Case Study 1: MSDS 7333 Spring 2020

Sean Kennedy

Sterling Beason

Emil Ramos

Souther Methodist University

Introduction

Knowing the location of business-critical machiney and/or workers is essential to optimize productivity for organizations that live and die by shipping times, workflow prioritization, deliverable timelines, and cost minimization. Real-time location systems (RTLS) have enabled some business to be in a position to monitor their assets through the production cycle.

In this case study, we are evaluating the use of RTLS over wifi for an organization. Specifically, our authors (Nolan and Lang) posit that by distributing various wifi-enabled RTLS devices across the facility to assets, one might be able to use clustering methods to ascertain the predicted position of those assets based on past behavior.

Objective: Using the OFFLINE data and two different clustering methods predict the location of the ONLINE data set.

Our two methods will be:

- weighted kNN
- unweighted kNN

Imports

Parameters

Determines which macIds to include in analysis. Not all macs have enough readings to be observed. The following list meets the minimum threhold established in the EDA section. It will be updated later in the analysis to consider a different group of macs in the creation of our kNN graph.

Code/Functions

Please view .ipynb and toggle the "Show Code" button below to view all code used to generate output

Data Dictionary

The columns from the raw OFFLINE dataset

Variable	Description
t	timestamp (ms) of scan
id	MAC address of scanning device
pos	comma separated position (x, y, z)
degree	orientation of scanning device
(mac)*	MAC address(es) of access points as key with comma separated value (signal, channel, type)

Show Code

Create DataFrame

Objective: Describe how you prepared the data.

df : training set. Not aggregated by angle. **Will be aggregated by XY-loc and mac across orientation angles on demand as needed**

df_online : testing set. Aggregated at outset, not touched until after cross validation.

mac addresses with fewer than 1,018,838 entries have been removed

All angles have been shifted to the nearest 45 degree interval (i.e all angles have been mapped to [0,45...135,180,..315,360]

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1018838 entries, 0 to 1181626
Data columns (total 12 columns):
time                  1018838 non-null object
scanMac               1018838 non-null object
posX                  1018838 non-null object
posY                  1018838 non-null object
posZ                  1018838 non-null object
orientation           1018838 non-null float64
mac                   1018838 non-null object
signal                1018838 non-null int64
channel               1018838 non-null object
type                  1018838 non-null object
mapped_orientation    1018838 non-null int64
xy-loc                1018838 non-null object
dtypes: float64(1), int64(2), object(9)
memory usage: 101.1+ MB
```

Out[8]:

	time	scanMac	posX	posY	posZ	orientation	mac	signal
0	1139643118358	00:02:2D:21:0F:33	0.0	0.0	0.0	0.0	00:14:bf:b1:97:8a	-38
1	1139643118358	00:02:2D:21:0F:33	0.0	0.0	0.0	0.0	00:14:bf:b1:97:90	-56
2	1139643118358	00:02:2D:21:0F:33	0.0	0.0	0.0	0.0	00:0f:a3:39:e1:c0	-53
3	1139643118358	00:02:2D:21:0F:33	0.0	0.0	0.0	0.0	00:14:bf:b1:97:8d	-65
4	1139643118358	00:02:2D:21:0F:33	0.0	0.0	0.0	0.0	00:14:bf:b1:97:81	-65

```
[  0  45  90 135 180 225 270 315 360]
[360 135  90 270   0 180 315  45 225]
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 60 entries, 0.0-0.05 to 9.86-3.88
Data columns (total 9 columns):
00:0f:a3:39:dd:cd    60 non-null float64
00:0f:a3:39:e1:c0    60 non-null float64
00:14:bf:3b:c7:c6    60 non-null float64
00:14:bf:b1:97:81    60 non-null float64
00:14:bf:b1:97:8a    60 non-null float64
00:14:bf:b1:97:8d    60 non-null float64
00:14:bf:b1:97:90    60 non-null float64
02:00:42:55:31:00    60 non-null float64
dummy_angle          60 non-null int64
dtypes: float64(8), int64(1)
memory usage: 4.7+ KB
```

Out[9]:

mac	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
xy-loc					
0.0-0.05	-63.207207	-52.227273	-62.948980	-61.813953	-40.068966
0.15-9.42	-66.117117	-55.275229	-73.961905	-72.701031	-47.813084
0.31-11.09	-67.054054	-51.709091	-70.082474	-70.098901	-54.088235
0.47-8.2	-74.153153	-49.500000	-64.258065	-72.597701	-45.652893
0.78-10.94	-71.403670	-53.263636	-66.960000	-66.809524	-48.413793

Online Test Set Should Flatten out to 60 rows per textbook example

Exploratory Data Analysis (EDA)

Mac address counts in OFFLINE data set

Out[14]:

```
00:0f:a3:39:e1:c0    145862
00:0f:a3:39:dd:cd    145619
00:14:bf:b1:97:8a    132962
00:14:bf:3b:c7:c6    126529
00:14:bf:b1:97:90    122315
00:14:bf:b1:97:8d    121325
00:14:bf:b1:97:81    120339
02:00:42:55:31:00    103887
Name: mac, dtype: int64
```

Distance Measurements and Error

Objective: Describe how you estimated your error and found the best fit ASSUMING you CANNOT USE THE ONLINE DATA.

Distance between two points in space will be defined as the **euclidean** distance between the point we wish to classify and the collection of points in our sample **signal space** (a n-dimensional space consisting of each mac address' signal reading).

$$d_{signalspace} = \sqrt{\sum_{i=1}^{i=n} (S_i - T_i)^2}$$
$$S_i = Signal@mac_{i,train}$$
$$T_i = Signal@mac_{i,test}$$

Error will similarly be evaluated, but in the more traditional euclidean distance represented in 2-d space (i.e the **pythagorean theorem**). Using this as a metric for scoring (lower is better) will allow us to distinguish the best algorithm parameters.

$$Error_{prediction} = \sqrt{(X_{target} - X_{predicted})^2 + (Y_{target} - Y_{predicted})^2}$$

Additionally, we will use an inverse distance weighting to weight our XY predictions in our nearest neighbor algorithm to see if we can smooth out the effect of increasing values of k.

$$Weight_i = \frac{1/d_i}{\sum_{j=1}^{j=k} 1/d_j}$$

Due to the fact that we cannot build our model with knowledge of the online dataset, we will use a train/test split and cross validation to select the best model for locating the OFFLINE set

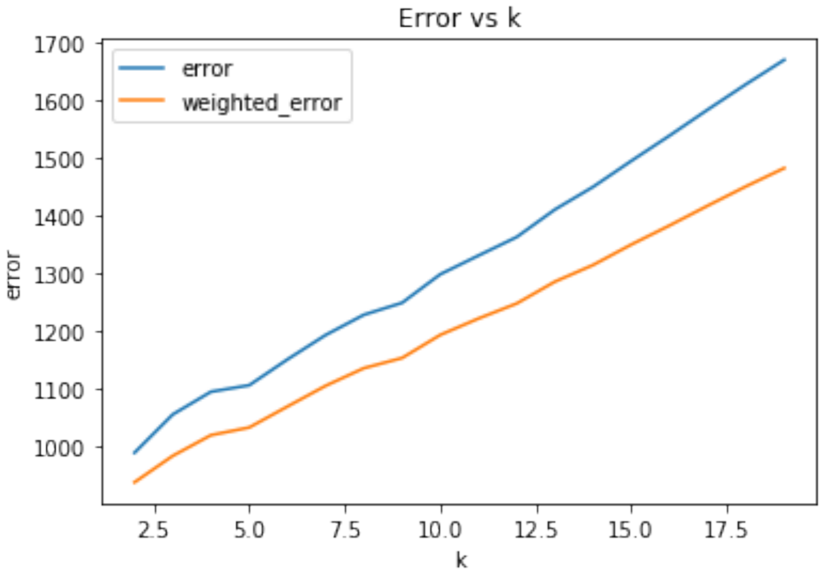
Train/test split

Optimal k - we'll search for the optimal value of k (the number of neighbors to be included in making our prediction) by performing a **5 fold cross validation** on a range of **ks from 2-20**. Each fold will consist of an **80/20 train/test split** for the entire offline dataset. 80% will be used to form our nearest neighbor graph which will then be used to estimate the X-Y location of the remaining 20% of points.

The parameter m will be set to **three** for this particular selection of k. Setting m to 3 will run this cross val using a sample from 3 angle orientations that bracket the dummy angle provided in the validation/testing sets (**which are randomly assigned**). If an angle of 45 is passed into the filter, the training set will consist of angles in [0, 45, 90].

```
Fold: 1 completed
Fold: 2 completed
Fold: 3 completed
Fold: 4 completed
Fold: 5 completed
```

Out[16]:
```
[Text(0, 0.5, 'error'), Text(0.5, 1.0, 'Error vs k')]
```



For visualization purposes, the cross validation results are presented in aggregate by fold, each line is the aggregate error generated at k for each fold

Cross Val Results

Objective: Describe the best fit for the data.

As expected, the weighted predictions generally performed better, having lower values for error across the board. Unfortunately, it's difficult to assess where the elbow is on the chart and **it seems to be counterintuitive that the errors increase almost monotonically with increasing k**. Since there appears to be bump at around 3 or 4, we'll try that as a value. Anything lower than that seems too small. We will continue our fitting process below by swapping out a few of the mac addresses that may not be reliable.

cD or c0?

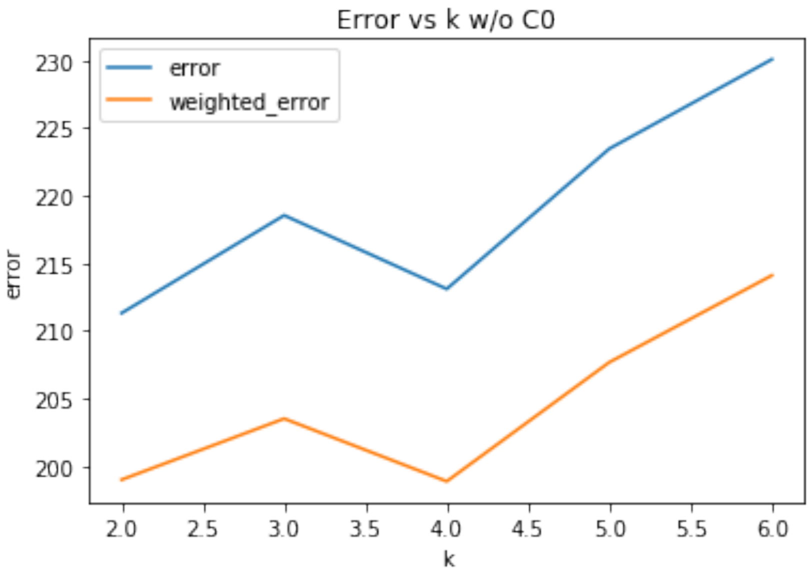
Objective: There are 2 macIDs located at the same position. Does one give better performance than the other?

It has been suggested that one of these two devices may be faulty or that they may be less useful when used in tandem. Let's see if our modeling improves if we leave one of the devices in and the other out.

No C0

Limiting k search space to 7 and foregoing CV to keep run time low

```
Out[19]: [Text(0, 0.5, 'error'), Text(0.5, 1.0, 'Error vs k w/o C0')]
```

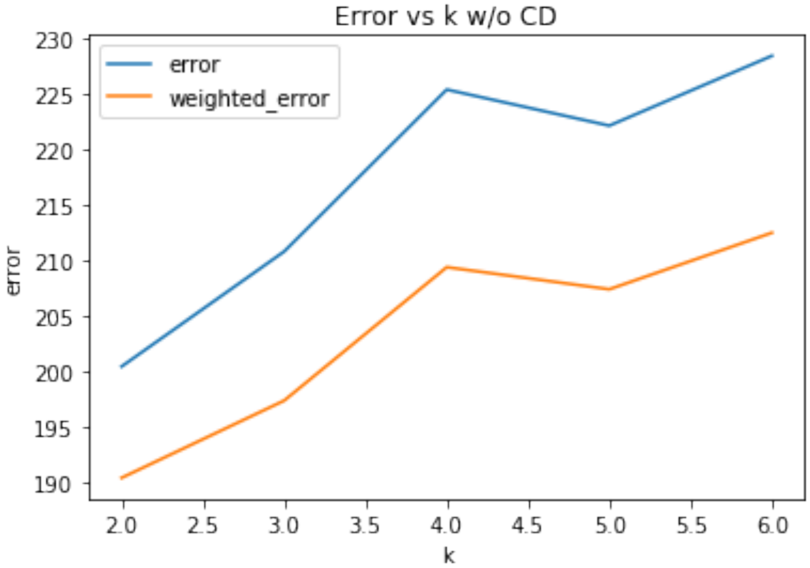


Elbow plot shows a turn right around the k = 3 mark. Again, weighted error was lower across the search space.

No CD

Limiting k search space to 7 and foregoing CV to keep run time low

```
Out[21]: [Text(0, 0.5, 'error'), Text(0.5, 1.0, 'Error vs k w/o CD')]
```



Elbow plot shows a turn right around the k = 3 mark. Again, weighted error was lower across the search space.

Objective: What about using them both?

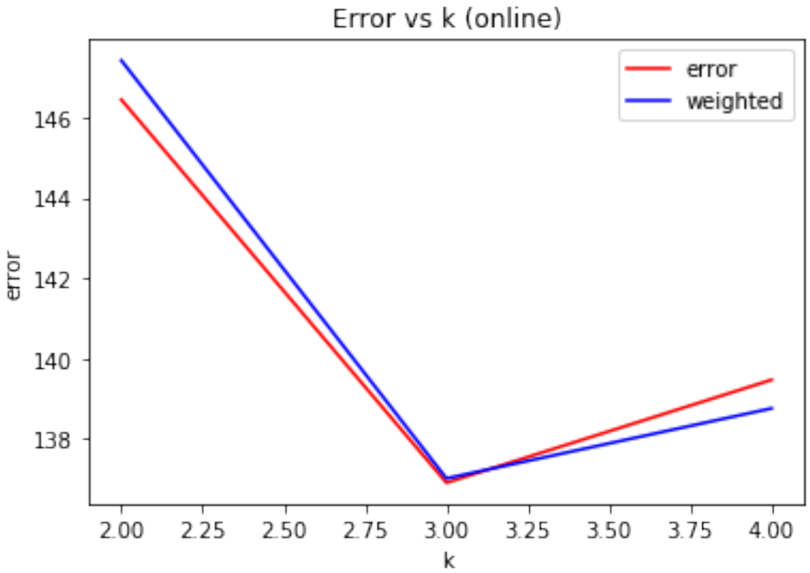
Overall, predicting with one or the other seems better than predicting with both though there was not much of a difference in the best error rate achieved by leaving one out instead of the other. *Error rates for both methods at the optimal k of 3 were nearly identical*.

In our final analysis we will use the data without CD and with k=3.

Online Predictions

Using the optimal values of k selected in cross validation, we will make predictions on the **online** testing set using the same methodology outlined above. We'll search a bit outside our targeted value of k to see if errors improve or worsen as k changes. Ideally, our choice of k should be an elbow point as it was in our analysis with C0/CD swapping.

```
Out[23]: [Text(0, 0.5, 'error'), Text(0.5, 1.0, 'Error vs k (online)')]
```



Conclusions/Final Analysis

Objective: What is the drawback (if any of using this method to real-time locate an object)?

- The main drawback of using this method is speed and in order to account for that we have traded off a great deal of accuracy by aggregating with respect to angle of orientation. While the speed boost was necessary, it still was not enough to make this a viable real time locations tracking system. **Using the dummy angle as decribed in the book is also rather dubious, given more time we would've like to make the testing set larger by disaggregaton and then cross reference that with a more robust training set**.
- The angles didn't seem like useful data to use in this method because the device will always be held in multiple angles based on the user preference inside the warehouse. Whether the user is holding the device up, down or to the side, the main objective is to identify the actual location of the device in real time. So as long as we can determine the actual location of the device, the angles are uneccessary.
- The positioning of the actual receivers can also play a part because it is hard to determine if the device is located in multiple floors. Such as being in the 3rd floor because of a mezzanine inside the warehouse compared to being on floor level. It is possible the solution will provide you with the wrong location/distance.

Objective: Describe a method that may be an improvement based on your perceived drawbacks.

Various improvements could include:

- Using a different metric for distance such as **manhattan distance** could lead to different/improved results. Alternatively, we could have attempted to use a different weighting scheme rather than **distance**, perhaps **exponential smoothing** or some other type of decay methodology.
- Caching our training data in variety of angle sweeps so that it does't have to be created every time the algorithm is called would dramatically improve speed. In theory, if we increased the size of the testing set by including an angle then we could use the cached data to quickly create a multitude of kNN graphs and select from those to create our final kNN graph. This would ultimately improve speed and accuracy.
- Fully utilizing the heat mapping, using the median signal at two access points and two angles. The heat map nicely illustrates representation of signal stength. So you can see how people move around your venue. This allows you to gain important insights on what’s good about your layout.
- Varying the parameter m for the number of angles included in training aggregation would be an interesting further analysis to undertake. A strong argument could be made that including all angles in the aggregation would be ideal though there would be a performance impact if that change were made. Searching the space from 2-5 similiary to how we searched for an appropriate k would've helped assess the effect of that parameter in a more direct fashion.