Assignment 2 : Predict 454

Rahul Sangole

2018-10-24

The Modeling Problem & Data Prep

The objective of this exercise is to develop an internal positioning system, which will try to predict the location of a mobile user based off of the relative strengths of the Wi-Fi signal measured on the mobile device as it connects to 6 Wi-Fi routers placed in the building.

The input data is a flat file in a messy format. It contains intersperced comment lines which start with #, semi-colon separated key-value pairs, with some = separated key-value pairs containing further comma-separated list of values, one of the key-value having a varying number of values in the list. This level of messiness causes some consternation during data preparation, but can be quickly and elegantly solved by using the appropriate functions from `tidyverse`. Sample code for data prep is shown in the appendix.

Furthermore, while the objective of the problem is predict the location of the user based on the strengths of 6 routers, there are additional router mac addresses in the original data. These were identified based on the meta data for the routers, and appropriately filtered.

The positions of each of the six routers is identified from the diagram provided, joined with the training dataset, and a eucledian distance is calculated for each point.

The result of the data prep is a training data-frame which is (769,332 x 10), with the test dataset being (34,778 x 6). So, essentially, we have 769,332 observations to a build a model from, to predict the location of 34,778 datapoints. The data are manipulated so that each row in the table corresponds to a one observation for one location, for one angle, for one of the Wi-Fi routers.

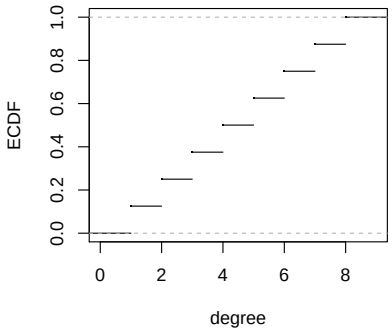


Figure 1: ECDF

mac	strength	t	pos_x	pos_y	degree	pos_xy	mac_x	mac_y	dist
00:14:bf:b1:97:8a	-38	2006-02-11 07:31:58	0	0	0	0_0	2.5	-0.8	2.624881
00:14:bf:b1:97:90	-56	2006-02-11 07:31:58	0	0	0	0_0	1.0	14.0	14.035669
00:0f:a3:39:e1:c0	-53	2006-02-11 07:31:58	0	0	0	0_0	7.5	6.3	9.794897
00:14:bf:b1:97:8d	-65	2006-02-11 07:31:58	0	0	0	0_0	33.5	9.3	34.766938
00:14:bf:b1:97:81	-65	2006-02-11 07:31:58	0	0	0	0_0	33.5	2.8	33.616811
00:14:bf:3b:c7:c6	-66	2006-02-11 07:31:58	0	0	0	0_0	12.8	-2.8	13.102671

The degree at which the mobile phone is held was varied at increments of 45 degrees. However, the data did show angles not equal to these increments, but quite close to them. One of the first data clean up activities was to adjust the `degrees` column to make the angles equal 45 degree increments. The ECDF plot shows the cumulative distribution post-correction.

Exploratory Data Analysis

The data are explored in various ways - from univariate to multivariate graphical methods. Two of the more interesting plots are shown here. The first is a marginal plot which shows the distributions of 4 key variables conditional on the 6 routers. Right away we can see that strength has very varying distributions for each router.

```
## Warning in formals(fun): argument is not a
## function
```

For each coordinate, the average strength can be calculated. I have done this for all the degrees combined. This strength is plotted on a raster plot which shows interesting distributions. The triangle shows the position of router 2, and clearly the strength of signals is the highest in the opposite corridor. Wi-fi doesn't travel well perpendicular to corridors, and we can see the sharp drop in strength as shown in black.

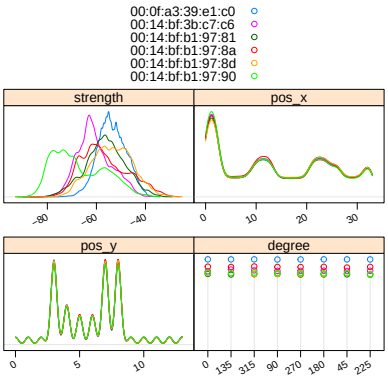
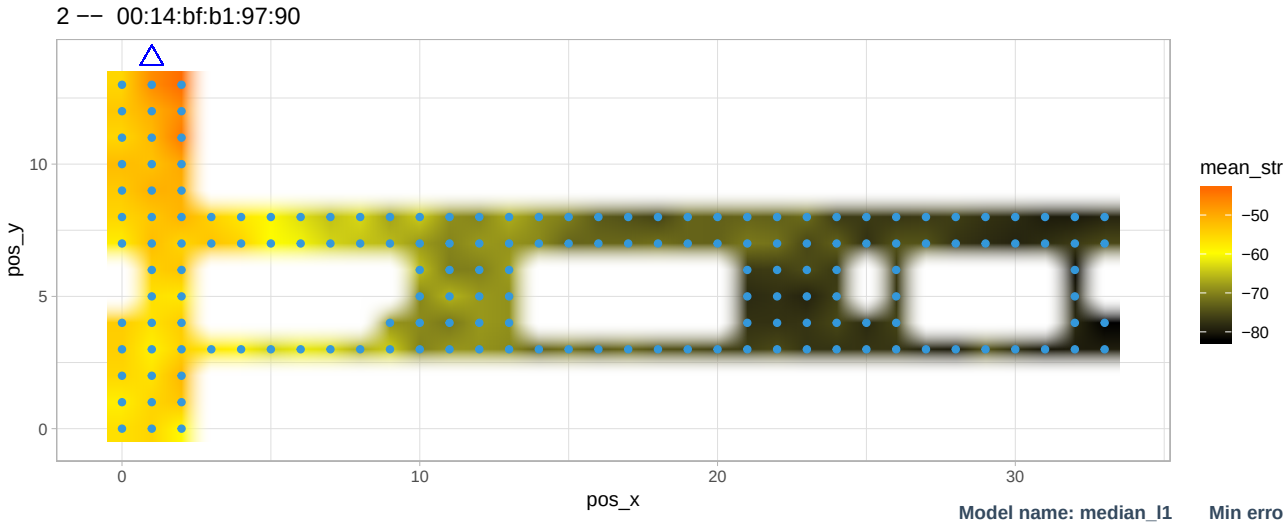


Figure 2: Margin plot



Model Comparison

The model built is a k-nearest neighbours model. In essence, the model tries to identify the "k" "closest" points in the training dataset to the point we wish to classify in the test dataset. "k" is selected by cross-validation. The definition of "closest" can be done in a few ways - eucledian distance, manhattan distance, weighted distances. Also, to process the top "k" choices, we can calculate the mean distances, or the median distances etc.

Here, I have developed a few models, and performed cross validation on all of them.

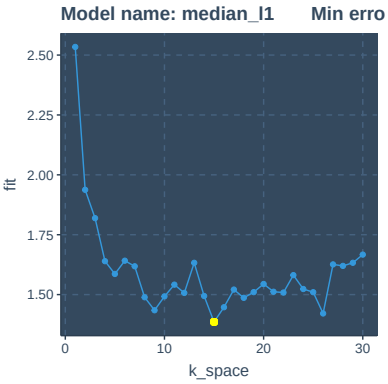


Figure 4: Search for k Median L1

- Median of distances, l1 distance
- Mean of distances, l1 distance
- Median of distances, eucledian distance
- Mean of distances, eucledian distance
- Mean of weighted distances, l1 distance
- Mean of weighted distances, eucledian distance

The functions to solve the models are custom developed and documented in the appendix. The plots to the right show the cross-validation searches for optimal value of k for two of the models. The performance for all the models is shown below. The best model is the median fit with the L1 distance, with a k value of 15 points, which results in an error of 1.38. (This error is calculated as the average error of the x_hat and y_hat position estimations.) The second best model is the median fit with the eucledian distance which results in an error of 1.59 for a k value of 6.

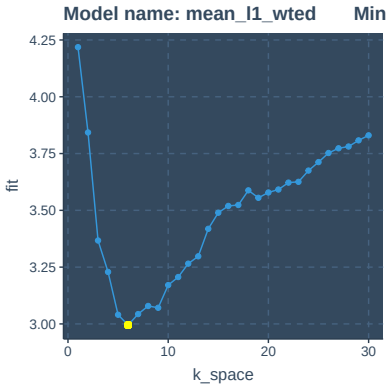


Figure 5: Search for k Mean Weighted L1

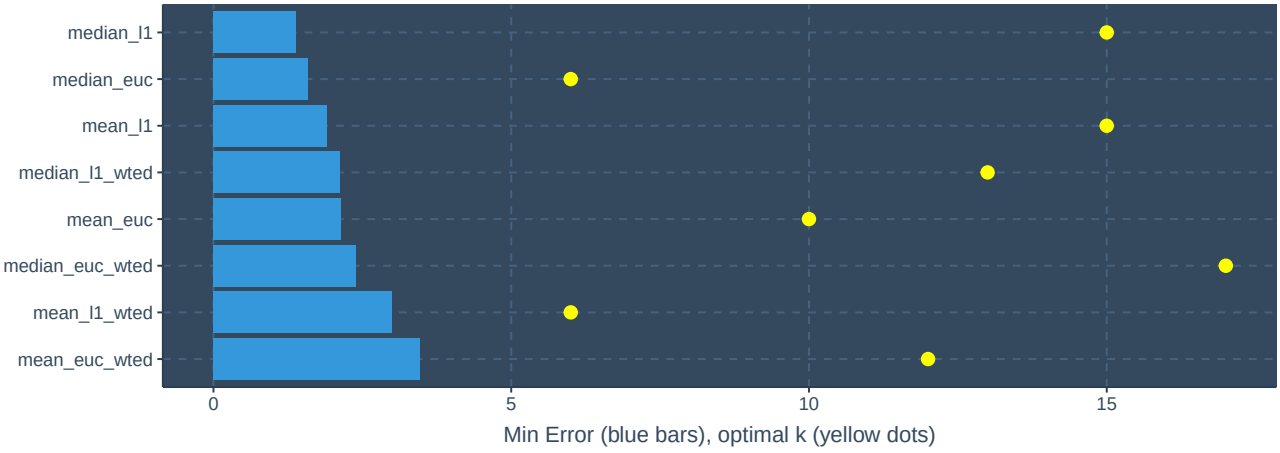


Figure 6: Comparison of the final models on train and test datasets

Selecting the best model, the following graph shows the comparison of the fitted values to the true values. Points in blue are for the y-location, while points in yellow are for the x-location. The model performs quite well in it's predictive power.

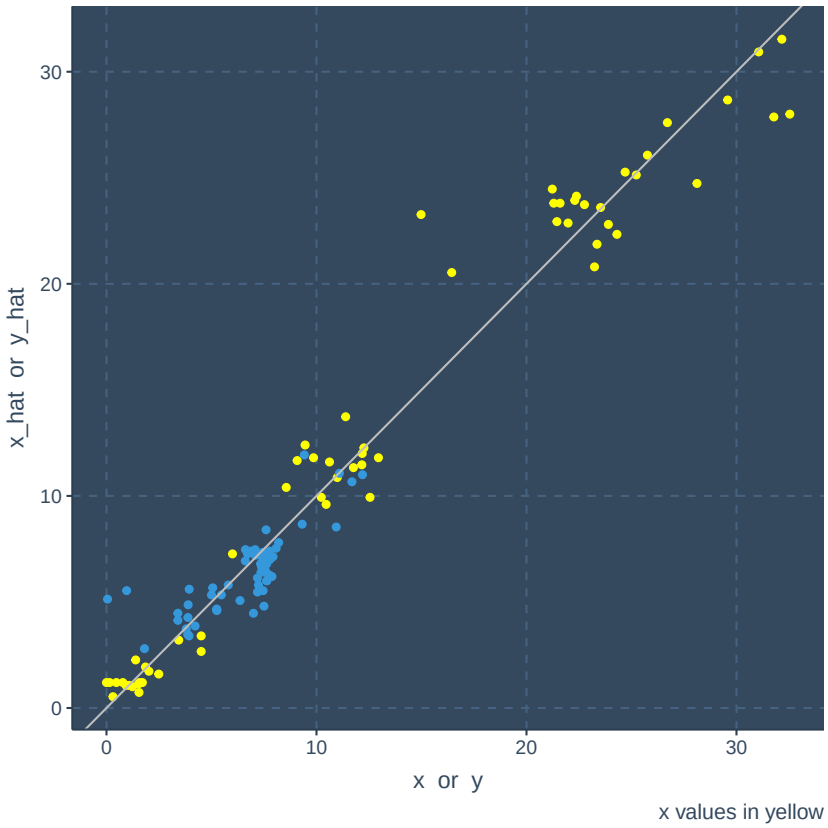


Figure 7: Prediction Plots

Appendix

Quick data ingestion using `tidyverse`

```
make_df <- function(i, x) {
    tibble(row_id = i, measurements = unlist(x)) %>%
        separate(measurements, sep = ",", into = c("mac",
            "freq", "mode")) %>% separate(col = "mac",
        sep = "=", into = c("mac", "strength"))
}
raw_lines <- read_lines("data/offline.final.trace.txt")
raw_lines <- raw_lines[!stringr::str_detect(raw_lines,
    pattern = "^#")]
raw_lines_list <- map(raw_lines, ~stringr::str_split(.x,
    pattern = ";", simplify = T))
mask <- map_int(raw_lines_list, ~length(.x)) <
    5
raw_lines_list <- raw_lines_list[!mask]
df <- tibble(t = map_chr(raw_lines_list, ~.x[[1,
    1]]), id = map_chr(raw_lines_list, ~.x[[1,
    2]]), pos = map_chr(raw_lines_list, ~.x[[1,
    3]]), degree = map_chr(raw_lines_list, ~.x[[1,
    4]])) %>% separate(col = "pos", sep = ",",
    into = c("pos_x", "pos_y", "pos_z")) %>% map_df(~stringr::str_remove_all(.x,
    pattern = "[a-z]*=")) %>% mutate(t = as.POSIXct(x = as.numeric(t)/1000,
    origin = "1970-01-01 UTC"), pos_x = as.numeric(pos_x),
    pos_y = as.numeric(pos_y), pos_z = as.numeric(pos_z),
    degree = as.numeric(degree)) %>% rownames_to_column("row_id")

names(raw_lines_list) <- df$row_id
measurements <- map(raw_lines_list, ~.x[-1:-4])
df_measurements <- seq_along(measurements) %>%
    map_df(~make_df(.x, measurements[.x]))
df_raw <- df_measurements %>% left_join(df)
```

knn Model Code

```
find_knn <- function(df_train, test_row, k, d_type) {
    train_dist_mat <- as.matrix(df_train[, -1:-2])
    test_strengths <- as.matrix(test_row[, -1:-2])
    distance <- numeric()
    switch(d_type, euc = {
```

```
        for (i in 1:nrow(train_dist_mat)) {
            distance <- c(distance, sqrt(sum((train_dist_mat[i,
                ] - test_strengths)^2)))
        }
    }, l1 = {
        for (i in 1:nrow(train_dist_mat)) {
            distance <- c(distance, (sum(abs(train_dist_mat[i,
                ] - test_strengths))))
        }
    })
    df_train %>% dplyr::mutate(distance = distance) %>%
        dplyr::select(pos_x, pos_y, degree, distance) %>%
        dplyr::arrange(distance) %>% head(k)
}
get_xy_hats <- function(df_train, test_row, k,
    d_type, wted) {
    if (wted) {
        nearest_neighbours <- find_knn_wted(df_train,
            test_row, k, d_type)
    } else {
        nearest_neighbours <- find_knn(df_train,
            test_row, k, d_type)
    }

    x_hat <- mean(nearest_neighbours$pos_x)
    y_hat <- mean(nearest_neighbours$pos_y)
    paste(x_hat, y_hat, sep = "_")
}
get_errors <- function(df_train, df_test, k, type,
    d_type, wted) {
    euc_dist <- character()
    for (i in 1:nrow(df_test)) {
        euc_dist <- c(euc_dist, get_xy_hats(df_train,
            df_test[i, ], k, d_type, wted))
    }
    df_test$euc_dist <- euc_dist
    df_test <- df_test %>% separate(euc_dist,
        sep = "_", into = c("x_hat", "y_hat"))
    df_test <- df_test %>% select(matches("[xy]")) %>%
        map_df(~as.numeric(.x)) %>% mutate(err = sqrt((x_hat -
        pos_x)^2 + (y_hat - pos_y)^2))
    if (type == "mean") {
        return(mean(df_test$err))
```

```
    }
    if (type == "median") {
        return(median(df_test$err))
    }
}
get_predictions <- function(df_train, df_test,
    k, d_type, wted) {
    distance <- character()
    for (i in 1:nrow(df_test)) {
        distance <- c(distance, get_xy_hats(df_train,
            df_test[i, ], k, d_type, wted))
    }
    df_test$distance <- distance
    df_test <- df_test %>% separate(distance,
        sep = "_", into = c("x_hat", "y_hat"))
    df_test %>% select(matches("[xy]")) %>% map_df(~as.numeric(.x)) %>%
        mutate(err = (sqrt((x_hat - pos_x)^2) +
            sqrt((y_hat - pos_y)^2))/2)
}
```

Solving the knn model to search for K

```
k_search_mean_l1 <- 1:30 %>% map_dbl(~get_errors(df_train,
    df_test, .x, "mean", "l1", FALSE))
k_search_mean_euc <- 1:30 %>% map_dbl(~get_errors(df_train,
    df_test, .x, "mean", "euc", FALSE))
```