## **Introduction**

Real-Time Location Systems (RTLS) or Indoor Positioning Systems (IPS) are devices that allow for the monitoring, locating, and tracking of items or people [1]. Traditional Global Positioning Systems (GPS) do not provide the ability to locate or track items inside of buildings, but RTLS utilize signals from Local Area Networks (LANs) to detect access points allowing for the tracking and locating of items indoors in real-time [2]. The use of these systems within businesses or communities has the potential to improve safety, encourage efficiency, and mitigate the risk of theft or lost items [3].

In the book *Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving*, the authors Deborah Nolan and Duncan Temple Lang describe the use of RTLS to estimate the location of a device using signal strength detected by various access points throughout one level of an office building. A k-Nearest Neighbors (k-NN) approach is used to predict positions of the device using a method that calculates the closest set of signal strengths to the current set of signal strengths detected by the access points [2]. The basis of this statistical method is to find the k number of nearest observations to the observation of interest. Then, the observation can be predicted based on the mean of the neighbors [4].

In this analysis, we briefly review the approach and expand upon the work done in the text by conducting a thorough analysis of the location of access points with corresponding MAC addresses to determine the most accurate address to predict location. This analysis uses location and access point data. We use both a standard k-NN approach as discussed in the text as well as a k-NN approach where weight is placed on the received signal strength. This allows us to estimate the position of a device given the signal strength between said device and various access points throughout the building. We will use the findings of these procedures to determine which access points and corresponding MAC addresses should be used for RTLS in this building based on the accuracy of device location prediction.

## **Background**

The data collected for this analysis is from the text and offers location information for one floor of a building at the University of Mannheim. There are 6 access points (represented by the black squares) in the building according to the 15-meter by 36-meter floor plan, which can be seen in **Figure 1**. The data provides the coordinates that a hand-held device was located along with the orientation of the device. Signal strength data exists for both known locations and unknown locations, which provides the opportunity for predictive models to be created and tested.

In **Figure 1**, the building only includes 6 access points however the data provides 7 access points. It is necessary to determine which MAC address to remove from the data for accuracy.

|  |
| --- |
| **Figure 1. Floor Plan of the Building from which Data was Collected** |
|  |
|  |

The authors of the text, Nolan and Lang, arbitrarily determine that the access point with MAC address 00:0f:a3:39:e1:c0 (this address will be referred to as E1:C0 from this point forward) should be kept in the data and the other access point data from MAC address 00:0f:a3:39:dd:cd (DD:CD) should be removed. They then run a *k*-NN analysis to match the access points to location information.

In the following sections, we will detail our exploration of this data to determine if we support the decision of the authors. We also provide a more detailed account of the accuracy obtained by predicting location between the two access points.

## **Method**

**Data description**

In the offline and online datasets, there are the same initial variables to describe time, MAC address, location, and orientation. The data have been cleaned considerably for the analysis as detailed in Nolan and Lang’s work. **Table 1** depicts descriptions of the variables used for this analysis.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1: Cleaned Data to be used in this Study** | | | |
| **Variable** | **Type** | **Description** | **Range/Values** |
| time | POSIXt | Timestamp in milliseconds since 01/01/1970 UTC | 2006-02-10 23:31:58 to 2006-03-09 12:41:10 |
| scanMac | character | MAC address of the scanning device | include MAC, others like 183831 |
| posX | numeric | The physical coordinate of the scanning device | 0 to 33 |
| posY | numeric | The physical coordinate of the scanning device | 0 to 13 |
| degree |  | Orientation of the user carrying the scanning device in degrees | 0 to 360 |
| MAC | string | MAC address of the wireless access point | 00:14:bf:b1:97:8a, 00:14:bf:b1:97:90, 00:0f:a3:39:e1:c0, 00:14:bf:b1:97:8d, 00:14:bf:b1:97:81, 00:14:bf:3b:c7:c6, 00:0f:a3:39:dd:cd |
| signal | numeric | Measured signal strength in dBm | -99 to -25 |
| rawTime | numeric | Date and time of the signal strength without conversion | 1,139,643,118,358 to 1,141,936,870,456 |
| angle | numeric | Angle of the scanning device | 0, 45, 90, 135, 180, 225, 270, 315 |
| posXY | string | Concatenation of X and Y positions with a hyphen | 0-0, 0.31-9.42, 0.93-11.69, etc. |

**Nearest neighbor methods to predict location**

There are many different statistical methods to estimate the location of a device from the strength of the signal detected between the device and several access points. In this case study, we take two separate approaches to determine which MAC address should be removed for location prediction.

The first method applied in this analysis is an unweighted *k*-NN model, which is also used by Nolan and Lang [2]. By calculating the *k* observations that are the closest to new signal strengths for an unknown location, we are then able to predict the location given any new signal strengths in our data using the positions of the closest *k* training observations. In this analysis, the closeness is defined as the distance between the recorded signals of the access points and the unknown device location [2].

The second approach is a weighted *k*-NN model which uses the received signal strengths for the weighting. This alternate approach differentiates between access points given how close they actually are from the new observation’s signals. Stronger signal strengths from a certain MAC address are given higher priority for locating the device in question than other access points.

These two approaches will be used to address the following questions:

1. Which of these two MAC addresses (E1:C0 or DD:CD) should be used and which should not be used for RTLS?
2. Which MAC address yields the best prediction of location?
3. Does using data for both MAC addresses simultaneously yield more, or less, accurate prediction of location?
4. Does using a weighted k-NN model provide a different or more accurate result than a non-weighted *k*-NN approach?
5. For what range of values of weights are you able to obtain better prediction values than for the unweighted *k*-nearest neighbor approach?

## **Results**

In Nolan and Lang’s work, 12 known access points have available data; the experimental signal strength measurements for 6 of these access points were taken at the same floor [2]. Using field strength maps similar to those in **Figure 3**, the authors were able to match MAC addresses to access points with no ambiguity with the exception of the two MAC addresses E1:C0 and DD:CD. These have similar heat maps and both correspond to the access point near the center of the building (i.e., x = 7.5 and y = 6.3). In addition, the field map for DD:CD appears mostly uniform regardless of the measurement orientation, while the map for E1:C0 distinctly varies with angle. It is reasonable to test whether including the access point CC:CD or substituting it for E1:C0 might yield more accurate predictions of position than using E1:C0 alone. In this case study, we will evaluate the impacts in location prediction with E1:C0 or DD:CD to address our questions of interest.

**Signal strength distribution by angle for access points DD:CD and E1:C0**

To compare the two questionable MAC addresses E1:C0 or DD:CD, we will first analyze the distributions of signal strengths based on orientation.

**Figure 2** shows the distributions of signal strength by angle for access point DD:CD (left panel) and E1:C0 (right panel) for position orientations in 45° increments between 0° and 315°. This reveals that several of the distributions for address DD:CD are fairly normally distributed, but many of the distributions for address E1:C0 are extremely left-skewed indicating that the majority of the signals are fairly high. Overall, the signals of the access point with MAC address E1:C0 have a higher strength than access point DD:CD.

|  |
| --- |
| **Figure 2. Distribution of Signal Strengths by Angle** |
|  |
|  |

In **Table 2**, we can further confirm this with the summary statistics of access points. The minimum, mean, and maximum signal strengths for address E1:C0 are higher than those for address DD:CD, while the measurement numbers and standard deviations are almost identical.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 2. Summary Statistics of the Access Points** | | | | | |
| Access Point | N | SD | Min | Max | Mean |
| 00:0f:a3:39:dd:cd | 888 | 3.5 | -87 | -60 | -71 |
| 00:0f:a3:39:e1:c0 | 879 | 3.3 | -78 | -48 | -55 |

**Topographical representation of signal strength for addresses DD:CD and E1:C0**

The signal strength for each access point at different angles reveals that the access points in question are in the relatively same location. There are available data for 7 access points, while there are only 6 access points based on the floor plan of the building (**Figure 1**), indicating the two access points (E1:C0 and DD:CD) should be close. In order to address this contradiction, we will run a *k*-NN analysis to determine which MAC address data provides the most accurate location prediction.

In **Figure 3**, we examine the contour maps created at each angle for determining the location of the two access points E1:C0 or DD:CD. The signal strength fields for E1:C0 (B, right panel) show more significant variations than corresponding DD:CD fields (A, left panel), suggesting both access points might not be on the same floor.

|  |  |
| --- | --- |
| **Figure 3. Signal Strength by Angle for (A) DD:CD and (B) E1:C0** | |
|  |  |
| **A:** DD:CD | **B:** E1:C0 |

**Un-weighted *k*-NN method to predict location**

For our first approach, we will find the *k* closest training points in the signal strength domain where *k* is larger than 1 and estimate the new observation’s position by an aggregate of the positions of the *k* training points [2]. In this case study, we run a *k*-NN model to predict the locations of the measurements for three scenarios using the online dataset (whether to keep access point E1:C0, DD:CD, or both).

In **Figure 4,** the 11-fold cross validation results for three scenarios are shown. The pink line shows the sum of squared errors (SSE) for excluding access point DD:CD is 1,150, nearly matching the result of 1,268 in the textbook, both with the number of neighbors *k*=5 [2]. The slight difference in SSE is most likely due to the unspecified random seed in textbook.

|  |
| --- |
| **Figure 4. The Sum of Squared Errors for Different MAC Address Combinations.** |

The blue line reveals that excluding the access point E1:C0 performed the best due to the fact that this model provides the lowest SSE of 890 and requires the fewest neighbors (number of neighbors *k*=4). This gives evidence that by removing MAC address E1:C0 while keeping address DD:CD can improve the overall location prediction accuracy. This result disagrees with the result Nolan and Lang arrived at which stated one should keep MAC address E1:C0 and remove DD:CD.

The black line shows the SSE for using the data from both DD:CD and E1:C0. This has the highest squared error (SSE=1,317) and requires the most neighbors (number of neighbors *k*=6). In other words, including both E1:C0 and CC:CD does not improve prediction accuracy.

In order to get the most accurate location prediction, data yielded from MAC address E1:C0 should be removed and that from MAC address DD:CD should be kept in the dataset.

**Weighted versus un-weighted k-NN method to predict location**

The un-weighted *k*-NN approach described above using ordinary means is a good approach to determine the unknown location of devices. However, an alternative approach using a weighted *k*-NN can potentially improve the location prediction accuracy. In this approach, weight is placed on signal strengths based on the distance between the device and the access point. Higher weights are placed on stronger signal strengths, allowing for devices closer to access points to have a stronger contribution to the model than those that are further away [2].

In **Figure 5,** we test this alternative approach using the same random number seed to determine whether there are improvements in the two scenarios. The pink line shows the SSE is consistent with using ordinary means in textbook and **Figure 4** (SSE = 1,150) [2]. This result is almost identical with the SSE using the weighted means as shown in red line.

While the weighted model outperforms the unweighted model when one increases the number of neighbors, for a lower number of neighbors the two perform virtually the same. This gives evidence that the accuracy of location prediction is not improved upon by using a weighted *k*-NN model as opposed to an unweighted model for a small number of neighbors (k≤5).

|  |
| --- |
| **Figure 5. The Sum of Squared Errors for Weighted vs. Unweighted k-NN** |
|  |

## **Conclusions**

**MAC address selection**

From our study, access points excluding MAC address E1:C0 performed the best in overall location prediction accuracy. The use of access point DD:CD for best performance suggests the MAC address was incorrectly matched to the access point E1:C0 on the study floor, or slightly further away in the z-direction from up or down one floor. These possibilities are worthy of further analysis. In conclusion, MAC address DD:CD (not E1:C0) should be used for RTLS because of its best location prediction. This conclusion does not support keeping MAC address E1:C0 and removing DD:CD as handled in textbook [2].

In addition, using both MAC addresses (DD:CD and E1:C0) simultaneously did not improve the predictions produced by our location system. In fact, this scenario performed the worst of the three combinations tested.

**Unweighted vs. weighted *k*-NN**

The location prediction accuracy is not improved upon by using a weighted *k*-NN model as opposed to an unweighted model for 5 or fewer neighbors. This is most likely due to the fact that not all access points on the grid have similar signal profiles. In further studies, we could also consider the use of different metrics [5]. For example, we could use medians rather than averages when combining neighbors, especially for skewed data (e.g. E1:C0) for prediction.

**Applications of RTLS technologies**

The use and importance of RTLS goes beyond the application presented in this analysis. The ability to predict the location of hand-held devices allows for companies to track devices full of potentially confidential data, saving the company money and mitigating the risk of losing data. This technology can also be used to track people, ensuring efficiency and safety. These smart buildings are just one aspect of the continued development of smart cities. On a broader scale, RTLS systems that exist within buildings can share information with the city to ensure safety and quality for citizens. For example, access points in buildings such as the one used for this study could share location information with first responders and city officials in the case of a break-in, shooting, fire, or medical emergency [6]. The ability to predict the real-time location of unknown objects or people can have immense impact on the quality of life for citizens and the safety and proficiency of businesses.

## **References**

1. “What Is a Real Time Location System (RTLS)?” Sonitor, www.sonitor.com/blog-1/what-is-rtls-real-time-location-system.
2. Nolan, D. and Temple Lang, D. (2015). Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving. Boca Raton, FL: CRC Press (Chapter 1).
3. “Three Core Benefits of RTLS Technologies.” AiRISTA Flow, 11 Aug. 2016, www.airistaflow.com/3-core-benefits-of-rtls-technologies/.
4. Srivastava, Tavish. “Introduction to KNN, K-Nearest Neighbors : Simplified.” Analytics Vidhya, 27 Mar. 2018, [www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/](http://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/).
5. Science.gov, www.science.gov/topicpages/k/k-nearest+neighbor+graphs.
6. Journal, RFID. “A Look Into 2019: Smart Cities and Smart Buildings.” The History of RFID Technology - 2005-01-16 - Page 1 - RFID Journal, [www.rfidjournal.com/articles/view?18136](http://www.rfidjournal.com/articles/view?18136).

## **Appendix - R code**

# R codes for Real-Time Location System Case Study

# -----------------------------------------------------------------------

# **Part 1**: **Functions** for Data File (online.final.trace.txt and offline.final.trace).txt

# -----------------------------------------------------------------------

### processLine() to process each row in the input file.

processLine = function(x)

{

tokens = strsplit(x, "[;=,]")[[1]]

if (length(tokens) == 10)

return(NULL)

tmp = matrix(tokens[ - (1:10) ], , 4, byrow = TRUE)

cbind(matrix(tokens[c(2, 4, 6:8, 10)], nrow(tmp), ncol = 6,

byrow = TRUE), tmp)

}

### roundOrientation() to create the rounded angles

roundOrientation = function(angles){

refs = seq(0, by = 45, length = 9)

q = sapply(angles, function(o) which.min(abs(o - refs)))

c(refs[1:8], 0)[q]

}

### readData() to read data files

readData = function(

filename = './offline.final.trace.txt',

subMacs = c("00:0f:a3:39:e1:c0", "00:0f:a3:39:dd:cd", "00:14:bf:b1:97:8a",

"00:14:bf:3b:c7:c6", "00:14:bf:b1:97:90", "00:14:bf:b1:97:8d",

"00:14:bf:b1:97:81")

)

{

txt = readLines(filename)

lines = txt[ substr(txt, 1, 1) != "#" ]

tmp = lapply(lines, processLine)

offline = as.data.frame(do.call("rbind", tmp), stringsAsFactors= FALSE)

names(offline) = c("time", "scanMac","posX", "posY", "posZ", "orientation","mac", "signal", "channel", "type")

offline = offline[ offline$mac %in% subMacs, ]

# convert numeric values

numVars = c("time", "posX", "posY", "orientation", "signal")

offline[ numVars ] = lapply(offline[ numVars ], as.numeric)

# convert time to POSIX

offline$rawTime = offline$time

offline$time = offline$time/1000

class(offline$time) = c("POSIXt", "POSIXct")

# round orientations to nearest 45

offline$angle = roundOrientation(offline$orientation)

# Create a special factor that contains all of the unique combinations # of the observed (x,y) pairs for the locations

offline$posXY = paste(offline$posX, offline$posY, sep = "-")

return(offline)

}

### surfaceSS() has 3 arguments: data for the offline summary data frame, and mac and angle,

### which supply the MAC address and angle to select the subset of the data that we want smoothed and plotted.

### We call surfaceSS() with a couple of MAC addresses and angles to compare them.

surfaceSS = function(data, mac, angle = 45)

{

# extract subset of the data for specified MAC and angle:

oneAPAngle = data[ data$mac == mac & data$angle == angle, ]

# Fit a thin plate spline (TPS) surface to the data:

smoothSS = Tps(oneAPAngle[, c("posX","posY")], oneAPAngle$avgSignal)

# Evaluate fitted model on a 2D grid so we can plot it:

vizSmooth = predictSurface(smoothSS)

# And... plot:

plot.surface(vizSmooth, type = "C", xlab = "", ylab = "", xaxt = "n", yaxt = "n")

# Add the measurement points

points(oneAPAngle$posX, oneAPAngle$posY, pch=19, cex = 0.5)

# Annotate the plot

text(32,12,paste(angle,intToUtf8(176),sep=''), adj=c(1,0.5))

text(32,1,toupper(mac), adj=c(1,0.5))

}

### reshapeSS() to aggregate the signal strengths from these angles and create a data structure

reshapeSS = function(

data, varSignal = "signal",

keepVars = c("posXY", "posX","posY"), sampleAngle = FALSE,

refs = seq(0, 315, by = 45))

{

nCol = length(unique(data$mac))

byLocation = with(data, by(data, list(posXY),

function(x) {

if (sampleAngle) {

x = x[x$angle == sample(refs, size = 1), ]}

ans = x[1, keepVars]

avgSS = tapply(x[ , varSignal ], x$mac, mean)

y = matrix(avgSS, nrow = 1, ncol = nCol,

dimnames = list(ans$posXY, names(avgSS)))

cbind(ans, y)

}))

newDataSS = do.call("rbind", byLocation)

return(newDataSS)

}

### selectTrain() function has 3 parameters:

### angleNewObs, the angle of the new observation;

### signals, the training data, i.e., data in the format of offlineSummary;

### and m, the number of angles to include from signals.

### The function returns a data frame that matches trainSS.

selectTrain = function(angleNewObs, signals = NULL, m = 1)

{

# m is the number of angles to keep between 1 and 5 refs = seq(0, by = 45, length = 8)

nearestAngle = roundOrientation(angleNewObs)

if (m %% 2 == 1)

angles = seq(-45 \* (m - 1) /2, 45 \* (m - 1) /2, length = m)

else {

m=m+1

angles = seq(-45 \* (m - 1) /2, 45 \* (m - 1) /2, length = m)

if (sign(angleNewObs - nearestAngle) > -1)

angles = angles[ -1 ]

else

angles = angles[ -m ]

}

angles = angles + nearestAngle

angles[angles < 0] = angles[ angles < 0 ] + 360

angles[angles > 360] = angles[ angles > 360 ] - 360

angles = sort(angles)

offlineSubset = signals[ signals$angle %in% angles, ]

reshapeSS(offlineSubset, varSignal = "avgSignal")

}

### findNN() function to calculate the distance from the new point to all observations in the training set

findNN = function(newSignal, trainSubset)

{

# Signal differences at THIS location between training and observed, for each access point

diffs = apply(trainSubset[,4:9], 1, function(x) x - newSignal)

dists = apply(diffs, 2, function(x) sqrt(sum(x^2)))

closest = order(dists)

return(trainSubset[closest, 1:3 ])

}

### predXY() to formalize and make predictions for all of the test data

predXY = function(newSignals, newAngles, trainData, numAngles=1, k=3)

{

closeXY = list(length = nrow(newSignals)) # an empty list

for (i in 1:nrow(newSignals)) {

trainSS = selectTrain(newAngles[i], trainData, m = numAngles)

closeXY[[i]] = findNN(newSignal = as.numeric(newSignals[i,]), trainSS)

}

estXY = lapply(closeXY, function(x) sapply(x[,2:3], function(x) mean(x[1:k])))

estXY = do.call("rbind", estXY)

return(estXY)

}

### findNNalt(newSignal,trainSubset)

findNNalt = function(newSignal, trainSubset)

{

# Signal differences at THIS location between training and observed, for each access point

diffs = apply(trainSubset[,4:9], 1, function(x) x - newSignal)

dists = apply(diffs, 2, function(x) sqrt(sum(x^2)))

closest = order(dists)

retSS = trainSubset[closest, 1:3]

retSS$dist = dists[closest]

retSS$wt = 1/retSS$dist

return(retSS)

}

### predXYalt(newSignals,newAngles,trainData,numAngles,k) for weighted k-NN

predXYalt = function(newSignals, newAngles, trainData, numAngles=1, k=3){

closeXY = list(length = nrow(newSignals))

for (i in 1:nrow(newSignals)) {

trainSS = selectTrain(newAngles[i], trainData, m = numAngles)

closeXY[[i]] = findNNalt(newSignal = as.numeric(newSignals[i,]), trainSS)

}

estXY = lapply(closeXY,function(a)sapply(a[,2:3],function(b)weighted.mean(b[1:k],a[1:k,5])))

estXY = do.call("rbind", estXY)

return(estXY)

}

### calcError(estXY,actualXY) to find the sum of squared errors

calcError = function(estXY, actualXY)

{

sum( rowSums( (estXY - actualXY)^2) )

}

# -----------------------------------------------------------------------

# **Part 2: Exploratory Data Analysis (EDA)**

# -----------------------------------------------------------------------

options(digits = 2)

library(dplyr)

library(lattice)

library(fields)

allMacs = c("00:0f:a3:39:dd:cd", "00:0f:a3:39:e1:c0", "00:14:bf:3b:c7:c6", "00:14:bf:b1:97:81",

"00:14:bf:b1:97:8a", "00:14:bf:b1:97:8d", "00:14:bf:b1:97:90")

offline = readData(subMacs=allMacs)

# create a list of data frames for every combination of (x,y), angle, and access point

byLocAngleAP = with(offline, by(offline, list(posXY, angle, mac), function(x) x))

# calculate summary statistics on each of these data frames

signalSummary = lapply(byLocAngleAP,

function(oneLoc) {

ans = oneLoc[1, ]

ans$medSignal = median(oneLoc$signal)

ans$avgSignal = mean(oneLoc$signal)

ans$num = length(oneLoc$signal)

ans$sdSignal = sd(oneLoc$signal)

ans$iqrSignal = IQR(oneLoc$signal)

ans

})

offlineSummary = do.call("rbind", signalSummary)

# filter to only the two interested MACs: dd:cd and e1:c0

off.filt <- filter(offline, mac=='00:0f:a3:39:dd:cd' | mac=='00:0f:a3:39:e1:c0', posX==2, posY==12)

# signal strength by angle for these two access points: dd:cd and e1:c0

png(filename="./f21.png", width = 12, height = 6, units="in", res=300, pointsize=1/600)

bwplot(signal ~ factor(angle) | mac, data=off.filt, layout=c(2,1))

dev.off()

# reproduce Fig 1.8: plot standard deviation of signal strength (y) by mean signal strength (x)

png(filename="./f22.png", width = 12, height = 6, units="in", res=300, pointsize=1/600)

opar = par(mar = c(3.1, 3, 1, 1))

bwplot(sdSignal ~ cut(avgSignal, breaks=seq(-90,-30,by=5)),

data = offlineSummary,

subset = mac != "00:0f:a3:39:dd:cd",

xlab = "Mean Signal", ylab = "SD Signal")

par(opar)

dev.off()

# Show summary statistics for these two MACs

off.filt %>% group\_by(mac) %>% summarize(n=n(), sd=sd(signal), min=min(signal), max=max(signal), mean=mean(signal))

# side-by-side visual comparisons with signal strength field plots at 8 angles for two access points: DD:CD and E1:C0

for ( m in allMacs ){

opar = par(mfrow = c(4,2), mar = rep(2, 4))

# make 4 calls to our surfaceSS() function using mapply()

mapply(surfaceSS, mac = rep(m,8),

angle = seq(0, 315, by=45),

data = list(data = offlineSummary))

# reset the plotting parameters

par(opar)

}

# -----------------------------------------------------------------------

# **Part 3: line plot for the cross validated selection of k** in 3 scenarios:

## 6 access points (not including DD:CD)

## 6 access points (not including E1:E0)

## 7 access points (including DD:Cd and E1:C0)

# -----------------------------------------------------------------------

allMacs = c("00:0f:a3:39:dd:cd", "00:0f:a3:39:e1:c0", "00:14:bf:3b:c7:c6", "00:14:bf:b1:97:81",

"00:14:bf:b1:97:8a", "00:14:bf:b1:97:8d", "00:14:bf:b1:97:90")

keepVars = c("posXY", "posX", "posY", "orientation", "angle")

set.seed(12345)

# printf <- function(...) invisible(cat(sprintf(...)))

### 1. reproduce predictions from the text (not including access point DD:CD)

# preparing the Test Data

macs1 = allMacs[2:7] # as in the text, without access point DD:CD

offline = readData(subMacs=macs1)

online = readData("./online.final.trace.txt", subMacs=macs1)

# create a list of data frames for every combination of (x,y), angle, and access point

byLocAngleAP = with(offline, by(offline, list(posXY, angle, mac), function(x) x))

# calculate summary statistics on each of these data frames

signalSummary = lapply(byLocAngleAP,

function(oneLoc) {

ans = oneLoc[1, ]

ans$medSignal = median(oneLoc$signal)

ans$avgSignal = mean(oneLoc$signal)

ans$num = length(oneLoc$signal)

ans$sdSignal = sd(oneLoc$signal)

ans$iqrSignal = IQR(oneLoc$signal)

ans

})

offlineSummary = do.call("rbind", signalSummary)

# as with the offline data, we create a unique location identifier with

online$posXY = paste(online$posX, online$posY, sep = "-")

byLoc = with(online,

by(online, list(posXY),

function(x) {

ans = x[1, keepVars]

avgSS = tapply(x$signal, x$mac, mean)

y = matrix(avgSS, nrow=1, ncol=6, dimnames=list(ans$posXY, names(avgSS)))

cbind(ans, y)

}))

onlineSummary = do.call("rbind", byLoc)

# cross validation with v=11 for the 166 locations (6 MAC addresses and 8 orientations)

v = 11

permuteLocs = sample(unique(offlineSummary$posXY))

permuteLocs = matrix(permuteLocs, ncol=v, nrow=floor(length(permuteLocs)/v))

onlineCVSummary = reshapeSS(offline, keepVars=keepVars, sampleAngle=TRUE)

# find the k-NN estimates for each fold, here set k = 20

K = 20

err = rep(0, K)

for (j in 1:v)

{

onlineFold = subset(onlineCVSummary, posXY %in% permuteLocs[,j])

offlineFold = subset(offlineSummary, posXY %in% permuteLocs[,-j])

actualFold = onlineFold[ , c("posX", "posY")]

for ( k in 1:K )

{

estFold = predXY(newSignals = onlineFold[,6:11], newAngles=onlineFold[,4], offlineFold, numAngles=3, k=k)

thisError = calcError(estFold, actualFold)

# printf("j=%d, k=%d, err=%f\n", j, k, thisError)

err[k] = err[k] + thisError

} }

# put err in a doggie bag for later err.txt = err

err.txt = err

### 2. Use the DD:CD access point instead of E1:C0

# preparing the Test Data

macs2 = allMacs[c(1,3:7)] # without access point E1:C0

offline = readData(subMacs=macs2)

online = readData("./online.final.trace.txt", subMacs=macs2)

# Create a list of data frames for every combination of (x,y), angle, and access point

byLocAngleAP = with(offline, by(offline, list(posXY, angle, mac), function(x) x))

# calculate summary statistics on each of these data frames

signalSummary = lapply(byLocAngleAP,

function(oneLoc) {

ans = oneLoc[1, ]

ans$medSignal = median(oneLoc$signal)

ans$avgSignal = mean(oneLoc$signal)

ans$num = length(oneLoc$signal)

ans$sdSignal = sd(oneLoc$signal)

ans$iqrSignal = IQR(oneLoc$signal)

ans

})

offlineSummary = do.call("rbind", signalSummary)

# as with the offline data, we create a unique location identifier with

online$posXY = paste(online$posX, online$posY, sep = "-")

byLoc = with(online,

by(online, list(posXY),

function(x) {

ans = x[1, keepVars]

avgSS = tapply(x$signal, x$mac, mean)

y = matrix(avgSS, nrow=1, ncol=6, dimnames=list(ans$posXY, names(avgSS)))

cbind(ans, y)

}))

onlineSummary = do.call("rbind", byLoc)

# cross validation with v=11 for the 166 locations (6 MAC addresses and 8 orientations)

v = 11

permuteLocs = sample(unique(offlineSummary$posXY))

permuteLocs = matrix(permuteLocs, ncol=v, nrow=floor(length(permuteLocs)/v))

onlineCVSummary = reshapeSS(offline, keepVars=keepVars, sampleAngle=TRUE)

# find the k-NN estimates for each fold, here set k = 20

K = 20

err = rep(0, K)

for (j in 1:v){

onlineFold = subset(onlineCVSummary, posXY %in% permuteLocs[,j])

offlineFold = subset(offlineSummary, posXY %in% permuteLocs[,-j])

actualFold = onlineFold[ , c("posX", "posY")]

for ( k in 1:K ){

estFold = predXY(newSignals = onlineFold[,6:11], newAngles = onlineFold[,4], offlineFold, numAngles=3, k=k)

thisError = calcError(estFold, actualFold)

# printf("j=%d, k=%d, err=%f\n", j, k, thisError)

err[k] = err[k] + thisError

} }

# put err in a doggie bag for later err.txt = err

err.ddcd = err

### 3. Use all 7 access points

# preparing the Test Data

macs3 = allMacs # with 7access points including E1:C0 and DD:CD

offline = readData(subMacs=macs3)

online = readData("./online.final.trace.txt", subMacs=macs3)

# Create a list of data frames for every combination of (x,y), angle, and access point

byLocAngleAP = with(offline, by(offline, list(posXY, angle, mac), function(x) x))

# calculate summary statistics on each of these data frames

signalSummary = lapply(byLocAngleAP,

function(oneLoc) {

ans = oneLoc[1, ]

ans$medSignal = median(oneLoc$signal)

ans$avgSignal = mean(oneLoc$signal)

ans$num = length(oneLoc$signal)

ans$sdSignal = sd(oneLoc$signal)

ans$iqrSignal = IQR(oneLoc$signal)

ans

})

offlineSummary = do.call("rbind", signalSummary)

# as with the offline data, we create a unique location identifier with

online$posXY = paste(online$posX, online$posY, sep = "-")

byLoc = with(online,

by(online, list(posXY),

function(x) {

ans = x[1, keepVars]

avgSS = tapply(x$signal, x$mac, mean)

y = matrix(avgSS, nrow=1, ncol=7, dimnames=list(ans$posXY, names(avgSS)))

cbind(ans, y)

}))

onlineSummary = do.call("rbind", byLoc)

# cross validation with v=11 for the 166 locations (6 MAC addresses and 8 orientations)

v = 11

permuteLocs = sample(unique(offlineSummary$posXY))

permuteLocs = matrix(permuteLocs, ncol=v, nrow=floor(length(permuteLocs)/v))

onlineCVSummary = reshapeSS(offline, keepVars=keepVars, sampleAngle=TRUE)

# find the k-NN estimates for each fold, here set k = 20

K = 20

err = rep(0, K)

for (j in 1:v)

{

onlineFold = subset(onlineCVSummary, posXY %in% permuteLocs[,j])

offlineFold = subset(offlineSummary, posXY %in% permuteLocs[,-j])

actualFold = onlineFold[ , c("posX", "posY")]

for ( k in 1:K )

{

estFold = predXY(newSignals = onlineFold[,6:11], newAngles=onlineFold[,4], offlineFold, numAngles=3, k=k)

thisError = calcError(estFold, actualFold)

# printf("j=%d, k=%d, err=%f\n", j, k, thisError)

err[k] = err[k] + thisError

} }

# put err in a doggie bag for later err.txt = err

err.all = err

# cross validated selection of k for all 3 scenarios

png(file="./f4.png",width=600, height=400)

opar = par(mar = c(4, 4, 1, 1))

err.combined = c(err.txt, err.ddcd, err.all)

y\_limits = c(round(min(err.combined),-2)-100, round(max(err.combined))+100)

# plot errors for 1st run from the textbook (not including access point DD:CD)

rmseMin = min(err.txt)

kMin = which(err.txt == rmseMin)[1]

my.col = 'pink'

plot(y=err.txt, x = (1:K), col=my.col, type = "l", lwd= 2, ylim = y\_limits,

xlab = "Number of Neighbors",

ylab = "Sum of Square Errors")

segments(x0=0, x1=kMin, y0=rmseMin, col=my.col, lty=2, lwd=2)

segments(x0 = kMin, x1=kMin, y0=y\_limits[1], y1=rmseMin, col=my.col, lty=2, lwd=2)

mtext(kMin, side=1, line=1, at=kMin, col=my.col)

text(x=2, y=rmseMin+40, label=as.character(round(rmseMin)), col=my.col)

# plot errors for 2nd run (not including access point E1:C0)

rmseMin = min(err.ddcd)

kMin = which(err.ddcd == rmseMin)[1]

my.col = 'blue'

lines(err.ddcd, col=my.col)

segments(x0=0, x1=kMin, y0=rmseMin, col=my.col, lty=2, lwd=2)

segments(x0 = kMin, x1=kMin, y0=y\_limits[1], y1=rmseMin, col=my.col, lty=2, lwd=2)

mtext(kMin, side=1, line=1, at=kMin, col=my.col)

text(x=2, y=rmseMin+40, label=as.character(round(rmseMin)), col=my.col)

# plot errors for 3nd run (including access points DD:CD and E1:C0)

rmseMin = min(err.all)

kMin = which(err.all == rmseMin)[1]

my.col = 'black'

lines(err.all, col=my.col)

segments(x0=0, x1=kMin, y0=rmseMin, col=my.col, lty=2, lwd=2)

segments(x0 = kMin, x1=kMin, y0=y\_limits[1], y1=rmseMin, col=my.col, lty=2, lwd=2)

mtext(kMin, side=1, line=1, at=kMin, col=my.col)

text(x=2, y=rmseMin+40, label=as.character(round(rmseMin)), col=my.col)

legend(9, 2100, c('6 access points (not include DD:CD)',

'6 access points (not include E1:C0)',

'7 access points (include DD:CD and E1:C0)'),

lty=c(1,1,1), col=c('pink', 'blue', 'black'))

par(opar)

dev.off()

# -----------------------------------------------------------------------

# **Part 4: implement the alternative prediction method with weight**

# -----------------------------------------------------------------------

allMacs = c("00:0f:a3:39:dd:cd", "00:0f:a3:39:e1:c0", "00:14:bf:3b:c7:c6", "00:14:bf:b1:97:81",

"00:14:bf:b1:97:8a", "00:14:bf:b1:97:8d", "00:14:bf:b1:97:90")

keepVars = c("posXY", "posX", "posY", "orientation", "angle")

set.seed(12345)

# 1. reproduce predictions from the text (not including access point DD:CD)

macs\_text = allMacs[2:7] # as in the text, without access point DD:CD

offline = readData(subMacs=macs\_text)

online = readData("./online.final.trace.txt", subMacs=macs\_text)

# create a list of data frames for every combination of (x,y), angle, and access point

byLocAngleAP = with(offline, by(offline, list(posXY, angle, mac), function(x) x))

# calculate summary statistics on each of these data frames

signalSummary = lapply(byLocAngleAP,

function(oneLoc) {

ans = oneLoc[1, ]

ans$medSignal = median(oneLoc$signal)

ans$avgSignal = mean(oneLoc$signal)

ans$num = length(oneLoc$signal)

ans$sdSignal = sd(oneLoc$signal)

ans$iqrSignal = IQR(oneLoc$signal)

ans

})

offlineSummary = do.call("rbind", signalSummary)

# as with the offline data, we create a unique location identifier with

online$posXY = paste(online$posX, online$posY, sep = "-")

byLoc = with(online,

by(online, list(posXY),

function(x) {

ans = x[1, keepVars]

avgSS = tapply(x$signal, x$mac, mean)

y = matrix(avgSS, nrow=1, ncol=6, dimnames=list(ans$posXY, names(avgSS)))

cbind(ans, y)

}))

onlineSummary = do.call("rbind", byLoc)

# cross validation with v=11

v = 11

permuteLocs = sample(unique(offlineSummary$posXY))

permuteLocs = matrix(permuteLocs, ncol=v, nrow=floor(length(permuteLocs)/v))

onlineCVSummary = reshapeSS(offline, keepVars=keepVars, sampleAngle=TRUE)

# try K-nearest neighbors prediction for values of k from 1 to 20

K = 20

err = rep(0, K)

for (j in 1:v)

{

onlineFold = subset(onlineCVSummary, posXY %in% permuteLocs[,j])

offlineFold = subset(offlineSummary, posXY %in% permuteLocs[,-j])

actualFold = onlineFold[ , c("posX", "posY")]

for ( k in 1:K )

{

estFold = predXY(newSignals = onlineFold[,6:11], newAngles=onlineFold[,4], offlineFold, numAngles=3, k=k)

err[k] = err[k] + calcError(estFold, actualFold) }

}

# Put err in a doggie bag for later err.txt = err

err.txt = err

# 2. same date, but use weighted nearest neighbors for predicting scenario of not including access point DD:CD

# try K-nearest neighbors prediction for values of k from 1 to 20

K = 20

err = rep(0, K)

for (j in 1:v)

{

onlineFold = subset(onlineCVSummary, posXY %in% permuteLocs[,j])

offlineFold = subset(offlineSummary, posXY %in% permuteLocs[,-j])

actualFold = onlineFold[ , c("posX", "posY")]

for ( k in 1:K )

{

estFoldalt = predXYalt(newSignals = onlineFold[,6:11], newAngles=onlineFold[,4], offlineFold, numAngles=3, k=k)

err[k] = err[k] + calcError(estFoldalt, actualFold) }

}

# Put err in a doggie bag for later err.alt = err

err.alt = err

# line plot for cross validated selection of k for all 2 scenarios (unweighted and weighted)

png(file="./f5.png",width=600, height=400)

opar = par(mar = c(4, 4, 1, 1))

err.combined = c(err.txt, err.alt)

y\_limits = c(round(min(err.combined),-2)-100, round(max(err.combined))+100)

# plot errors for 1st run from unweighted scenario of not including access point DD:CD

rmseMin = min(err.txt)

kMin = which(err.txt == rmseMin)[1]

my.col = 'pink'

plot(y=err.txt, x = (1:K), col=my.col, type = "l", lwd= 2, ylim = y\_limits,

xlab = "Number of Neighbors",

ylab = "Sum of Square Errors")

segments(x0=0, x1=kMin, y0=rmseMin, col=my.col, lty=2, lwd=2)

segments(x0 = kMin, x1=kMin, y0=y\_limits[1], y1=rmseMin, col=my.col, lty=2, lwd=2)

mtext(kMin, side=1, line=1, at=kMin, col=my.col)

text(x=2, y=rmseMin+40, label=as.character(round(rmseMin)), col=my.col)

# plot errors for 2nd run from weighted scenario of ncluding access point DD:CD

rmseMin = min(err.alt)

kMin = which(err.alt == rmseMin)[1]

my.col = 'magenta'

lines(err.alt, col=my.col)

segments(x0=0, x1=kMin, y0=rmseMin, col=my.col, lty=2, lwd=2)

segments(x0 = kMin, x1=kMin, y0=y\_limits[1], y1=rmseMin, col=my.col, lty=2, lwd=2)

mtext(kMin, side=1, line=1, at=kMin, col=my.col)

text(x=2, y=rmseMin+40, label=as.character(round(rmseMin)), col=my.col)

legend(7, 1950, c('unweighted, 6 access points (not include DD:CD)',

'weighted, 6 access points (not include DD:CD)'),

lty=c(1,1), col=c('pink', 'magenta'))

par(opar)

dev.off()

# **- The End -**