Location-based Predictive User Activity Models

Capstone Project

November 2017

Project Overview

In this project, a predictive model is developed that can power users' intelligent personal assistants. The model enables use cases which need to predictively anticipate a user's desired activities and preferences at each point in time. Here are several examples of such use cases:

- Predict where a user will go next based on their current location and historical movement data;
- Predict what application a user is most likely to run on their phone next given their current location and context and past behavior;
- Predict who a user will call on their phone next given current context and historical actions.

This project will focus on the first use case (i.e., location prediction) but the methodology outline here can be extended to other use cases via the generalization of the notion of *activity*, i.e., what action will come next given some present context and historical data.

This way, the notion of activity effectively captures the user's personal *habits* over a period of time during which the user's historical data has been collected, and analyzed. Furthermore, the modeling of user's activities is context-aware and is dependent on present set of circumstances, e.g., time, location, or presence of a missed call from a friend as an example.

The predictive model is *personalized* for individual users and can be continuously trained as future training data becomes available for that user. Finally, the predictive model offers a simple and standard interface allowing it to be plugged programmatically to a variety of "intelligent" systems. Such systems (not covered in this project) could use the model's knowledge of user's habits for scenarios including social planning (e.g., when a group of friends are available for a social gathering), or opportunistic marketing (e.g., identifying the needs of a user given his/her anticipated future activity/needs) and alike.

Problem Statement

The problem addressed in this project is to determine what activity a user is inclined to do at a given point in time. More specifically, this project focuses on "where" the user will be next given his/her current whereabouts (e.g., region and geographical coordinates) and other contextual data such as time of day. This determination is based on the user's past behavior and their context at that particular point in time.

Since individual users are likely to be following widely varying sets of day-to-day activities and habits, the approach taken in this project is therefore to predict each user's activities in a personalized fashion.

As such, the user data conveying the following pieces of information is presented to the model for training purposes:[1]

```
UserId: [list of <DateTimeBag, LocContext>]
```

where the `UserId` identifies the user; `TimeDateBag` is the recorded time an activity takes place; and `LocContext` identifies the user location. Few points are of note here:

- For simplicity, this document focuses on only one user (`UserId=0000`).
- The `DateTimeBag` element is decomposed into individual `'year', 'month', 'hour', 'partofday', 'weekday'` features thought to be helpful according to domain knowledge.
- Finally, `LocContext` consists of latitude, longitude readings. The altitude of the readings (while being present) is purposefully ignored due to the fact that alt is largely dependent on (lat,lon) coordinates. Furthermore, an online service (offered by Google Map API) will be used to translate location to landmarks using nearby search API, e.g., `Central-Park`.

The model then performs a multi-label classification using a decision tree classifier where each predicated label is a previous `Activity` (in this case, already-seen location) of the user.

Metrics

The metric used to evaluate the performance of predictions is the accuracy score. This metric is chosen since the false predictions may become an important impediment to user experience. As such it is most important to ensure accuracy of predictions by thoroughly evaluating the performance of the predictive models (and naive model as benchmark).

Finally, the accuracy score is computed over a test dataset which was not used for training purposes. This is done by applying `sklearn.metrics.accuracy_score()` to test and predicted activity labels.

Analysis

Data Exploration

First, let us focus on the user location data points in the dataset. As discussed earlier, location data consists of latitude and longitude of GPS reading for each user. Although latitude and longitude are radial coordinates measured as degrees from center of the earth, a simple 2D plot of these readings over x and y axes (ignoring their radial nature) immediately gives valuable insights about the user's movements.

The graph in **Figure 1** shows concentration of

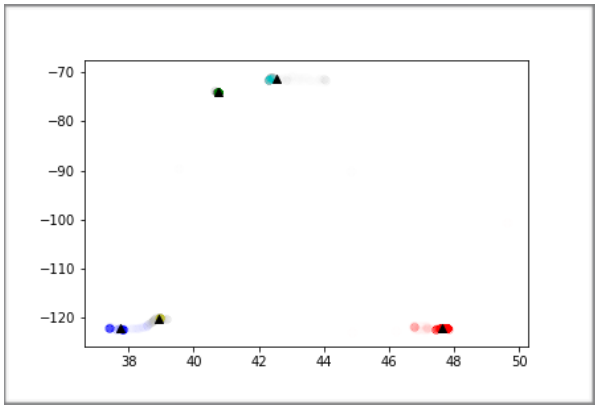

Figure 1. 2D plot of latitude/longitude data shows concentration of user movements.

[1] The data is collected with user consent using an iOS application over a period of 5 months.

Cluster (record count)	Latitude	Longitude	Address
Cluster center 1 (8344)	47.65569203238255	-122.13006269447509	15722 NE 53rd St, Redmond, WA 98052, USA
Cluster center 2 (408)	40.74688682745098	-74.05754189950981	255 Hutton St, Jersey City, NJ 07307, USA
Cluster center 3 (548)	37.73064549543796	-122.16183264744525	201-213 Leo Ave, San Leandro, CA 94577, USA
Cluster center 4 (322)	42.53186271645963	-71.27615696397517	Northwest Expy, Billerica, MA, USA
Cluster center 5 (284)	38.91116934929577	-120.16368754683099	Pacific Crest Trail, South Lake Tahoe, CA 96150, USA

Table 1. KMeans is used to identify 5 centers of user movements.

user's movements clustered around 5 centers.[2] This is expected, since individuals often live and work in certain areas, but may still travel to other cities or states occasionally.

Given the visual validation of clustered concentration of user movements, an unsupervised KMeans clustering technique is applied to the data. The coordinates of the identified cluster centers designate the whereabout of where user activities are conducted. As we shall see later, these cluster centers will prove valuable in improving the prediction accuracy of the trained classification model. As such, a pre-processing step is applied to augment `LocContext` in the raw data with the designated coordinates of the cluster centers (see Data Processing section).

Table 1 illustrates the detailed coordinates of the cluster centers and their corresponding addresses. Visual inspection of this table confirms that majority of datapoint are from Redmond, WA, with fewer datapoint in remaining cluster centers where the user has only been visiting for a short time. Note that the exact address of the cluster center offers little value but the city/state/country portion of these addresses offer sufficient detail about the whereabouts of the user.

Given that much of the data points is geographical latitude and longitude, producing a geographical map of the user's movements will further shed more light into the behavior of the user. This is illustrated in **Figure 2** which maps the user's movements around the 5 cluster centers separately.

[2] As mentioned earlier, data exploration discussed in this section pertain to `UserId=0000`.

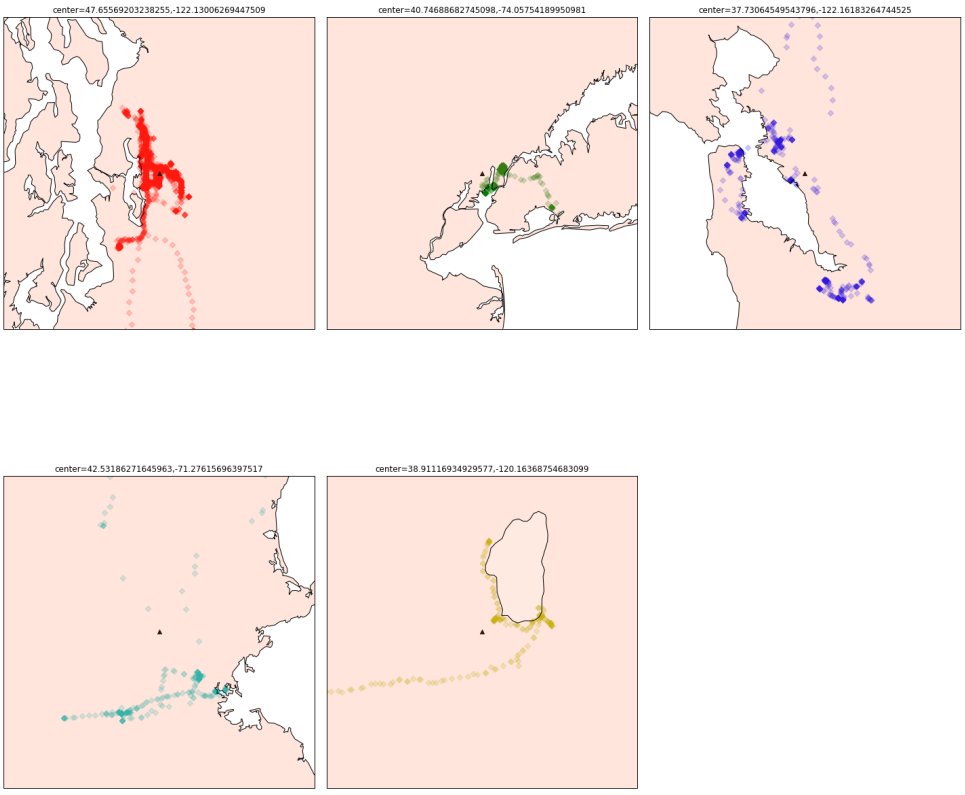

Figure 2. Mapping of user movements.

The maps further shed light on the fact that many of the datapoint are along highways and pertain to the times when the user is commuting/driving. Such data points are unlikely to be of much value to the goals of this project for two reasons:

- The highest frequency of readings in the raw data set is about 2 minutes. While the user is in motion on a long stretch of road at a high speed, it is almost guaranteed that raw data points pertaining to the same geographical location are unlikely to be re-captured over successive travels.

- Moreover, a long travel can be more appropriately modeled by the starting and ending locations where the user has been stationary (before and after the movement). These stationary points often better capture the user's desired activity. For example, being at work or at a coffeeshop is a key activity. In contrast, the motion to drive to work or walk to a coffeeshop are merely transitionary activities which bear less significance in this analysis.

As such, a pre-processing step is carried out to first capture the "duration of stay" at each point, and second filter out the locations where the duration of stay is less than 5 minutes. To further facilitate the latitude and longitude readings that within the same location vicinity, a mathematical rounding step is carried out to keep only 6 decimal points of the latitude/ longitude values (the raw data comes with 7 digits). The effect of this rounding is also to cobble up readings within a 10-meter vicinity to be perceived as identical.[3]

[3] This loss of accuracy is perfectly acceptable for real world scenarios targeted in this project and has the added advantages outlined above.

Duration of stay is not directly available in the raw dataset but can be easily derived by subtracting the timestamp of successive readings (readings are already sorted over time). Furthermore, the 5 minute stationary criteria of being at one location is chosen to account for and gracefully handle cases where the user travels slowly, e.g., in heavy traffic. In reality, many of daily activities take longer than 5 minutes, which makes the choice of this value a reasonable one.

The result of the pre-preprocessing of data at this point produces 3463 readings pertaining to locations where the user has been stationary. However, a quick review of the frequency of visits in each location reveals that not all these locations are visited sufficiently frequent to be of significance for purposes of this project. As a result, a further filtering steps is carried out to limit the data to those locations with a minimum of 10 visits in the dataset. This results in the "frequent stationary locations" which capture much of the daily habits and activities of the user.

Finally, Google maps API for nearby search is applied to produce a label for each frequent stationary locations. **Figure 3** illustrates these labels and their "frequency of visit" and "total duration of stay" are illustrated.

The graphs are highly skewed with only 3 locations being visited over 100 times - these most likely pertain to the home and workplace of the user (in fact the user has confirmed that he has moved between two houses!).

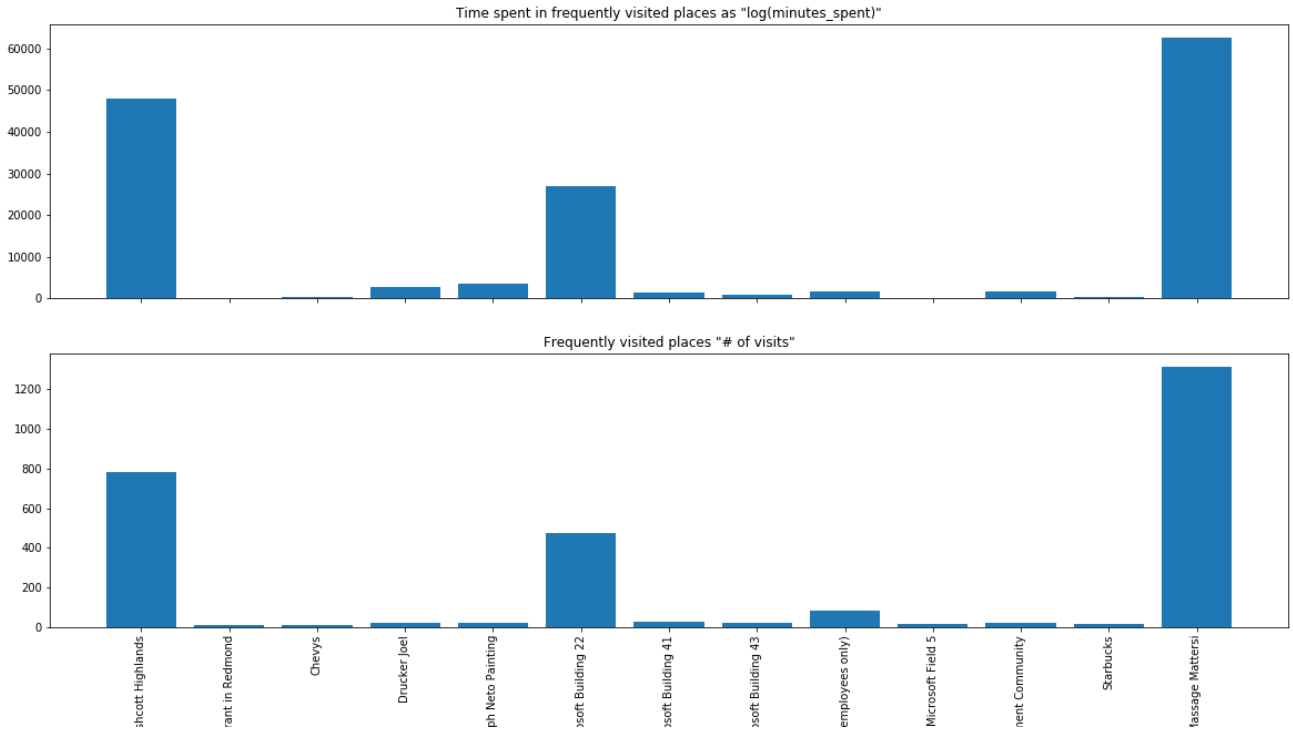

Figure 3. Duration of stay (top) and frequency of visit (bottom) for user's frequently visited stationary locations.

Algorithms and Techniques

The data processing step uses the K-Means clustering algorithm. This choice is based on the fact that the raw dataset does not contain any labels and an early application of unsupervised clustering technique produces further insights into the data. However, the use of the cluster centers identified by K-Means is not limited to data exploration, rather the dataset is augmented by he cluster ID's so to provide this insight as a feature to the classification algorithm (discussed below). The choice of number of clusters to use (i.e., 5) has been largely driven by visual inspection of the dataset, as depicted in **Figure 1**. The implementation used is `sklearn.tree.DecisionTreeClassifier`. K-means works by randomly picking the designated number of cluster centers and iteratively assigning data points to each cluster and readjusting the center based on the mean of data points assigned to that cluster at the end of each iteration.

The classification algorithm (implemented in the `LocationPredictor` class) is based on a multi-class decision tree classifier available through `sklearn.tree import DecisionTreeClassifier`. This choice is motivated by the need for a simple supervised learning technique to predict future user locations (as predicted labels). The classifier is feed a mixture of raw and derived features (i.e., features present in the raw dataset vs. features derived as part of data processing). The classification labels are also based on the derived feature labels produced by nearby search using Google Maps API. The decision tree classifier works by finding best split at each node a decision tree it constructs iteratively. This project uses the gini metric which is roughly speaking a measure of impurity of a given tree node. The algorithm choose the split in such a way to minimize the impurity and still respecting various algorithm parameters such as "max depth" or "min sample split".

The naive classifier predicts user location randomly and based on the cumulative distribution of past visits.

Benchmark Model

The benchmark model developed in conjunction to this project predicts user location based on the distribution of user's visits to frequently visited location. As such, this model ignores the user's current location and predicts the user's next location using the frequency of previous visits as weights. Hence, this benchmark is considered to be a *naive* model.

The initial evaluation of the naive model over 100 runs shows the accuracy of the predictions is on average 7% ranging from min of 0% to max of 23%.

```
Number of runs: 100
Min accuracy: 0.0
Avg accuracy: 0.07615384615384616
Max accuracy: 0.23076923076923078
```

Methodology

Data Processing

The raw data points of over 9900 readings for the user is pre-processed through the following steps:

1. Identify stationary user locations (small movements over successive readings in a 5 minute interval). Filter out the data points where the user is not stationary. This brings the total number of records to 3463.

2. Filter out the infrequently visited locations using a threshold of 10 over the number of visits. This brings the number of records to 2820.

3. Apply KMeans clustering to latitude and longitude data points to identify cluster centers (i.e., user's whereabouts). Furthermore, augment each record with the cluster center it is assigned to.

4. Augment the timestamp of each record to capture `'year', 'month', 'hour', 'partofday', 'weekday'` as separate features (where `partofday={morning, afternoon, evening}`).

In this project, the majority of development time was devoted to the data processing and exploration phase (which were tightly intertwined and were carried out over many iterations.

Implementation

The predictive model chosen for this project is a decision tree classifier. The `LocationPredictor` class encapsulates the classifier and provides an interface to train, use, and evaluate it. Using a single function call, a caller can easily run the training, and prediction steps and retrieve the evaluation results of the classification tasks (as accuracy and logloss) on an instance of this class by passing the desired dataset (splitting the dataset to training and test data is carried out by the class). The caller can also choose to convert the dataset to one-hot encoded values although doing so may cause some loss of information for several features such as time and date.

```
    loc_predictor = LocationPredictor(one_hot_encode)
    accuracy, logloss = loc_predictor.evaluate_classifier(loc_data[feature_list],
loc_data[['loc_name']])
```

Training is carried out over subset of data containing 80% of the records while evaluated over a test subset which contains 20% of the data. Furthermore, substring of the data to training and test is done randomly, without introducing unintended bias despite the fact that the dataset appears to be a time series. This is due to the fact that pre-processing of data has already captured the duration of stay and stationary criteria of the user over the original dataset (containing successive readings both either of test/training subsets). Apart from these, the training of the model (further elaborated below) does not rely on the time series nature of the readings.

An early version of the naive classifier (implemented in the `NaiiveLocationPredictor` class) was implemented entirely from scratch to compute the cumulative distribution of user locations. This version was later replaced with the final version that achieves the same goal using Pandas libraries via a call to `pandas.DataFrame.cumsum` instead. This proved to produce a code that was much cleaner and far more concise.

one of the decisions that was made is to “condense” many data points associated with each stationary locations over a period of time to only 1 entry and a corresponding “time spent” value. This approach is in contrast to using input data reported at fixed time frequency where for example a location A at which the user spends 4 times the amount of time (contagiously) compared to location B would appear as 4 times the number of entries in the input data. Given the large skew in time spent at different locations this helps keep the significance of locations at which less time was spent (e.g., a user’s home location where the user may spend between one third to half of the day would completely overshadow much of the other visited locations where much less time is spent).

Refinements

Given the small size of the dataset, various sets of features are examined with preference to get best prediction score using fewer features (to escape the curse of dimensionality). In the code, this is carried out using the following lines with feature set used being **highlighted** below:

```
_ = report_results(frequent_stationary_loc_data, ['month', 'hour', 'weekday'], False)
_ = report_results(frequent_stationary_loc_data, ['cluster', 'month', 'hour', 'weekday'],
False)
_ = report_results(frequent_stationary_loc_data, ['cluster', 'hour', 'weekday'], False)
_ = report_results(frequent_stationary_loc_data, ['cluster', 'year', 'month', 'hour',
'weekday'], False)
```

Results

Table 2 below illustrates the results of evaluation of the model over the test subset using the accuracy score. The best result is obtained when the training feature set contains ‘cluster’, ‘month’, ‘hour’, ‘weekday’.

This performs significantly better compared to the average accuracy of **7.61%** obtained from the naive classifier (see section on **Benchmark Model**).

Feature set	Accuracy (test over data)
['month', 'hour', 'weekday']	76.41%
['cluster', 'month', 'hour', 'weekday']	**78.01% (best result)**
['cluster', 'hour', 'weekday']	56.91%
['cluster', 'year', 'month', 'hour', 'weekday']	78.19%

Table 2. Accuracy of classifiers using different feature sets.

Furthermore, the best-performing feature set shows robust performance over 10 runs of training and evaluation. The accuracy ranges from a low of 75.17% and high of 79.78%, and average of 77.40%.

```
[0.75177304964539005, 0.76063829787234039, 0.76773049645390068,
0.76773049645390068, 0.77127659574468088, 0.78191489361702127,
0.78191489361702127, 0.78546099290780147, 0.7978723404255319]
```

Figure 4 plots the feature significance produced by decision tree using the best-performing feature set. It appears that the time is the best and most effective predictor of user location.

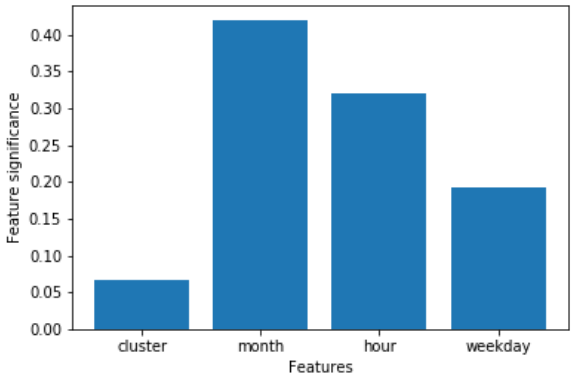

Figure 4. Feature significance of decision tree trained with best-performing feature set.

Finally, **Table 3** shows the result of the decision tree classifier under various model parameters (using `['cluster', 'month', 'hour', 'weekday']` as the best-performing feature set). The best accuracy of **78.01%** is achieved with **`min_sample_split=5`** and **`max_depth=16`**.

Min_sample_split	Max_depth	Accuracy
2	2	66.49%
2	4	76.95%
2	8	77.66%
2	16	77.13%
5	2	66.49%
5	4	76.95%
5	8	77.66%
5	**16**	**78.01%**
10	2	66.49%
10	4	76.95%
10	8	77.66%
10	16	77.30%

Table 3. Performance of the decision tree classifier under different model parameters.

Conclusions and Future Improvements

The problem of predicting user activities is challenging in many ways. In this project, the attempt is to overcome these challenges using a privately collected dataset. Several explicit choices where made to focus on user's stationary and frequently visited locations only. Furthermore, another design choice was made to condense data points using time-spent feature as summary in order to prevent long visited locations to overshadow the remaining data points.

There is much room for improvements, however. These include:

- Generalization of the prediction model to work not only over location data but over other types of actives (e.g., predicting what app the user runs on his/her cellphone next).

- Expansion of the prediction to consider multiple users (perhaps those who are related as friends may perform similar activity when together).

- Improve the classification process using new 'hand-crafted' features, perhaps ones based on user profession, age, or gender.

Appendix A - Code Organization

The code is organized in two files:[4]

- **Capstone-project.ipynb** contains the code for data pre-processing and exploration, as well as implementation of the naive and decision tree classifier classes.
- **find_address.py** implements auxiliary functions to lookup an address and to find a label for a coordinate using Google nearby search.

Appendix B - Software Libraries and Dependencies

- Python 3.5.2
- numpy 1.13.1
- pandas 0.20.3
- sklearn
- matplotlib 2.1.0
- mpl_toolkits.basemap 1.0.7 (see http://basemaptutorial.readthedocs.io/)
- geopy 1.11.0 (see https://github.com/geopy/geopy)
- seaborn 0.8.1

[4] Both files contain code that is developed in Python 3.5.2.