pracma)
ll<-1
tindex <- data.frame(matrix(data=0,nrow=162,ncol=101))
testfunc<-function(ll,ii){
  tindex[ll,ii]<<-tindex[ll,ii]+1
}
testlbl<-function(ll,lbl){
  tindex[ll,101]<<-lbl
}
tn<-length(unique(twenty_test$z))
for (f in 1:tn) {
      dacc<-subset(twenty_test[,1:3], twenty_test[,4] == as.character(f))
      ttl<-subset(twenty_test[,5], twenty_test[,4] == as.character(f))
      tvector<-vfunc(dacc)
      nrw<-nrow(tvector)
      for (feature in 1:nrw) {
        if (nrw != 0) {
            dist<-distmat(kmlist$centers,tvector[feature,])
```

```
            ii<-which.min(dist)
            testfunc(ll,ii)
            testlbl(ll,ttl[1])
          }
        }
        ll<<-ll+1
}
```

```
library(h2o)
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     &&, %*%, %in%, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

```
s<-0
train = as.data.frame(index[,-101])
test = as.data.frame(tindex[,-101])
h2o.init(nthreads = -1, max_mem_size = '4g', ip = "127.0.0.1", port = 50001)
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         35 minutes 14 seconds
##     H2O cluster timezone:       America/Los_Angeles
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.26.0.2
##     H2O cluster version age:    2 months and 7 days
##     H2O cluster name:           H2O_started_from_R_amishukl_vsg553
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   4.00 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
```

```
##     H2O Connection ip:          127.0.0.1
##     H2O Connection port:        50001
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.5.2 (2018-12-20)
```

```
train$class = as.factor(index[,101])
test$class = as.factor(tindex[,101])

train.h2o <- as.h2o(train)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
test.h2o <- as.h2o(test)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
T.dep <- 101
T.indep <- c(1:100)

#Tree 16, depth 10
model <- h2o.randomForest(y=T.dep, x=T.indep, training_frame = train.h2o,
                          ntrees = 16, max_depth = 10, seed = 10)
```

```
##
  |
  |                                                                      |   0%
  |
  |====                                                                  |   6%
  |
  |======================================================================| 100%
```

```
preds <- as.data.frame(h2o.predict(model, newdata=test.h2o))
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
confusionMatrix <- table(test$class, as.vector(preds$predict))
```

#Confusion Matrix

```
 confusionMatrix
```

```
##
##       1 10 11 12 13 14  2  3  4  5  6  8
##   1   3  0  0  0  0  0  0  0  0  0  0  0
##   2   0  0  0  0  0  0 19  1  0  0  0  0
```

```
##   3   0  0  0  0  0  0  0  5  0  1  0  0
##   4   0  0  0  0  0  0  3  0  4  0  0  1
##   5   0  1  0  0  0  0  0  0  0 18  0  0
##   6   0  0  0  0  0  0  0  0  0  0  1  0
##   7   0  0  0  0  0  0  0  0  0  0  0  1
##   8   0  0  1  1  0  0  0  0  0  0  0 18
##   9   0  1  2  0  0  0  0  0  0  0  0  2
##   10  0 19  0  0  0  0  0  0  0  0  0  0
##   11  0  0 15  2  0  0  0  0  0  0  0  2
##   12  0  0  5 12  1  0  0  0  0  0  0  2
##   13  0  0  0  0  1  1  0  0  0  0  0  0
##   14  0  0  2  0  0 15  2  0  0  0  0  0
```

```
correct_prediction <- preds$predict == tindex[,101]
accuracy <- sum(correct_prediction)/
  (sum(correct_prediction)+sum(!correct_prediction))
```

#Accuracy

```
accuracy
```

```
## [1] 0.8024691
```

##Experimented with different cluster size and chunk size. Below are the results

##32 chunk size 200 clusters: Accuracy - 0.6790123

##48 chunk size 200 clusters: Accuracy - 0.6666667

##Based on the above two observation it seems as the chunk size decreses the accuracy increases a bit. ##Same observation with my final run with 15 chunk size and 100 clusters, the accuracy improved. Below is the accuracy with this combination

```
accuracy
```

```
## [1] 0.8024691
```

##Based on the below two observation it seems as the cluster size is increased, chunk size remaining the same, the accuracy did not change much but slight decrease in the accuracy is seen. ##32 chunk size 480 clusters: Accuracy - 0.66875

##32 chunk size 200 clusters: Accuracy - 0.6790123

##Another factor which varied the results slightly is the choice of seed size.