Pervasive and Mobile Computing 9 (2013) 772–783

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Feature engineering for semantic place prediction

Yin Zhu [a,*], Erheng Zhong [a], Zhongqi Lu [a], Qiang Yang [a,b]

[a] Computer Science and Engineering Department, Hong Kong University of Science and Technology, Hong Kong
[b] Huawei Noah's Ark Research Lab, Hong Kong

ARTICLE INFO

Article history:
Available online 19 July 2013

Keywords:
Feature engineering
Domain knowledge
Feature selection
Classification

ABSTRACT

We present in this paper our winning solution to Dedicated Task 1 in Nokia Mobile Data Challenge (MDC). MDC Task 1 is to infer the semantic category of a place based on the smartphone sensing data obtained at that place. We approach this task in a standard supervised learning setting: we extract discriminative features from the sensor data and use state-of-the-art classifiers (SVM, Logistic Regression and Decision Tree Family) to build classification models. We have found that feature engineering, or in other words, constructing features using human heuristics, is very effective for this task. In particular, we have proposed a novel feature engineering technique, *Conditional Feature* (CF), a general framework for domain-specific feature construction. In total, we have generated 2,796,200 features and in our final five submissions we use feature selection to select 100 to 2000 features. One of our key findings is that features conditioned on fine-granularity time intervals, e.g. every 30 min, are most effective. Our best 10-fold CV accuracy on training set is 75.1% by Gradient Boosted Trees, and the second best accuracy is 74.6% by L1-regularized Logistic Regression. Besides the good performance, we also report briefly our experience of using F# language for large-scale (∼70 GB raw text data) conditional feature construction.

1. Introduction

MDC Task 1 [1] is to infer the semantic category of a place. A good solution to this task can help various applications in Mobile Computing and Location-based social networks (LBSNs). For example, location recommendation on either web or mobile phone can be greatly enhanced with the understanding of the semantic meaning of the places [2]. Nokia Mobile Data Challenge (Nokia MDC) provides the largest and the richest public dataset for researchers to study this problem. The dataset contains 112 users' mobile sensing data and each type of data may contribute discriminative power to Semantic Place Prediction. For example, using time context alone, we would be able to distinguish working places and homes accurately because after midnight we usually stay at home. In addition to time context, call log and other sensor data recorded in the user's smartphone during the user's stay at a place may also indicate the type of it.

We formulate this task as a classification problem, where each user–place pair is presented as a feature vector and the location categories are considered as targets. Our main strategy is to extract as many useful features as we can from the sensor data and then build accurate classifiers using these features. By useful, we mean features that have discriminative information among the ten predefined location categories: *Home, Home of a friend, My workplace/school, Transportation, Workplace of a friend, Outdoor sports, Restaurant or bar, Shopping* and *Vocation spot*. Once the features are generated, we use state-of-the-art classifiers, e.g. SVM and Decision Trees, to decide how to form models for place category classification.

* Corresponding author. Tel.: +852 54107200.
E-mail addresses: yinz@cse.ust.hk (Y. Zhu), ezhong@cse.ust.hk (E. Zhong), zluab@cse.ust.hk (Z. Lu), qyang@cse.ust.hk (Q. Yang).

1574-1192/$ – see front matter
http://dx.doi.org/10.1016/j.pmcj.2013.07.004

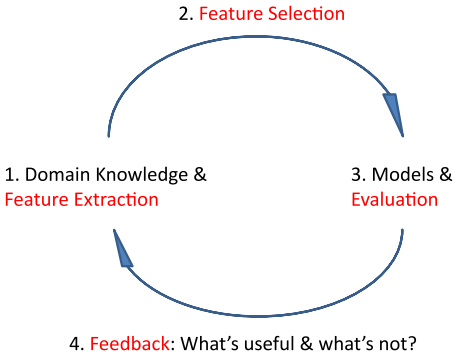

Fig. 1. The workflow for feature engineering.

This classification task is different from some classification tasks in previous data competitions. For instance, KDDCUP2009 organized by Orange[1] released a data table of 50 000 data samples by 15 000 attributes. However, neither the meaning of these 15 000 attributes is known to the contestants, nor any auxiliary data is available for exploring the domain knowledge to generate more features. In such a competition, the challenge is limited to train the best classifier, or the best ensemble of classifiers [3], rather than to find the most effective way to tackle a specific data mining task. Nokia MDC releases the mobile phone data in their raw forms, which provide us the possibility to know more about the data mining task itself and to explore feature engineering ourselves.

Garbage in and garbage out—no machine learning method is able to construct an accurate classifier with low discriminative features. We believe that feature construction would be much more important than other techniques, e.g. classifier ensemble. If we missed several important features, then no matter what classifier we use, we would fail to achieve a good performance in classification. Therefore our effort is mainly on feature engineering—for each kind of sensing data, we use our domain knowledge and the state-of-the-art feature construction methods in the literature to generate as many discriminative features as possible.

In particular, we have designed a novel method, *Conditional Feature* (CF), to explore the dependency and correlation between sensor data and the conditions under which they are recorded. For example, for each place, we can calculate the average WiFi signal strength during the user's multiple stays at it. This is a single value feature, and it does not differentiate under what conditions the WiFi strengths are collected. Thus we call it an *Unconditional Feature* (UF). A user's smartphone receives multiple WiFi signal strengths during a single stay, and the user usually visits this place many times. Compressing many signal strengths into a single value, the average signal strength obviously loses information. The Conditional Feature method helps calculate multiple variations of average WiFi signal strengths under different contexts. Each version is on some conditions, e.g. whether the WiFi strength is in a trusted stay/interval[2] or not. The WiFi strength recorded at the trust intervals are more reliable to use, while the strength at untrust intervals may still provide some information. The best way is then to use both of them, but split the conditions. The most important condition is probably time—whether the WiFi strength is recorded during weekend, or in a morning, and so on. By using the CF method, we are exploring the relationship between the average WiFi strength and the conditions when WiFi strength is recorded. In our experiments, we find that conditional features, especially those conditioned on fine-granularity time intervals, are far more effective than their unconditioned versions in classification.

Our contributions are summarized as follows.

1. We propose the Conditional Feature method to explore the relationship among sensor readings and the conditions under which they are recorded. And in our experiments, we find that time conditions are very useful in semantic place classification.
2. We analyze what features are useful for place type prediction. Although most of the features are already proposed in the literature of mobile computing, our work is a systematic analysis on their usefulness in this specific task, therefore provides important insights on semantic place prediction.

We show the whole feature engineering workflow in Fig. 1, which also indicates the structure of the rest of our paper: after a review of related works, we will focus on feature construction and Conditional Feature method first, and then describe how we do feature selection to improve the performance of the classifiers, and at last show the experimental results and more importantly some insights on what kinds of features are very useful for semantic place classification. As the arrows in Fig. 1 indicate, feature engineering is an iterative process. For the clarity of presentation, what we have written in the following is the final product of that many iterations. Meanwhile, we do include a small section on the programming practices that

1 http://www.kddcup-orange.com/.

2 A visit to a place is *trusted* if there are short transitions before and after the visit; otherwise the starting time and ending time of the visit cannot be confirmed or trusted.

helped us to make this iterative process effective and efficient. Our experience, especially in tooling and programming such as Asynchronous IO and data parallelism in F#, may be more valuable to those who endeavor on similar large-scale data mining tasks.

2. Related work

When talking about semantic place prediction, we usually expect to be given relatively accurate location coordinates of the places so that external information such as Point-of-Interests (POIs) or even check-ins from social networks can be used to infer the location type [4–8]. The semantic place prediction in Nokia MDC Task 1 is very different from these existing research works in its motivation—the semantic meaning of a place is inferred from the smartphone usage patterns rather than from GPS locations. The result of our work would be useful when the accurate location coordinates are not available and would be a good complementary to existing methods using accurate location coordinates and social media enrichments.

There are two related research areas: sensor-based activity recognition and practices in data mining competitions.

1. Activity recognition. Early research on activity recognition [9–11] usually studied on the sensor data from a single sensor and a single person. For example in [10], WiFi sensor readings from one subject are used to infer the place category and the goal of a route. Smartphones have matured in the recent few years, and the rich sensors they equip allow heterogenous sensing with multiple sensors. Also the increasing popularity of smartphones makes collecting large-scale mobile sensing data much easier than before. For example, CenceMe [12] project studied how multiple sensors in the smartphone can jointly predict human activity. The mobile dataset shared in Nokia MDC is by far the largest, and the dedicated tasks provide researchers a platform to compete on important mobile computing problems with wide applications. In particular, this paper provides a first-hand experience report on Semantic Place Prediction, including the most useful features and best practices in building classifiers. Our Conditional Feature Engineering method is also novel in the existing activity recognition research literature (most of the papers do not focus on features while in practice feature engineering is usually the black magic for systems to achieve the best performance [13]).
2. Practices in data mining competitions. When we prepare Nokia MDC, we read the reports from past data mining competitions that our research group participated in [14–16] and a few by other researchers [3,17]. Some of the past competitions also require extensive feature engineering, e.g. KDDCup'05 [14], while others focus on prediction model building only, e.g. KDDCup'09 [3]. Similar to feature enrichment in [14], our Conditional Feature method is also a semi-automatic method for feature construction—once the conditions and the base features are specified, different versions of base features can be extracted without any human interference. We believe our method will be very suitable for large-scale machine learning [18,19] with more than tens of millions of training instances where linear classifiers are always applied.

3. Conditional Feature Engineering

As stated above, before exploiting supervised learning algorithms, we need to construct features from raw data and represent each user–place pair as a feature vector, i.e., $\mathbf{x} = \{x_i\}_{i=1}^{m}$, where x_i represents the i-th value of the instance $\mathbf{x}$ and m is the number of features. We first briefly introduce the Nokia MDC dataset, then formally define a feature construction framework, and finally present constructed features in detail.

3.1. Raw data from Nokia

The raw data are provided by Lausanne Data Collection Campaign as described in [1] and organized in the format of.csv files. Each user has a profile including 12 files. A file contains a table, in which each row represents a record such as a phone call. The fields of each table file are described in Table 1.

3.2. Feature construction framework

Without explicit prior knowledge on inferring the semantic meaning of a place, we aim to construct as many features as possible and then perform feature selection and extraction. Since there are infinite ways to perform feature construction, to make the feature construction process tractable, we formulate it as a unified paradigm: Conditional Feature (CF). CF defines each feature as the probability that one user action happens under given conditions. Formally, each feature is $Pr(a|u, p, \Theta)$, where a is one action, u is a given user, p is a place and Θ is the set of conditions, such as weekend/weekday. Then, the task of feature construction can be divided into three subtasks:

- Enumerating as many user actions as possible.
- Defining appropriate conditions.
- Computing the conditional probabilities.

Firstly, we categorize the actions in the raw data into three cases: continuous variables, such as the time length of calling (in calllog.csv) and the avdelt value (in accel.csv); independent discrete variables which are different for different users, such

Table 1
Raw data provided by Nokia.

File name	Fields description
accel.csv	user ID, time, motion measure, and accelerometer samples.
application.csv	user ID, time, event, unique identifier of the application, and name of the application.
bluetooth.csv	user ID, time, first 3 bytes of MAC address, anonymized MAC address, anonymized name of the Bluetooth device.
calendar.csv	user ID, time, entry ID, status (tentative/confirmed), entry start time, anonymized title, anonymized location, entry type (appointment/event), entry class (public/private), last modification time of the entry.
calllog.csv	user ID, call time, call type (voice call/show message), SMS status (delivered, failed, etc.), direction (incoming, outgoing, missed call), international and region prefix of phone number, anonymized phone number, indicator if number is in phone book, call duration.
contacts.csv	user ID, creation time, anonymized name, international and region prefix of phone number, last modification time.
gsm.csv	user ID, time, country code and network code, anonymized cell id, anonymized location area code, signal strength.
mediaplay.csv	user ID, time, album name, artist, track, track title, track location, player state, track duration.
media.csv	user ID, record time, media file time, anonymized media file name, file size.
process.csv	user ID, record time, path name of running process.
sys.csv	user ID, time, current profile (normal, silent, etc.), battery level, charging state, free drive space, elapsed inactive time, ringing type (normal, ascending, etc.), free ram amount.
wlan.csv	user ID, time, first 3 bytes of MAC address, anonymized MAC address of WLAN device, anonymized SSID, signal level, channel, encryption type, operational mode.

as the number of places[3] visited by users (in visit_sequence_10min.csv); and common discrete variables which are shared by every user, such as the call type, e.g., short message and phone call (in calllog.csv). For the continuous variables, we extract their means and variances. For example, we extract the mean and variance of the call length for each user. For the common discrete variables, we record the frequency of each discrete value for each user. For example, we extract the number of calls, number of messages, and the like. For the independent variables, we retain the ratio of the top-5 values as features. For example, we extract the running time ratio of top-5 frequent applications for each individual user. We summarize the extracted actions or statistics in each .csv file as follows:

- accel: the mean and variance of acceleration records,
- application: the three most frequently used applications,
- bluetooth: the three most frequently used Bluetooth prefixes,
- calendar: private appointment,
- call log: the most frequent call regions, the outgoing, incoming and missing calls, the message and voice calls, the duration of calls,
- contacts: the phone prefix,
- gsm: the time staying at home,
- media: the mean and variance of the sizes of media files,
- mediaplay: listening to songs from a particular artist and listening to a particular song,
- process: usage of applications,
- distance: moving length and maximum moving speed,
- sys: space left in the drive D and time being in silent mode,
- wlan: most frequently used brands and average WiFi signal strength,
- visit: most frequently visited places.

Secondly, we are aware that people behave quite differently in different time. For example, in the weekdays people go to work, while they rest at home during the weekends. Therefore, we proposed three conditions based on the temporal information as follows:

- *dw*, weekday/weekend;
- *td*, time of the day;
- *trust*, under trust interval or not, which is given in the raw data as an indicator of whether the record is trustful.

These three conditions are independent to each other. Motivated by "bag of words" [20], we estimate the probability $Pr(a|u, p, \Theta)$ based on counting. In other words, we compute this probability by calculating the number of given actions for each user under given conditions. Using the Bayesian rule, we obtain the following computing equations:

$$Pr(a|u, p, \Theta) = \frac{Pr(a, \Theta, u, p)}{Pr(\Theta, u, p)} \propto \frac{n_{a,u,p,\Theta}}{n_{u,p,\Theta}} \tag{1}$$

where $n_{u,p,\Theta}$ is the number of records under conditions Θ for a given user–place pair and $n_{a,u,p,\Theta}$ denotes the number of action a that the user u takes at the place p under conditions Θ. These statistics can be calculated directly from the raw data in csv files. Based on the conclusion in [20], this estimation equals maximum likelihood estimation.

[3] According to MDC task description [1], a place is a cluster of near GPS points and is denoted by a unique ID. In this way, the privacy of users is preserved for the real GPS locations are substituted by anonymous place IDs.

Let us use the *average WiFi signal strength* feature to illustrate the idea of conditional feature construction. This feature is simply the mean value of strength from all the WiFi points that the user's smartphone encountered at place p. We denote this value as $v_{\text{wifi\_str}}$ and call it an *Unconditional Feature*. To calculate conditional versions of $v_{\text{wifi\_str}}$, we split the whole WiFi records from user u's at place p into groups defined by the conditions: *td*, *dw* and/or *trust*, and use the following notations to represent them: $Pr(v_{\text{wifi\_str}}|u, p, \{td, dw, trust\})$, where $\{td, dw, trust\}$ is the combination of these conditions. There are three conditions, therefore $2^3 = 8$ combinations of conditioning in total. One of them, when all three conditions are turned off, is the Unconditional Feature. Obviously *dw* and *trust* are binary conditions. The third condition, time of the day, requires more discussion. We discretize time of the day by splitting a whole day into a number of equally-sized intervals. We have tried different numbers from 4 (every 6 h) to 144 (every 10 min), and we choose the best number by cross validation (CV). For one feature, the 8 combinations of conditions generate $(8 + 9\ ntd)$ new features, where *ntd* is the number of conditional time intervals during a day, e.g. when the time interval is set to 30 min, $ntd = 48$.

3.3. Feature engineering

In the previous section, we use the average WiFi signal strength feature to introduce Conditional Feature Engineering. In this section, we describe in detail more features that we find to be useful in experiments. We define a *super interval* for a pair (u, p) as the intervals in *visit_sequence_10min.csv* that are labeled as place ID p for user u. *visit_sequence_10min.csv* contains all the intervals with over 10 min duration for some user. Thus all features for a user–place pair (u, p) are constructed from the contextual raw data within this super interval. In the following, we list important features for each type of contextual data.

3.3.1. Time features

We first calculate the summary statistics of the staying durations at this place. They are mean, variance, min, max, and the second maximum. One of the summary statistics is the average value of duration, which is an obvious feature to classify many places, say home and a coffee shop. The motivation is that people may stay in different kinds of places with different time durations.

3.3.2. Accelerometer features

Each data point from the 3D accelerometer is an (x, y, z) tuple, and it is common practice [21] to use its signal strength to calculate features:

$$\text{strength} = \sqrt{x^2 + y^2 + z^2} - 680,$$

where 680 is the unit gravity in the Nokia phone's accelerometer.[4] For a one-minute sequence of accelerometer strength, we extract the following five features:

- Average of strength.
- Variance of strength.
- Energy of strength (average of strength squares).
- Sum of FFT coefficients of the strength sequence.
- Weighted sum of FFT coefficients of the strength sequence.

In our previous study [22], we found that these six features are very effective to discriminate people's daily activities.

3.3.3. Application features

The numbers of application "close", "foreground", "started" and "view" in application.csv files are counted, as people may use applications under different status when they are visiting different places.

3.3.4. Bluetooth and WLan features

The number of different Bluetooth/WLan devices is recorded. For the top-five frequently encountered devices, their ratios relative to the total number of scanned devices are also calculated. The summary statistics of the signal strength are also used as features.

3.3.5. Call log features

We extract most of the call log features as described in Table 3 in [23], a previous study by Nokia Research and Idiap on this dataset. The features include the ratio between incoming and outgoing, the ratio of missed calls, etc. Intuitively, these features capture important calling/messaging behavior at a place.

3.3.6. System features

There are nine statuses of a phone (e.g. charging and silence). The ratio of the status at a place has a high indication of the place type. For example, during working, we usually switch the phone to meeting/silent.

[4] Some other phones use 9.8 or 0.98 as the unit gravity, that depends on the accelerometer and OS the phone uses.

3.3.7. Media features

We calculate the number of songs the user has listened to at a place. We also calculate the time length. For example, in a working place, people may listen to less music than at home.

3.3.8. Bag-of-words (BoW) features

The counts such as how many times the user has used the "Message" application and how many times the user has scanned a specific MAC address during his/her stay at a place may also be quite useful. Following the conventions in text processing and classification, we call *application names*, *country prefixes of call numbers*, and *MAC prefixes of Bluetooth and WiFi* as *words*, and we use their counts within a specific place to represent a *document*.

3.4. Feature postprocessing

There are three postprocessing steps for the features defined above.

1. Feature normalization. Some of the above features are already normalized, e.g. the average WiFi signal strength; some of them are not, especially those raw counts, e.g. the number of calls at a place. If a place is often visited for a long time, such as home or office, then some count features at that place are not comparable to those at a short visited place such as a coffee shop. Therefore besides the original count features, we also create two more normalized versions of them by *dividing* them by two factors: the number of visits to place p by user u and the total duration of the user's visits.
2. Construct features via the Conditional Feature method. As calculated before, each feature has $(8 + 9\ ntd)$ conditional versions.
3. Filtering and standardization. In addition, some of the constructed features may have non-zero values in only one or two users, which has no discriminability. Thus, we set a threshold for each feature. For a given feature, if there are only two non-zero values, it is removed. Then, for the rest features, we standardize them into $[-1, 1]$ using

$$x_i^t = 2\frac{x_i - x_{\min}}{x_{\max} - x_{\min}} - 1 \tag{2}$$

where $x_{\max}$ and $x_{\min}$ represent the maximal and minimal values of the correspondent feature.

4. Feature selection and classifier building

4.1. Feature selection

After conditional feature construction, we generate amounts of features, approximately two million, when the conditional time interval is set to 30 min. Compared to the limited numbers of instances, this high dimensionality of the generated features may cause the model to overfit. In addition, building a classifier with many features costs much computational time. Therefore we need feature selection to eliminate the less relevant features, and we perform the following two stages of feature selection.

First, we drop those features with more than 99% percent of zeros. This filter typically reduces the number of features to about $6k$ to $30k$ depending on the size of conditional time interval. Because the number of training samples is only 366, the resulting data table can be fed to most classifiers.

Second, we use the Relief feature selection method [24] to select 50–2000 most relevant features. We also use L1-regularized Logistic Regression (L1-LogReg) as a feature selection method. L1-LogReg is effective at forcing many redundant features to have zero weights, i.e., these features have no effect in the prediction model and therefore can be discarded. We can control the number of features with non-zero coefficients by changing the regularization coefficient C in L1-LogReg.

4.2. Classifiers and parameters

To build prediction models, we use four state-of-the-art classifiers, Logistic Regression (LogReg) [25], Support Vector Machines (SVM) [26], Random Forests (RF) [27], and Gradient Boosted Trees (GBT) [28]. These four classifiers are among the best classifiers in practice [29]. In the following, we describe how the parameters of the classifiers are tuned.

4.2.1. Logistic Regression (LogReg)

We use the LibLinear package [30], which implements both L1-regularized and L2-regularized Logistic Regressions. Logistic Regression has only one parameter, the regularization coefficient C. We vary this value from 10^{-6} to 10^3. When C is set to be a large value, most of the regression coefficients are set to zero by the optimization.

4.2.2. Support Vector Machines (SVM)

We use kernel SVM implemented in the LibSVM package [31]. We use three kernels: linear kernel, polynomial kernel, and RBF kernel. The regularization coefficient C follows the same setting as that in Logistic Regression. For kernel-specific parameters, we vary the degree parameter in polynomial kernel from 2 to 10 and vary the window parameter in RBF kernel from 10^{-3} to 10^3.

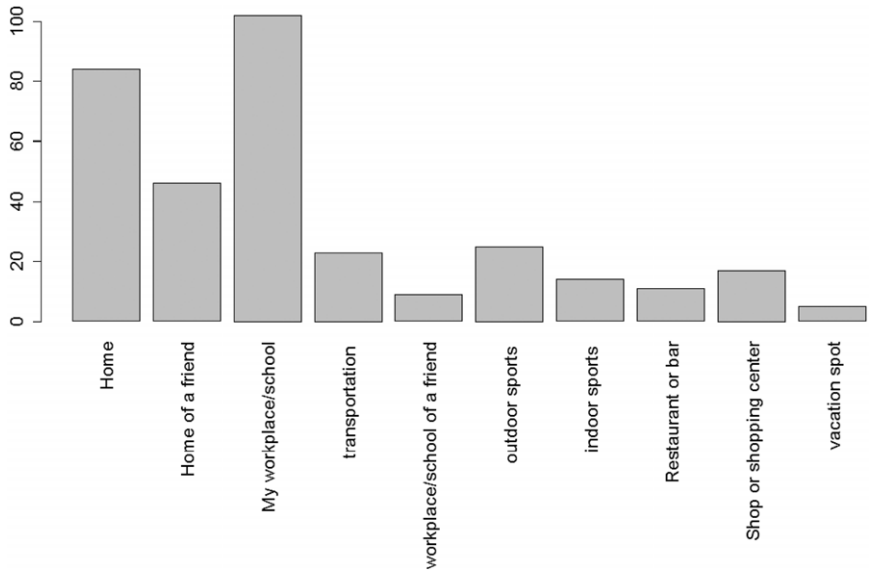

Fig. 2. Class distribution among the 366 training samples.

4.2.3. Gradient Boosted Trees (GBT)

We used an in-house gradient boosted tree implemented by Nathan N. Liu, who also used it successfully in his winning solutions in KDDCup'2011 [15] and WSDM Challenge 2012 [16]. There are two main parameters in GBT: the max depth of a tree and the number of boosting iterations. We vary the first parameter from 2 to 10. The GBT tool calculates the cross validation accuracy at each iteration as well; therefore we may choose the best cross validation accuracy from the first 1000 iterations.

4.2.4. Random Forest (RF)

We use the Random Forest implementation in Weka [32]. We set the number of random features from $0.1\sqrt{m}$ to $10\sqrt{m}$, where m is the number of features.

4.3. Imbalanced classification

In Fig. 2, we plot the label distribution from the total 336 training place IDs. There are three major classes: *Home, Home of a Friend*, and *My Workplace/School*, which occupy about 2/3 of the total samples. Although the data are imbalanced, we still use unit weight over class labels. The main reason is that the evaluation metric is classification accuracy, which is defined as

$$\text{accuracy} = \frac{\#\text{ of corrected classified}}{\#\text{ of total test samples}}.$$

Assuming that the places in the testing data have similar labeling distribution with that of the training data, we should assign the same weight for majority classes and minority classes.

5. Experiments

Because Nokia MDC gives no feedback or leaderboard on the submitted predictions of the testing data, we must rely on the training data for model evaluation. Following the common practice, we use the accuracy from 10-fold cross validation (CV) on the training data as the evaluation criterion to select the best models. We also notice that the testing data come from 32 users who are different from the 80 users in the training data. So when we split the training data samples to 10 folds, we restrict that the labeled places from one user all go to one fold. In this way, the prediction model for each testing fold is trained using labeled data of different users in that fold.

5.1. The effectiveness of feature selection

When we set the conditional time interval to 30 min, we got totally 2 796 200 features. With so many features, not only the computational time of classifiers increases, but the accuracy may also drop. In particular, we find that when there are many features, the performance of the classifiers may be very sensitive and unstable to its parameters. In Fig. 3, we plot the 10-CV accuracies of LogReg models on two sets of features, the full set and 2000 of them selected by the Relief method.

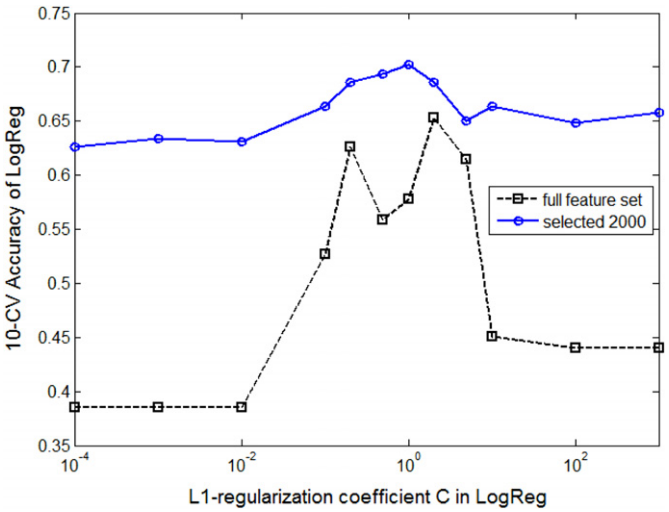

Fig. 3. Effectiveness of feature selection.

Table 2
Performance of different classifiers.

Datasets	# of selected features			
	Relief		L1-LogReg	
	1000	2000	1132 ($C = 10$)	1925 ($C = 1$)
LogReg	0.697	0.702	0.735	**0.746**
SVM-RBF	0.645	0.639	0.669	0.672
SVM-poly	0.656	0.648	0.669	0.637
SVM-linear	0.680	0.678	0.675	0.683
GBT	0.727	**0.751**	0.730	0.740
RF	0.702	0.719	0.724	0.716

When we change the regularization coefficient C in LogReg models, the models trained on the full feature set are unstable—a small change in C may result in a big fluctuation on the 10-CV accuracy. In this case, we are not sure about which is the best parameter for the test data. The performances of the models trained on the good feature subset are quite stable even for extreme C values.

We also compared two feature selection methods, Relief and L1-LogReg, and we found that the performance of L1-LogReg can be greatly boosted by using a feature selection of L1-LogReg first, while other classifiers perform closely with these two selection methods. Table 2 shows the 10-fold CV results in detail.

We use 10-fold CV on the 366 training samples to select the best parameters for each classifier. We set the conditional time interval to 30 min in all the experiments.

As in Table 2, we find that LogReg and GBT perform the best, RF a close third, but all SVM classifiers perform much worse than others. One possible reason is that both feature selectors work in a linear manner and hence the selected features may not be suitable for kernel-based methods.

5.2. The size of the conditional time interval

In all the above experiments, we set the conditional time interval to 30 min. To study the influence of the size of it, we change it from 6 h to 10 min, and plot the accuracy changes of two classifiers LogReg and GBT in Fig. 4. The feature selection method used in both classifiers is L1-LogReg with $C = 1$. To illustrate the usefulness of conditional features, we also plot the accuracy of the unconditional features as a reference. Classifiers trained on conditional features clearly outperform those trained on unconditional features. And when the time interval is too coarse, e.g. 6 h per interval, the accuracy drops close to that of unconditional features.

5.3. Most useful features

Besides the accuracy from 10-fold cross validation, it is probably more interesting and illustrative to show the most useful features. And by checking these features, we can have a better understanding of the mobile data.

In Table 3, we show the feature ranked by GBT, the classifier with the best 10-fold CV accuracy. The algorithm for feature ranking is described in [33]. Table 3 shows the most useful features and their relative scores. We can find that the number

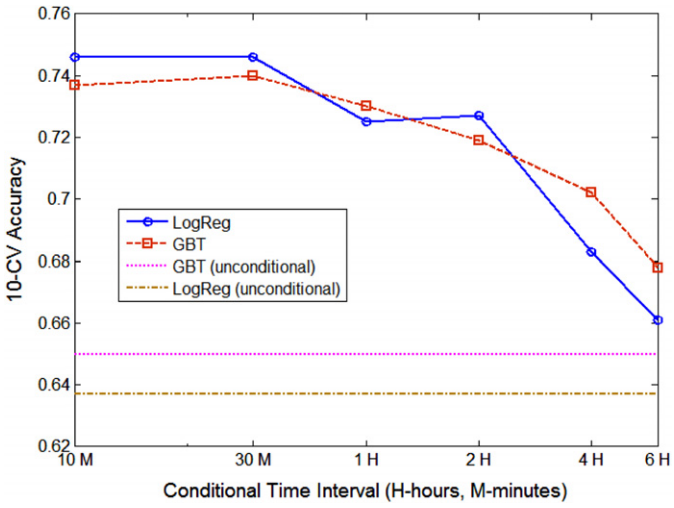

Fig. 4. Performance with different lengths of time interval.

Table 3
Weights of different features in GDT.

Morning 8–10 am	100.0
mac_r1	71.57
Ratio of Sunday	67.47
Evening 8–10 pm	53.93
Accelerometer FFT	53.76
Num Bluetooth	44.74
N night	39.57
N morning	39.41
Duration div by max	39.30
Ave signal	36.6
Acc mean	34.26

of visits to a place during 8–10 am and 8–10 pm are very useful. Two unconditional features from accelerometer records (FFT and mean signal strength) are also very useful. The second feature with codename mac_r1 means that the ratio of the most frequent MAC address the user ever encountered has occurred in that particular place. Intuitively, the place label My workplace/school has higher values and other labels have lower values.

It would be illustrative to show some explainable classification rules. We build a classification model for the three majority classes (*Home*, *WorkPlace* and *Others' Home*) using R's *party* package and plot the decision tree in Fig. 5. From the top nodes in the decision tree, we can see that time-related features, an accelerometer feature, and a phone profile feature have more discriminative information for the three classes.

5.4. Five submitted models

The contest rule allows us to submit five different predictions and the one with the best performance will be used for ranking. We submit the four best models from the three classifiers (L1-LogReg, GBT and RF[5]). For the fourth one, we submit an ensemble of their results using simple bagging [34]. For each classifier, we use three models trained by different parameter settings. Therefore, we have $4 * 3 = 12$ models in total. The ensemble is simply a majority voting among the 12 predictions; in case of a tie, we use the label predicted by the best GBT model. We also submit a sub-optimal prediction as the fifth prediction, which is predicted by a GBT model on only 50 unconditional features. The best 10-fold CV accuracy by GBT on the unconditional feature set is 65.3%. We include this sub-optimal model because all other four submissions use too many features and there is a small possibility that the training set and testing set have some distribution shift so that models using many features can easily overfit the training data. Therefore we also submitted a model using only 50 unconditional feature well hedges the risk of overfitting. Table 4 shows the final accuracy on the test dataset released by Nokia Research.[6] Our accuracy is 66%, though much better than the second place's 61%. As presented in [35,36], the top three teams used different strategies to extract features. In both [35,36], the authors manually defined features, which are a subset of our unconditional features. Our improvement over theirs can be viewed as the improvement of the Conditional Feature method over the unconditional features.

5 SVM performs worse than other classifiers; thus we did not submit any single model from SVM.

6 http://research.nokia.com/page/12362.

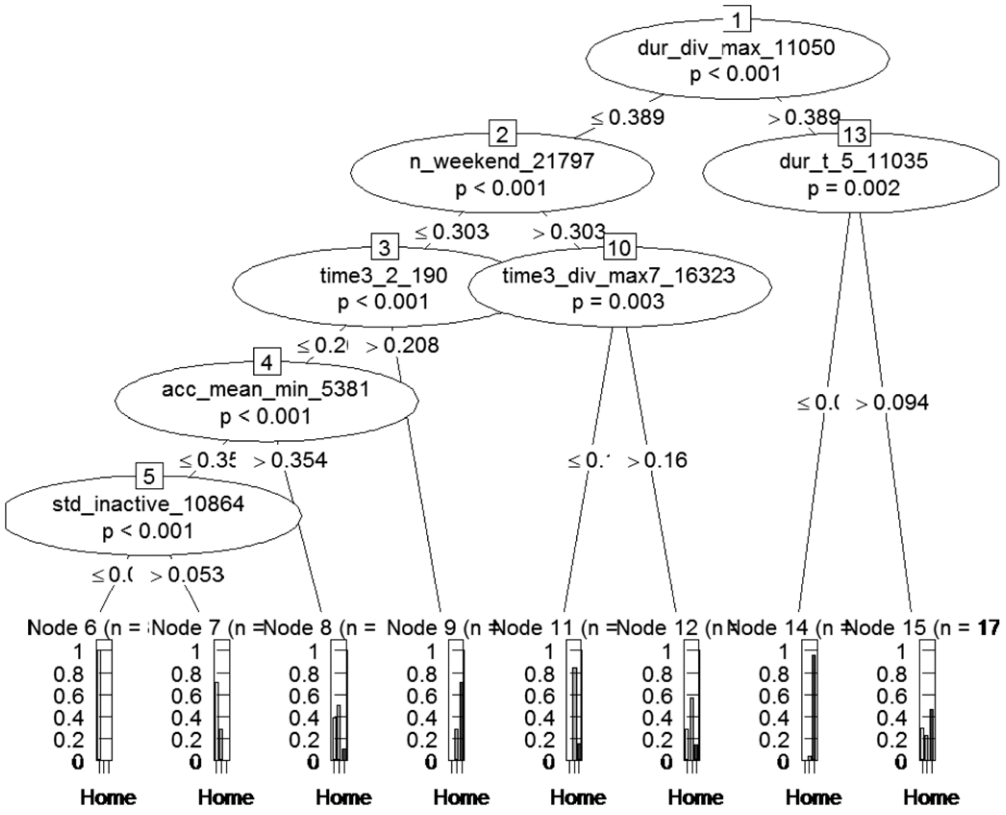

Fig. 5. A decision tree for three major class labels. Illustration of what are the most useful features.

Table 4
Accuracy on the test dataset.

Team	Accuracy (%)
HKUST [37]	66
National Cheng Kung University [35]	61
Institute of New Imaging Technologies [36]	60

In Fig. 6, we plot the histogram of our prediction over the 3043 testing samples. The number of homes is about 100, which is reasonable for 30 users if we consider that there are many places in the 3043 that belong to none of the 10 classes. We found that a lot of instances are classified as 4/"transportation". After checking these instances, we find that most of these places actually have no sensor data records, e.g., no WiFi signals and no accelerometer values are recorded during the stay of these places. For these places, only time-related features, e.g. average interval length, are used for classification. In training data, places with short time intervals are labeled as transportation, and this seemingly abnormal finding well confirms the statistics in the training data.

6. Tools and computation

As we have shown in Fig. 1, there are usually several, sometimes many, iterations through the four steps. It is very hard to extract a good set of features at the first iteration, and writing the correct numerical code for feature construction is not straightforward either. Considering that the data size in Nokia MDC is approximately 70 GB of raw text, we describe briefly in this section how our feature construction program is written and why it is very fast.

Our feature construction program is written in F# and C#. Because the feature construction is done user by user, we can easily archive data parallelism, which is extremely convenient in F# [38]. Generating all the 2 796 200 features for both training and testing dataset costs about 1 h on an 8-core Intel Xeon server.

The size of uncompressed data files is 70 GB, with training 50 GB and testing 20 GB. Except for the FFT for calculating accelerometer features, computing other features is IO-bounded, not CPU-bounded. Modern IO devices usually support hundreds, if not thousands, concurrent accesses. To boost the IO speed, we use F#'s asynchronous programming model [39] to start many parallel async tasks. The number of tasks is decided by .NET Runtime and is far more than the number of CPU cores. By using this programming model, we are able to load IO and CPU both near to 100%. While if we use a simple

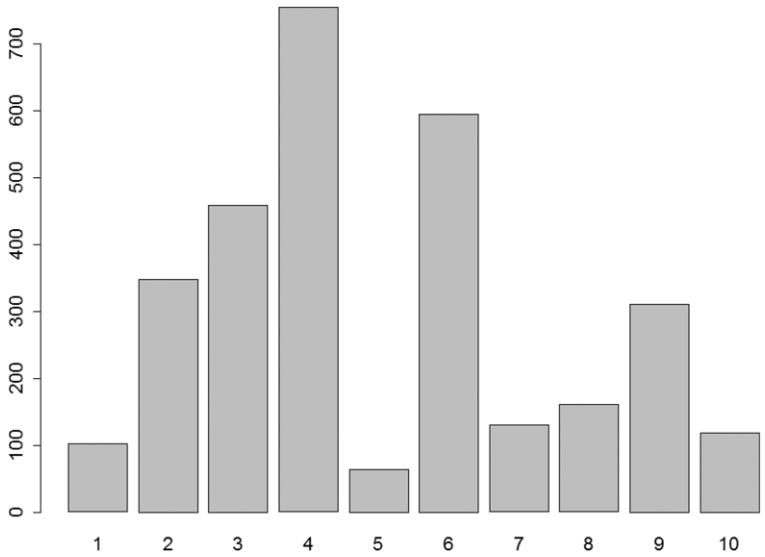

Fig. 6. The 3043 testing predictions over 10 categories by L1-LogReg.

multi-threaded programming model, e.g., using 8 parallel threads and using the single thread for both IO and CPU, the CPU is not fully utilized. A similar strategy is also used in svdfeature,[7] a software developed by the SJTU team to win the recommendation task in KDDCup 2012 [17].

The fast feature construction program in F# gives us the advantage to quickly iterate the whole feature engineering process, from debugging the programs, to constructing more discriminative features, and to evaluating more model building strategies.

7. Conclusion

The task 1 of Nokia MDC requires a board range of data mining skills: Feature Construction, Feature Selection, and Classifier Building. In this task, we found that each step is vital for accurate prediction, and working through all the three steps greatly sharpens our ability to apply data mining algorithms to real world applications.

1. In the feature construction phase, we use the Conditional Feature Engineering technique to construct many features under different conditions, mostly time.
2. In the feature selection phase, we need to reduce the tens of thousands of features to a small set so that the subsequent classifier building is robust and fast.
3. In the classifier building phase, we have found that different classifiers have a noticeable difference in classification accuracy. We have found that L1-LogReg and GBT get the best 10-fold CV accuracy on training data. LogReg classifiers are suitable for situations where the number of instances is smaller than the number of attributes [40].

In summary, we find that the features conditioned on fine-granularity time intervals are very helpful for place category classification and our conditional feature construction effectively constructs these features in a principled way.

Acknowledgments

We thank the support of Hong Kong RGC GRF Projects 621010 and 621211. We thank Xing Xie, Nathan N. Liu, Liuhang Zhang, and Derek Hao Hu for discussions.

References

[1] J.K. Laurila, D. Gatica-Perez, I. Aad, J. Blom, O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, M. Miettinen, The mobile data challenge: big data for mobile computing research, in: Mobile Data Challenge by Nokia Workshop, in conjunction with International Conference on Pervasive Computing, 2012.
[2] J. Zhuang, T. Mei, S.C.H. Hoi, Y.-Q. Xu, S. Li, When recommendation meets mobile: contextual and personalized recommendation on the go, in: Ubicomp, 2011, pp. 153–162.
[3] A. Niculescu-Mizil, C. Perlich, G. Swirszcz, V. Sindhwani, Y. Liu, P. Melville, D. Wang, J. Xiao, J. Hu, M. Singh, W.X. Shang, Y.F. Zhu, Winning the kdd cup orange challenge with ensemble selection, Journal of Machine Learning Research—Proceedings Track 7 (2009) 23–34.

[7] http://apex.sjtu.edu.cn/apex_wiki/svdfeature.

[4] D. Lian, X. Xie, Learning location naming from user check-in histories, in: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS'11, ACM, New York, NY, USA, 2011, pp. 112–121.
[5] M. Ye, D. Shou, W.-C. Lee, P. Yin, K. Janowicz, On the semantic annotation of places in location-based social networks, in: KDD, 2011, pp. 520–528.
[6] X. Cao, G. Cong, C.S. Jensen, Mining significant semantic locations from GPS data, Proceedings of the VLDB Endowment 3 (1) (2010) 1009–1020.
[7] J. Liu, O. Wolfson, H. Yin, Extracting semantic location from outdoor positioning systems, in: MDM, 2006, p. 73.
[8] Y. Chon, N.D. Lane, F. Li, H. Cha, F. Zhao, Automatically characterizing places with opportunistic crowdsensing using smartphones, in: Proc. 14th Int. Conf. Ubiquitous Computing, UbiComp'12, ACM, 2012.
[9] E.M. Tapia, S.S. Intille, K. Larson, Activity recognition in the home using simple and ubiquitous sensors, in: Pervasive Computing, 2004, pp. 158–175.
[10] J. Yin, X. Chai, Q. Yang, High-level goal recognition in a wireless lan, in: Proc. of National Conference on Artificial Intelligence, AAAI, 2004, pp. 578–584.
[11] D.J. Patterson, L. Liao, D. Fox, H.A. Kautz, Inferring high-level behavior from low-level sensors, in: Proc. of International Conference on Ubiquitous Computing, UbiComp, 2003, pp. 73–89.
[12] E. Miluzzo, N.D. Lane, K. Fodor, R.A. Peterson, H. Lu, M. Musolesi, S.B. Eisenman, X. Zheng, A.T. Campbell, Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application.
[13] P. Domingos, A few useful things to know about machine learning, Communications of the ACM 55 (10) (2012) 78–87.
[14] D. Shen, R. Pan, J.-T. Sun, J.J. Pan, K. Wu, J. Yin, Q. Yang, Query enrichment for web-query classification, ACM Transactions on Information Systems 24 (3) (2006) 320–352.
[15] T. Chen, N.N. Liu, Q. Yang, et al. Kddcup 2011 Workshop, in: Informative Ensemble of MultiResolution Dynamic Factorization Models, 2011.
[16] B. Hu, N.N. Liu, W. Chen, Workshop on web Search Click Data, WSCD'12, in: Learning from Click Model and Latent Factor Model for Relevance Prediction Challenge, 2012.
[17] T. Chen, L. Tang, Q. Liu, D. Yang, S. Xie, X. Cao, C. Wu, E. Yao, Z. Liu, Z. Jiang, C. Chen, W. Kong, Y. Yu, Combining factorization model and additive forest for collaborative followee recommendation, in: KDDCup'12 Workshop, 2012.
[18] J. Lin, A. Kolcz, Large-scale machine learning at twitter, in: SIGMOD Conference, 2012, pp. 793–804.
[19] J. Langford, Parallel machine learning on big data, ACM Crossroads 19 (1) (2012) 60–62.
[20] W. Pan, E. Zhong, Q. Yang, Transfer learning for text mining, in: Mining Text Data, 2012, pp. 223–257.
[21] S. Reddy, M. Mun, J. Burke, D. Estrin, M.H. Hansen, M.B. Srivastava, Using mobile phones to determine transportation modes, ACM Transactions on Sensor Networks (TOSN) 6 (2) (2010).
[22] Y. Zhu, Y. Arase, X. Xie, Q. Yang, Bayesian nonparametric modeling of user activities, in: The 13th International Conference Ubiquitous Computing, UbiComp 2011, TDMA'11: Workshop on Trajectory Data Mining and Analysis, 2011.
[23] G. Chittaranjan, J. Blom, D. Gatica-Perez, Personal and ubiquitous computing (published online), in: Mining Large-Scale Smartphone Data for Personality Studies, December, 2011.
[24] K. Kira, L.A. Rendell, A practical approach to feature selection, in: Proceedings of the Ninth International Workshop on Machine learning, ML'92, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992, pp. 249–256.
[25] S. Menard, Applied Logistic regression Analysis, Vol. 106, Sage Publications, Incorporated, 2001.
[26] J.A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Processing Letters 9 (3) (1999) 293–300.
[27] A. Liaw, M. Wiener, Classification and regression by randomforest, R News 2 (3) (2002) 18–22.
[28] J. Ye, J.-H. Chow, J. Chen, Z. Zheng, Stochastic gradient boosted distributed decision trees, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, ACM, 2009, pp. 2061–2064.
[29] R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in: ICML, 2006, pp. 161–168.
[30] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: a library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.
[31] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27. Software available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[32] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, ACM SIGKDD Explorations Newsletter 11 (1) (2009) 10–18.
[33] F. Pan, T. Converse, D. Ahn, F. Salvetti, G. Donato, Feature selection for ranking using boosted trees, in: CIKM, 2009, pp. 2025–2028.
[34] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[35] C.-M. Huang, J.J.-C. Ying, V.S. Tseng, Mining users' behavior and environment for semantic place prediction, in: Mobile Data Challenge by Nokia Workshop, in conjunction with International Conference on Pervasive Computing, 2012.
[36] R. Montoliu, A. Martinez-Uso, J. Martinez-Sotoca, Semantic place prediction by combining smart binary classifiers, in: Mobile Data Challenge by Nokia Workshop, in conjunction with International Conference on Pervasive Computing, 2012.
[37] Y. Zhu, E. Zhong, B. Wu, Feature engineering for place category classification, in: Mobile Data Challenge by Nokia Workshop, in Conjunction with International Conference on Pervasive Computing, 2012.
[38] Y. Zhu, T. Petricek, Real-World Functional Programming (Online Book), in: Numerical Computing in F#, Manning Publications Co. and Microsoft, 2011 (Chapter 4).
[39] D. Syme, T. Petricek, D. Lomov, Padl'11, in: The F# Asynchronous Programming Model, 2011.
[40] S. Rosset, G. Swirszcz, N. Srebro, J. Zhu, l1 regularization in infinite dimensional feature spaces, in: Proceedings of the 20th Annual Conference on Learning Theory, COLT'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 544–558.