Indoor Localization with RSS kNN Classification

Goal: develop an application that will achieve localization using available signals. Options in your kit:

- Ultrasonic range sensors
- IR proximity sensors
- Digital compass
- Received radio strength signal indication (RSSI) from XBee module or from wifi (Photon)

Like trilateration, the kNN method uses data gleaned from ranging to identify an estimated position for a device. kNN requires calibration (understanding the RSS signal), and training, to identify the set of viable outcomes (bins). The cool thing is that the output bins do not need to cover the entire physical space. In our example this means that we do not need to map anything but the hallways.

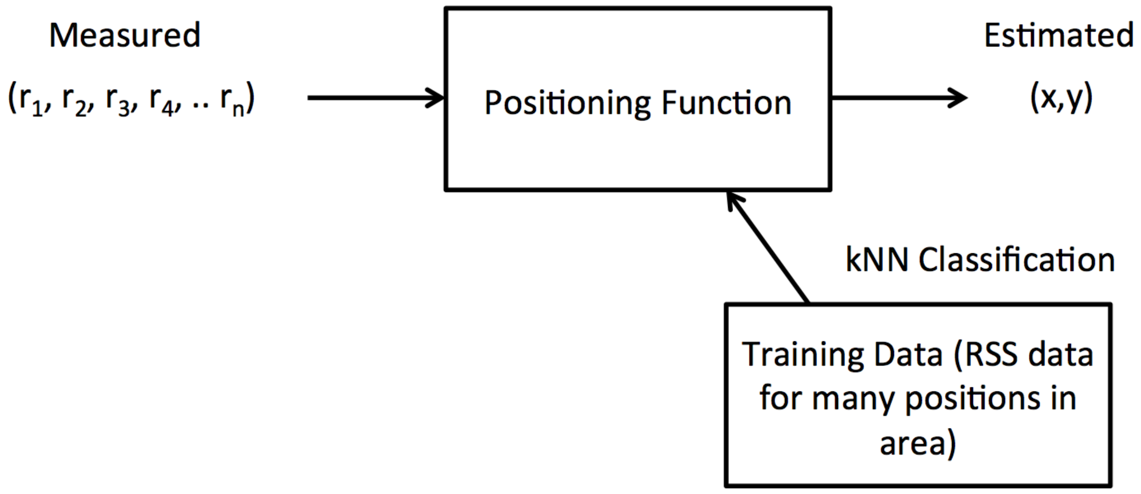

Measured
$(r_1, r_2, r_3, r_4, .. r_n)$
Positioning Function
Estimated
(x,y)
kNN Classification
Training Data (RSS data for many positions in area)

Figure 1: kNN Schema

Reference Materials

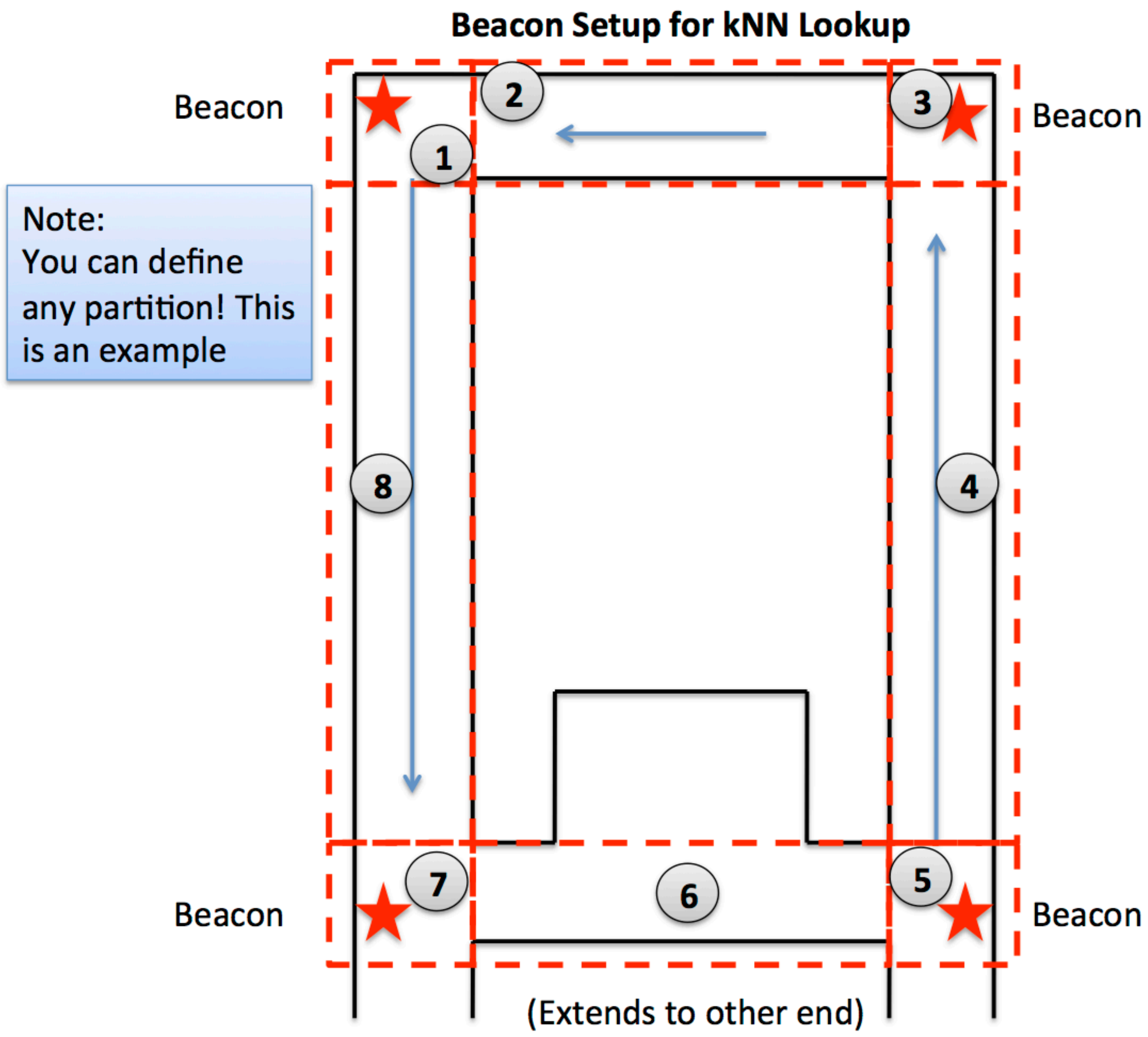
- K-nearest neighbors algorithm http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- Sohn, Localization performance analysis of KNN in IEEE 802.11 TGn channel. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6082583

Localization with kNN Classification

k-NN is a widely-used machine learning algorithm, that requires two-stage data analysis. The first stage is to collect training data. The second is to use the trained classifier to classify new (unlabeled) data. The hardware/sensor setup remains the same in both training and testing stages. The steps for kNN are enumerated below.

1. <u>Create a Map of the space.</u>

Divide the entire floor plan into partitions and assign each a label $(1, 2, \ldots, y)$. The size of bins will decide the resolution of localization. However a large number of small-size bins may lead to overfit problem from more possible misreading. The decision on bin size and number requires experimentation.


Beacon Setup for kNN Lookup
Beacon
Beacon
Note:
You can define any partition! This is an example
1
2
3
4
5
6
7
8
Beacon
Beacon
(Extends to other end)

2. <u>Set up your RF beacons</u>

Place a number of $\mathcal{N}$ beacon XBees at fixed locations. The remote (target) mote measures signal strength (RSS) of RF signals from each of the beacons. The beacon locations remain the same in both training and testing stages.

The more beacons used, the more accurate localization results are. At the minimum, 3 beacons will yield a decent result (as we discuss in class, fewer than 3 can result in ambiguity in location identification.

3. <u>Collect the training data</u>

At each region (bin) of the partition on the floor plan, collect a sufficient number of RSSI readings from each beacon. The more readings acquired at the training stage, the better classification will be made in the application stage.

Store the RSS readings from each beacon at a region instance in a text, csv or excel file. Each row represents a data frame instance: the signal value from each beacon, and the ground-truth region label. The training data set is comprised of multiple data frames at each region in the partition of the floor space.

Example:

Beacon 1	Beacon 2	Beacon …	Beacon $\mathcal{N}$	Location Bin
12.1	22	…	12.6	1
…	…	…	…	1
13.1	22.5	…	13	1
12.5	22.2	…	13.2	1
5.7	18	…	7.1	2
5.3	18.5	…	8	2
…	…	…	…	…
…	…	…	…	y

Multiple data frames of RSS measurements at each region

X Y

4. <u>Train kNN classifier in Matlab</u>

Convert stored training data into a Matlab data format. Matlab has a built-in k-NN classification function, `ClassificationKNN.fit(X,Y)` in MatLab2012/2013 or `fitcknn(X,Y)` in MatLab2014. Clicking the function name leads you to the Mathworks documentation.

In both functions `ClassificationKNN.fit(X,Y)` and `fitcknn(X,Y)`, `X` is the numeric matrix of features, `Y` is the label or classification output, indicated in the above table. By default, without specifying `NumNeighbors`, k equals 1. Read the documentation for further information on setting the classifier's arguments.

`mdl = ClassificationKNN.fit(X,y,Name,Value)` or
`mdl = fitcknn(X,y,Name,Value)`
`mdl` is the trained classifier model that can be used for classification in the second stage.

Of course, you can write your own kNN classification function other than in Matlab.

5. **Localization with the Classifier Model**

Once the classifier is trained, we can proceed to predict the location by feeding in the RSS measurements at a current location. The k-NN classifier will output the predicted location partition label based on its nearest neighbors.

`Location_label=predict(md1,current)` where `current` represents the data frame of RSS values at the current location.

Note: Averaging a number of signals before feeding into the predictor will yield better results.

The current RSS value can be fed into Matlab by using a TCP/IP connection between Matlab and the java program running on the host which collects the radio signals from the devices.

The returned label can be transformed into the area information that can be overlaid on top of the floor plan image to represent the current location of the object.

Summary of Matlab functions

`ClassificationKNN.fit()` or `fitcknn`
`predict`
`load`
`fscanf()` tcp connection from java program...read in stream of data at a port
`plot(),imshow()` to plot the current location on top of the floor plan

Sample Code

- Creating a TCP/IP connection in MATLAB

```
obj=tcpip('localhost',8080,'InputBufferSize',8000) %where
8080 is the port number
fopen(obj)
pause(10)
while(obj.BytesAvailable)
     data=fscanf(obj)
```

- Drawing a circle on top of the image at coordinate x,y

```
FloorPlan=imread('.\FloorPlan.jpg');
imshow(FloorPlan);
hold on;
plot(x,y,'r*');
ang=0:0.01:2*pi;
xp=100*cos(ang);
yp=100*sin(ang);
plot(x+xp,y+yp);
drawnow;
hold off;
```