mini proj wk8-9

Aaron Palumbo

October 23, 2015

Dependencies

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# global chunk options
knitr::opts_chunk$set(warning=FALSE)
```

Data

We're going to work with a dataset from the UCI Machine Learning Repository on diabetes patients (https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008).

```r
dd <-read.csv("dataset_diabetes/diabetic_data.csv",
              stringsAsFactors = FALSE, na.strings = "?")
# Right away we can get rid of columns without much info
dd = dd[ , -caret::nearZeroVar(dd)]

# We can also drop any id columns, since those should be
# unique identifiers and not useful for clustering
idCols <- c("encounter_id", "patient_nbr")
dd = dd[ , !(names(dd) %in% idCols)]
```

Data Cleaning

We are going to apply the hamming distance to these points so we want to convert all our columns to categorical data (factors). Let's take a look at what we have to start with.

```r
getInfo <- function(dd) {
  return(do.call(rbind, lapply(names(dd), function(n) {
    data.frame(name=n,
               firstrow=dd[1, n],
               class=class(dd[1, n]),
               numfactors=length(table(dd[,n])))
  }))
}

getInfo(dd)
```

```
##                          name                 firstrow     class numfactors
## 1                        race                Caucasian character          5
## 2                      gender                   Female character          3
## 3                         age                   [0-10) character         10
## 4                      weight                     <NA> character          9
## 5           admission_type_id                     <NA>   integer          8
## 6    discharge_disposition_id                     <NA>   integer         26
## 7         admission_source_id                     <NA>   integer         17
## 8            time_in_hospital                     <NA>   integer         14
## 9                  payer_code                     <NA> character         17
## 10          medical_specialty Pediatrics-Endocrinology character         72
## 11         num_lab_procedures                     <NA>   integer        118
## 12             num_procedures                     <NA>   integer          7
## 13            num_medications                     <NA>   integer         75
## 14          number_outpatient                     <NA>   integer         39
## 15           number_emergency                     <NA>   integer         33
## 16           number_inpatient                     <NA>   integer         21
## 17                     diag_1                   250.83 character        716
## 18                     diag_2                     <NA> character        748
## 19                     diag_3                     <NA> character        789
## 20           number_diagnoses                     <NA>   integer         16
## 21                  A1Cresult                     None character          4
## 22                  metformin                       No character          4
## 23                  glipizide                       No character          4
## 24                  glyburide                       No character          4
## 25               pioglitazone                       No character          4
## 26              rosiglitazone                       No character          4
## 27                    insulin                       No character          4
## 28                     change                       No character          2
## 29                diabetesMed                       No character          2
## 30                 readmitted                       NO character          3
```

Now there are four columns that need to be binned to be useful.

```
numBins <- 21
colsToBin <- c("num_lab_procedures", "diag_1", "diag_2", "diag_3")
for (c in colsToBin){
  vec <- as.integer(dd[ ,c])
  from <- min(vec, na.rm=TRUE)
  to   <- max(vec, na.rm=TRUE)
  bins <- seq(from, to, length.out=numBins)
  binned <- cut(vec, breaks=bins)
  # we'll make this a factor a little later
  binned <- as.character(binned)
  dd[ ,c] <- binned
}
```

```r
# let's look at our summary again
getInfo(dd)
```

```
##                          name                  firstrow     class numfactors
## 1                        race                 Caucasian character          5
## 2                      gender                    Female character          3
## 3                         age                    [0-10) character         10
## 4                      weight                      <NA> character          9
## 5           admission_type_id                      <NA>   integer          8
## 6  discharge_disposition_id                      <NA>   integer         26
## 7         admission_source_id                      <NA>   integer         17
## 8            time_in_hospital                      <NA>   integer         14
## 9                  payer_code                      <NA> character         17
## 10          medical_specialty Pediatrics-Endocrinology character         72
## 11           num_lab_procedures              (40.3,46.9] character         20
## 12              num_procedures                      <NA>   integer          7
## 13             num_medications                      <NA>   integer         75
## 14            number_outpatient                      <NA>   integer         39
## 15             number_emergency                      <NA>   integer         33
## 16             number_inpatient                      <NA>   integer         21
## 17                      diag_1                 (202,252] character         20
## 18                      diag_2                      <NA> character         20
## 19                      diag_3                      <NA> character         20
## 20            number_diagnoses                      <NA>   integer         16
## 21                   A1Cresult                      None character          4
## 22                   metformin                        No character          4
## 23                   glipizide                        No character          4
## 24                   glyburide                        No character          4
## 25                pioglitazone                        No character          4
## 26               rosiglitazone                        No character          4
## 27                     insulin                        No character          4
## 28                      change                        No character          2
## 29                 diabetesMed                        No character          2
## 30                  readmitted                        NO character          3
```

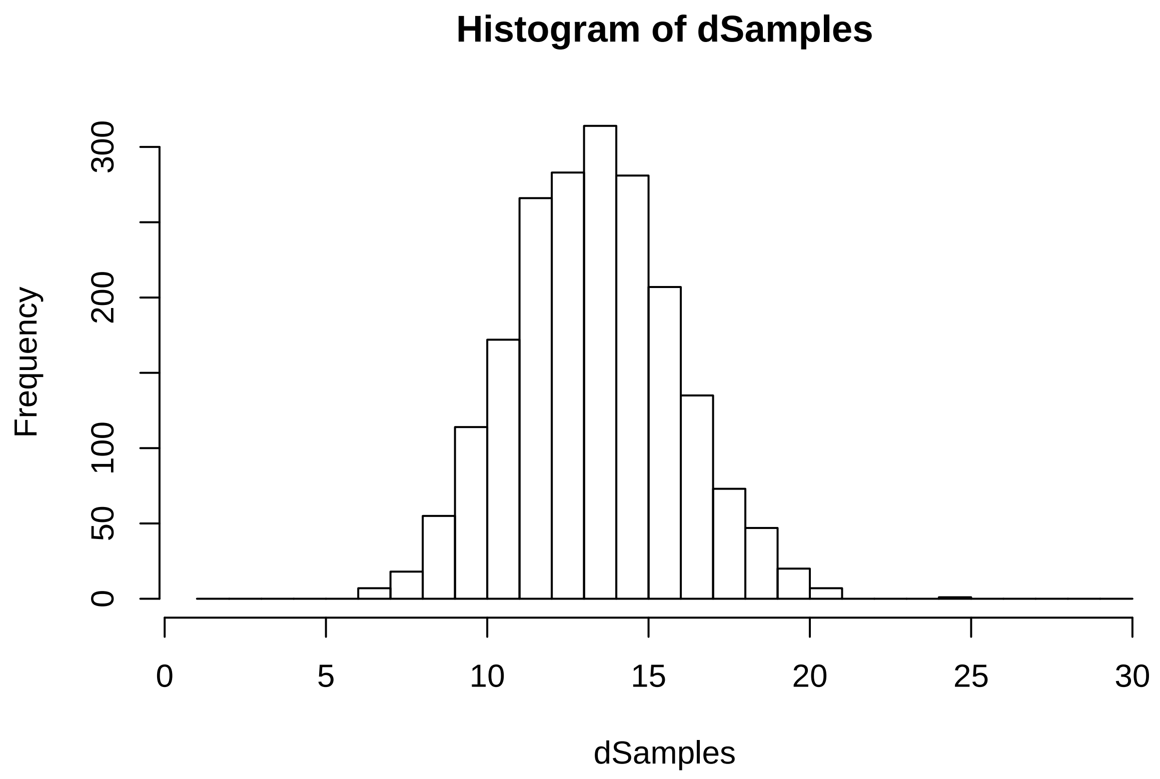
Last, we will treat all the NAs as a factor category by replaceing with 'missing' and then coercing each column to a factor.

```r
for (c in names(dd)) {
  vec <- dd[ ,c]
  vec[is.na(vec)] <- "missing"
  dd[ ,c] <- as.factor(vec)
}

getInfo(dd)
```

```
##                          name                  firstrow  class numfactors
## 1                        race                 Caucasian factor          6
## 2                      gender                    Female factor          3
## 3                         age                    [0-10) factor         10
## 4                      weight                   missing factor         10
## 5           admission_type_id                         6 factor          8
## 6  discharge_disposition_id                       25 factor          26
## 7       admission_source_id                        1 factor          17
## 8          time_in_hospital                        1 factor          14
## 9                payer_code                  missing factor          18
## 10        medical_specialty Pediatrics-Endocrinology factor          73
## 11        num_lab_procedures              (40.3,46.9] factor          21
## 12           num_procedures                        0 factor           7
## 13          num_medications                        1 factor          75
## 14        number_outpatient                        0 factor          39
## 15         number_emergency                        0 factor          33
## 16         number_inpatient                        0 factor          21
## 17                   diag_1                (202,252] factor          21
## 18                   diag_2                  missing factor          21
## 19                   diag_3                  missing factor          21
## 20         number_diagnoses                        1 factor          16
## 21                A1Cresult                     None factor           4
## 22                metformin                       No factor           4
## 23                glipizide                       No factor           4
## 24                glyburide                       No factor           4
## 25             pioglitazone                       No factor           4
## 26            rosiglitazone                       No factor           4
## 27                  insulin                       No factor           4
## 28                   change                       No factor           2
## 29              diabetesMed                       No factor           2
## 30               readmitted                       NO factor           3
```

Now let's get an idea of the distribution of distances we have.

```r
hamDist <- function(r1, r2) {
  # The hamming distance between two rows
  return(sum(r1 == r2))
}

# We can't compute the distance between all points
# so we will just sample a large number of distances
nSamples <- 2000
n <- nrow(dd)
dSamples <- unlist(lapply(1:nSamples, function(i) {
  s <- sample(1:n, 2)
  hamDist(dd[s[1], ], dd[s[2],])
}))
hist(dSamples, breaks=1:30)
```



Initializing the Cluster Tree

Okay, I think we can handle about 100 points in main memory (this creates about 5000 combinations). So let's go ahead and sample from our data and create some clusters.

```r
# ######### #
# Functions #
# ######### #

hamDist <- function(r1, r2) {
  # The hamming distance between two rows
  return(sum(r1 == r2))
}

nextPoint <- function(clusters, dist.df,
                      index1='index1', index2='index2',
                      dist.col='dist') {
  potential.set <- potentialDF(clusters, dist.df)
  # find the candidate furthest from existing seeds
  minClusterDist <- potential.set %>%
    group_by(candidate) %>%
    summarize(dist = min(dist))

  return(minClusterDist$candidate[which.max(minClusterDist$dist)])
}
```

```r
assignToCluster <- function(clusters, dist.df,
                            index1='index1', index2='index2',
                            dist.col='dist') {
  potential.set <- potentialDF(clusters, dist.df)
  # Find the point closest to a cluster
  minClusterDist <-
    potential.set %>% group_by(cluster) %>% slice(which.min(dist))
  assignment <- minClusterDist[which.min(minClusterDist$dist), ]
  clusters[assignment$candidate[1]] <- assignment$cluster[1]
  if (sum(clusters == 0) > 0) {
    cdf <- assignToCluster(clusters, dist.df, index1, index2, dist.col)
    assignment <- rbind(assignment, cdf)
  }
  return(assignment)
}

hcluster <- function(clusters, dist.df,
                     index1='index1', index2='index2',
                     dist.col='dist') {
  dist.df <- dist.df[order(dist.df[dist.col]), ]
  assign.order <- do.call(rbind, apply(dist.df, 1, function(r) {
    id1 <- dist.df[1, index1]
    id2 <- dist.df[1, index2]
    cluster.union <- union(which(clusters == clusters[id1]),
                           which(clusters == clusters[id2]))
    clusters[cluster.union] <<- clusters[cluters[id1]]
    return(clusters)
  }))
  return(do.call(rbind, assign.order))

}

potentialDF <- function(clusters, dist.df,
                        index1='index1', index2='index2',
                        dist.col='dist') {
  # Break out which points have been assigned and which are candidates
  candidates <- which(clusters == 0)
  assigned <- which(clusters != 0)
  # id vectors
  id1 <- dist.df[index1]
  id2 <- dist.df[index2]
  # create combinations of candidates and assigned
  potential.set <- expand.grid(candidates, assigned)
  names(potential.set) <- c('candidate', 'assigned')
  # add distance information
  potential.set['dist'] <-
    apply(apply(potential.set, 1, sort), 2, function(c) {
      dist.df[(id1 == c[1] & id2 == c[2]), dist.col]})
  # add cluster information
  potential.set['cluster'] <-
    unlist(lapply(potential.set$assigned, function(i) {clusters[i]}))
  return(potential.set)
}
```

```r
# ### #
# Run #
# ### #

# Cluster parameters
k <- 10
n <- 5

# Sample from data
sample.df <- dd[sample(1:nrow(dd), n), ]

# Initialize assignment vector
tocluster <- rep(0, n)

# Enumerate all combinations of points
combos.df <- as.data.frame(t(combn(1:n, 2)))
names(combos.df) <- c("index1", "index2")
combos.df['dist'] <- unlist(apply(combos.df, 1, function(c){
    hamDist(sample.df[c[1], ], sample.df[c[2], ])
  }))
combos.df['dist.sq'] <- combos.df$dist**2

# Assign first two clusters seeds
maxd <- combos.df[which.max(combos.df$dist), ]
tocluster[maxd[1, 1]] <- 1
tocluster[maxd[1, 2]] <- 2

# Assign the rest of the initial k clusters seeds
trash <- lapply(3:k, function(i) {
  nextPt <- nextPoint(tocluster, combos.df)
  tocluster[nextPt] <<- i
})

# Now we have our seeds, we need to assign points to clusters
assignments <- assignToCluster(tocluster, combos.df)
tocluster[assignments$candidate] <- assignments$cluster
```

At this point we have assigned all our initial points to a cluster. Now we need to create a tree to hold hierarchical relationship between clusters, and we need to store information about each cluster.

```r
nodes <- list()
for (i in unique(tocluster)) {
  nodes[[as.character(i)]] <- list(
    parent=NA,
    type='leaf',
    N=table(tocluster)[i],
    clustroid=NA,
    clustroid.rowsum=NA,
    closest=list(),
    farthest=list()
  )
}
```