Classification Challenge
for
MLC 16-17

Indoor Localization Using WiFi

by Luis I. Lopera

Date: October 25, 2016
Supervisors: Prof. Dr. Oliver Amft, Luis I. Lopera

1 Introduction

Some methods of Indoor localisation propose to use the already installed WiFi access points to pin point the location of a user. For example, trilateration based on the power of the received signal was used by Aboodi et al. [1]. However, for the trilateration approach to be useful, the precise location of the access point must be known. In large buildings, the presence of access points is often arbitrary and not properly documented. As a result, trilateration can not be successfully applied.

Alternatively to trilateration, the power levels of all access points visible in a single location can be recorded. The set of power level values for a location is called a WiFi fingerprint. To overcome the problem of documenting every access point, Kawajari et al. [2] propose a method using WiFi fingerprints to describe locations of interest, and study different methods of classification to accurately detect the location based on the measured WiFi fingerprint. Additionally, Kawajari et al. [3] used crowd sensing to collect measurements.

2 Challenge description

In this challenge, you are asked to apply classification algorithms to correctly map WiFi fingerprints to location labels. The dataset recorded by Kawajari et al. [3] will be used for the analysis. You should explore the classification algorithm performance using classification metrics, as well as distance between the estimated location and the labelled location. Compare your results with those achieved by Kawajari et al. [2], [3]. Here are recommended steps on how to proceed:

1. Set up your working environment. Load the dataset and apply the code snippets provided below to obtain a first classification result.
2. Perform a statistical analysis of the dataset to derive relevant features. You will need to deal with missing data too. Find the best access points to use in classification.
3. Develop hypotheses on what features, data preprocessing (e.g., standardisation, normalisation), classification algorithm, etc. to choose such that the classification performance is maximised.
4. Proceed in iterations to test and revise your hypotheses. Document each iteration in your report (configuration of the classification, performance achieved, e.g. diagrams/tables) and provide a short discussion on how to interpret the results with respect to the initial hypothesis.

3 Results

Analyse the classifier accuracy. In addition, you can measure the error in meters from your classification result to the actual label, using the provided equation. Document all your work and results in the report.

4 Dataset

The data set consists of annotations of 16 participants who recorded fingerprints for 190 locations during a 10 week period. To simplify the classification task, consider all participants jointly and apply validation splits of training and testing over the entire dataset.

The dataset can be found in the zip file under the folder SCSUT2014v1. Folder SCSUT2014v1/eng2 contains pictures and geographical data. The experiment data is located in folder SCSUT2014v1/EXP201312. The fingerprint data is stored in timestamped json files in individual folders per participant, e.g., SCSUT2014v1/EXP201312/StampRally_e0/Uploads/TraeinData. More information about the dataset can be found on SCSUT2014v1/README.html.

To speed up your development time, two python modules have been prepared to parse the dataset: configuration.py and repository.py. The following code snippet illustrates how to load the entire dataset in data-labels format:

```
from repository import Repository
from configuration import config
repository = Repository(config)
dataset, labels = repository.get_dataset_and_labels()
```

It is recommended that you use cross validation. Make sure to apply appropriate validation when tuning hyper-parameters. The following code snippet illustrates how to separate the dataset in training and testing. Furthermore, it uses cross validation for parameter tuning. You are encouraged to extend the code to also use different splits of the dataset to produce final results.

```
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.svm import SVC
from sklearn.cross_validation import ShuffleSplit
from sklearn.grid_search import GridSearchCV

# Ensure that there are no NaNs
dataset = dataset.fillna(-85)

# Split the dataset into training (90 \%) and testing (10 \%)
X_train, X_test, y_train, y_test = train_test_split(dataset, labels,
                                                    test_size = 0.1 )

cv = ShuffleSplit(X_train.shape[0], n_iter=10, test_size=0.2,
                  random_state=0)
```

```
16
17 # Define the classifier to use
18 estimator = SVC(kernel='linear')
19
20 # Define parameter space.
21 gammas = np.logspace(-6, -1, 10)
22
23 # Use Test dataset and use cross validation to find bet hyper-parameters.
24 classifier = GridSearchCV(estimator=estimator, cv=cv,
25                            param_grid=dict(gamma=gammas))
26 classifier.fit(X_train, [repository.locations.keys().index(tuple(l)) for
27                          l in y_train])
28
29 # Test final results with the testing dataset
30 classifier.score(X_test, [repository.locations.keys().index(tuple(l)) for
31                           l in y_test])
```

4.1 Labels and error in meters

Every location is labelled with longitude and latitude coordinates. Therefore, when comparing the classification output with the label you can estimate how far the estimated location is from the actual labelled location. Equation 1 shows an approximation of how to convert the difference between two labels' longitude and latitude into an error E_m in meters using the earth average radius R in meters.

$$E_m = R\sqrt{\left(\frac{\pi}{180}\Delta latitude\right)^2 + \left(\frac{\pi}{180}\Delta longitude\right)^2} \tag{1}$$

The recording took place trough several floors of the evaluation building. Thus, in addition to longitude and latitude, every location label also contains the corresponding floor. The additional information can be used to reduce the number of candidate locations. For example, using hierarchical classifiers and given a new fingerprint, first select a floor and then the corresponding location from those available in the selected floor.

For some classifiers, labels have to be numerical values. If required, use the index of the label as numerical value. The values can subsequently be used to convert the classifier's output back into the correct label format. The following code snippet shows how to use the label list found in the repository for label conversions.

```
1 # Convert to numerical labels
2 numerical_labels = [repository.locations.keys().index(tuple(l)) for
3                     l in y_train]
4
5 # Convert back to coordinate labels
6 coordinate_labels = [repository.locations.keys()[i] for i in c_output])
```

5 Administrative aspects

5.1 Working in student groups

The challenge shall be approached by teams of 3-4 students.Please register your group before Oct 31, 2016 by e-mail to Giovanni Schiboni <Giovanni.Schiboni@uni-passau.de> stating student names and IDs. Discuss your approach in the team, work in parallel where useful and consolidate your findings in the report.

5.2 Relevant literature material

References for initial literature are provided in this project description. If needed, additional literature should be searched and considered independently. It is strongly recommended to keep a literature database and add comments on conclusions of individual papers, e.g. using JabRef.

5.3 Software tools

The following list of software packages are useful to solve the challenges, but not mandatory to use. The code snippets presented in this guide utilise these packages.

- Python [4].
- IPython [5].
- Pandas [6].
- Scikit-Learn [7][8]

5.4 Challenge advisor consultations

Team meetings with the challenge advisor shall be planned to discuss achieved work results, define next steps. The team is asked to attend with all members and *thoroughly prepare* for the meeting, including presenting the work already performed and proposed next steps. Meeting have to be scheduled at least two days in advance. Walk-in consultation is discouraged.

5.5 Project documentation, presentations, and report

It is recommended to keep an adequate record (e.g. in a draft report) of project results, conclusions, and decisions made. This information can be used to prepare the final report and the presentation.

A report should be submitted by or before the deadline. See Section 5.6 for details on the presentation dates. The report shall describe the team's effort in the challenge (prepared using LaTeX, in English language). The report should be max. 5 pages long, plus cover page. Submissions which exceed the number of pages will NOT be accepted. A report template will be provided on Ilias. The challenge report shall be accompanied by an electronic appendix (e.g., zip file), containing all literature used, code developed, or otherwise acquired during the challenge. It should be possible to reproduce your results with the code. A brief documentation shall be added to navigate in the electronic appendix.

Assessment criteria for the report are (1) quality (specifically including the following aspects: hypotheses, rationale, analysis, conclusions), (2) clarity & systematic working style, (3) quantity of results.

At the end of the course, teams are invited to give a presentation of their results for a selected challenge. Which team will present the challenge as well as detailed instructions on the presentation will announced after all challenges are completed. The presentation shall illustrate the method used, the strong and weak points of the approach, the results obtained, and conclusions.

5.6 Important dates for this challenge

- The Challenge starts as the task description is made available.
- Challenge report and electronic appendix is due by Nov 13, 2016. Please upload the report and the electronic appendix to illias. No submission will be accepted after the deadline.
- The final presentation will be on Feb 08, 2016. The presenting teams for this challenge will be announced once all the challenges are completed.

References

[1] A. Aboodi and T.-C. Wan, "Evaluation of wifi-based indoor (wbi) positioning algorithm," in *Mobile, Ubiquitous, and Intelligent Computing (MUSIC), 2012 Third FTRA International Conference on*, June 2012, pp. 260–264.

[2] R. Kawajiri, M. Shimosaka, R. Fukui, and T. Sato, "Frustratingly simplified deployment in WLAN localization by learning from route annotation," in *Proceedings of the 4th Asian Conference on Machine Learning, ACML 2012, Singapore, Singapore, November 4-6, 2012*, 2012, pp. 191–204. [Online]. Available: http://jmlr.csail.mit.edu/proceedings/papers/v25/kawajiri12.html

[3] R. Kawajiri, M. Shimosaka, and H. Kashima, "Steered crowdsensing: Incentive design towards quality-oriented place-centric crowdsensing," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14. New York, NY, USA: ACM, 2014, pp. 691–701. [Online]. Available: http://doi.acm.org/10.1145/2632048.2636064

[4] Python. [Online]. Available: https://www.python.org/

[5] Ipython. [Online]. Available: http://ipython.org/

[6] Pandas. [Online]. Available: http://pandas.pydata.org/

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] Scikit learn: Machine learning in python. [Online]. Available: http://scikit-learn.org/

[9] M. Pilgrim, *Dive into Python*. Berkeley, CA; New York, NY: Apress ; Distributed to the Book trade in the United States by Springer-Verlag New York, 2004. [Online]. Available: http://www.diveintopython.net/

[10] Pycharm. [Online]. Available: https://www.jetbrains.com/pycharm/