Machine Learning Engineer Nanodegree

Capstone Proposal

Luca Veronese
February 28th, 2017

Domain Background

This project aims at the solution of a real world problem I've found working for a startup where I am one of the founders. We have developed a people/object localization system in buildings that uses Bluetooth Low Energy (BLE) devices, commonly known as beacons. We use Raspberry Pi 3 devices connected to a network inside a building to sense the beacons in a given area, report the data to a Cloud server and display the location of people/objects in the building on a map. I'd like to use machine learning techniques to improve the localization algorithm in order to provide much better results to the customer.

I have analyzed various approaches to similar problems. Many of these approaches are suitable for the case where multiple observations coming from multiple sensors are used to learn a single spatiotemporal function[1] or where multiple input features are provided[2]. As we will see, my problem is more constrained and simple, so I decided to make a personal proposal for a solution.

Problem Statement

BLE[3] devices are known for their inherent imprecision. The signal strength (RSSI[4]) is extremely unstable over time and in changing environments. Our application needs to state if a person is present inside a room in a building with high precision and low latency. Oftentimes a beacon is perceived by multiple antennas[5] on nearby rooms or the signal is intermittent or absent for a significant period of time. Also the thresholds values for IN/OUT vary from room to room and depend on the furniture inside the room and the location of the Raspberry device.

Our problem is to identify when a beacon enters a given room and when it exits the room from the signals received from the antenna located in the room. The beacon emits a signal nearly every second[6], so the sequence of signals received is a time series. We need to detect the enter/exit events indexes in the sequence of observations. Many beacons will concurrently move inside a building and each room will have one antenna. We consider all beacons equal but we need to

[1] D. Gu and H. Hu, "Spatial Gaussian process regression with mobile sensor networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 23, no. 8, pp. 1279–1290, 2012.

[2] A. Shareef, Y. Zhu, and M. Musavi, "Localization using neuralnetworks in wireless sensor networks," in Proceedings of the 1st International Conference on Mobile Wireless Middleware, Operating Systems, and Applications, 2008, pp. 1–7.

[3] https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy

[4] https://blog.bluetooth.com/proximity-and-rssi

[5] Antennas in our case are Raspberry Pi 3 devices

[6] This is a parameter. Higher signal emission frequency causes more battery drain.

learn a different model for each room since each room has different features (size, furniture, materials, ...). If two beacons are in the same room at the same time they represent independent time series, each one referring to a different beacon. Our problem is a classification problem where each observation in the time series can be labeled in two ways: IN or OUT, IN meaning inside the room, and OUT outside it. A single time series is defined by a single beacon on a single room from the time the beacon is first detected by the antenna and the time the beacon is no longer detectable by it.

In our current implementation we are using simple moving averages computed over the series to clean the values a bit and stabilize them but this solution is limited. Currently we label all detected signals as IN and only detect an OUT event when the beacon is no longer detected (using a timeout). This is a problem since beacons can be detected when they are significantly far from the room (or in nearby rooms) and signals are sometimes interrupted for unpredictable time intervals even when the beacon is stationary inside the room. This causes many spurious events to be emitted.

Datasets and Inputs

I will be using the RSSI values measured from antennas as well as the delay in milliseconds between successive measures. I can obtain an arbitrary number of labeled measures for a few use cases making a few changes to the current code (JavaScript on node.js inside the Raspberry Pi). So our input files will have three features: RSSI value, delay from last measure, beacon inside/outside. The last feature is the feature to be predicted and the class labels are IN and OUT as describe in the previous section. This data can easily be obtained from our current instrumentation and there is no privacy problem related to it. This is the data I can obtain on the field. I could think of additional features to add but the purpose of this system is to create an easy to use setup procedure for installers that set these devices up, so I can’t force them to provide additional information like exact positions of objects on a coordinate system.

Solution Statement

I we had a coordinate system with precise locations for the Raspberries we may use known algorithms for localization like Kalman filters and particle filters[7]. This may be a solution for other kind of situations but for this specific use case we are trying to detect an IN/OUT event, we have no coordinates and a single antenna per room. Classical localization algorithms seem not to be a good fit.

The solution I suggest is to use a probabilistic model based on Gaussian distributions. The data I get, being repeated observations of the RSSI over time, should be reasonably distributed as a Gaussian (at least the distribution is symmetric and unimodal if the antenna is placed centrally in the room).

Each data sample is a time series where the observations are ordered and time dependent. We are actually interested in the point in time in which the beacon changes position (inside to outside

[7] https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373

and vice versa). My idea is to learn the parameters of two normal distributions from a set of IN labelled observations and a set of OUT labelled ones and compare the joint probabilities of N subsequent observations under the two assumptions (represented by the IN and OUT gaussians), computing a comparison metric over a sliding window of time until the metric reaches a threshold that signals the event.

Benchmark Model

As a benchmark model I will use the results of our current implemented solution which is not using any probabilistic model and uses some heuristics to detect the event. This will be our baseline for comparing the new model and evaluating its performance.

Evaluation Metrics

The metric I suggest to implement for the solution model is:

$M = P(IN_1, IN_2, \ldots, IN_N) / P(OUT_1, OUT_2, \ldots, OUT_N)$

where IN_j is the probability of the j-th observation in the current window measured on the IN Gaussian, and OUTj, is the equivalent for the OUT one. P(..,...,...) is the joint probability of the observations (in the window). While the beacon is "IN", M should be much greater than 1. If the beacon is "OUT", M should be much smaller than 1. So M = 1 should be our desired threshold to use to detect the event. In practice we will predict IN when M >> 1[8] and OUT when M << 1.

From our training data we know the exact index in the observations sequence at which the IN/OUT event happens. The model must be able to predict that index. Any deviation between predicted and observed index has to be penalized, as well as any spurious event prediction (predicted by the model but not happened in the observations). Spurious events should be penalized more than lags or leads in event detection since we don’t want the system to behave erratically. In our domain it is acceptable to have a few seconds of delay in the detection of the event. The best model is the one that minimizes this delay without producing spurious events. This two requirements are unfortunately in conflict: the shorter the allowed delay, the higher is the probability to detect a spurious event.

For the benchmark model we can compute the predictions with the current code. For the solution model we will compute the predictions (IN, OUT) from the sequence of observations using a sliding window of N observations. I will compute a cost function that accumulates a cost whenever the predicted value differs from the observed one at the same index. Whenever there is a "bounce" in the predicted values (e.g. IN, OUT..., IN) but the actual values are stable for the same indexes (IN...) we will add a spurious event cost. The costs we should use for computation are parameters we will have to define somewhat arbitrarily since they are domain specific. The resulting function can be applied to both the benchmark and solution model. This makes it easy to compare their relative performance.

The value of N in the solution model is a hyperparameter that must be learned in order to

[8] The symbols >> and << here are used with the meaning: much greater than, much smaller than.

minimize the cost function for the solution model. I will evaluate the solution model (on the same validation samples) for various values of N from 1 to 10 and will choose the value of N that minimizes the cost function.

Project Design

These are the steps I intend to follow:

1. implement the data collection so that I can obtain useful samples of the data captured by the antennas. This is a fairly simple programming task but has some requirements outside this project (I need to send the data to a server and have the process controlled from a client device in the hands of field personnel). I can't release this code but it is not part of the solution. It is intended only to support the data collection phase;
2. Field personnel will use this simple application to train the system. While inside the room they will walk around while the system collects a sample of observations. After this collection they will make a new collection outside the room in the neighboring areas (corridors and rooms). So we will have two labeled sets of observations to train (estimate) the Gaussians parameters;
3. We will then collect a few sequences where the personnel will enter/exit the room at various speeds, signaling the exact moment they enter/exit the room[9]. I will use this data for hyperparameter tuning and validation;
4. I will then test the model on the field at a customer's site to see how the system behaves once the model is in operation. I will collect predictions from the benchmark model as well as the new ML based model in order to create a comparison report to determine the improvements provided by the ML based model.

One last consideration: since we have two features we need to estimate bivariate normal distributions. The two features (RSSI and delay from previous observation) are probably significantly correlated. If the correlation comes out to be very high we may decide to remove the delay feature and go for a univariate normal distribution based on RSSI alone which simplifies things[10]. On the other hand, if the delay provides significant additional information we will make use of it and implement bivariate normal distributions.

[9] They will use the configuration application from a portable computer or mobile phone to signal the position change event

[10] Choosing to keep/remove the delay represents a trade off between potentially higher variance and potentially higher bias.