Feature Engineering for Place Category Classification

Yin Zhu, Erheng Zhong, Zhongqi Lu, and Qiang Yang
Hong Kong University of Science and Technology
{yinz, ezhong, zluab, qyang}@cse.ust.hk

ABSTRACT
MDC Task 1 is to infer the category of a place using the smartphone sensing data obtained at the place. We formulate this problem as a standard supervised learning task: 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 and feature selection are very effective for this task. In particular, we have proposed a feature engineering technique, Conditional Features (CF), a general method for domain-specific feature extraction. 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 minutes, 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-regularied Logistic Regression. Besides the good performance, we also report briefly our experience of using F# language in dealing with large-scale (~70GB raw data) feature processing.

Author Keywords
Feature Engineering, Domain Knowledge, Feature Selection, Classification.

INTRODUCTION

MDC Task 1 [1] is to infer the category of a place, therefore is a classification task in the general sense. The discriminative information lies in the context data recorded at the place. For example, using time context alone, we would be able to classify working places and homes quite accurately because after midnight we usually stay at home. In addition to time context, calllog 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.

Our main strategy is to extract as many useful features as we can from the sensor data, and build good classifiers using these features. By useful, we mean features that have discriminability among the ten location categories. Once the features are generated, we use state-of-the-art classifiers, e.g. SVM and Decision Trees, to decide how to form rules for place category classification.

This classification task is different from classification tasks in previous data competitions. For instance, KDDCUP 2009 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, rather than to find the most effective way to tackle a specific data mining task. Nokia MDC releases all the sensor data in the raw form, which provides us the possibility to extract best features ourselves.

Because this competition does not have an online leaderboard or a discussion forum, we have no knowledge of what other teams are doing. If most of other teams choose to use traditional classification[2] as we do, then feature extraction 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 time is mainly spent on feature engineering – for each kind of sensing data, we use our domain knowledge and the state-of-the-art feature extraction methods in the literature to generate as many features as possible.

In particular, we have designed a method, *Conditional Feature* (CF), to explore the dependency and correlation between features. 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 we call it as 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. Conditional feature method helps to calculate multiple versions of average WiFi signal strengths. Each version is on some condition, e.g. whether the WiFi strength is in a trusted stay/interval. The most important

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

[2] We have considered using other methods, e.g. Conditional Random Fields (CRFs) to explore the dependency structure among the class labels. However, due to time constraint, we were not able to generate a model with close accuracy to our best classification model.

condition is time – whether the WiFi strength is recorded during weekend, or during a morning, etc. By using the CF method here, 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 that are conditioned on fine-granary 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 features. And in our experiments, we find that time conditions are very useful in place type classification.

2. We analyze what features are useful for place type prediction. Although most of the features are already proposed in the literature, our work is a systematic analysis on their usefulness in this specific task, therefore provides important insights on place type prediction.

CONDITIONAL FEATURES

In this section, we formally describe the Conditional Feature method.

For a user *u* and a place *p*, we may collect all the smartphone data recorded during user *u*'s visits to place *p*. The sensor data include accelerometer strength, call logs system status, etc. We extract features from these sensor logs using common sense knowledge. In particular, we distinguish the condition under which the sensor log is recorded. These conditions include:

- Weekday/weekend, dw
- Under trust interval or not, $trust$
- Time of the day, td

These three conditions are independent to each other. Let's continue to use the *average WiFi signal strength* feature to illustrate the idea of conditional features. 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_{wifi_str}, and call it an *Unconditional Feature*.

To calculate *conditional* versions of v_{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:

$v_{wifi\_str}|(td, dw, trust)$, $v_{wifi\_str}|(dw, trust)$

$v_{wifi\_str}|(td, trust)$, $v_{wifi\_str}|(td, dw)$

$v_{wifi\_str}|(td)$, $v_{wifi\_str}|(dw)$, $v_{wifi\_str}|(trust)$

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 the different numbers from 4 (every 6 hours) to 144 (every 10 minutes), and we choose the best number by cross validation. For one feature, the 8 combinations of conditions generate $(8 + 9\,ntd)$, where ntd is the number of conditnal time intervals during a day, e.g. when the time interval is set to 30 minutes, $ntd = 48$.

FEATURE ENGINEERING

In the previous section, we use the average WiFi signal strength feature to introduce Conditional Features Method. In this section, we describe in details more features.

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-minute duration for some user. Thus any feature for user-place pair *(u, p)* is extracted from the context raw data within this supper interval.

In the following, we list important features for each type of context data.

Time features

We first calculate the summary statistics of the staying durations at this place. They are average, variance, min, max, and the second maximum. One of the summary statistics is average value of duration, which is an obvious feature to classify many places, say home and a coffee shop.

Accelerometer features

Each data point from 3D accelerometer is a (*x,y,z*) tuple, and it is common practice to use its signal strength:

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

where *680* is the unit gravity in the Nokia phone's accelerometer [3]. For a one-minute sequence of accelerometer strength, we extract the following six features:

- average
- variance
- energy (average of squares)
- sum of FFT coefficients
- weighted sum of FFT coefficients

In a previous study [4], we found that these six features are very effective to discriminate people's daily activities.

Application features

The numbers of application "close", "foreground", "started", and "view" are counted.

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

Bluetooth and Wlan features

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

Calllog features

We extract most of the calllog features described in Table 3 in [2], a previous study by Nokia. 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.

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.

Media feature

We calculate the number of songs the user has listened to at a place. We also calculate the time length.

Bag-of-Words (BoW) features

The counts such as how many times the user has used "Message" application and how many times the user has scanned a specific MAC address during his/her stay at a place may also 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*.

There are two important post-processing steps on the number features listed above.

- **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. For these count features, we create two extra features 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.
- **Construct Conditional Features**. Each feature has 8 conditional variants. As calculated before, each feature has $(8 + 9\,ntd)$ conditional versions.

FEATURE SELECTION AND CLASSIFIER BUILDING

Feature selection

Using the conditional feature technique, we generate many features, approximately two million when the conditional time interval is set to 30 minutes. Unrelated features may hurt the classification accuracy for many classifiers; and building a classifier with many feature costs a long computational time. Therefore we need feature selection, 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. Considering the number of training samples is 366, the resulting data table is manageable by most of the classifiers.

Second, we use the Relief feature selection method in Weka[4] to select 50 to 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. We can control the number of features with non-zero coefficients by changing the regularization coefficient C in L1-LogReg.

Classifiers and parameters

We use four state-of-the-art classifiers, one of which is from linear-classifier family, one of which is a kernel-based method, and two of which are from decision tree family.

Logistic regression (LogReg). We use LibLinear package, which implements both L1-regularized and L2-regularized Logistic Regressions. LogReg has only one parameter, the regularization coefficient *C*. We vary this value from 10^{-6} to 10^{3}.

Support Vector Machines (SVM). We use kernel SVM implemented in LibSVM package. We use three kernels, linear kernel, polynomial kernel, and RBF kernel. The regularization coefficient C follows the same setting as that in LogReg. 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}.

Gradient Boosted Trees (GBT). We used an in-house gradient boost tree implemented by Nathan N. Liu, who also uses it successfully in his winning solutions in KDDCup'2011 [6] and WSDM Challenge 2012 [7]. 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 CV accuracy at each iteration; therefore we may choose best CV accuracy from the first 1000 iterations.

RandomForest (RF). We use the implementation in Weka. We set the number of random features from $0.1\sqrt{m}$ to $10\sqrt{m}$, where m is the number of features.

Unbiased classification

In Figure 1, we plot the label distribution from the total 336 training place IDs. Although the data is unbalanced, we decided to use unit weight over class labels. The main reason for using unit weight over all class labels is that the evaluation metric is classification accuracy, which is defined as

[4] www.cs.waikato.ac.nz/ml/weka/.

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

Assuming that the places in the testing data have similar labeling distribution with that of the training data, we should take majority classes and minority classes with equal weights.

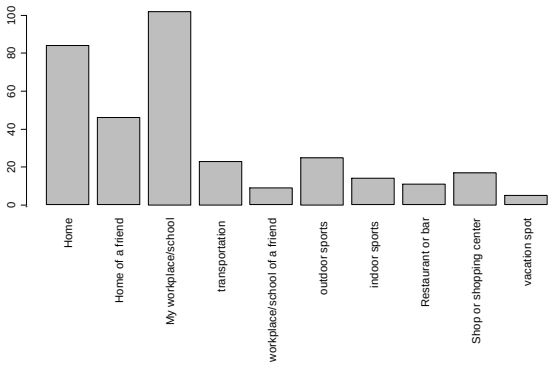

Figure 1. Class Distribution Among the 366 Training Samples.

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 as the evaluation criterion to select 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 predicted model for each fold when it is used as testing is trained from data of different users in that fold.

The effectiveness of feature selection

When we set the conditional time interval to 30 minutes, 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 Figure 2, we plot the 10-CV accuracies of LogReg models on two sets of features, the full set and 2000 of them selected by Relief method. When we change the regularization coefficient C in LogReg models, the models trained on the full feature set is very 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 on the models trained on the good feature subset are quite stable even for extreme C values.

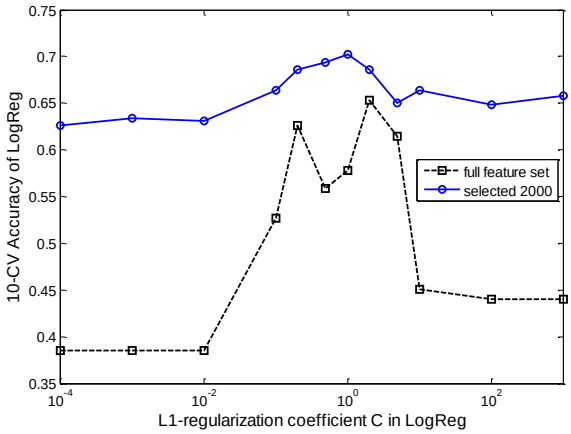

Figure 2. Effectiveness of Feature Selection.

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. Please see the 10-fold CV results below for details.

10-fold CV results

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

Table 1. Performance of Different Classifiers

Classifiers	# 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

In the table, 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 be not suitable for kernel-based methods.

The size of the conditional time interval

In all the above experiments, we set the conditional time interval to 30 minutes. To study the influence of the size of

it, we change it from 6 hours to 10 minutes, and plot the accuracy changes of two classifiers LogReg and GBT in Figure 3. The feature selection method used in both classifiers are LogReg with C=1. To illustrate the usefulness of conditional features, we also plot the accuracy of the unconditional features as a reference.

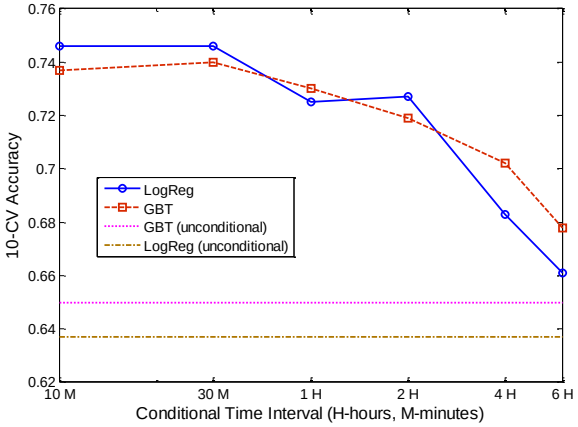

Figure 3. Performance with Different Lengths of Time Interval.

Most useful features

The GDT classifier supports to rank the contribution of features by looking at their occurring frequency in the ensembed trees. Table 2 shows the most useful features and their scores.

Table 2. Weights of Different Features.

Morning 8-10am	100.0
mac_r1	71.57
Ratio of Sunday	67.47
Evening 8-10pm	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

It would be illustrative to show some explainable classification rules. We build a classification model for the three majority classes (Home/WorkPlace/Other's Home) using R's **party** package, and plot the decision tree in Figure 4.

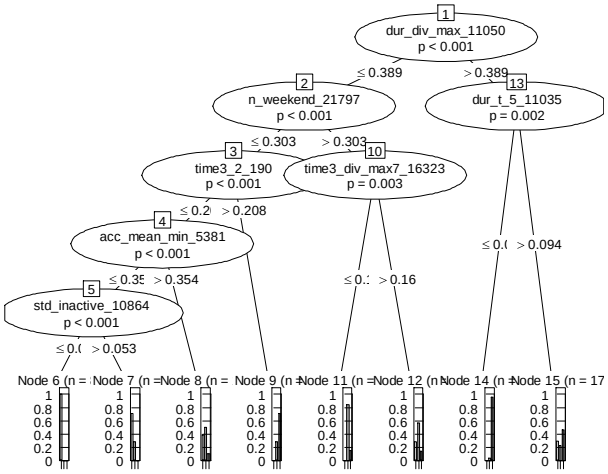

Figure 4. A Decision Tree for Three Major Class Labels. Illustration of What Are The Most Useful Features.

FIVE SUBMISSION MODELS

The contest rule allows us to submit five different predictions and the one with 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. 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 this dataset is 65.3%. We include this suboptimal model because all other four submissions use too many features and there is a sight possibility that the training set and testing set have some distribution shift so that models using many features can easily overfit the training data.

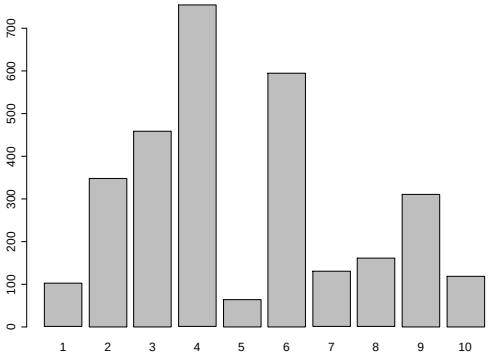

Figure 5. The 3043 Testing Predictions over 10 Categories by L1-LogReg.

[5] SVM performs worse than other classifiers, thus we don't submit any single model from SVM.

In Figure 5, we plot the histogram of our prediction over the 3043 testing samples. We found that a lot of instances are classified as 4/"transportation". By looking at those instances, we found that most of them actually have no sensor data recorded in their time intervals defined `visit_sequence_10min.csv` files. 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.

TOOLS AND COMPUTATION

Our feature extraction program is written in F# and C#. Because the feature extraction is done user by user, we can easily archive data parallelism, which is extremely convenient in F# [5]. Generating all the 2,796,200 features for both training and testing dataset costs about 1 hour on an 8-core Windows server.

The size of uncompressed data files is ~70GB, 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 [8] 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 traditional concurrent programming model, e.g. simply using 8 threads and use the same thread for both IO and CPU, the CPU is not fully loaded.

The fast feature extraction gives us the advantage to quickly debug the programs, construct more discriminative features and evaluate more model building strategies.

CONCLUSION AND DISCUSSIONS

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

1. In feature extraction phase, we use conditional feature engineering technique to extract many features under different conditions, mostly time.
2. In 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 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 [9].

In summary, we find that time dependent features are very helpful for place category classification and our conditional feature extraction effectively extracts these features in a principled way.

ACKNOWLEDGEMENT

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. Juha K. Laurila, Daniel Gatica-Perez, Imad Aad, Jan Blom, Olivier Bornet, Trinh-Minh-Tri Do, Olivier Dousse, Julien Eberle, and Markus Miettinen. The Mobile Data Challenge: Big Data for Mobile Computing Research. In Proc. Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf.. on Pervasive Computing, Newcastle, June 2012.
2. G. Chittaranjan, J. Blom, and D. Gatica-Perez. Mining Large-Scale Smartphone Data for Personality Studies. Personal and Ubiquitous Computing. Published online Dec 2011.
3. Rich Caruana, and Alexandre Niculescu-Mizil. An empirical comparison of supervised learning algorithms. ICML'06.
4. Yin Zhu, Yuki Arase, Xing Xie and Qiang Yang. Bayesian Nonparametric Modeling of User Activities. The 13th International Conference Ubiquitous Computing (UbiComp 2011), TDMA'11: Workshop on Trajectory Data Mining and Analysis.
5. Yin Zhu and Tomas Petricek. Numerical Computing in F#. Chapter 4 of Real-World Functional Programming Online Book, Manning Publications Co. and Microsoft, 2011.
6. Tianqi Chen, Nathan N. Liu, Qiang Yang, et. al. Informative Ensemble of MultiResolution Dynamic Factorization Models. KDDCup 2011 Workshop, 2011.
7. Botao Hu, Nathan N. Liu, and Weizhu Chen. Learning from Click Model and Latent Factor Model for Relevance Prediction Challenge. Workshop on Web Search Click Data (WSCD'12), 2012.
8. Don Syme, Tomas Petricek, Dmitry Lomov. The F# Asynchronous Programming Model. PADL'11.
9. Saharon Rosset, Grzegorz Swirszcz, Nathan Srebro, Ji Zhu: l1 Regularization in Infinite Dimensional Feature Spaces. COLT 2007: 544-558.