# Taxi Data

*Aaron Palumbo*

*9/6/2015*

## Contents

Per Rohan's suggestion I'm following the example here: https://github.com/RevolutionAnalytics/rmr2/blob/master/docs/tutorial.md

I started by downloading data for 2015 green taxis, but just that was 1.5GB. I started looking here: http://hortonworks.com/hadoop-tutorial/using-commandline-manage-files-hdfs/ to try to figure out how to get that file into hdfs (there's no way I can load that into memory and use R to push it into hdfs), but I'm not clear on how to do this.

I then downloaded the green taxi data for just January 2015 (https://storage.googleapis.com/tlc-trip-data/2015/green_tripdata_2015-01.csv). This was a lot smaller so I will attempt to move forward with that.

Here's the sample code from the tutorial:

```
# Copyright 2011 Revolution Analytics
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

library(rmr2)

## @knitr kmeans-signature
kmeans.mr =
  function(
    P,
    num.clusters,
    num.iter,
    combine,
    in.memory.combine) {
## @knitr kmeans-dist.fun
    dist.fun =
      function(C, P) {
        apply(
          C,
          1,
          function(x)
            colSums((t(P) - x)^2))}
## @knitr kmeans.map
```

```
    kmeans.map =
      function(., P) {
        nearest = {
          if(is.null(C))
            sample(
              1:num.clusters,
              nrow(P),
              replace = TRUE)
          else {
            D = dist.fun(C, P)
            nearest = max.col(-D)}}
        if(!(combine || in.memory.combine))
          keyval(nearest, P)
        else
          keyval(nearest, cbind(1, P))}
## @knitr kmeans.reduce
    kmeans.reduce = {
      if (!(combine || in.memory.combine) )
        function(., P)
          t(as.matrix(apply(P, 2, mean)))
      else
        function(k, P)
          keyval(
            k,
            t(as.matrix(apply(P, 2, sum))))}
## @knitr kmeans-main-1
    C = NULL
    for(i in 1:num.iter ) {
      C =
        values(
          from.dfs(
            mapreduce(
              P,
              map = kmeans.map,
              reduce = kmeans.reduce)))
      if(combine || in.memory.combine)
        C = C[, -1]/C[, 1]
## @knitr end
#      points(C, col = i + 1, pch = 19)
## @knitr kmeans-main-2
      if(nrow(C) < num.clusters) {
        C =
          rbind(
            C,
            matrix(
              rnorm(
                (num.clusters -
                   nrow(C)) * nrow(C)),
              ncol = nrow(C)) %*% C) }}
        C}
## @knitr end

## sample runs
```

```
##

out = list()

for(be in c("local", "hadoop")) {
  rmr.options(backend = be)
  set.seed(0)
## @knitr kmeans-data
  P =
    do.call(
      rbind,
      rep(
        list(
          matrix(
            rnorm(10, sd = 10),
            ncol=2)),
        20)) +
    matrix(rnorm(200), ncol =2)
## @knitr end
#  x11()
#  plot(P)
#  points(P)
  out[[be]] =
## @knitr kmeans-run
    kmeans.mr(
      to.dfs(P),
      num.clusters = 12,
      num.iter = 5,
      combine = FALSE,
      in.memory.combine = FALSE)
## @knitr end
}
```

Now let's try to point this at the taxi data:

```
library(rmr2)
```

```
## Warning: S3 methods 'gorder.default', 'gorder.factor', 'gorder.data.frame',
## 'gorder.matrix', 'gorder.raw' were declared in NAMESPACE but not found
```

```
## Please review your hadoop settings. See help(hadoop.settings)
```

```
##########################################################
# We'll keep the kmeans map reduce function in tact. #
##########################################################

## @knitr kmeans-signature
kmeans.mr =
  function(
    P,
    num.clusters,
    num.iter,
```

```
    combine,
    in.memory.combine) {
## @knitr kmeans-dist.fun
    dist.fun =
      function(C, P) {
        apply(
          C,
          1,
          function(x)
            colSums((t(P) - x)^2))}
## @knitr kmeans.map
    kmeans.map =
      function(., P) {
        nearest = {
          if(is.null(C))
            sample(
              1:num.clusters,
              nrow(P),
              replace = TRUE)
          else {
            D = dist.fun(C, P)
            nearest = max.col(-D)}}
        if(!(combine || in.memory.combine))
          keyval(nearest, P)
        else
          keyval(nearest, cbind(1, P))}
## @knitr kmeans.reduce
    kmeans.reduce = {
      if (!(combine || in.memory.combine) )
        function(., P)
          t(as.matrix(apply(P, 2, mean)))
      else
        function(k, P)
          keyval(
            k,
            t(as.matrix(apply(P, 2, sum))))}
## @knitr kmeans-main-1
    C = NULL
    for(i in 1:num.iter ) {
      C =
        values(
          from.dfs(
            mapreduce(
              P,
              map = kmeans.map,
              reduce = kmeans.reduce)))
      if(combine || in.memory.combine)
        C = C[, -1]/C[, 1]
## @knitr end
#      points(C, col = i + 1, pch = 19)
## @knitr kmeans-main-2
      if(nrow(C) < num.clusters) {
        C =
```

```
          rbind(
            C,
            matrix(
              rnorm(
                (num.clusters -
                   nrow(C)) * nrow(C)),
              ncol = nrow(C)) %*% C) }}
        C}
```

Now we'll load in out data:

```
green <- read.csv("../../data/green_tripdata_2015-01.csv")
```

We're just supposed to look at clustering something so I guess we'll start with pickup location:

```
## Points
points <- as.matrix(green[ , c("Pickup_longitude", "Pickup_latitude")])

## Now let's see what happens

out <- kmeans.mr(
          to.dfs(points),
          num.clusters = 12,
          num.iter = 5,
          combine = FALSE,
          in.memory.combine = FALSE)
```

Let's take a look at what we got:

```
out
```

```
##       Pickup_longitude Pickup_latitude
##  [1,]        -73.93608        40.75024
##  [2,]        -73.93600        40.74974
##  [3,]        -73.93603        40.74993
##  [4,]        -73.93598        40.74989
##  [5,]          0.00000         0.00000
##  [6,]        -73.93620        40.74994
##  [7,]        -73.93614        40.74987
##  [8,]        -73.93606        40.74966
##  [9,]        -73.93620        40.75018
## [10,]        -73.93620        40.74994
## [11,]        -73.93606        40.74997
## [12,]        -73.93627        40.74977
```

This doesn't look right at all ...